基于springboot的音乐分享平台

摘　　要

随着人们生活节奏的加快，如何利用碎片化时间进行最大化的放松身心，成为了每个人比较关心的问题。听音乐已然成为了很多人的选择。怎么更好的找到自己喜欢的音乐，在合适的场景下去怎么选择合适的音乐，遇到自己喜欢的音乐，怎么管理等问题，都是人们在听歌经常遇到的问题。

音乐在线分享研究的重点就是收集用户听音乐的喜好，帮助用户更好的管理自己喜欢的音乐，用户也可以根据自己的喜好来选择想听的音乐，也可以将自己的听过的音乐进行分类，方面在不同的场景下进行切换。本系统也会记录用户的歌曲评论，用户的浏览历史，用户的听歌类别等信息。在这个设计中使用的springboot和vue分别为前后端的框架来构建整体的系统。使用mysql来对数据的内容进行存储。主要开发的工具使用的是idea编辑器。

本系统主要还是基于RBAC模型来进行权限控制的。主要区分三个角色，游客，普通用户，管理员三个角色。 客户端主要包含的功能有： 登录，注册，搜索，收藏音乐，新建歌单，评论，播放音乐，查看浏览历史等功能。后台主要功能： 用户信息管理，歌曲信息管理，歌单信息管理，歌手信息管理，角色授权，评论信息管理以及信息统计等功能。

关键词**：springboot mysql vue RBAC 音乐在线分享**

**ABSTRACT**

With the acceleration of people's life rhythm, how to use fragmented time to maximize relaxation has become a problem that everyone is more concerned about. Listening to music has become the choice of many people. How to find the music you like better, how to choose the right music in the right scene, how to manage the music you like, and how to manage them are all problems that people often encounter when listening to music.

The focus of online music sharing research is to collect users’ preferences for listening to music, to help users better manage their favorite music, users can also choose the music they want to listen to according to their preferences, and they can also classify the music they’ve listened to. , The aspect is switched in different scenarios. The system will also record the user's song reviews, user's browsing history, user's listening song category and other information. The springboot and vue used in this design are the front-end and back-end frameworks to build the overall system. Use mysql to store the content of the data. The main tool developed is the idea editor.

This system is mainly based on the RBAC model to carry out authority control. It mainly distinguishes three roles, tourists, ordinary users, and administrators. The client's main functions include: login, register, search, favorite music, create a new playlist, comment, play music, view browsing history and other functions. Main background functions: user information management, song information management, playlist information management, singer information management, role authorization, comment information management, and information statistics.

Keywords: springboot mysql vue RBAC music online sharing

目录

[一 可行性分析 6](#_Toc71670923)

[1.技术可行性 6](#_Toc71670924)

[2 经济可行性 7](#_Toc71670925)

[3 资源可用性 7](#_Toc71670926)

[4 社会因素可行性(市场和政策) 8](#_Toc71670927)

[2 需求分析 8](#_Toc71670928)

[2.1 项目概述 8](#_Toc71670929)

[2.1.1开发意图 8](#_Toc71670930)

[2.1.2 应用目标 8](#_Toc71670931)

[2.1.3 产品描述 9](#_Toc71670932)

[2.1.4 产品功能 9](#_Toc71670933)

[2.2 具体需求 9](#_Toc71670934)

[2.2.1 项目模块划分 9](#_Toc71670935)

[2.2.2 系统模块描述 10](#_Toc71670936)

[2.2.3 系统流程图 11](#_Toc71670937)

[2.2.4 数据流图 13](#_Toc71670938)

[2.2.5 数据字典 13](#_Toc71670939)

[2.2.6 支持信息 16](#_Toc71670940)

[3 系统概要设计 17](#_Toc71670941)

[3.1 系统的整体结构设计 17](#_Toc71670942)

[3.2 系统功能设计 18](#_Toc71670943)

[3.2.1 E-R图 18](#_Toc71670944)

[3.2.2 系统具备功能结构（画一个层次图） 21](#_Toc71670945)

[3.3 系统功能需求和程序之间的关系 21](#_Toc71670946)

[3.4人工处理过程 21](#_Toc71670947)

[3.5接口设计 22](#_Toc71670948)

[3.5.1 用户接口 22](#_Toc71670949)

[3.5.2 外部接口 22](#_Toc71670950)

[3.5.3 内部接口 22](#_Toc71670951)

[3.6 运行设计 23](#_Toc71670952)

[3.6.1 运行模块组合 23](#_Toc71670953)

[3.6.2运行控制 24](#_Toc71670954)

[3.7 系统数据结构设计 25](#_Toc71670955)

[3.8 系统出错设计(待完善) 28](#_Toc71670956)

[3.8.1 出错信息 28](#_Toc71670957)

[3.8.2 系统的补救措施 28](#_Toc71670958)

[4 系统详细设计 28](#_Toc71670959)

[4.1 程序系统的结构(需要一个系统的结构图) 28](#_Toc71670960)

[4.2 系统功能实现 28](#_Toc71670961)

[4.2.1 用户信息角色模块 29](#_Toc71670962)

[4.2.2 歌曲模块实现 31](#_Toc71670963)

[4.2.3 歌单模块实现 33](#_Toc71670964)

[4.2.4 评论模块实现 34](#_Toc71670965)

[4.2.5 信息统计模块实现(这个统计信息的接口还是没有实现) 36](#_Toc71670966)

[4.2.6 系统监控模块实现 36](#_Toc71670967)

[4.2.7 异常设计实现 36](#_Toc71670968)

[5 系统的单元测试和集成测试 36](#_Toc71670969)

[6 总结 37](#_Toc71670970)

# 一 可行性分析

## 1.技术可行性

1 功能说明：

本次系统采用b/s的架构来实现基于springboot的音乐在线分享平台，分为客户端和管理后台。

客户端具备的主要功能： 用户的登录注册，收藏音乐，多维度搜索音乐，音乐的分类别展示，提交评论，创建歌单，上传自己的作品等功能。

后台管理系统具备的功能：用户信息的管理，用户信息的统计。歌曲信息动态管理，歌曲信息的统计，歌手信息管理，热门歌单的推送，用户评论的动态管理，以及系统的监控，用户行为分析等功能。

2 技术分析：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 关键技术项 | 要求 | 现有水平 | 对照分析 | 选择原因 |
| mysql | 熟悉 | 熟悉 | 可以满足需求 | 1.性能卓越服务稳定，很少出现异常宕机 2 开放源代码且无版权制约，自主性强、使用成本低 |
| redis | 熟悉 | 了解 | 可以满足需求 | 1 存储结构丰富  2 基于内存的数据库，速度快 |
| springboot | 熟悉 | 熟悉 | 可以满足需求 | 1 创建独立的Spring应用程序直接嵌入Tomcat，Jetty或Undertow（无需部署WAR文件）  2 尽可能自动配置Spring和第三方库提供生产就绪的功能，例如指标， 运行状况检查和外部化配置完全不需要代码生成，也不需要XML配置（引用官网） |
| Vue | 熟悉 | 了解 | 学习文档，可以达到要求 | 1 渐进式框架，扩展性强  2 比较轻量，可以自底向上逐层应用(引用官网) |

## 2 经济可行性

本次系统主要是为用户提供音频的服务，所以存储和网络就比较重要，所以本次打算使用腾讯云的对象存储来来存储具体的音频文件，使用一个云主机来进行部署。在经济的可用性是可以负担的。

## 3 资源可用性

1.3.1 软件资源

服务器系统平台 Windows10 及以后

客户端系统平台 Windows10 及以后

1.3.2 硬件资源

服务器硬件要求：支持Intel 平台、AMD 平台。双CPU 2.0G 以上，内存2.0G 以上，100M 网卡、硬盘250G 以上，带液晶显示。

客户端硬件要求：wifi功能，双CPU 2.0G 以上，内存2.0G 以上，100M 网卡、硬盘200G 以上，具有音频输出.

