123

**毕业设计(论文)**

**题 目： 基于springboot的音乐在线系统**

**学生姓名： 陈伟伟**

**学 号： 120250329**

**专业班级： 软件1703**

**学 院： 信息与电气工程学院**

**指导教师： 汤永新**

**2021年 6月 6 日**

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。听音乐作品成为了很多人的消遣方式。怎么更加高效的找到自己比较喜欢的音乐类型，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他们经常听的音乐，发掘用户喜欢的音乐类型。用户也可以根据自己喜欢的音乐风格来选择想听的音乐，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。运用mysql来对音频文件以及用户的相关信息的进行持久化存储。主要开发的工具使用的是idea编辑器。

本系统主要是基于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。 客户端主要包括的功能有： 用户的登录，用户的注册，音乐作品的多维度搜索，收藏音乐，新建歌单，评论，播放音乐，查看浏览历史等功能。后台管理系统的主要功能： 用户基本信息管理，歌曲基本信息的管理，歌单信息管理，歌手信息管理，角色的动态授权，评论信息动态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 a pastime for many people. How to find the type of music you like more efficiently, 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 the music they often listen to, and to discover the types of music users like. Users can also choose the music they want to listen to according to their favorite music style, or they can classify the music they have listened to, and switch between 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 persistently store audio files and user-related information. 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: user login, user registration, multi-dimensional search of music works, music collection, new playlist, comment, play music, view browsing history and other functions. The main functions of the background management system: user basic information management, song basic information management, playlist information management, singer information management, role dynamic authorization, comment information dynamic management, and information statistics.

**Keywords:** springboot; mysql ;vue ;RBAC; music online sharing

目 录

[摘 要 I](#_Toc72683912)

[**Abstract** II](#_Toc72683913)

[第1章 绪论 1](#_Toc72683914)

[1.1 研究背景与意义 1](#_Toc72683915)

[1.2 国内外研究现状 1](#_Toc72683916)

[1.2.1 国外研究现状 1](#_Toc72683917)

[1.2.2 国内研究现状 1](#_Toc72683918)

[1.3 研究内容与组织结构 2](#_Toc72683919)

[1.3.1 研究内容 2](#_Toc72683920)

[1.3.2 组织结构 2](#_Toc72683921)

[1.4 本章小结 2](#_Toc72683922)

[第2章 系统分析 3](#_Toc72683923)

[2.1 关键技术分析 3](#_Toc72683924)

[2.1.1 SpringBoot 3](#_Toc72683925)

[2.1.2 Vue 3](#_Toc72683926)

[2.1.3 SpringSecurity 3](#_Toc72683927)

[2.2 系统可行性分析 4](#_Toc72683928)

[2.2.1 技术可行性 4](#_Toc72683929)

[2.2.2 经济可行性 4](#_Toc72683930)

[2.2.3 社会可行性 5](#_Toc72683931)

[2.3 系统需求分析 5](#_Toc72683932)

[2.3.1 功能性需求 5](#_Toc72683933)

[2.3.2 非功能性需求 8](#_Toc72683934)

[2.4 本章小结 8](#_Toc72683935)

[第3章 系统总体设计 10](#_Toc72683936)

[3.1 设计目标 10](#_Toc72683937)

[3.2 系统总体架构图 10](#_Toc72683938)

[3.3 系统流程图 11](#_Toc72683939)

[3.4 数据流图 12](#_Toc72683940)

[3.5 数据字典 12](#_Toc72683941)

[3.6 系统模块设计 15](#_Toc72683942)

[3.7 系统E-R图设计 16](#_Toc72683943)

[3.8 接口设计 19](#_Toc72683944)

[3.9 运行设计 21](#_Toc72683945)

[3.9 系统出错设计 22](#_Toc72683946)

[3.9.1 出错信息处理 22](#_Toc72683947)

[3.9.2 系统补救措施 22](#_Toc72683948)

[3.10 本章小结 22](#_Toc72683949)

[第4章 系统详细设计及实现 23](#_Toc72683950)

[4.1 系统数据结构设计 23](#_Toc72683951)

[4.2 客户端功能模块设计及实现 26](#_Toc72683952)

[4.3 后台功能模块设计及实现 30](#_Toc72683953)

[4.3.1 用户信息角色模块 30](#_Toc72683954)

[4.3.2 歌曲模块 32](#_Toc72683955)

[4.3.3 歌单模块 33](#_Toc72683956)

[4.3.4 评论模块 33](#_Toc72683957)

[4.3.4 歌手信息模块 35](#_Toc72683958)

[4.3.5信息统计模块 35](#_Toc72683959)

[4.3.6 系统监控模块 36](#_Toc72683960)

[4.4 异常设计 36](#_Toc72683961)

[4.5 本章小结 36](#_Toc72683962)

[第5章 系统测试 37](#_Toc72683963)

[5.1 测试意义 37](#_Toc72683964)

[5.2 测试主要方法 37](#_Toc72683965)

[5.5 本章小结 39](#_Toc72683966)

[结 论 40](#_Toc72683967)

[致 谢 41](#_Toc72683968)

[参考文献 42](#_Toc72683969)

第1章 绪论

## 1.1 研究背景与意义

智能手机高度普及，以及人们生活节奏的加快，人们碎片化时间的利用显得越发的重要。当然，信息高速发展的同时，也给我们检索信息带来很大不便，听音乐也不例外。本系统研究的重点就是收集用户听音乐的喜好，可以使用户更好的收集到自己喜欢的音乐。节省用户检索音乐的时间，帮助用户更好的管理自己音乐，更快速的找到自己喜欢的音乐作品。更好的利用碎片化的时间来放松自己的身心。

1.2 国内外研究现状

1.2.1 国外研究现状

本次主要是针对国外的app music ：也就是Apple Music的相关技术进行了调研。不同于其它手机自带的原生音乐APP，Apple Music的起源要早得多，主要使用Objective-C语言开发。优点：音乐软件统一，特别是界面UI色彩很是出众，前端页面简约大气，全球曲库范围 版权最丰富。缺点：没有评论，没有每天的推荐功能。

1.2.2 国内研究现状

国内的音乐app比较多，本次主要调研了国内有代表性的两个app,分别是网易云和qq音乐。

网易云的热评比较好，评论真的很走心，用户的粘性高。评论实现的基础架构采用了Mysql的分布式集群存储。使用redis来进行缓存，提高了网站的响应的速度。推荐算法主要使用的是协同过滤的算法。网易云音乐还通过数据中台建设来实现为8亿用户服务，做到了服务的高可用，高可靠。缺点： 就是推荐系统粒度不够细，存在推荐的内容同质化的现象。推荐的形式还不够灵活和多样，对用户的评论内容并没有进行过滤。

qq音乐，有着非常丰富的歌曲资源，音质提升功能强大，流畅性高，分类比较齐全。主要的功能识别歌曲，相比其他的音乐软件，识曲的速度很快，使用的音频指纹技术也更加的先进。另外QQ音乐的转码服务已经在用docker容器服务，具有更好的性能。也使用了CDN技术为用户提供更好的听歌体验。缺点：歌曲歌手整体曲风分类上存在不足，效果预见性不好。

1.3 研究内容与组织结构

1.3.1 研究内容

主要采用统计和对比的调研方法，对国内外主流的音频网站进行调研和分析。并采用访谈的调研方法，了解目前大众对音乐软件的具体需求。

采用springboot以及Mysql主要对数据进行处理，前端主要使用vue来呈现相关的内容。主要解决问题： 用户对音乐的自定义分类，评论的过滤功能，以及歌曲信息的统计功能，歌单的多维度展示。

1.3.2 组织结构

本文主要包含了5个章节，每个章节的具体研究的内容如下：

第1章，绪论，主要对当前国内外音乐系统进行了研究和分析。注重分析了各个系统的优缺点，确定了本次系统主要研究的内容和意义。

第2章，可行性分析以及需求分析，可行性分析主要在技术，经济，社会等方面进行了分析。需求分析主要从功能的实现和用户的易用性两方面进行分析。

第3章，系统概要设计。确定系统实现的总体框架。梳理系统的主要流程以及关键的数据模型。根据业务的不同划分系统的实现模块。

第4章，系统的详细设计及实现，这一阶段，主要是对总体设计中的功能进行落地实现，具体包括数据处理，前端的界面呈现等内容。

第5章，对系统进行测试，主要使用了黑盒测试和白盒测试相结合的方式，对系统的功能实现进行了总体的测试。

1.4 本章小结

本章主要介绍了音乐分享系统主要解决的问题，并且调研了国内外主流的音乐网站并分析当前音乐系统的现状，为下步系统的实现了提供了理论支撑。

第2章 系统分析

2.1 关键技术分析

音乐在线分享网站是以SpringBoot后端框架、Vue前端框架以及SpringSercurity安全框架为主要技术点来实现的。数据存储主要使用的是mysql和redis以及腾讯云cos的对象存储。下面对几个技术进行介绍:

2.1.1 SpringBoot

SpringBoot是的核心原则是约定大于配置，使用比较灵活。SpringBoot摆脱了繁琐的文件配置，通过为实体类添加注解将其标识为特定类型。Springboot的使用方式也比较简单，在Maven项目中通过在pom.xml引入项目所依赖的软件包，通过注解的方式代替XML配置。提高了代码的可读性和项目的可维护性。提高了项目的开发效率和项目整体的稳定性。

2.1.2 Vue

随着系统功能的复杂以及用户对软件体验感的要求，传统的前端的单页面开发已经很难适应开发需要。Vue的出现就很好的解决了这个问题。使用Vue可以基于组件化开发，不但可以提高代码的复用率，而且可以提高项目的可维护性。与后端的交互，Vue也提供了axios，很好的解决跨域的问题，基于axios可以实现请求的拦截，请求头的定义等自定义的功能。另外，Vue的文档健全，代码开源，对中文支持友好，学习门槛低，社区活跃，国内使用人数比较多。出现问题，基本都能找到解决方案。这些都是本次系统选择vue作为前端框架的依据。

