#### 密级：



**NANCHANG UNIVERSITY**

**学 士 学 位 论 文**

THESIS OF BACHELOR

（2017 —2021 年）



题 目 在线音乐点播系统

学 院： 软件学院 系 软件工程

专业班级： 软件工程1910班

学生姓名： 卞因杰 学号： 8002119304

指导教师： 刘海 职称： 教授

**南 昌 大 学**

**学士学位论文原创性申明**

本人郑重申明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式表明。本人完全意识到本申明的法律后果由本人承担。

作者签名： 日期：

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权南昌大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在 年解密后适用本授权书。

本学位论文属于

不保密□。

（请在以上相应方框内打“√”）

作者签名： 日期：

导师签名： 日期：

**在线音乐点播系统**

专 业： 软件工程 学 号：8002119304

学生姓名：卞因杰 指导教师： 刘海

# 摘要

随着互联网的普及和发展，音乐行业发生了巨大的变化，许多人听歌的方式也发生了变化。由一开始的购买CD然后使用DVD播放，到如今在各个音乐APP中搜索并点播自己喜欢的音乐，还可以在歌曲中发表自己的听歌感想等。同时，在巨大的市场需求下，许多音乐平台如雨后春笋般涌出，用以满足不同用户对音乐的个性化需求和日益扩张的社交需求。

但是，蓬勃发展的背后也是乱象丛生，许多音乐平台都需要去下载他们家的APP才能正常使用其功能，这与用户想要轻轻松松来上一首歌的想法是相违背的。有些音乐网站提供了免下载APP就可以在线听歌的方案，但又需要用户去登录注册等步骤，在我看来也是徒增繁琐。为了能让用户轻轻松松，没有后顾之忧的听上令人舒心的音乐，我结合大学所学习到的知识，完成了对这个在线音乐点播系统的需求分析和数据库的相关设计，并抽象出了用户、歌曲、评论等最基本的实体间的关系。使用MySql数据库建表，进行数据的存储。后端采用Nodejs快速搭建出能够处理用户请求的管理系统。前端则使用Vue3+ElemenPlus组件化开发的方式，完成了简洁干练的用户界面的设计和编写。

目前，在线音乐点播系统经过设计、编写、运行和测试后，已经能在本地服务器上运行，在云服务器上部署后，就能提供给用户一个畅通无阻的在线听歌体验。

**关键词**：在线音乐；后台管理；MySql数据库

#### Online Music On Demand System

# Abstract

With the popularization and development of the Internet, the music industry has undergone tremendous changes, and the way many people listen to music has also changed. From purchasing CDs and playing them on DVDs at the beginning, to now searching and playing your favorite music on various music apps, and even expressing your listening thoughts in the songs. At the same time, under the huge market demand, many music platforms have sprung up like mushrooms after rain to meet the personalized needs of different users for music and the expanding social needs.

However, behind the booming development, there is also chaos, and many music platforms need to download their apps to use their functions properly, which goes against the idea of users wanting to easily play a song. Some music websites offer a solution that allows users to listen to music online without downloading an app, but it also requires users to log in and register, which in my opinion is just too cumbersome. In order to allow users to listen to comfortable music without any worries, I combined the knowledge learned in university to complete the requirements analysis and database design of this online music on demand system, and abstracted the most basic relationships between users, songs, comments, and other entities. Create tables using the MySql database for data storage. The backend uses Nodejs to quickly build a management system that can handle user requests. The front-end uses the Vue3+ElemenPlus component based development approach, completing the design and writing of a concise and concise user interface.

At present, the online music on demand system has been designed, written, run, and tested, and can now run on local servers. After deployment on cloud servers, it can provide users with a seamless online listening experience.