1.3.3 时间资源

预计项目开发、研究，与产品测试，所需时间为2个月。

## 4 社会因素可行性(市场和政策)

4.1 法律可行性。

该系统没有侵权或者抄袭等违法情况，也没有被申请过专利，是可行的。

4.2 政策可行性

遵守国家对于音频版权的要求，保护用户作品的版权使用，遵守软件开发的相关的守约以及用户隐私。

4.3 使用可行性

对于系统的使用会涉及到各个类型的人群，所以客户端的界面设计简约友好，满足用户精准搜索音乐，收藏音乐，听音频的需求，操作简单，易用。

服务端的操作人员： 主要是对用户信息， 歌曲信息，歌手信息以及用户评论进行动态管理，后端的可视化界面操作方便，操作人员只要熟悉基本的电脑知识即可，不需要进行专门的培训。

# 2 需求分析

**编写目的**： 随着人们生活节奏的不断加快，对生活的品质要求越来越高，如何利用碎片化的时间来最大化的愉悦身心，放松心情就显得很重要，本系统就是音频管理系统，为用户提供比较丰富的音频资源，用户也可以创建自己的歌单来收集自己喜欢的音乐，最主要的一个功能，就是根据用户听歌习惯，生成用户的听歌分析报告，让用户能实时的了解自己的听歌类型，更好的放松自己的身心。

## 2.1 项目概述

该音乐软件目前已经有完善的功能和需求，研制这个系统是为了给用户以更好的生活体验，和更好的利用音频资源分享自己的生活。

### 2.1.1开发意图

提高人们的碎片化使用时间的效率，提高用户的音频体验。

### 2.1.2 应用目标

通过本系统，能快速帮助用户找到自己喜欢的音乐，并使创作者的音频能进行更好的管理和分析。使好的作品能得到更广泛的传播。使人们能更好的利用音乐放松自己的身心，方便用户更好对自己喜欢的音乐进行分类。

### 2.1.3 产品描述

本产品是一个b/s架构的产品的音乐平台，用户可以方便的使用浏览器来进行访问。存储使用的mysql数据库。并使用vue和用户进行交互，为用户提供良好的交互体验。

### 2.1.4 产品功能

系统主要是根据不同的角色赋予不同的功能：

主要分为游客，普通用户，创作者，系统管理员四个角色，

游客可以查询歌曲，听歌，浏览热门歌曲，歌手等信息。但是不能评论和创建自己的歌单，不能收藏歌曲。

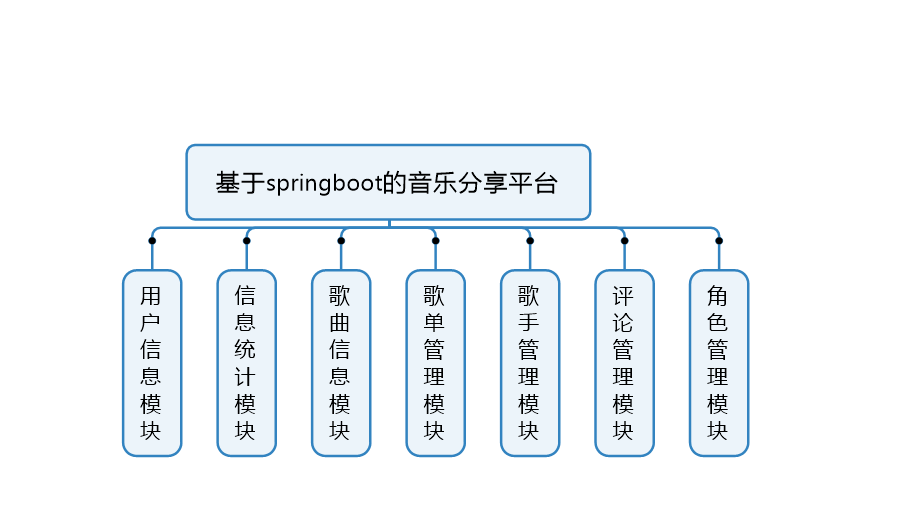
普通用户可以登录，注册，浏览歌曲，收藏歌单，根据自己的喜好创建属于自己歌单，浏览自己的听歌历史，评论歌曲，查看自己的听歌报告等功能。

创作者除了具有普通用户的功能之外，还可以上传自己的作品，并且可以通过登录后台，查看自己的作品的访问量，评论，以及综合评分等维度的信息，可以方便的为创作者提供有关作品比较直观的数据。便于后期的创作。

系统管理员具有最高的权限，具体的可以实现对用户信息的管理，对歌曲的管理，歌单的管理，评论的管理，收藏的管理，歌手信息管理，后端的接口管理，对用户的授权管理，角色管理等功能。可以查看每首歌曲的一个访问情况以及、歌手的地域分布，用户的分布情况。并可以查看系统的实时的监控信息和设置系统的一些配置信息等功能。

## 2.2 具体需求

### 2.2.1 项目模块划分



### 2.2.2 系统模块描述

##### 2.1 用户信息模块：

主要包括用户信息的增加，修改，更新和删除。普通用户可以进行注册和修改自己的信息。查看自己的浏览歌曲的历史等的信息，管理员可以对用户的基本信息进行分析，具体的包括用户的地域，以及用户的性别，出生年龄等的信息。

##### 歌曲管理模块：

对系统中的所有歌曲进行管理，包括歌曲的风格，歌曲地域分布，歌曲的增加，修改和删除，以及歌曲的文件的存储。由于歌曲的文件比较大，本系统采用存储到腾讯云的第三方平台上来进行实现的。增加的了歌曲的各个维度的分类，方便用户进行方便的搜索使用。对部分的热门歌曲的显示做了一层缓存，提高用户的检索效率，有一个更好的系统体验。

歌手管理模块：

歌手信息的增删改查和统计实现，用户可以查询歌手的基本信息，包括简介，代表作，出生地等的信息。每个歌手会关联自己的音乐作品。用户可以根据歌手来搜索自己喜欢的音乐。系统后台也可以统计每个歌手的作品信息以及歌手分布等维度的统计信息。

##### 角色管理模块:

本系统是基于RABC模型来对用户进行权限管理的。角色管理模块主要是对具体的角色信息进行动态的管理，具体包括角色的创建，角色的授权，角色的查询等功能，实现不同角色的可以访问不同的资源。这里的角色授权分为两个维度，一个是菜单的授权，一个是具体的接口的授权。具有更好的灵活性。

##### 歌单管理模块:

将不同的歌曲组成歌单，进行展示，给用户提供更好的检索体验和歌曲资源，系统也会根据用户的听歌内容，每天给用户推送个性化的歌单。实现歌曲内容的最大共享。

##### 评论管理模块:

用户可以对自己喜欢的歌曲进行评论，后台集中对一首歌曲的评论进行管理，包括对评论敏感词汇的过滤，一首歌曲的评论的具体数量。用户评论的点赞数等信息。以及对歌曲评论的动态管理。