2.1.3 SpringSecurity

Spring Security是一个比较优秀的安全框架，由于其简单及易用性吸引了诸多的目光。Spring Security主要用于系统开发中的用户信息认证，用户角色授权，用户信息加密，以及会话管理等开发需求。

Spring Security 主要是为应用程序提供身份验证和授权。他独有的优势于在于他的扩展能力和需求定制的能力。特别是它可以对用户的登录信息，以及用户的角色进行天然集成，不需要开发人员来管理，只需要继承其相关的类就可实现相关的功能。

2.2 系统可行性分析

2.2.1 技术可行性

本系统主要采用前后端分离的设计方式进行开发的，前端的功能主要是页面的渲染以及为用户提供良好的交互体验，后端主要负责对数据的数据逻辑处理。前端访问后端数据主要使用axios库，后端主要基于mvc的方式响应前端的请求，将相关的数据传递给前端。

本次系统主要使用到的关键开发技术以及要求如下表：

表2-1 关键的技术分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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  3 社区活跃，使用用户人数多 |
| 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 配置简单，没有繁琐的xml配置，对开发人员比较友好  2 社区丰富，自动化配置大大提高了开发的效率。 |
| SpringSercurity | 熟悉 | 了解 | 学习管方文档，可以满足要求 | 1 接入简单，是基于角色管理的框架  2 使用md5对用户的密码加密，安全性能高。  3 与springboot可以无缝整合 |
| 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) |
| Echars | 了解 | 了解 | 结合官方文档，可以达到要求 | 1 界面简洁，友好  2 开源组件，社区活跃 |
| ElementUI | 了解 | 了解 | 可以满足开发需求 | 1 性能稳定  2 文档齐全 |

2.2.2 经济可行性

本系统在开发过程中使用的具体的开发环境如下：

（1）硬件设备：windows10操作系统、i7-7200U处理器。

（2）软件设备：Vscode、IDEA、Navicat for MySQL，Xshell。

因为本次开发过程中使用的编译器以及各类框架均为开源，使用的腾讯云的对象存储cos免费额度也满足此次的开发需求。因此，综合分析，在经济上是可行的。

2.2.3 社会可行性

（1）法律可行性

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。

（2）操作可行性

由于系统是面向个人服务的。所以系统的使用会涉及到各个类型的人群。客户端的界面设计简约友好，以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。简单易上手。

后台系的管理的操作人员： 主要是对用户信息， 歌曲的基本信息，歌手基本信息以及用户评论进行一个动态的管理和监控。后台管理系统提供了交互良好的web界面，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即可上手。

2.3 系统需求分析

本系统结合前期的系统调研，主要从当前大众的需求以及系统的实用性出发，确定本系统的具体需求和要实现主要功能。

2.3.1 功能性需求

音乐分享系统是基于RBAC模型来实现的，主要设置有三个角色信息，游客，普通用户，后台管理员，具体的用例图入下图。如图2-1、2-2、所示。

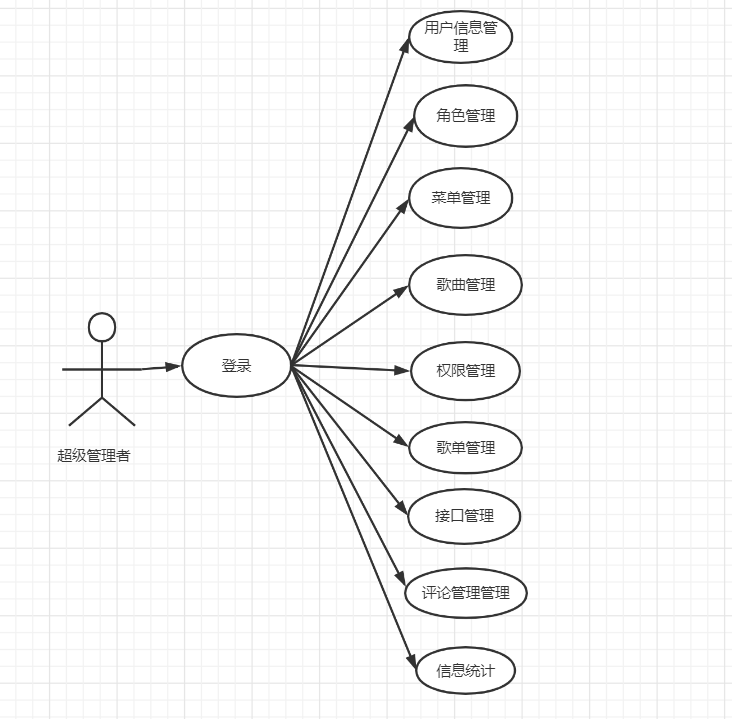


图2-1 管理员的用例图

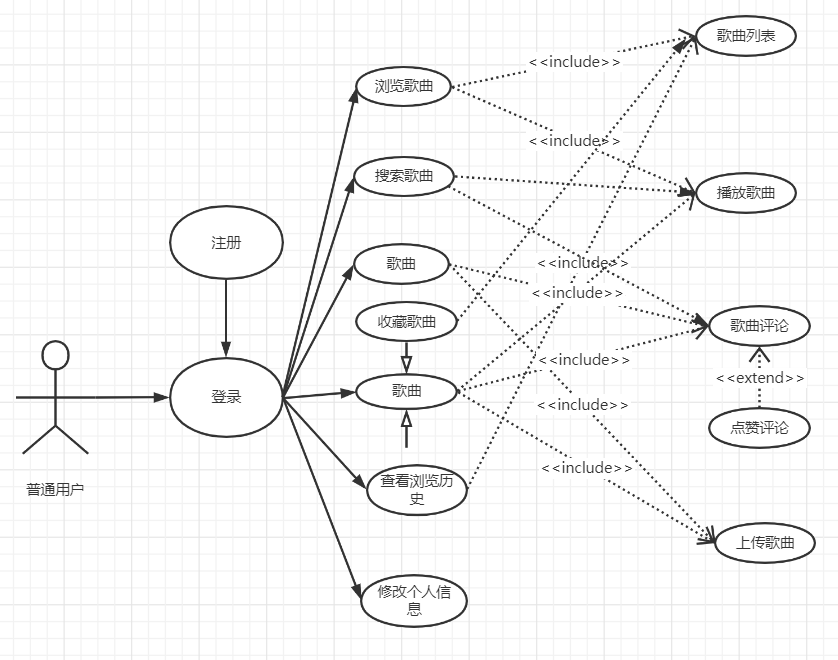


图2-2 普通用户的用例图

为了便于后期系统的扩展以及系统的稳定性，本次系统主要划分了七个子模块：

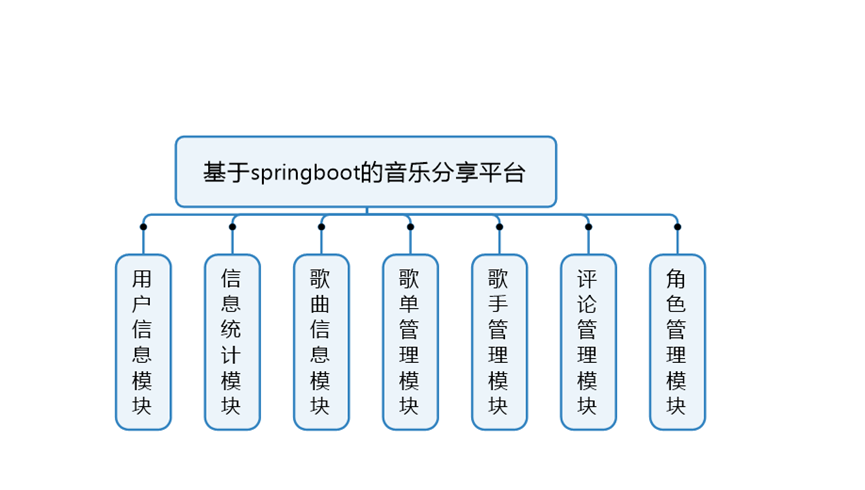


图2-4 系统的模块划分

（1）用户基本信息模块：主要包含用户基本信息的动态管理。用户通过注册，就可以成为本系统的用户。用户登录本系统后，可以在个人中心中，查看自己的听歌历史以及音乐的收藏信息，也可以创建自己的歌单以及基本信息的修改等。后台管理员可以登录后台管理系统，对所有用户的信息进行管理。

（2）歌曲管理模块：主要对本系统中的所有音频文件进行管理，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cos对象存储中的。增加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，使用户有一个更好的系统体验。

（3）歌手管理模块：主要功能是系统中歌手基本信息的增删改查和统计实现。用户可以根据自己的需求动态的查询歌手的基本信息。具体包括简介，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

（4）角色管理：基于权限管理模型RBAC对权限进行管理。本系统的权限主要是基于两个方面： 一是菜单权限，二是接口权限。只有系统管理员可以根据不同的需要，给用户进行动态的授权。具体包括角色的创建，各种角色的授权，角色的动态查询等功能，以实现不同角色的可以访问不同的资源。使系统具有更好的灵活性。

（5）歌单管理模块：将不同的歌曲组成具体的歌单，进行展示。给用户提供更好的检索体验和音频资源。系统也会根据每个用户的听歌类型，每天生成个性化的歌单。实现歌曲内容的最大共享。

（6）评论管理模块：用户可以对自己喜欢的歌曲发表评论。后台会统计每一首歌曲的评论并进行管理，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。

（7）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。对数据资源有一个比较好的隔离效果。

（8）统计信息模块：主要是收集用户的基本信息和行为信息。具体包含用户的播放历史，用户收藏歌曲的数量，以及歌曲的播放量等信息进行统计，并使用Echars进行可视化的展示。

2.3.2 非功能性需求

本次的音乐分享系统的非功能性需求如下：

（1）界面需求

界面要求简约，符合用户的操作习惯，给用户良好的交互体验。必要的提示信息很有必要，比如用户查询歌曲不存在时，给出必要的提示，用户发表评论时，提示用户评论成功等。

（2）易用性

系统应该具有易用性，使用户操作简单，使用方便。

（3）安全性需求

本系统的安全性方面主要体现在两个方面：一是对用户的信息使用md5不可逆的方式进行加密。保证用户的账号安全。另一方面，基于角色的控制，达到后端接口的灵活配置，保证不同的角色访问的资源不同。