**KeyWords**：Online music; Back-stage management; MySql database

**目录**

[摘要 I](#_Toc134745250)

[Abstract II](#_Toc134745251)

[第一章 绪论 1](#_Toc134745252)

[1.1选题的依据及意义 1](#_Toc134745253)

[1.2国内外研究现状及发展趋势 1](#_Toc134745254)

[1.3本课题研究内容与技术路线 2](#_Toc134745255)

[1.4论文结构 3](#_Toc134745256)

[第二章 开发环境和主要技术 4](#_Toc134745257)

[2.1开发环境 4](#_Toc134745258)

[2.2开发语言 4](#_Toc134745259)

[2.3框架技术 4](#_Toc134745260)

[2.3.1 Vue及其相关技术 5](#_Toc134745261)

[2.3.2 Koa及其相关技术 6](#_Toc134745262)

[2.3.3 Sequelize 6](#_Toc134745263)

[2.3.4 其他 6](#_Toc134745264)

[2.4开发工具 7](#_Toc134745265)

[第三章 需求分析 8](#_Toc134745266)

[3.1系统概述 8](#_Toc134745267)

[3.2可行性分析 8](#_Toc134745268)

[3.3系统用例分析 8](#_Toc134745269)

[3.4系统用例规约 10](#_Toc134745270)

[3.4.1 播放用例的用例规约 10](#_Toc134745271)

[3.4.2 登录用例的用例规约 10](#_Toc134745272)

[3.4.3 添加歌曲用例的用例规约 11](#_Toc134745273)

[第四章 系统设计 13](#_Toc134745274)

[4.1系统功能模块设计 13](#_Toc134745275)

[4.1.1 用户登录模块 13](#_Toc134745276)

[4.1.2 用户修改头像模块 14](#_Toc134745277)

[4.1.3 用户点赞评论模块 15](#_Toc134745278)

[4.1.4 用户或管理员删除评论模块 16](#_Toc134745279)

[4.2系统数据库设计 17](#_Toc134745280)

[4.2.1 用户表 17](#_Toc134745281)

[4.2.2 歌手表 18](#_Toc134745282)

[4.2.3 歌曲表 18](#_Toc134745283)

[4.2.4 评论表 18](#_Toc134745284)

[4.2.5 收藏歌曲表 19](#_Toc134745285)

[4.2.6 历史记录表 19](#_Toc134745286)

[4.2.7 评论点赞状态表 20](#_Toc134745287)

[4.2.8 歌单表 20](#_Toc134745288)

[4.2.9 歌单歌曲表 20](#_Toc134745289)

[4.2.10 数据库实体关系图 20](#_Toc134745290)

[第五章 系统实现 22](#_Toc134745291)

[5.1登录模块 22](#_Toc134745292)

[5.2个人信息管理模块 23](#_Toc134745293)

[5.3歌曲管理模块 24](#_Toc134745294)

[5.4界面截图展示 26](#_Toc134745295)

[第六章 系统测试 30](#_Toc134745296)

[6.1引言 30](#_Toc134745297)

[6.2测试环境 30](#_Toc134745298)

[6.3测试结果 30](#_Toc134745299)

[6.3.1 登录注册模块测试表 30](#_Toc134745300)

[6.3.2 个人信息管理模块测试表 31](#_Toc134745301)

[6.3.3 播放模块测试表 32](#_Toc134745302)

[6.3.4 评论模块测试表 33](#_Toc134745303)

[6.3.5 用户管理模块测试表 33](#_Toc134745304)

[6.3.6 歌手管理模块测试表 33](#_Toc134745305)

[第七章 总结与展望 35](#_Toc134745306)

[7.1总结 35](#_Toc134745307)

[7.2展望 35](#_Toc134745308)

[参考文献 36](#_Toc134745309)

[致谢 37](#_Toc134745310)

# 第一章 绪论

## 1.1选题的依据及意义

随着Web技术的不断发展，各种各样的网站也随之产生了。比如视频网站、游戏网站、在线制作证件照网站等等。从曾经只能阅读枯燥的文字和图片，到如今享受各种眼花缭乱的功能，人们浏览网站的体验也是获得了巨大的提升。而在如今这样一个信息化的网络时代，音乐变得无处不在，尤其是多媒体的出现让音乐更加大众化、普及化了[1]。并伴随着数据的压缩编码和用于发布媒体的网络技术的诞生，制作一个网页端的音乐平台便成为了可能[2]。