##### 接口管理模块:

一般是由管理员进行管理，可以对后台菜单和具体的访问接口进行动态的添加和授权。方便后期的扩展

##### 统计信息模块:

主要用户信息统计，歌手的信息统计，歌曲的信息统计等子功能。

##### 系统监控模块:

主要是监控系统的硬盘信息，系统的cpu的信息，系统的存储等信息。实时的查看整个系统的运行状况。

### 2.2.3 系统流程图

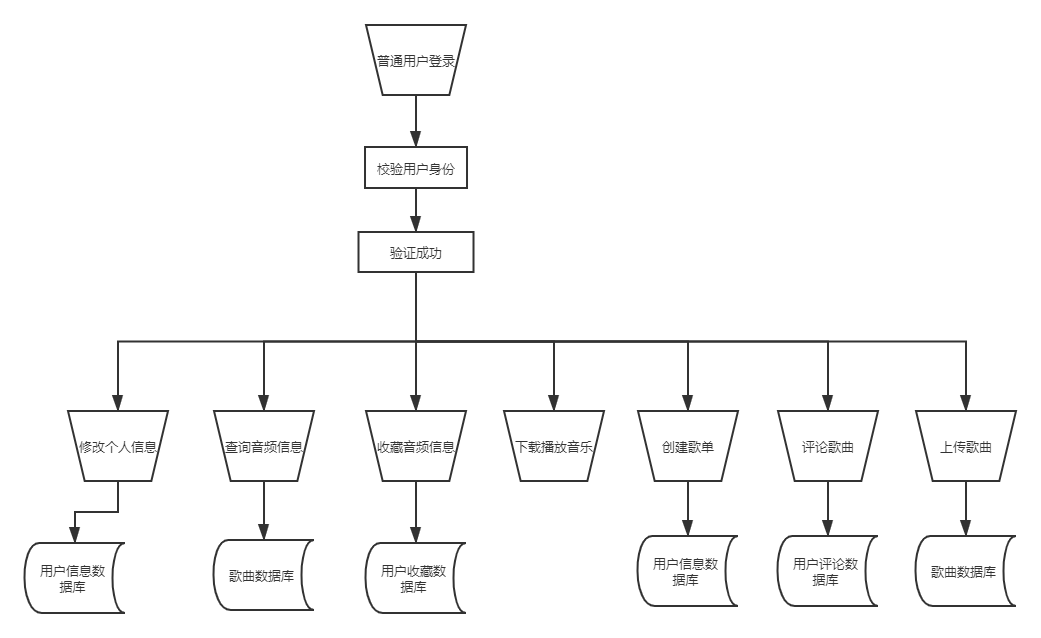


图2-2-1普通用户的流程图

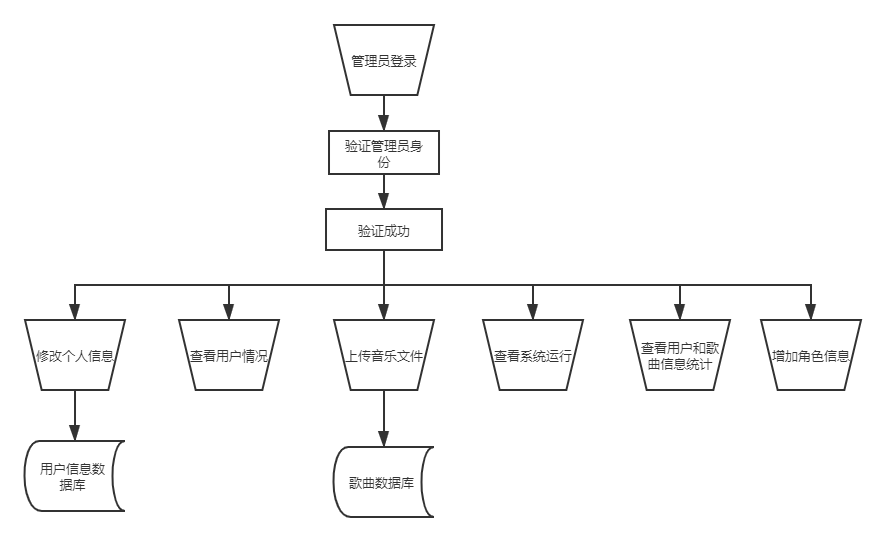


图2-3-2 管理员流程图

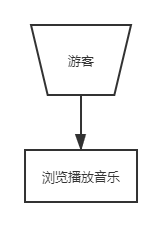


图2-3-4 游客的流程图

### 2.2.4 数据流图

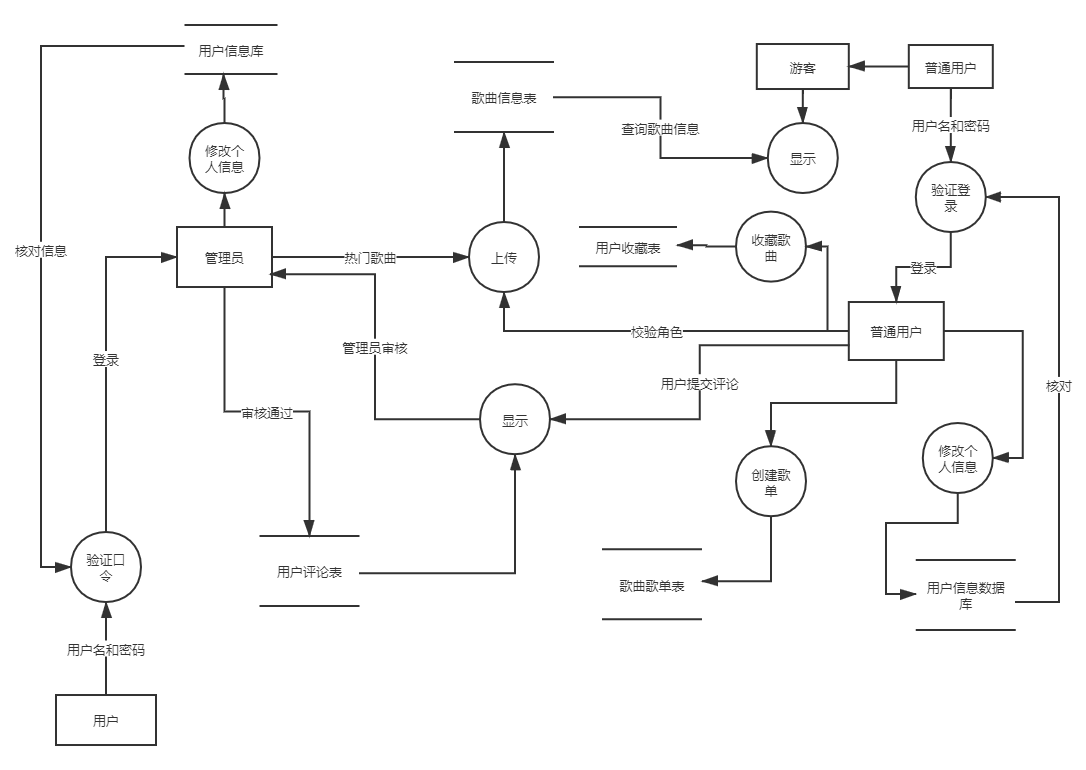


图2-3-5 系统数据流图

### 2.2.5 数据字典

1.数据流描述：

数据流名：用户名密码

数据来源：用户登陆处理

数据去向：验证信息处理。

说明：用户输入用户名和密码

数据组成：用户名+密码

数据流量：小

数据流名：登陆

数据来源：验证信息处理正确

数据去向：根据验证信息和角色返回给用户的显示菜单信息

说明：系统登陆到管理员界面。

数据组成：用户名

数据流量： 小

数据流名：歌曲文件

数据来源：用户上传接口和管理员后台上传界面

数据去向：上传处理

说明：管理员上传热门歌曲或者用户上传作品信息

数据组成： 歌曲文件+歌曲图片

数据流量： 大

数据流名： 用户评论

数据来源：用户评论界面

数据去向：歌曲评论

说明：用户对歌单和歌曲进行评论

数据组成： 用户名+评论内容

数据流量： 小

数据流名：创建歌单

数据来源：用户创建歌单界面

数据去向：用户创建的歌单

说明：用户可以创建自己歌单，将歌曲进行分类

数据组成：用户名+创建歌单信息。

数据流量： 小

数据流名：下载音乐

数据来源： 用户的下载界面