（4）系统的可扩展性和可维护性

本系统设计以高内聚，低耦合原则来实现的。系统的各个模块独立又统一，方便后期对每个模块功能的扩展，而不影响别的模块。另一方面，本系统主要采用面向对象的设计思想，并进行了必要的出错设计以及数据库的备份等的相关工作，方便系统的后期维护和使用。

2.4 本章小结

本章主要对系统的可行性和系统的需求分析进行了说明。在可行性分析中，主要从技术可行性，社会道德可行性，以及经济可行性三个方面进行分析。在技术可行性分析中，主要考虑了技术的选型，项目对技术的要求，当前对该技术的掌握程度以及选择该技术的原因进行了重点的分析。在系统的需求分析中，主要确定了系统的模块以及功能，并对各个模块的功能进行了简要的说明。通过本章的分析，可以确定开发音乐软件系统是有必要的，也是能达到人们听音乐的基本需求的。

第3章 系统总体设计

3.1 设计目标

音乐分享系统是一款融合听音频与管理为一体的音频平台，旨在通过本系统使用户能更快速的寻找到自己喜欢的音乐，并对自己的音乐能进行较好的管理。本系统利用角色属性来控制系统的不同资源的访问：主要分为游客，普通用户，系统管理员三个角色，

游客可以查询歌曲，听歌，浏览热门歌曲，歌手等信息。但是不能对具体歌曲进行评论和创建自己的歌单，不能收藏歌曲。不能对歌曲进行评论。

普通用户可以登录，注册，浏览歌曲，收藏歌单，并且根据自己的喜好创建属于自己歌单信息。也可以浏览自己的听歌历史，评论歌曲，查看自己的听歌报告，下载音乐等功能。

系统管理员可以实现对音频文件的管理，系统歌单的管理，用户评论的管理，用户收藏音乐的管理，歌手信息管理，后端的接口管理以及对用户的授权管理，角色信息管理等功能。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。