现今生活节奏加快，外出游玩对部分人来说可能是种奢求，于是欣赏音乐无疑成为了一种好的娱乐方式。技术与互联网的进步，让智能手机与无线网络飞速发展[3]，新时代下几乎人手一部智能手机，公共场所无线网络遍布，与此同时音乐爱好者对数字音乐的需求方式越来越高[4]，为了提供人们一个无需下载应用，即搜即用的音乐平台，于是我计划做一个在线音乐点播系统，以满足人们听歌放松的需求。

## 1.2国内外研究现状及发展趋势

音乐应用分为很多种，有APP端的，如酷狗、QQ音乐等，还有一类就是网页版，代表如网易云音乐，客户在电脑上安装浏览器，终端提供地址栏输入网站地址，即可进入网页。网络音乐由于得益于互联网的便利，大大减少了音乐产业价值链中的中介数量，以低成本的数字化替代了现有高成本的物质生产、包装、传播及售后服务等环节，使得网络音乐市场得到了快速发展，对传统音乐产生了巨大的冲击[5]。并且现今国内外各大音乐APP的模式也变得大同小异，基本上都会根据用户的一些爱好推荐用户所喜爱的歌曲、歌手，或是包含特定风格的歌单等，用以加强用户黏性，提高用户浏览的时间[6]。同时，现在的音乐APP融入了社交属性，你可以与志趣相投的人一起欣赏音乐，在歌曲下留下你的感想等。当前我国众多的网上音乐平台仍仅限于互联网的一个层次，未能在同一平台上提供个性化的业务。现在能够完成此类网上音乐库的系统并不多见。只有能在网页上实现对音乐系统的综合管理规范，只有这样，我们的网上音乐网站，才能更好的适应使用者的需要。因此，网上的音乐网站很有潜力，也是一种流行的休闲方式。

众所周知，社交是人类社会重要的组成部分。当前，游戏社交化已成为网络游戏的发展趋势，在游戏中进行互动交流与人际交往已成为玩家进行游戏体验的关键需求之一[7]。为什么类如王者荣耀的网络游戏能成为国民手游，而一些制作精良的单机游戏却很难将用户覆盖的如此之广，因为游戏中的社交不仅仅能促进情感的交流，加强彼此的共鸣，同时也能将游戏迅速扩散出去，相当于用户给游戏免费打了广告。音乐也是相同的道理，音乐下的评论连接了一个又一个的陌生人，你或许会在一个冷门的歌里找到一个感同身受的听众，记住这首歌，从而延长这首歌曲的寿命。由此可见，音乐中的社交是很有必要的，而且未来的歌曲社交的属性也会越来越多。

因此，制作一个包含社交属性的音乐网站才能跟上时代的潮流，让用户享受音乐的同时，能够满足用户的社交需求。

## 1.3本课题研究内容与技术路线

在线音乐点播系统，主要用于提供给用户方便快捷、简单易用的听歌方式，同时带上一些的社交属性，让用户听歌时能与其他听众擦出思维的火花，提高用户的听歌体验。由此，本课题的主要研究内容如下：

建立合适的数据库，存储用户听歌时产生的数据，实现自动登录，用户喜爱歌曲的收藏，听歌历史记录等。

优化后端代码，降低代码耦合性，提高代码的复用率，可适当采用“以空间换时间”的方式，加快服务器的响应速度，降低用户等待时的焦虑，提升用户的体验。

设计并实现一个符合用户操作逻辑的前端界面，给用户舒适的听歌体验并提供一个评论系统用以实现用户间的交流。

本课题的技术路线如下：后端采用Nodejs编写，采用Koa框架解析并处理前端发送的请求，使用Sequelize形成与Mysql数据库的关系映射，由此方便的进行数据库中表的操作。使用koa-static进行文件的动态托管，主要是音乐文件和用户的头像。前端使用Vue3框架，结合ElementPlus，快速构建出符合大多数人审美的用户界面。发送请求时采用axios，向后端发送get或是post请求，然后根据后端的返回值改变响应式数据的值，从而实现页面的更新。