数据去向： 本地存储

说明：用户可以对歌曲下载到本地

数据组成： 用户名+下载歌曲文件

数据流量： 小

数据流名： 用户信息统计

数据来源： 用户操作界面

数据去向：用户信息分析

说明：用户产生的行为数据进行分析

数据组成： 用户名+用户播放歌曲+用户收藏+用户下载歌曲

数据流量： 大

数据流名： 歌曲信息统计

数据来源： 歌曲展示界面

数据去向：歌曲信息统计界面

说明： 对歌曲的信息进行统计

数据组成： 歌曲基本信息+下载的次数+播放的次数

数据流量： 大

**2 数据元素词条描述：**

元素名：角色名

描述：用来区分不同的用户

定义：角色 = 角色+权限

元素名：用户

描述：使用该系统的用户

定义：用户=游客+普通用户+管理员

### 2.2.6 支持信息

元素名：评论

描述：用户的评论内容

定义：评论=评论人+评论时间+评论内容+评论的点赞数

位置：数据库

元素名：歌单

描述：将歌曲进行分类

定义：歌单=歌单名+歌单所属歌曲+歌单描述+歌单风格

位置：数据库

元素名：管理员

描述：系统的管理员，最高的权限

定义：普通用户=用户名+密码+用户头像+用户地域+用户出生日期

位置：数据库

元素名：密码

描述：用来保护用户的信息安全

定义：密码=管理员密码+用户密码

元素名：普通用户

描述 有注册过系统账号的用户

定义：普通用户=用户名+密码+用户头像+用户地域+用户出生日期

位置： 数据库

元素名：歌曲

描述：定义歌曲的基本信息

定义：歌曲=歌曲名+歌曲风格+歌曲的大小+歌曲图片+歌曲的创作者

位置：数据库

元素名：歌手

描述：用来描述歌手的基本信息

定义：歌手 =歌手名+歌手的住址+歌手出生日期+歌手简介+歌手性别

位置：数据库

#### 2.2.6.1硬件环境：

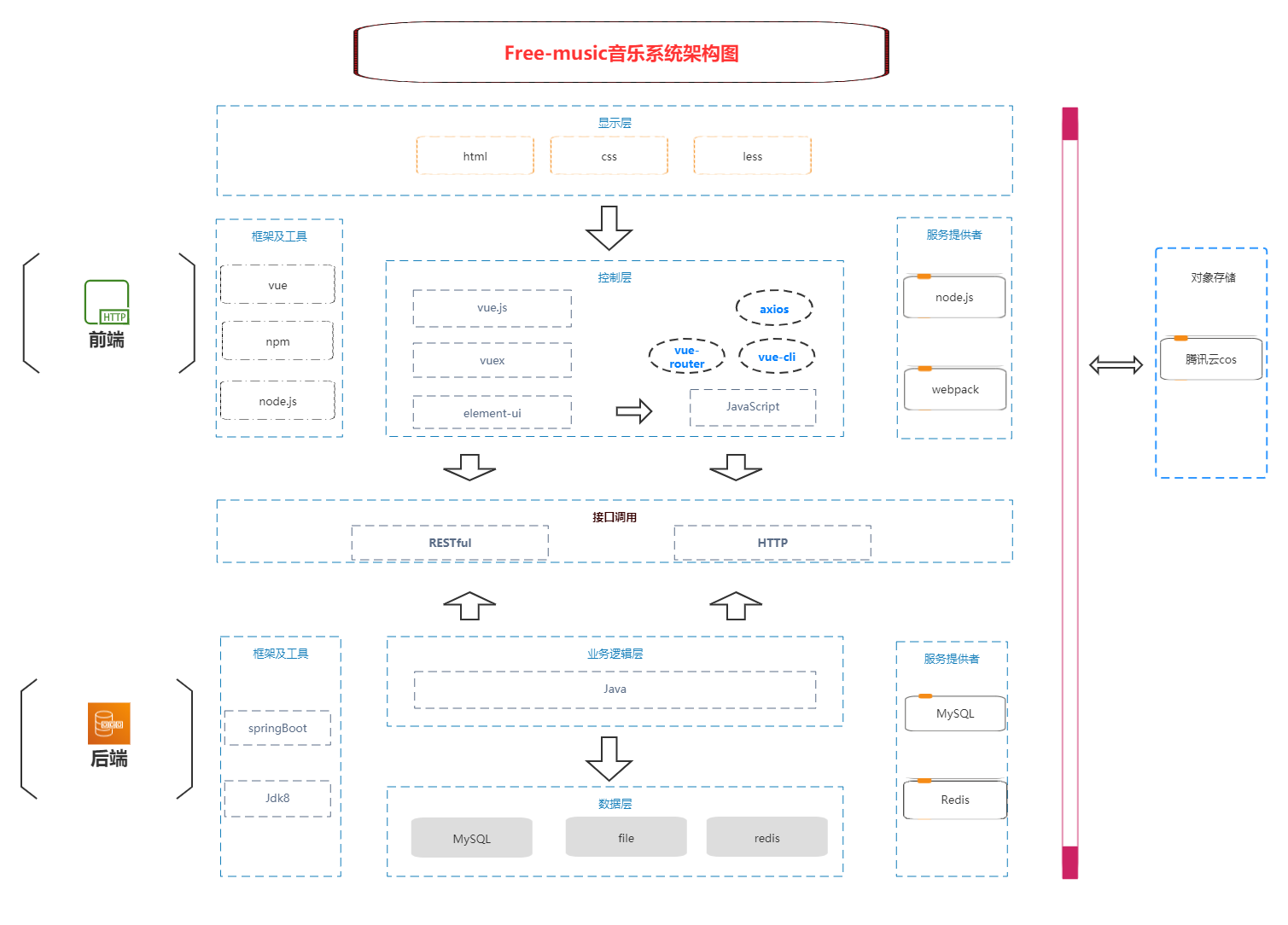
（1）支持Intel 平台、AMD 平台。双CPU 2.0G 以上，内存2.0G 以上，100M 网卡、硬盘250G 以上，带液晶显示。   
2.2.6.2 软件环境