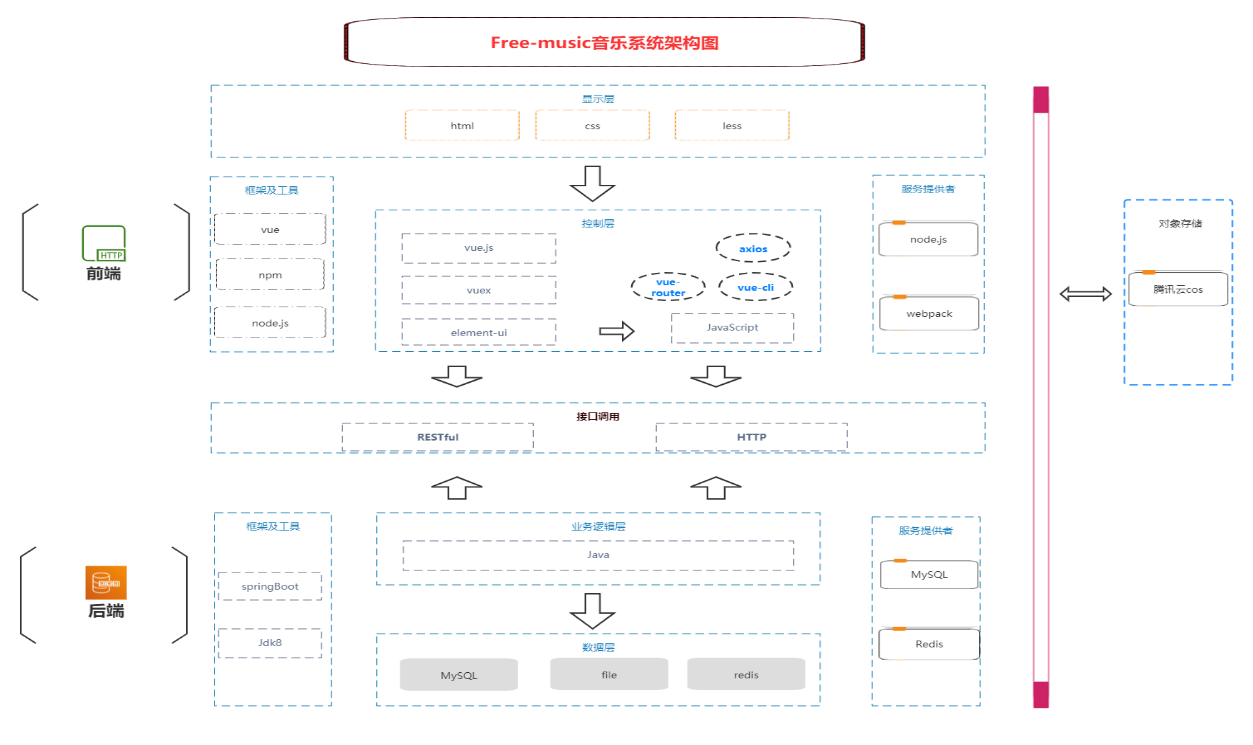
3.2 系统总体架构图

图3-1 系统的总体架构图

3.3 系统流程图

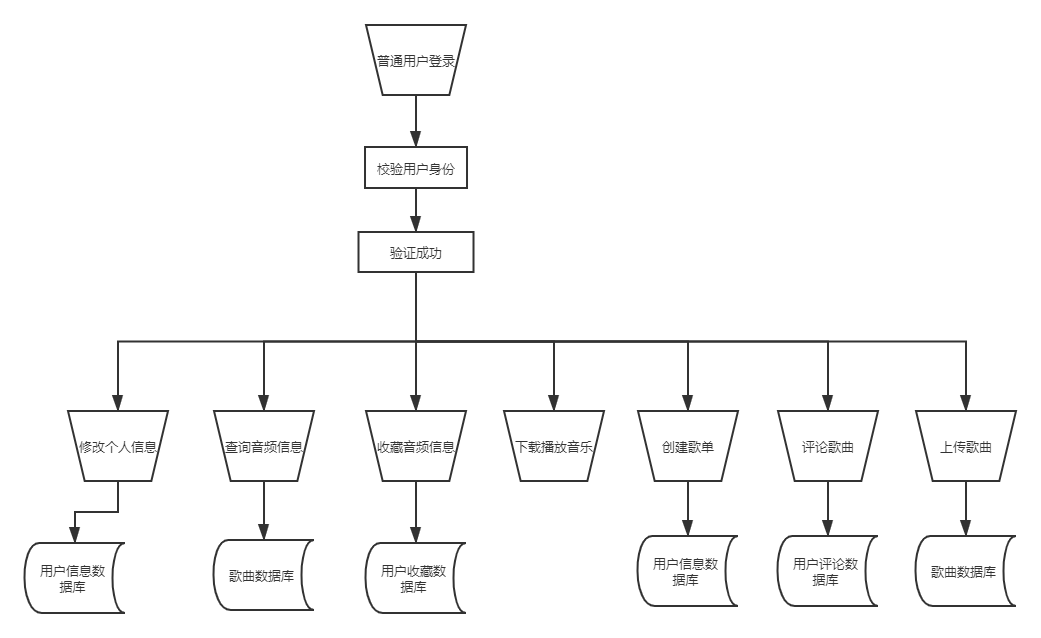


图3-2 普通用户的流程图

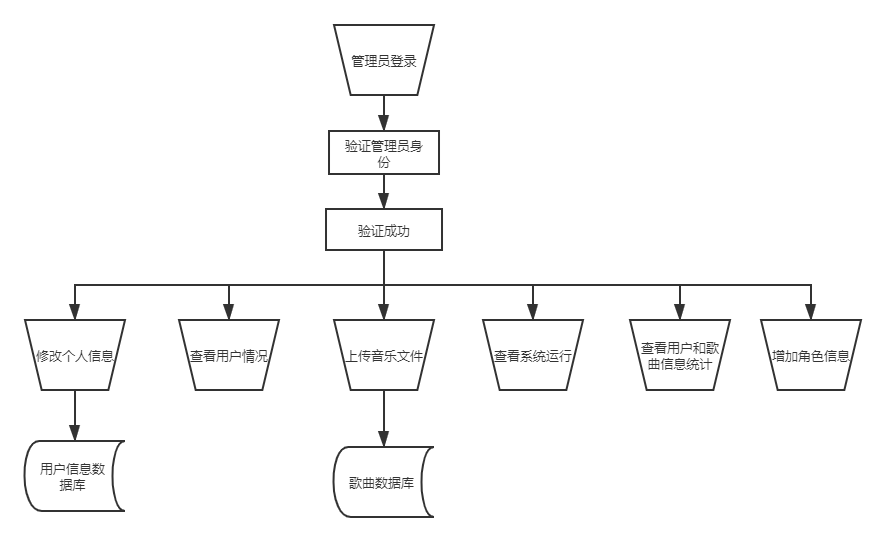


图3-3 管理员流程图

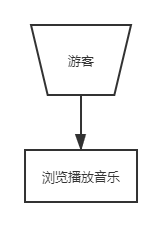


图3-4 游客的流程图

3.4 数据流图

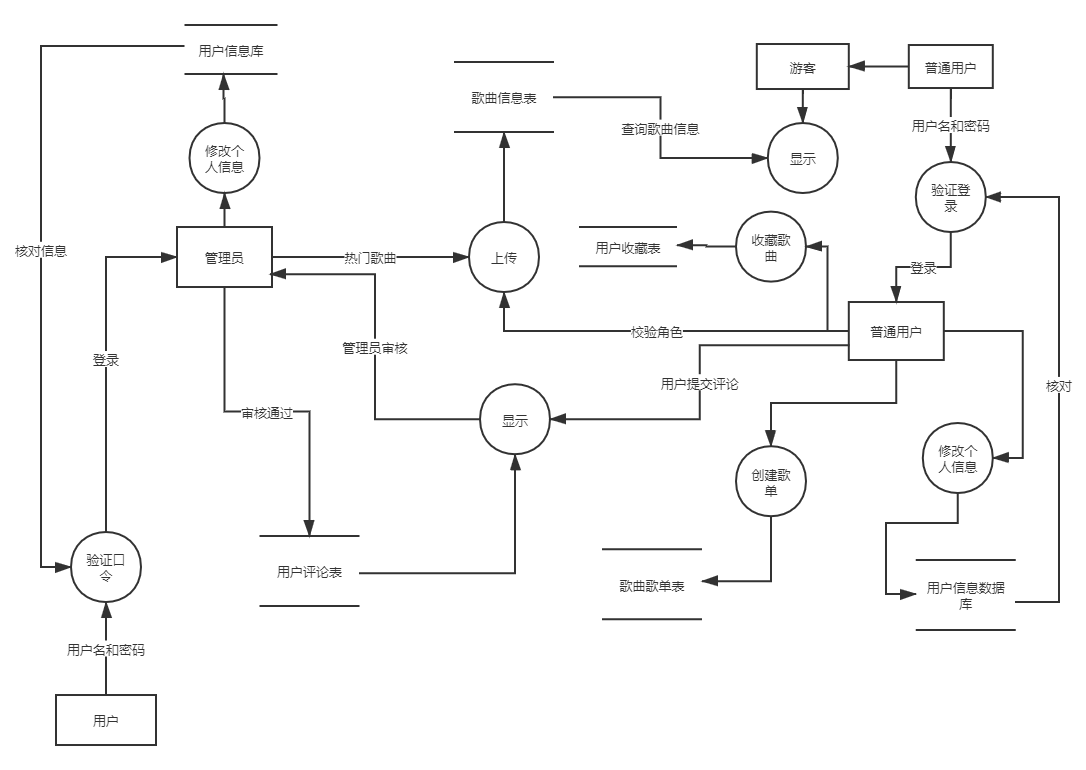


图3-5 系统的数据流图

3.5 数据字典

（1）数据流描述：

数据流名：登陆

数据来源：用户信息登录

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：根据角色不同跳转不同的页面。

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：创建歌曲分类

数据来源： 创建歌单界面

数据去向： 新建歌单

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+歌单分类信息。

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： 音频信息统计

数据来源： 歌曲播放界面

数据去向：音频信息统计界面

说明： 对音频的信息进行统计

数据组成： 音频的基本信息+下载的次数+播放的次数+收藏次数

数据流量： 大

（2） 数据元素词条描述：

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：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

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：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

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

3.6 系统模块设计

音乐分享系主要根据不同用户提供不同的操作界面。普通用户主要使用客户端，该模块主要包含音频文件搜索、歌单信息展示、歌手、个人中心，歌曲评论功能等；管理员主要使用后台管理模块，该模块主要包含用户基本信息管理，音频文件管理，评论管理，歌单管理，歌手基本信息管理，信息统计等功能。

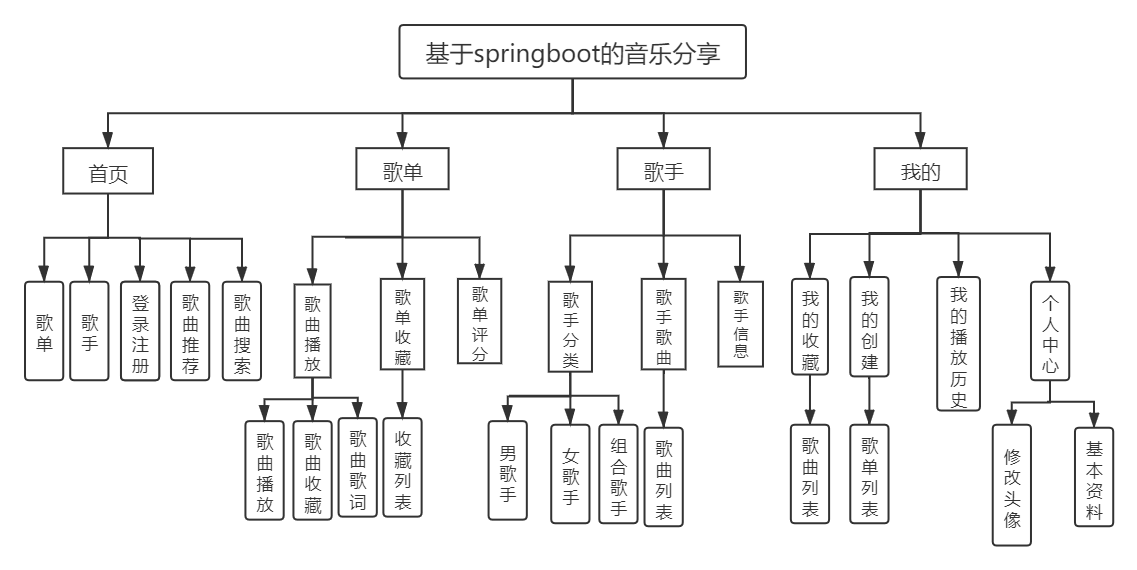


图3-6 系统客户端具备功能

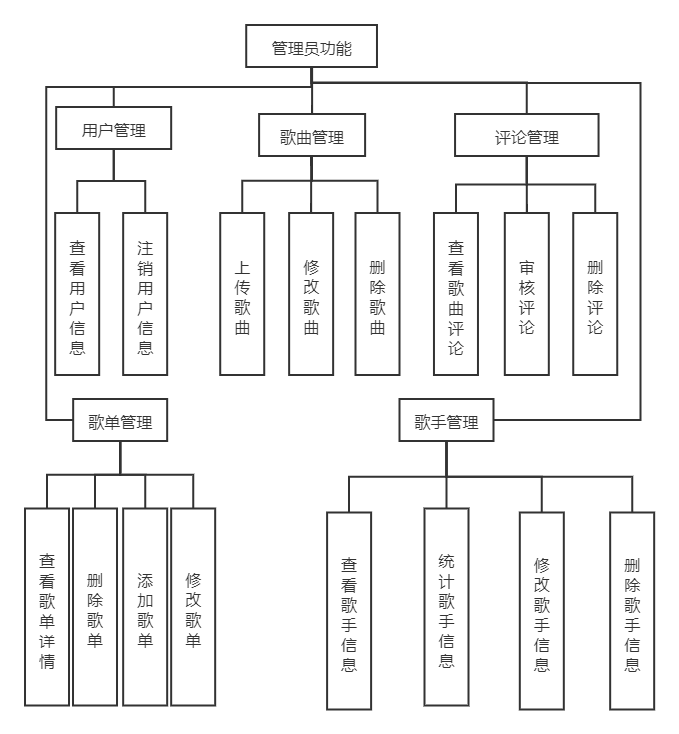


图3-7 系统后台功能结构图

3.7 系统E-R图设计

综合系统设计分析，确定系统的E-R图如下所示：

（1）用户-角色E-R图



图3-8 用户信息和角色E-R 图

（2）歌单-歌曲E-R图



图3-9 歌单-歌曲E-R图

（3）歌手-歌曲E-R图



图3-10 歌手-歌曲E-R图

（4）用户-歌单-收藏E-R图

图3-11 歌单-用户-收藏E-R图

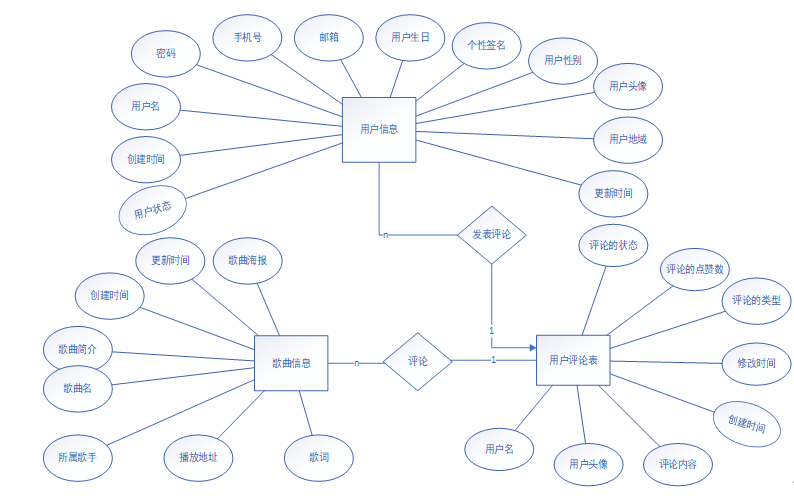


图3-12 用户-评论-歌曲E-R图

3.8 接口设计

（1）用户接口

表3-1 用户接口设计

|  |  |
| --- | --- |
| 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 |
| 查询播放历史信息 | 用户最近播放 100首歌曲 |

（2）外部接口

表3-2 外部接口设计

|  |  |  |
| --- | --- | --- |
| 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 | 用户基本信息，歌曲信息，歌手基本信息，用户的评论信息，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）内部接口

表3-3 内部接口设计

|  |  |  |
| --- | --- | --- |
| 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） |

3.9 运行设计

（1）运行模块组合

系统在外界不同操作下，各模块之间运行之间的相互影响组合情况如下表所示:

表 3-4 运行模块组合

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |  | √ |  |  |

（2）运行控制

表3-5 运行控制方法

|  |  |
| --- | --- |
| 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。 |

3.9 系统出错设计

3.9.1 出错信息处理

(1) 输入的用户名不存在，该用户未注册，提示用户进行注册。

(2) 用户名或密码不正确，说明用户的输入有误，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，提示上传的音频的大小超过限制，调整后在进行上传。

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

3.9.2 系统补救措施

(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)数据库的信息每周一次的备份，防止出现数据丢失的情况

3.10 本章小结

本章主要对基于springboot音乐在线分享系统的总体设计进行了说明。主要确定了系统的总体的架构设计，确定系统实体属性，并绘制系统的数据流图以及定义系统的流程图，从用户接口，外部接口，内部接口三个方面对相关的接口进行了设计，定义了系统运行时，各个模块之间的组合关系。最后定义了系统的出错设计。总体上确定了系统要实现的功能以及系统的边界设计等问题。进一步完善了系统的实现细节。为下一步的具体的编码工作提供了规范和依据。