## 1.4论文结构

第一章：绪论。本章阐述了在线点播系统的选题依据和当前关于本课题的国内外类似研究的现状，还结合其他学者的论文进行了本课题未来可能的发展趋势的预测。简要叙述了本课题的研究内容。

第二章：开发环境和主要技术。本章阐述了项目的开发环境，以及使用到的技术和其对应的版本。例如Vue和Node.js。

第三章：需求分析。本章主要对本系统的用户的需求进行了全面具体的分析。并且绘制了用户和管理员的用例图和部分用例规约。

第四章：系统设计。本章主要介绍了对Mysql数据库的表的设计，以及系统的功能模块的设计。

第五章：系统实现。本章主要对系统的一些功能进行展示并加以说明。

第六章：系统测试。对系统的各个功能进行测试，测试代码的健壮性与前端界面运行的流畅性。

第七章：总结与展望。本章主要阐述了我进行此项目时所遇到的困难和解决方法，还有从中得到的心得体会和收获。除此之外，还有我对现在完成项目未来的规划，指出其不足和能改进的地方，从而使其功能更加完善。

# 第二章 开发环境和主要技术

## 2.1开发环境

系统软硬件开发环境，如表2-1所示。

**表2-1 系统软硬件开发环境表**

|  |  |  |
| --- | --- | --- |
| 编号 | 名称 | 环境及版本 |
| 1 | 操作系统 | Windows10 家庭版 |
| 2 | 内存 | 8GB |
| 3 | 数据库 | Mysql8.0.32 |
| 4 | 开发工具 | VS Code |
| 5 | Web服务器 | Node16.16.0 |
| 6 | Vue脚手架 | Vue/cli 5.0.8 |

## 2.2开发语言

本系统采用Node.js[8]作为后端开发的环境。Node.js并不是一种语言，它是一种能够允许Javascript在服务端运行的环境。Node.js是以Google开源的V8引擎为基础进行编写的，我们能在浏览器按下F12并在里面控制台写下JavaScript的代码并运行，这就是JavaScript在浏览器环境中运行。而Node.js可以让JavaScript不在依赖浏览器而直接在本地中运行，从而实现利用JavaScript进行后端的开发。

Node.js的语法基本与Javascript相同，这使得一些从事前端的工作者们可以迅速的投入到后端的编写中。因为其弱类型的变量声明方式，导致其开发时特别的灵活，可以显著地提升开发效率。Node.js是基于事件驱动和无阻塞的，所以非常适合处理并发请求，但这并不是重点。现在await、async语法糖的加入使得Node.js处理异步代码更加的清晰和方便，考虑到系统中将会有许多因操作数据库而产生的异步代码，所以我选择了Node.js。