(1) JDK 1.8 mysql5.6 redis5.3 springboot2.2 vue3.0

(2) Window10的操作系统

# 3 系统概要设计

## 3.1 系统的整体结构设计

图3-1 系统整体架构图

## 3.2 系统功能设计

### 3.2.1 E-R图

用户信息和角色之间E-R 图



歌曲-歌单的E-R图



歌手-歌曲信息的E-R图



评论信息的E-R图



用户收藏信息的E-R图



用户和歌单-收藏之间E-R图



用户和歌单-收藏之间E-R图

### 3.2.2 系统具备功能结构（画一个层次图）

需要一个图。

## 3.3 系统功能需求和程序之间的关系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 创建 | 查找 | 修改 | 删除 |
| 维护系统 （管理员） | √ | √ | √ | √ |
| 维护用户信息（管理员） | √ | √ | √ | √ |
| 维护歌曲信息（管理员） | √ | √ | √ | √ |
| 维护歌单信息（管理员） | √ | √ | √ | √ |
| 维护评论信息（管理员） | √ | √ | √ | √ |
| 维护歌手信息（管理员） | √ | √ | √ | √ |
| 维护收藏信息（管理员） | √ | √ | √ | √ |
| 维护角色信息（管理员） | √ | √ | √ | √ |
| 维护统计信息（管理员） | √ | √ | √ | √ |
| 更新个人资料（用户） |  |  | √ |  |
| 上传歌曲 （用户） |  |  | √ |  |
| 查看评论内容（用户） |  | √ |  |  |
| 提交评论 （用户） | √ |  |  |  |
| 收藏歌曲 （用户） | √ |  |  | √ |
| 查询歌曲 (用户) |  | √ |  |  |
| 创建歌单 （用户） | √ |  |  |  |

## 3.4人工处理过程

1 创建歌曲： 用户需要将曲库中大量的歌曲进行导入。

2 创建预设系统歌单： 需要将不同的歌单的信息导入到系统中去。并关联到指定的歌曲中。

## 3.5接口设计

### 3.5.1 用户接口

|  |  |
| --- | --- |
| 和用户交互的命令 | 系统的响应信息 |
| 查询歌曲，歌手信息 | 检索匹配歌曲和歌手的信息 |
| 编辑用户资料 | 更新后的用户资料 |
| 上传歌曲 | 上传歌曲成功后，提示上传成功 |
| 收藏歌曲 | 收藏成功信息 |
| 播放歌曲 | 匹配歌曲资源，开始响应播放 |
| 下载歌曲 | 下载成功信息 |
| 评论歌曲 | 评论成功后，提示评论成功信息 |

### 3.5.2 外部接口

|  |  |  |
| --- | --- | --- |
| 接口类型 | 接口 | 传递信息 |
| mysql数据库接口 | MySQL数据库交互的接口 | 用户信息，歌曲信息，歌手信息，用户评论信息，用户收藏信息，歌单信息 |
| 对象存储接口 | 腾讯云存储交互接口 | 歌曲信息 |
| Redis数据接口 | redis数据库交互的接口 | 用户信息，歌曲信息 |

### 3.5.3 内部接口

|  |  |  |
| --- | --- | --- |
| 接口 | | 传递信息 |
| 维护用户信息 | 添加用户信息 | 用户信息（用户名，用户地区，邮箱，电话，性别） |
| 维护用户信息 | 删除用户信息 | 用户信息（用户名） |
| 维护用户信息 | 修改用户信息 | 用户信息（用户名，用户地区，邮箱，电话，性别） |
| 维护用户信息 | 查询用户信息 | 用户信息（用户名，用户邮箱，电话） |
| 维护用户信息 | 修改用户权限 | 用户信息：（用户名） |
| 维护歌曲信息 | 添加歌曲信息 | 歌曲信息：（歌曲名，歌词，歌曲简介，歌曲海报） |
| 维护歌曲信息 | 删除歌曲信息 | 歌曲信息：（歌曲名） |
| 维护歌曲信息 | 更新歌曲信息 | 歌曲信息：（歌曲名，歌词，歌曲简介，歌曲海报） |
| 维护歌曲信息 | 查询歌曲信息 | 歌曲信息：（歌曲名） |
| 维护歌手信息 | 添加歌手信息 | 歌手信息（歌手名，地区，出生日期，性别，专辑） |
| 维护歌手信息 | 删除歌手信息 | 歌手信息（歌手名） |
| 维护歌手信息 | 更新歌手信息 | 教师信息（教工号，姓名，职称，密码） |
| 维护歌手信息 | 查询歌手信息 | 歌手信息（歌手名） |
| 维护评论信息 | 添加评论信息 | 评论信息（歌曲名，用户名，评论内容） |
| 维护评论信息 | 删除评论信息 | 评论信息（歌曲名和用户名） |
| 维护评论信息 | 更新评论信息 | 评论信息（歌曲名，用户名，评论内容） |
| 维护评论信息 | 查询评论内容 | 评论信息（歌曲名） |
| 维护歌单信息 | 添加歌单信息 | 歌单信息（歌单名，歌单风格，歌单类型，歌单简介） |
| 维护歌单信息 | 删除歌单信息 | 歌单信息（歌单名） |
| 维护歌单信息 | 更新歌单信息 | 歌单信息（歌单名，歌单风格，歌单类型，歌单简介） |
| 维护歌单信息 | 查询歌单信息 | 歌单信息（歌单名） |
| 维护歌单信息 | 生成推荐歌单信息 | 歌曲信息统计（歌曲播放次数，歌曲收藏次数，下载次数） |
| 维护系统信息 | 查询系统运行信息 | 系统信息（内存占用，cpu使用） |

## 3.6 运行设计

### 3.6.1 运行模块组合

施加不同的外界运行控制时所引起的各种不同的运行模块组合如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 创建模块 | 查找模块 | 修改模块 | 删除模块 |
| 管理员添加歌曲信息 | √ |  |  |  |
| 管理员修改歌曲信息 |  | √ | √ |  |
| 管理员删除删除信息 |  | √ |  | √ |
| 管理员添加用户信息 | √ |  |  |  |
| 管理员修改用户信息 |  | √ | √ |  |
| 管理员注销用户信息 |  | √ |  | √ |
| 管理员给用户分配角色 | √ |  |  |  |
| 用户更新个人资料 |  |  | √ |  |
| 用户创建歌单 | √ |  | √ |  |
| 用户提交评论 | √ |  |  |  |
| 用户收藏歌曲 | √ |  |  |  |
| 用户上传歌曲 | √ |  |  |  |
| 用户播放歌曲 |  | √ |  |  |