第4章 系统详细设计及实现

本章主要是结合前几节的需求分析和系统的概要设计来进行系统的详细设计，主要完成的工作是数据库的物理设计和具体的功能的编码实现。

4.1 系统数据结构设计

本系统主要使用MySQL数据库对系统的基本信息进行持久化存储。共设计了12张数据表。sys\_user表、song表、singer表、song\_list表、collect表、sys\_role表、sys\_user\_role表、song\_play\_count表、songlist\_song表, rank表、user\_comment表、sys\_menu表。

本系统具体的业务表物理数据结构设计如下:

（1）sys\_user表示用户基本信息，详情如表4-1。

表4-1 sys\_user表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| 用户Id | id | bigint(10) | 是 | 否 | 否 |
| 用户名称 | username | varchar(11) | 否 | 否 | 否 |
| 密文密码 | password | varchar(40) | 否 | 否 | 否 |
| 手机号码 | phone | varchar(11) | 否 | 否 | 是 |
| 邮箱 | email | varchar(20) | 否 | 否 | 是 |
| 状态  创建时间  更新时间  用户性别  用户简介  用户头像  用户地域 | enabled  create\_time  update\_time  sex  introduction  avator  location | tinyint(4)  datetime  datetime  tinyint(4)  varchar(255)  varchar(64)  varchar(64) | 否  否  否  否  否  否  否 | 否  否  否  否  否  否  否 | 是  否  否  否  否  否  否 |

（2）song表表示歌曲实体信息，。数据结构设计见表4-2。

表 4-1 song表信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| 歌曲Id | id | bignt(10) | 是 | 否 | 否 |
| 歌手id | singer\_id | int (20) | 否 | 否 | 是 |
| 歌曲名 | name | varchar(50) | 否 | 否 | 是 |
| 歌曲简介 | introduction | varchar(255) | 否 | 否 | 是 |
| 歌曲海报 | pic | varchar(11) | 否 | 否 | 是 |
| 歌词 | lypic | varchar(11) | 否 | 否 | 是 |
| 歌曲播放地址 | url | Varchar(11) | 否 | 否 | 是 |
| 创建时间 | datetime | Varchar(20) | 否 | 否 | 是 |
| 更新时间 | Update\_time | varchar(20) | 否 | 否 | 是 |

（3）singer表中存储的歌手的信息，每个歌手的基本信息存储在这这张表中，详情见表4-3。

表4-3 singer表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| 歌手Id | id | int(10) | 是 | 否 | 否 |
| 歌手名 | name | varchar(30) | 否 | 否 | 是 |
| 歌手性别 | sex | tinyint(4) | 否 | 否 | 是 |
| 歌手海报 | pic | longtext | 否 | 否 | 是 |
| 歌手出生日期 | birth | varchar(100) | 否 | 否 | 是 |
| 歌手地域 | location | varchar(40) | 否 | 否 | 是 |
| 歌手简介 | introduction | varchar(255) | 否 | 否 | 是 |

（4）song\_list表歌单信息，具体包含用户创建的歌单和系统创建的歌单，数据结构设计见表4-4。

表4-4 song\_list表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| 歌单Id | id | int(10) | 是 | 否 | 否 |
| 歌单标题 | title | varchar(30) | 否 | 否 | 是 |
| 歌单简介 | introduction | varchar(10) | 否 | 否 | 是 |
| 歌单风格 | style | varchar(10) | 否 | 否 | 是 |
| 歌单类型 | type | varchar(20) | 否 | 否 | 是 |
| 歌单海报 | pic | varchar(20) | 否 | 否 | 是 |

（5）collect表示用户收藏表，该表记录的是用户收藏的歌曲和歌单的信息，具体的类型区分使用表字段type来进行区分的。数据结构设计见表4-5。

表4-5 collect表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| 收藏Id | id | int(11) | 是 | 否 | 否 |
| 用户id | user\_id | bigint(30) | 否 | 否 | 是 |
| 歌曲id | song\_id | bigint(30) | 否 | 否 | 是 |
| 歌单id | song\_list\_id | int(20) | 否 | 否 | 是 |
| 收藏类型 | type | tinyint(4) | 否 | 否 | 否 |

（6）sys\_role主要表示用户角色的基本信息，数据结构设计见表4-6。

表4-6 sys\_role表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| 角色Id | id | int(10) | 是 | 否 | 否 |
| 角色英文名称 | role\_code | varchar(30) | 否 | 否 | 是 |
| 角色中文名称 | role\_name | varchar(30) | 否 | 否 | 是 |
| 是否启用 | status | tinyint(1) | 否 | 否 | 是 |
| 排序 | sort | Int(10) | 否 | 否 | 是 |

（7）song\_play\_count表是用户的播放历史表，具体记录用户播放的时间和用户名以及歌曲名的信息。

表4-7 song\_play\_count表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| Id | id | int(10) | 是 | 否 | 否 |
| 用户Id | user\_id | bigint(10) | 否 | 否 | 是 |
| 歌曲Id | song\_id | int(10) | 否 | 否 | 是 |
| 播放次数 | play\_count | int(10) | 否 | 否 | 否 |
| 下载次数 | down\_count | int(10) | 否 | 否 | 否 |

（8）songlist\_song是歌单歌曲表，存储的是每个歌单和歌曲之间的关系，数据结构设计见表4-8。

表4-8 songlist\_song表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| Id | id | int(10) | 是 | 否 | 否 |
| 歌单Id | songlist\_id | int(10) | 否 | 否 | 否 |
| 歌曲Id | song\_id | bigint(20) | 否 | 否 | 否 |

（9）rank表歌单评分表，用户对某个歌单的评分信息就在这个表中来记录。数据结构设计见表4-9。

表4-9 rank表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| Id | id | int(10) | 是 | 否 | 否 |
| 歌单Id | songListId | int(10) | 否 | 否 | 是 |
| 用户Id | user\_id | bigint(10) | 否 | 否 | 否 |

（10）sys\_user\_role用户角色关联表，记录着每个用户所拥有的角色信息，当一个用户登陆时，就使用这个表来进行校验用户的角色信息，确定用户的访问资源。数据结构设计见表4-10。

表4-10 sys\_user\_role表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| 用户Id | user\_id | bigint(10) | 否 | 是 | 否 |
| 角色id | role\_id | varchar(50) | 否 | 否 | 否 |

（11）user\_comment表示歌曲的评论表，表中主要存储的是评论的内容，并存储该条评论所对应的用户名和歌曲名。数据结构设计见表4-11。

表4-11 user\_comment表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| 评论Id | id | int(10) | 是 | 否 | 否 |
| 歌曲id | song\_id | bigint(20) | 否 | 否 | 否 |
| 用户id | user\_id | bigint(20) | 否 | 否 | 否 |
| 用户名 | user\_name | varchar(10) | 否 | 否 | 否 |
| 用户头像 | avatar | varchar(20) | 否 | 否 | 否 |
| 评论内容 | content | varchar(255) | 否 | 否 | 否 |
| 评论状态 | comment\_status | int(10) | 否 | 否 | 否 |
| 点赞数量 | like\_num | int(10) | 否 | 否 | 否 |
| 评论的类型 | type | int(10) | 否 | 否 | 否 |

（12）sys\_menu表示系统的菜单，表中存储的是具体的菜单信息和路由信息，主要是通过权限来确定用户的路由信息。数据结构设计见表4-12。

表4-11 sys\_menu表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 字段 | 数据类型 | 主键 | 外键 | 允许为空 |
| id | id | int(10) | 是 | 否 | 否 |
| 父id | menu\_pid | bigint(30) | 否 | 否 | 否 |
| 父id集合 | menu\_pids | varchar(11) | 否 | 否 | 否 |
| 菜单名称 | menu\_name | varchar(10) | 否 | 否 | 否 |
| 菜单排序 | sort | int(10) | 否 | 否 | 否 |
| 是否为叶子 | is\_leaf | int(1) | 否 | 否 | 否 |
| 访问路径 | url | varchar(10) | 否 | 否 | 否 |
| 层级  菜单的状态 | level  status | int(10)  int(10) | 否  否 | 否  否 | 否  否 |

4.2 客户端功能模块设计及实现

音乐管理分享系统客户端部分主要包括首页、歌单、热门歌曲、每日推荐，我的音乐，登陆，注册，搜索框。并使用ElementUI前端框架设计，导航栏的背景以白色为主，页面的整体背景背景色彩为白色，使用组件化的方式进行实现，提高了代码的复用性。

（1）客户端的首页

首页采用栅格化的布局方式，通过图片列表的形式展示热门歌单和热门歌手的信息。用户可以选择自己喜欢的歌单和歌手，浏览并播放具体的音乐作品。



图4-1 客户端首页

（2）歌单

用户可以选择感兴趣的歌单，每个歌单的详情页展示了具体的歌曲以及该歌单的具体的评分。歌单是按照类别进行区分的，主要是方便用户进行搜索。用户也可以将喜欢的歌单添加到我的收藏，可在自己的收藏列表中查询具体的收藏内容。设计效果图如4-2 所示



图4-2 歌单效果图

（3）登陆 注册

用户登录注册实现:用户登录注册信息的校验，密码的加密。用户在登录时，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进行区分的，用户注册时，对用户填写的注册信息进行校验。后端主要采用spring security这个安全框架，利用不可逆的加密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。