## 3.6.2运行控制

|  |  |
| --- | --- |
| 运行控制 | 控制方法 |
| 管理员新增歌曲信息 | 管理员填写歌曲歌曲信息，并提交，数据库新增一条记录。 |
| 管理员修改歌曲信息 | 管理员修改指定的歌曲信息，并同步更新到数据库中 |
| 管理员删除歌曲信息 | 管理员通过检索指定的歌曲信息，并删除数据库中的一条记录 |
| 管理员添加歌手信息 | 管理员填写歌手信息并提交，系统在歌手信息表中新增一条歌手信息 |
| 管理员修改歌手信息 | 管理员查找指定的歌手信息并提交修改，系统在歌手信息表中写入修改后的歌手信息 |
| 管理员删除歌手信息 | 管理员找到指定的歌手信息并删除，系统删除该条记录 |
| 管理员添加评论信息 | 管理员提交评论内容，系统新增一条数据库记录 |
| 管理员修改评论信息 | 管理员提交修改内容，系统更新一条记录值 |
| 管理员删除评论新息 | 管理员检索评论内容，系统删除这条记录值 |
| 用户编辑个人信息 | 用户可重新编辑自己的个人信息，并提交，系统更新用户新的信息 |
| 用户创建歌单 | 提交歌单的相关信息 |
| 用户检索歌曲 | 用户输入想要检索的歌曲，系统返回歌曲的详细信息 |
| 用户提交评论 | 用户提交具体的评论内容，系统新增一条评论信息 |
| 用户收藏歌曲 | 用户检索歌曲，并进行收藏，系统新增一条收藏记录。 |

## 3.7 系统数据结构设计

逻辑结构设计要点

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 用户信息 | 用户名 | 文本 | 64 |  |
| 密码 | 文本 | 64 |  |
| 电话 | 文本 | 16 | 唯一标识 |
| 用户出生时间 | 日期 | 16 |  |
| 用户邮箱 | 文本 | 32 |  |
| 用户个性签名 | 文本 | 255 |  |
| 用户性别 | 数字 | 4 |  |
| 用户头像 | 文本 | 255 |  |
| 用户地域 | 文本 | 32 |  |
| 用户状态 | 数字 | 4 |  |
| 创建用户时间 | 日期 | 32 |  |
| 更新用户时间 | 日期 | 32 |  |

用户信息表的逻辑设置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 歌曲信息 | 歌曲名 | 文本 | 64 |  |
| 歌曲简介 | 文本 | 64 |  |
| 歌手名 | 文本 | 16 |  |
| 歌曲图片 | 日期 | 16 |  |
| 歌词 | 文本 | 32 |  |
| 歌曲播放地址 | 文本 | 255 |  |

歌曲信息表的设置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 收藏信息 | 歌曲名 | 文本 | 64 |  |
| 用户名 | 文本 | 64 | 唯一标识 |
| 歌曲名 | 文本 | 16 |  |
| 收藏类型 | 数字 | 16 |  |
| 收藏时间 | 日期 | 32 |  |
| 更新时间 | 文本 | 255 |  |

用户收藏表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 歌手信息 | 歌手名 | 文本 | 64 |  |
| 歌曲性别 | 文本 | 64 |  |
| 歌手出生日期 | 日期 | 16 |  |
| 歌曲图片 | 文本 | 128 |  |
| 所属地域 | 文本 | 32 |  |
| 歌手简介 | 文本 | 255 |  |

歌手信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 歌单信息 | 歌单名 | 文本 | 64 |  |
| 歌单标题 | 文本 | 64 |  |
| 歌单海报 | 文本 | 64 |  |
| 歌单风格 | 文本 | 16 |  |
| 歌单类型 | 数字 | 4 |  |
| 创建时间 | 日期 | 32 |  |
| 更新时间 | 日期 | 255 |  |

歌单信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 角色信息 | 角色名 | 文本 | 64 |  |
| 角色描述 | 文本 | 64 |  |
| 角色状态 | 数字 | 4 |  |
| 角色顺序 | 数字 | 16 |  |
| 创建时间 | 日期 | 32 |  |
| 更新时间 | 日期 | 255 |  |

角色信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 评论信息 | 用户名 | 文本 | 64 | 唯一标识 |
| 歌曲名 | 文本 | 64 |  |
| 用户头像 | 文本 | 64 |  |
| 评论内容 | 文本 | 255 |  |
| 点赞数 | 数字 | 16 |  |
| 创建时间 | 日期 | 32 |  |
| 更新时间 | 日期 | 32 |  |

评论信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 接口路由表 | 接口名 | 文本 | 64 |  |
| 接口父名称 | 文本 | 64 |  |
| 是否为叶子节点 | 数字 | 4 |  |
| 接口路径 | 文本 | 16 |  |
| 接口层级 | 数字 | 4 |  |
| 接口状态 | 数字 | 4 |  |
| 接口排序 | 数字 | 4 |  |
| 创建时间 | 日期 |  |  |
| 更新时间 | 日期 |  |  |

接口的信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | 数据项 | 类型 | 长度 | 备注 |
| 菜单信息表 | 菜单名 | 文本 | 64 |  |
| 菜单名称 | 文本 | 64 |  |
| 是否为叶子节点 | 数字 | 4 |  |
| 菜单路径 | 文本 | 16 |  |
| 菜单层级 | 数字 | 4 |  |
| 菜单状态 | 数字 | 4 |  |
| 菜单排序 | 数字 | 4 |  |
| 菜单图标 | 文本 | 64 |  |
| 是否隐藏 | 数字 | 4 |  |
| 创建时间 | 日期 | 32 |  |
| 更新时间 | 日期 | 32 |  |

菜单信息表

## 3.8 系统出错设计(待完善)

## 3.8.1 出错信息

1 输入的用户名不存在，说明用户的信息不存在，需要用户进行注册。

2 用户名和密码不争取，提示用户该用户可能尚未赋予角色或者密码错误。

3 查询歌曲不存在时，返回查询的信息不存在，请重新查询提示信息

4 用户提交评论出现的敏感词汇，提示信息用户的评论内容不合法，请重新提交评论。

5 用户上传的照片不符合指定的格式，提示信息，照片的格式不支持，仅支持指定的格式，请重新上传。

6 用户上传歌曲作品的文件过大，提示上传的文件的大小超过限制，调整后在进行上传。

7 系统出现网路原因，连接服务端失败，提示连接服务器失败，请重新连接的信息。

## 3.8.2 系统的补救措施

1 网络原因，导致歌曲播放失败，采取的措施： 增加本地缓存。

2 数据库的信息没周一次的备份，防止出现数据丢失的情况

# 4 系统详细设计

## 4.1 程序系统的结构(需要一个系统的结构图)

## 4.2 系统功能实现

由于系统的七个模块中都有涉及基本的增删改查。所以在具体的功能实现的设计。为避免内容重复阐述。只列出每个子模块与其他模块不同的功能实现。

### 4.2.1 用户信息角色模块

用户信息模块核心功能角色区分，用户信息的动态管理，具体包括普通用户的注册，登录，修改资料，管理员的登录，使用RBAC模型实现系统的权限控制。

（1）界面设计

用户登录界面设计(前端界面需要优化)

用户登陆成功后的界面设计

（2）基于RBAC权限模型实现角色的管理：

RBAC模型介绍：RBAC认为权限授权的过程可以抽象地概括为：Who是否可以对What进行How的访问操作，并对这个逻辑表达式进行判断是否为True的求解过程，也即是将权限问题转换为What、How的问题，Who、What、How构成了访问权限三元组。

RBAC模型的组成：在RBAC模型里面，有3个基础组成部分，分别是：用户、角色和权限。RBAC通过定义角色的权限，并对用户授予某个角色从而来控制用户的权限，实现了用户和权限的逻辑分离（区别于ACL模型），极大地方便了权限的管理。