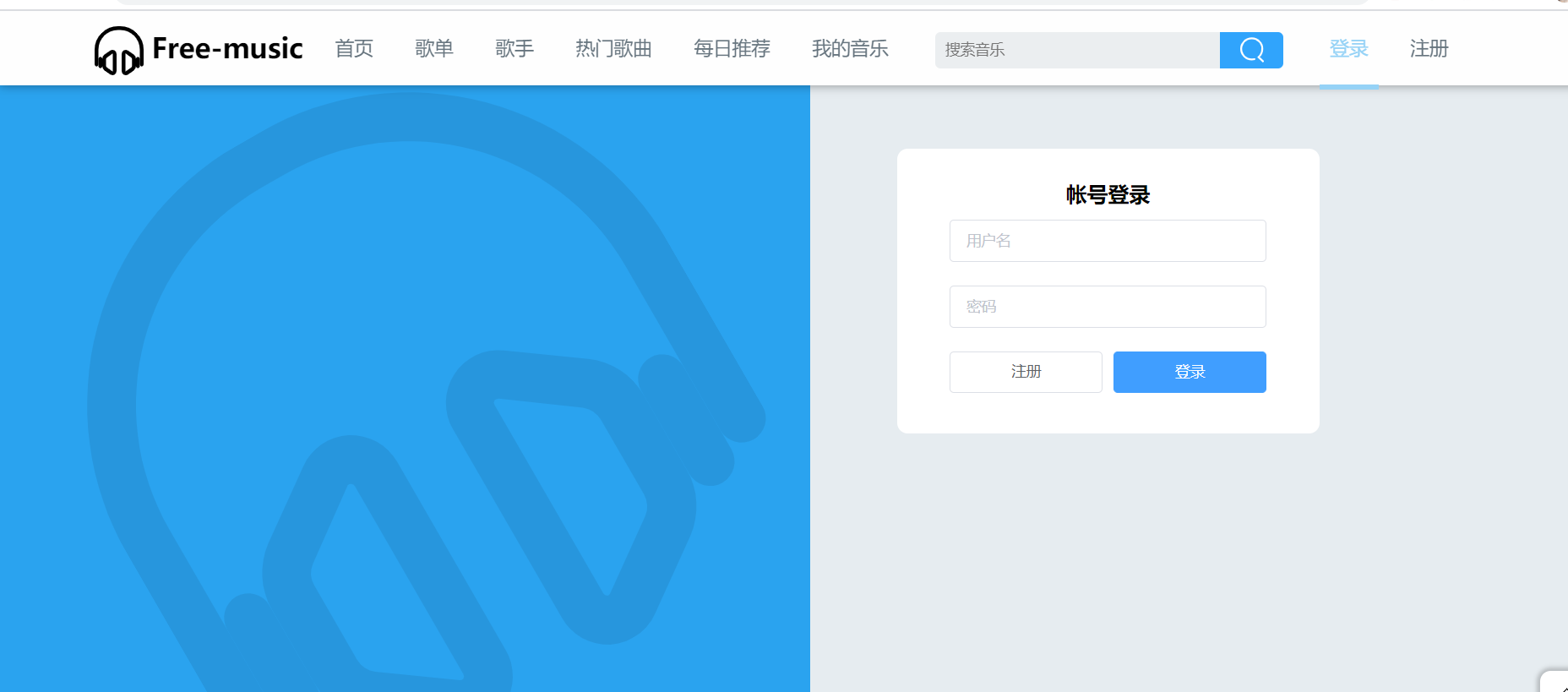


图4-4 登录运行效果图

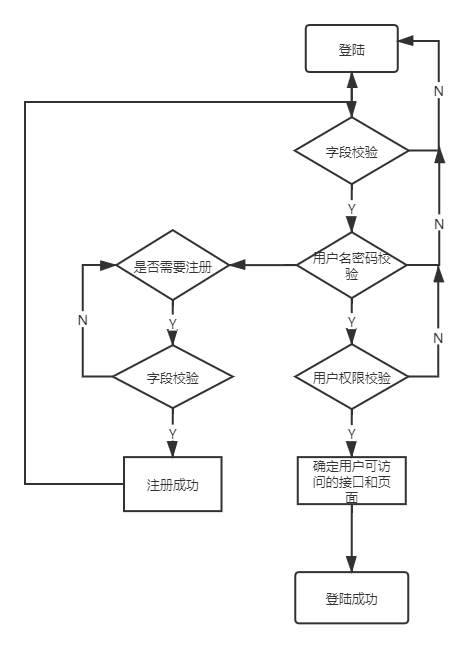


图4-5 用户登陆注册的流程图

（4）用户创建歌单

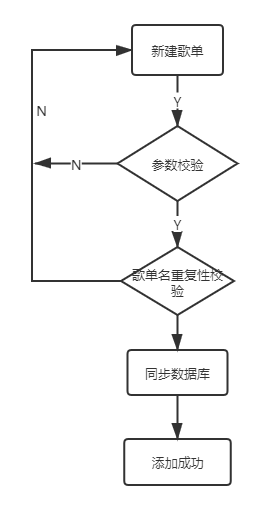


图4-6 用户创建歌单流程图

（5）用户听音乐实现

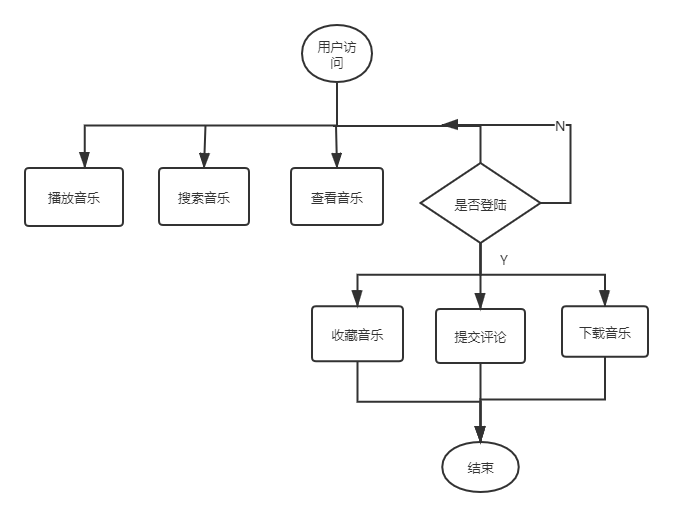


图4-7 用户播放音乐流程图

（6）用户个人中心

用户个人中心，用户可以查看自己的播放历史，自己创建的歌单，自己个人信息的修该，以及自己收藏歌单信息。具体设计效果图如下：



图4-8 用户个人中心的效果图

4.3 后台功能模块设计及实现

由于后台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。

4.3.1 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，具体包管理员的登录，管理员对用户信息动态观管理。并使用RBAC模型实现系统的权限控制。管理员可以给指定的用户进行授权。

（1）RBAC模型介绍

RBAC模型介绍：RBAC模型是在用户和权限之间定义了一个角色信息，在授权的过程中，具体的执行过程为：那些用户可以对那些内容进行那些操作。通过给不同的角色赋予不同的操作权限，进而将这些角色和用户进行绑定。提高了用户和权限之间的灵活性。

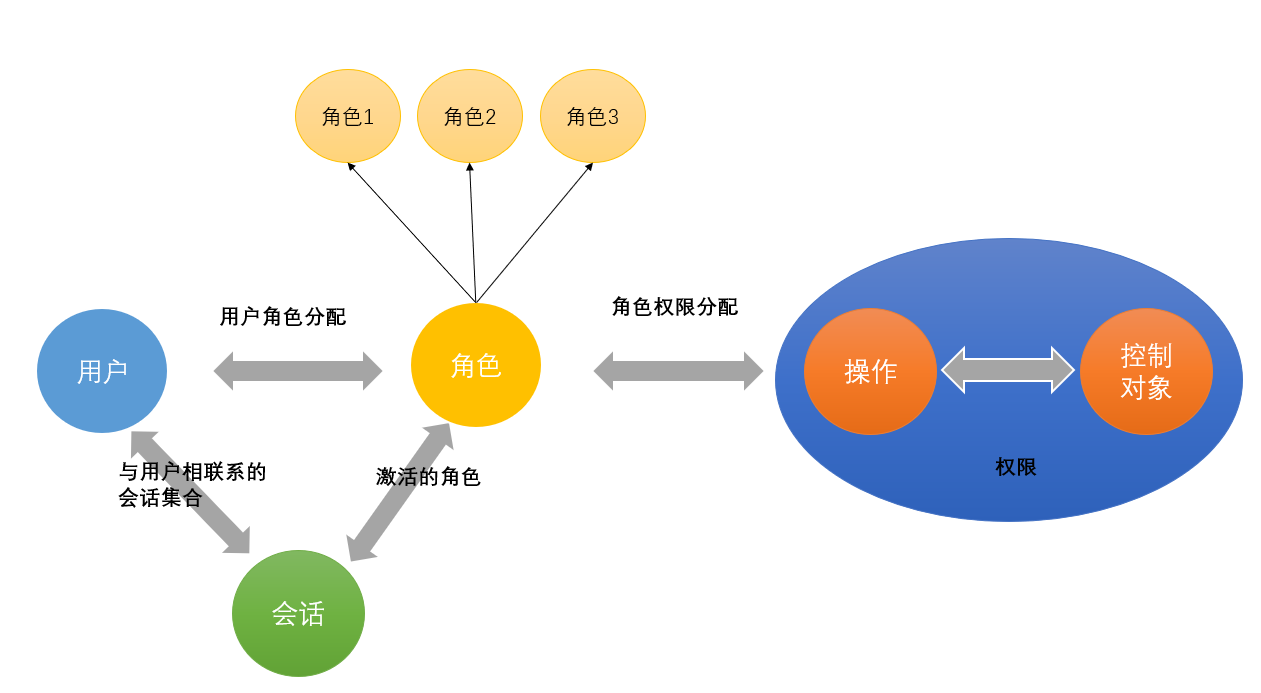


图4-9 基于RBAC模型的权限控制模型

（2）角色信息的动态管理

角色能使用户和权限灵活的进行组合。管理员可以根据不同的访问场景，动态的创建某个角色。并给特定的用户赋予指定的角色信息，达到权限控制的目的。管理员也可以对角色进行动态的管理，包括角色权限修改，角色的删除，以及角色访问接口和菜单的授权等功能。具体实现的效果图如下。



图4-10 角色赋予权限图

4.3.2 歌曲模块

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。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。