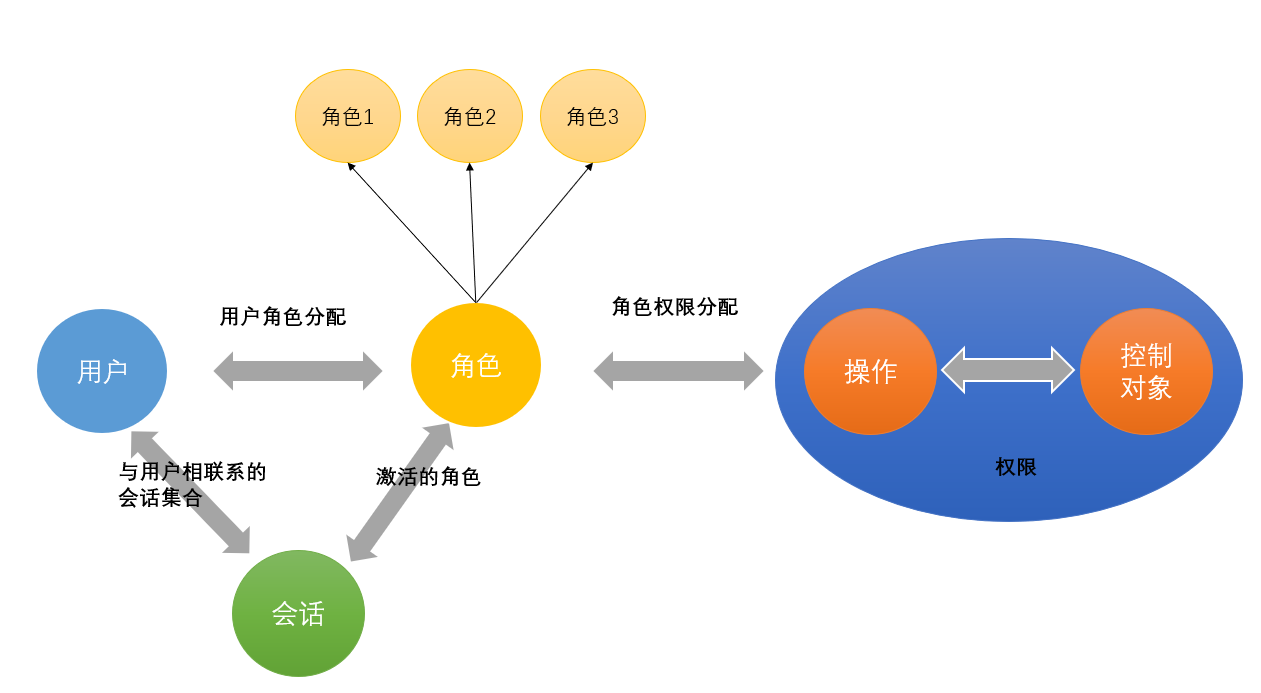
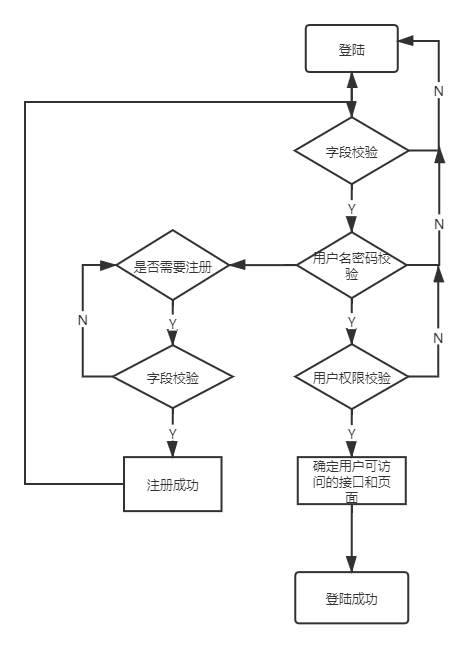


图4.2.1基于RBAC模型的权限控制模型

用户RBAC的用例图。

(3)用户登录注册的实现

用户登录注册主要的实现:用户登录注册信息的校验，密码的加密生成，以及用户登录时，对用户的角色校验等内容实现，本次系统是将用户名作为唯一的字段区分，用户在注册时，还会对用户名是否存在进行校验，如果存在，就给用户友好的提示，引导用户重新注册。后端主要采用spring security这个安全框架，使用不可逆的加密MD5进行加密，保证用户的信息安全，也是基于token对用户的发起的请求进行认证的，保证后端服务的安全性。



4.2.2 用户登录程序流程图

（4）类之间的依赖关系

（5）核心接口定义  
public class SysUserServiceImpl {  
 //根据登录用户名查询用户信息  
 public SysUserVO getUserByUserName(String userName)   
 //用户管理：分页查询  
 public IPage<SysUserOrg> queryUser  
 //用户管理：修改  
 public boolean updateUser(SysUser sysuser)

// 添加用户  
 public AjaxResponse addUser(SysUser sysUser)  
 //用户管理：删除  
 public void deleteUser(Long userId) }  
 //用户管理：重置密码  
 public void pwdreset(Long userId)   
 //个人中心：修改密码  
 public void changePwd(String username,String oldPass,String newPass)   
 //用户管理：更新用户的激活状态  
 public void updateEnabled(Long id,Boolean enabled)

// 根据用户Id查询用户信息

public SysUser getUserById(Long userId)

### 4.2.2 歌曲模块实现

歌曲模块的核心功能是为用户提供高效的检索功能，节省用户的搜索时间，提升响应的速度，为用户呈现想要的音乐作品。用户也可以上传自己的音乐作品。每天也会对每首歌曲进行不同维度的统计，来计算每首歌曲的热度，形成每天的推荐歌单呈现给用户。

（1）界面设计

（2）歌曲的维度，搜索的处理，每日推荐歌单

搜索主要区分了歌曲搜索和歌单搜索，默认为歌曲搜索，核心实现是根据输入内容进行模糊的匹配，如果搜索内容没有，就给用户的做一个友好的提示。

每日推荐功能的实现： 统计歌曲被收藏的次数，播放的次数，下载的次数，以及评论的次数，利用四个维度来对每首歌曲进行评分。每个维度分数是不同的，一首歌曲的总分是10分，下载一次 2分，播放一次1分 ，综合这些评计算出每首歌曲的评分，然后根据评分的对歌曲进行排序，形成每日的推荐歌单的信息。

热门歌曲推荐流程图：

每日推荐的流程图：

（3）类之间依赖关系(画一个依赖图)

（4） 核心的接口定义

public interface SongService {  
  
 /\*删除歌曲信息\*/  
 boolean deleteSong(Long id);  
  
 /\*插入歌曲的信息\*/  
 boolean insertSong(Song record);  
  
 /\*根据歌曲的id进行查询\*/  
 Song selectSong(Long id);  
  
 /\*更新歌曲的信息\*/  
 boolean updateSong(Song record);  
  
 /\*分页查询的实现\*/  
 IPage<SongVo> selectSongByPage(Integer pageNum, Integer pageSize);  
  
 /\*根据歌曲的url返回歌曲的id\*/  
 Long selectSongByUrl(String songUrl);  
  
 */\*\*  
 \* 根据歌曲名或歌手名返回信息  
 \** ***@param*** *pageNum 分页参数  
 \** ***@param*** *pageSize 分页的大小  
 \** ***@param*** *queryName 查询的参数  
 \** ***@return*** *{****@link*** *SongVo}  
 \*/* IPage<SongVo> querySongBySongNameOrSingerName(Integer pageNum, Integer pageSize, String queryName);  
  