歌曲的模块展示以及后台管理设计界面如4-11所示

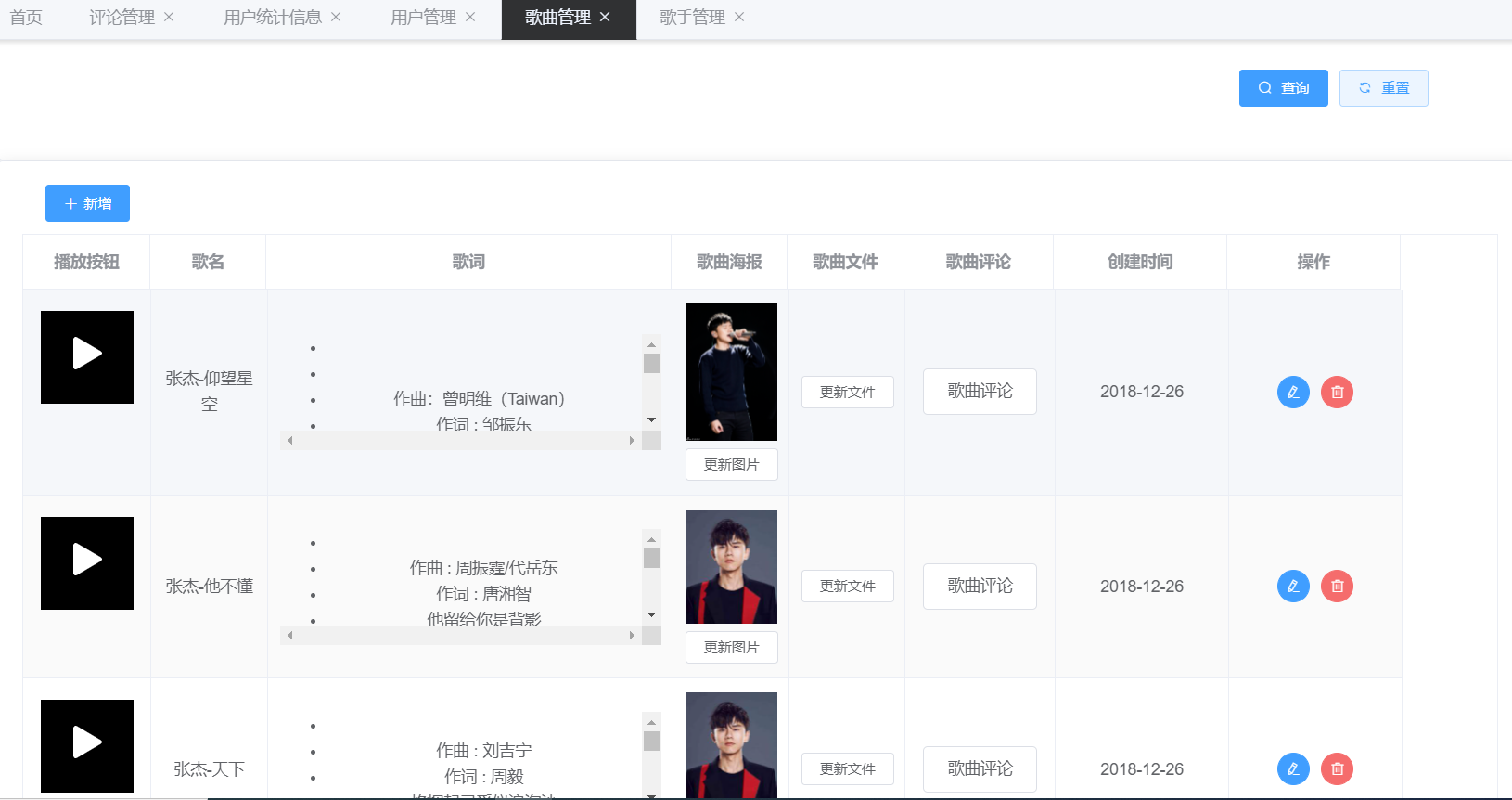


图4-11 歌曲的动态管理

具体操作有音频文件的添加，删除和查询，以及歌曲所关连的评论。歌曲搜索时根据歌曲名的模糊名进行匹配的，删除和添加歌曲的权限根据不同的角色具有不同的权限，只有管理员才能给指定的用户授权删除和添加的权限。这里重点描述每日歌曲推荐功能的实现。

（1）基于内容推荐算法简介

推荐算法通常是使用的有两种，一种是协同过滤，另一种是基于内容的推荐。本次系统主要采用的就是基于内容进行推荐的。具体就是根据用户每天播放歌曲的次数，用户对歌曲的评论，对歌曲的收藏次数等维度来确立相关歌曲信息。然后根据每首歌曲所属的类别确定推荐的歌曲。

（2）热门歌曲的推荐实现

本次推荐是通过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，利用这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，收藏4分，评论是3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。底层还是通过标签的方式将每首歌曲和用户进行关联。其实就是对用户进行画像，用户画像由两个部分构成，一部分是对用户端内容挖掘分析后得到用户的属性特征，另一部分是根据用户行为数据统计或学习到的对物品端结构化属性的偏好度量。

本次的音乐推荐的两种用户画像的方式都有用到，一方面通过用户收藏歌曲的数量，用户的对歌曲的评论次数以及用户的下载次数来综合确定每个用户的听歌的类别，来有区分的给每个用户推荐类似的歌曲信息。完成给用户的歌曲推荐，另一方面是后台可以根据用户的统计信息确定歌曲的热度的信息完成具体内容的推荐。

4.3.3 歌单模块

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的分类定义的歌单，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。后台数据库通过类型字段来区分不同歌单的类型。后台主要是对歌单进行动态的管理，包括查询用户具体创建了那些歌单，歌单中具体包含那些歌曲，歌单被收藏的次数等内容。系统会根据歌单的具体类型来区分不同是系统创建的歌单还是用户自己创建的歌单，系统创建的歌单一般是不允许被修改和删除的。

4.3.4 评论模块

（1）评论过滤敏感词的实现

用户提交评论时，为了避免用户的恶意评论，本次系统基于DFA算法，实现了敏感词过滤功能。

DFA算法简介： Deterministic Finite Automaton，即确定有穷自动机。其特征为：有一个有限状态集合和一些从一个状态通向另一个状态的边，每条边上标记有一个符号，其中一个状态是初态，某些状态是终态。但不同于不确定的有限自动机，DFA 中不会有从同一状态出发的两条边标志有相同的符号[8]。在过滤敏感词方面，效率比较高。本次系统中的评论过滤算法的实现，是基于DFA算法，具体使用Java中的HashMap来实现的。借鉴开源项目，从6W个敏感词进行匹配，并支持敏感词的脱敏，判断，返回和替换，支持全角和半角互换，支持用户自定义敏感词。

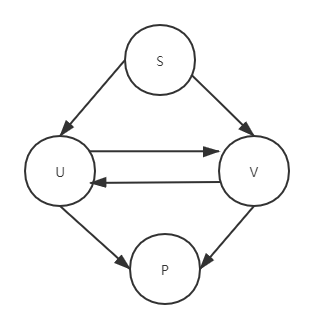


图4-12 DFA的算法模型

（2）评论功能的实现

用户在提交评论时，系统会将用户提交的评论内容进行过滤，如果含有敏感词汇，就会给用户一个提示信息，提示用户重新输入，这条评论内容也会被丢弃。每首歌曲的的评论内容是通过Mysql的数据库中一个关联表来和具体的用户来进行关联，并按照每个评论的点赞数来排序展示的，用户也可以回复别人的评论 。可以给喜欢的评论进行点赞。

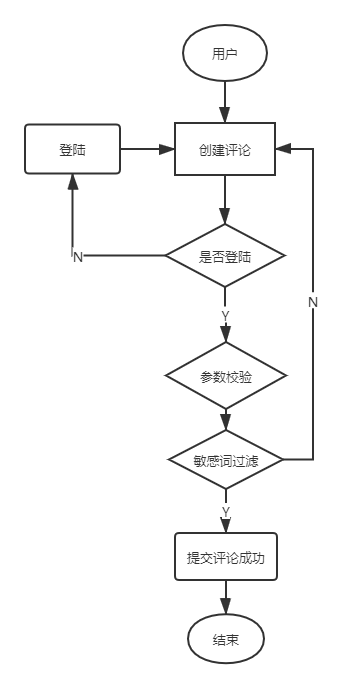


图4-13 用户提交评论程序流程图

4.3.4 歌手信息模块

歌手信息模块主要具备的功能： 歌手信息的分类展示，具体可分为男歌手，女歌手，组合，以及按照歌手的年龄段段进行区分。用户可通过歌手信息进行检索.歌手和歌曲的之间的关联通过mysql的一个关联表来进建立联系的。歌手和歌曲之间的关系是一对多的关系。点击某个歌手，会加载他所属的歌曲的信息。后台主要是对歌手信息进行动态的管理。

4.3.5信息统计模块

信息统计功能主要是为了统计歌曲，歌手信息以及用户信息的基本情况。前端主要采用Echars来进行可视化的展示，后端主要采用的是Ｍysql的联合查询来呈现相关的统计信息。

(1)综合界面设计

图4-14 首页信息展示

(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（系统每天会统计一首歌曲总的评论次数歌曲的收藏总次数，歌曲的播放的总次数，然后以可视化的方式来进行展示，系统管理员可以对每一天歌曲总的概况有一个直观的了解。

（3）用户信息统计设计实现

对用户的性别，用户的地域，用户的年龄段，来进行一个统计，实时的了解当前用户总数的概况 ，并对每天用户的活动状况有一个了解。并对用户总的情况有一个直观的展示。

图4-15 用户信息的统计实现

4.3.6 系统监控模块

系统的稳定性是很重要，系统监控模块就是实时监控系统的运行情况的。方便系统管理员实时的掌握系统的运行情况。

4.4 异常设计

（1）前端异常处理

前端的异常交互主要利用弹窗提示对后端异常信息的展示，通过axoios的拦截器对具体的错误码进行定义，具体定义了两个大的部分，以4XX开头，视为客户端的异常，5XX开头的为服务端的异常。

（2）后端异常处理。

后端异常处理分为运行异常和系统异常，运行异常使用枚举的方式来定义具体出错信息，前端通过错误码来渲染具体的出错信息，给用户一个良好的交互体验。系统异常是系统的出错信息，当系统出现系统错误，可以给管理员一个好的而提示并进行处理。

4.5 本章小结

本章详细描述了基于springboot的音乐在线分享系统的各个模块设计以及实现。主要完成了系统实现所需要的业务表设计，客户端的设计以及后台管理的设计，对每个子功能的执行流程图以及具体的实现效果进行了说明和展示。

第5章 系统测试

5.1 测试意义

软件测试的重要性不言而喻，任何一个系统的开发完成都离不开软件测试。完善测试是保证系统的功能是否减健全，系统的边界问题是否处理得当的重要依据。也是系统健壮性的重要保证。本章主要是对本系统实现的功能进行测试。具体测试内容：功能是否完善，边界问题是否处理得当，具体的输入校验是否得当等方面。

5.2 测试主要方法

测试主要采用的方法有静态测试、单元测试（黑盒测试和白盒测试）和组装测试[9]。

（1）静态测试： 不运行被测试的本身，仅通过分析和检查源程序的语法和结构，过程来检查程序的正确性[9]。

（2）单元测试： 单元测试又称模块测试，逻辑测试活结构测试，其目的在于检测每个程序单元能够正确的实现详细说明书中的功能[9]。

（3）要方法为黑盒测试。测试单元间的接口以及组装后的功能，使用黑盒测试方法测试组装的功能[9]。

5.3 测试内容

表5-1 功能需求表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 功能名称 | 功能描述 | 输入 | 系统响应 | 输出 |
| 1 | 用户信息存储 | 用户信息存取 | 用户信息 | 将全部用户信息存放到数据库相应的物理表中 | 提供用户的登录验证 |
| 2 | 建立并维护歌曲信息 | 建立歌曲信息表 | 歌曲的基本信息 | 歌曲的基本信息 | 添加歌曲信息成功。 |
| 3 | 注册登陆界面 | 用户登录注册 | 输入用户信息 | 系统调用数据库的基本信息，并生成token | 注册登录成功 |
| 4 | 提交评论功能 | 实现对歌曲评论 | 评论内容 | 评论过滤，存入到数据库中 | 评论成功 |

表5-2 性能需求表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 性能名称 | 性能描述 | 输入 | 系统响应 | 输出 |
| 1 | 信息查询 | 主要是对歌曲，歌单，用户的基本信息进行查询 | 输入搜索关键词 | 前后台系统根据关键词搜索存储的索引，再去调用数据库信息。后台则直接调用数据库数据 | 查询成功 |
| 2 | 信息修改 | 对数据库存储的信息进行修改 | 歌曲，歌单，歌手，用户收藏，用户评论的相关信息 | 将更新后的信息存储到指定的数据库的表结构中 | 更新成功 |
| 3 | 信息增加 | 信息增加是对数据库中的数据进行增加 | 录入歌曲，歌单，歌手等信息 | 增加数据库中相应表的记录信息 | 添加成功 |
| 4 | 信息删除 | 对数据库存储的相关信息进行删除 | 后台管理选择要删除的数据 | 系统对数据库中的数据进行删除 | 提示删除的信息 |

5.4 测试结果

表5-3 功能测试报告

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 功能名称 | 功能描述 | 输入 | 系统响应 | 输出 | 测试结果 |
| 1 | 用户信息存取 | 用户信息存取 | 用户信息 | 将全部用户信息存放到数据库相应的物理表中 | 提供用户的登录验证 | 通过 |
| 2 | 建立并维护资讯信息 | 建立歌曲信息表 | 歌曲的基本信息 | 歌曲的基本信息 | 添加歌曲信息成功。 | 通过 |
| 3 | 注册登陆界面 | 用户登录注册 | 输入用户信息 | 系统调用数据库的基本信息，并生成token | 注册登录成功 | 通过 |
| 4 | 提交评论功能 | 实现对歌曲评论 | 评论内容 | 评论过滤，存入到数据库中 | 评论成功 | 通过 |

表5-4 性能测试报告

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 性能名称 | 性能描述 | 输入 | 系统响应 | 输出 | 测试结果 |
| 1 | 信息查询 | 主要是对歌曲，歌单，用户的基本信息进行查询 | 输入搜索关键词 | 前后台系统根据关键词搜索存储的索引，再去调用数据库信息。后台则直接调用数据库数据 | 查询成功 | 通过 |
| 2 | 信息修改 | 对数据库存储的信息进行修改 | 歌曲，歌单，歌手，用户收藏，用户评论的相关信息 | 将更新后的信息存储到指定的数据库的表结构中 | 更新成功 | 通过 |
| 3 | 信息增加 | 信息增加是对数据库中的数据进行增加 | 录入歌曲，歌单，歌手等信息 | 增加数据库中相应表的记录信息 | 添加成功 | 通过 |
| 4 | 信息删除 | 对数据库存储的相关信息进行删除 | 后台管理选择要删除的数据 | 系统对数据库中的数据进行删除 | 提示删除信息 | 通过 |

5.5 本章小结

本章主要对系统的实现进行了测试，主要测试两个大的方面，一个是功能性测试，一个是性能测试，对于测试中不合格的地方，进行了纠正。测试的主要目的就是发现系统中的不足，完善系统的功能，使系统更加的稳定和健壮。

结 论

本课题主要以国内外现有的音乐软件为出发点，针对当前的音乐软件存在的优点和缺点，进行了有效的整合。使用当下比较流行的技术springboot和vue作为前后端开发的主要技术。本着安全，高效，使用，易操作的原则，旨在为用户提供一个稳定，易用的音乐平台。

系统作为音乐分享平台，着重考虑了用户的使用体验以及和用户信息安全。并基于RBAC模型实现系统的多角色管理。有利于项目后期的扩展和使用。在安全方面，主要采用的springSecurity安全框架，对用户的信息进行有效的保护。并使用DFA算法对用户的评论内容进行了过滤，保证了评论内容的质量。

客户端主要是和用户进行交互，核心的模块是歌单模块，歌曲模块，以及用户个人中心的模块等。为用户提供了较好的交互体验。用户可以在线听音乐，收藏音乐，以及下载音乐。使用户不但可以实现在线听音乐的功能，还可以创建歌单来管理自己音乐，为用户听音乐提供了很大的便捷性。

后台管理系统主要的功能是对整个系统的信息进行统计和管理。管理员可以实时的了解用户，歌单，歌曲，歌手，评论的信息等内容，也可对整个系统的运行情况有一个及时的掌握。便于后期对用户的习惯以及音乐的流行趋势进行分析。

本次系统虽然最终完成了需求分析阶段的有关功能。但是系统仍然存在着很多不完善地方。比如，用户播放歌曲时可能出现卡顿的现象，缓存key的设计不够合理，前端的部分图片的加载速度慢等问题。在今后的生活和学习中，也会进一步对系统进行完善和改进，使本系统能被更多的人所使用。

致 谢

大学的时光已然接近尾声。回顾大学的生活，有充实，有欢乐，也有考试前的忙碌。也有和同学之间相互合作完成大作业的成就感。经历的这一切都将是我宝贵的财富。

这次的毕业设计也是对大学生活一个比较好的总结。从确定选题到项目调研，从系统的设计到系统的实现，让我深切的感受到实现一个系统的不易。也对大学四年学习的内容有了比较好的运用。当然也在系统的实现过程中遇到了很多的困难，比如系统实现中用到的前端技术vue自己就不是很了解，后期通过官方文档学习，满足了系统实现的需求。在系统的实现过程中，也有操之过急的心理，导致部分功能的设计得不是很完善，后期实现起来比较麻烦。有些边界问题考虑的不是很到位，影响到系统的健壮性。

当然这次毕业设计要特别感谢我的指导老师汤永新老师，从帮助我选题到分析课题实现的细节以及难点，从项目调研到系统的总体实现，都给了我比较全面的指导。在这过程中也要感谢帮助过我的同学和老师。才能使我的毕业设计能按照预期完成。

大学生活的结束将意味着另一种的生活的开始，也希望同学们能不忘初心，继续前行，在自己擅长的领域发光发热。

参考文献

[1] 赵一凡，卞良，丛昕. 基于MD5的加盐消息摘要Java实现[J]. 软件导刊，2018,17（03）:214-216+220.

[2] 凯S.霍斯特曼.Java核心技术卷一.周立新.叶乃文.邝劲筠.杜永萍.10.机械工业出版社,2016(08).

[3] 梁灏.Vue.js实战.清华大学,2018(06).

[4] 蒋睿.MySQL数据库安全研究[J].电脑知识与技术,2020,16(09):3-4+21.

[5] 汪永松.JavaWeb开发技巧之项目模板[J].电脑编程技巧与维护，2020（02）：3-8+16.

[6] 段震.浅谈MySQL数据库有关数据备份的几种方法[J].山西电子技术,2020(02):17-18.

[7] 魏金岭,周苏.软件项目管理与实践.清华大学,2018(07).

[8] 吴珊,李英祥,徐鸿雁,张仕霞,施宜军.基于改进的Trie树和DFA的敏感词过滤算法[J/OL].计算机应用研究:1-6[2021-05-14]

[9] 吴迪,马世霞,吴义芝.软件测试基本原理与实践.电子科技出版社,2018(01)

[10] Ying Xin Liu,Xi Yuan Li. Design and Implementation of a Business Platform System Based on Java[J]. Procedia Computer Science,2020,166.

[11] Tian Ming Huang. Design and Implementation of App System for Legal Consulting Based on JAVA Technology[J]. Procedia Computer Science,2020,166.

[12] Amelie Gyrard, Amit Sheth IAMHAPPY: Towards an IoT knowledge-based cross-domain well-being recommendation system for everyday happiness[J]  Smart Health, 2020, 15.