 /\*根据歌手id返回歌曲的信息\*/  
 IPage<SongVo> selectSongBySingerId(Integer pageNum, Integer pageSize,Integer singerId);  
  
  
 /\*根据歌单的id返回歌曲的信息\*/  
 IPage<SongVo> selectSongBySongListId(Integer pageNum, Integer pageSize,Integer parseInt);  
  
 */\*\*  
 \* 根据歌曲查询歌曲的信息  
 \** ***@param*** *songName 歌曲的信息  
 \** ***@return*** *\*/* List<Song> songOfName(String songName);  
  
 /\*返回热门歌曲\*/  
 AjaxResponse getHotSong();  
  
 */\*\*  
 \* 根据个人信息返回推荐的音乐  
 \** ***@param*** *userId  
 \** ***@return*** *\*/* AjaxResponse getRecommendSong(Long userId);  
  
 */\*\*  
 \* 返回用户的浏览历史记录信息  
 \** ***@param*** *songIds 歌曲的id的信息  
 \** ***@return*** *\*/* AjaxResponse getHistorySong(List<Long> songIds);

### 4.2.3 歌单模块实现

歌单主要是组织不同的歌曲，分为系统生成歌单和用户自定义歌单，系统自定义歌单是由系统每日根据用户听歌的历史，区分不同维度生成的。用户自定义歌单是用户根据自己的需要类定义的歌单，可以添加自己喜欢的音乐。方便用户对自己音乐进行分类管理。

1. 前端的界面设计

(2) 用户可以创建自己的歌单，遇到自己喜欢的歌曲，就可以添加到指定的歌单，在添加歌曲到歌单前，系统会进行校验，如果歌曲已经存在歌单中，就插入数据库，并给用户友好的提示。

（3）核心接口设计以及依赖  
public interface SongListService {  
  
 /\*增加一个歌单\*/  
 boolean addSongListUserId (SongList songList,Long userId);  
  
 /\*增加的歌单信息\*/  
 boolean addSongList(SongList songList);  
  
 /\*更新一个歌单的信息\*/  
 boolean updateSongListMsg(SongList songList);  
  
 /\*更新歌单的图片信息\*/  
 boolean updateSongListImg(SongList songList);  
  
 /\*根据id删除歌单的信息\*/  
 boolean deleteSongList(Integer id);  
  
 /\*根据歌单的风格来查询歌单的信息\*/  
 IPage<SongList> likeStyle(Integer pageNum, Integer pageSize,String style);  
  
 /\*精确的查找歌单的信息\*/  
 IPage<SongList> songListOfTitle(Integer pageNum, Integer pageSize,String title);  
  
 /\*查询我的歌单信息\*/  
 IPage<SongList> findMySongList(Integer pageNum, Integer pageSize,Long userId);  
  
 /\*根据歌单的id查询单个歌单的信息\*/  
 SongList songListById(Integer parseInt);  
  
 /\*分页查询歌单信息\*/  
 IPage<SongList> getSongListByPage(Integer pageNum, Integer pageSize);  
  
 /\*获取10个热门的歌单信息\*/  
 List<SongList> getSongListHot();  
  
 /\*删除歌单中收藏的歌曲\*/  
 AjaxResponse deleteSongToSongList(Long songId, Integer songListId,Integer type);  
  
 /\*添加歌单中收藏的歌曲\*/  
 AjaxResponse addSongToSongList(Long songId, List<Integer> songListId,Integer type);  
}

### 4.2.4 评论模块实现

（1）前端UI设计

（2）用户评论的活动图

（3）评论的过滤功能实现算法说明DFA算法来实现，这里需要参考我文献

（4）核心方法定义

public interface UserCommentService {  
 */\*\*  
 \* 添加一条评论  
 \** ***@param*** *userComment {****@link*** *UserComment}  
 \** ***@return*** *\*/* AjaxResponse addComment(UserCommentSongRequest userComment);  
  
 */\*\*  
 \* 更新一条评论内容  
 \** ***@param*** *userComment {****@link*** *UserComment}  
 \** ***@return*** *\*/* AjaxResponse updateComment(UserComment userComment);  
  
 */\*\*  
 \* 删除一条评论内容  
 \** ***@param*** *id 评论的id  
 \** ***@return*** *\*/* Boolean deleteComment(Long id);  
  
 */\*\*  
 \* 根据的评论的id来查询评论  
 \** ***@param*** *id 评论的id  
 \** ***@return*** *\*/* UserComment selectByCommentId(Long id);  
  
 */\*\*  
 \* 分页查询评论内容  
 \** ***@param*** *page 具体分页对象  
 \** ***@return*** *\*/* IPage<UserCommentVo> getUserCommentByPage(Page<UserCommentVo> page);  
  
 */\*\*  
 \* 根据用户名查询用户的评论内容  
 \** ***@param*** *userName 用户名  
 \** ***@return*** *\*/* List<UserCommentVo> getCommentByUserName(String userName);  
  
 */\*\*  
 \* 根据歌曲名或者歌手名查询所有的评论信息  
 \** ***@param*** *songName 歌曲名  
 \** ***@return*** *\*/* IPage<UserCommentVo> getCommentBySongName(Page<UserCommentVo> page,String songName);  
  
 */\*\*  
 \* 根据歌曲的id查询所有的评论的内容  
 \** ***@param*** *songId 歌曲的id的信息  
 \** ***@return*** *{****@link*** *UserCommentVo}  
 \*/* IPage<UserCommentVo> getCommentBySongId(Page<UserCommentVo> page,Long songId);  
  
 */\*\*  
 \* 根据用户名查询用户的信息  
 \** ***@param*** *page 页数的大小  
 \** ***@param*** *userIdDb 用户的id信息  
 \** ***@return*** *{****@link*** *IPage}  
 \*/* IPage<UserCommentVo> getUserCommentByUserId(Page<UserCommentVo> page, Long userIdDb);  
  
 /\*返回评论的点赞内容\*/  
 AjaxResponse updateCommentLikeNum(Long commentId, Integer likeNum);  
}

### 4.2.5 信息统计模块实现(这个统计信息的接口还是没有实现)

信息统计功能主要是为了统计歌曲以及歌手信息，用户信息的基本情况。前端采用Echars来进行实现，

(1)前端界面设计：

综合统计的一个图例。

（2）歌曲信息统计设计实现

统计维度： 歌曲的评论次数， 歌曲被收藏的次数，歌曲播放的次数（这个在playBar中可以实现。）

（3）歌手信息统计设计实现

统计维度： 歌手性别，歌手的地域分布，歌手的年龄分布，歌手所拥有的歌曲 歌手数量

（4）用户信息统计设计实现

用户的性别，用户的地域，用户的年龄段，听歌排行，（统计用户的听歌次数，这个就看他的播放次数。播放历史（播放哪里统计。））

用户一定能上传歌曲，这个肯定要实现。

### 4.2.6 系统监控模块实现

（1）监控的方面

系统的内存运行，系统的cpu的使用，方便系统管理员实时的掌握当前系统的运行状况。

### 4.2.7 异常设计实现

（1）前端异常处理

（2）后端异常处理。活动图。

# 5 系统的单元测试和集成测试

主要类的设计

软件测试

# 6 总结