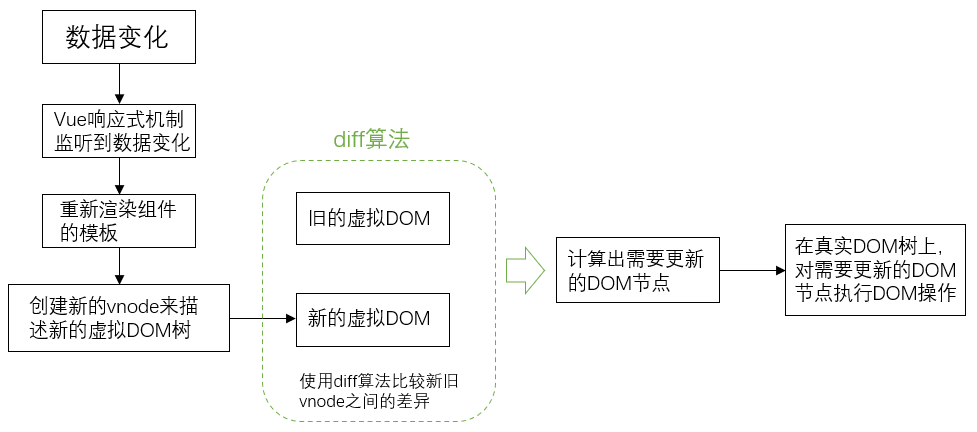
## 2.3框架技术

### 2.3.1 Vue及其相关技术

本系统采用前后端分离的开发方式进行开发。在前端的设计开发中，我使用了前端三大主流框架中简单易学且轻量级的Vue[9]框架，以用来实现我的单页面应用的开发。

Vue是一套用于构建用户界面的渐进式框架。在以往的前端开发中，如果从服务端传来的数据发生了变化，我们则需要获取到发生了变化的dom元素，亲自去更新它的值，这就意味着有时我们需要频繁的操作dom元素。这会引起浏览器页面频繁的渲染和重绘，将极大地消耗浏览器的性能，可能还会因此造成卡顿，影响用户的体验。而使用了Vue之后，我们可以将可能发生变化的数据存储在一个变量中，然后让Vue去监听这个数据，Vue会创建一层虚拟dom与浏览器的真实dom相映射，当数据发生变化后，Vue就会通过diff算法找出页面哪里发生了变化，从而单独的更新这一区域的元素。有了Vue，开发者们就只用关心变量的值，而不用自己去更新元素了，极大的提升了开发效率。

虚拟DOM实现原理如图2-2所示：



**图2-2 虚拟DOM实现原理图**

Vue还有一个强大的功能，那就是Component(组件)，有时我们浏览其他网站，看见其他开发者的界面，想要将里面的一小部分拿来自己用，则不仅需要复制html结构，CSS样式，还需要找到js所编写的事件处理函数，十分不便。而Vue生成的一个后缀为.vue的文件，相当于是html、CSS、js三大件的集合体，这样一个文件就包含了以上所需要的数据，这样一个.vue文件就可以称作为组件。组件可以让开发者将复杂的页面模块化，一小点一小点的去开发，然后像搭积木一样将一个个组件组装起来，这样也可以在其他地方用到这块“积木”，提高了代码的复用性。例如网站的头部导航栏，需要在多个页面显示，则最适合将其封装为组件。

VueRouter[10]是Vue官方打造的针对单页面应用程序的路由管理插件。可根据路由动态切换显示的组件，并可以设置路由守卫，例如：用户未登录则不能进入到“我喜欢”页面，可以通过路由守卫方便的实现。

Pinia是专为Vue打造的轻量化状态管理工具。可以在不同组件间和页面间传递变量的值，相当于一个全局变量，能够被所有组件读取和更改，且实现了响应式，会自动更新页面中使用到它的地方。

### 2.3.2 Koa及其相关技术

Koa[11]是服务于Node.js微型框架。在其中可以使用引入中间件的方式完成对其功能的扩展，将ctx(上下文)中的数据由中间件进行加工，然后传给下一个中间件，一层一层对数据进行处理，从而使逻辑清晰明了。async/await的使用也可以让异步程序以同步的代码风格编写，极大提高了开发效率。

koa-body是一个处理网络请求的中间件，可以通过ctx.request.body或是ctx.request.files获得请求中body部分的内容，包括json、表单、文本、文件等类型。

koa-static是一个能将静态资源托管到服务器上的中间件，在本项目中，我使用其来托管头像和歌曲。

### 2.3.3 Sequelize

Sequelize[12]是一个基于promise的Node.js ORM，其可以支持Mysql、SQLite等诸多关系型数据库。Sequelize会构建一个模型，相当于数据库中表的抽象，然后就可以通过调用模型上的方法实现对数据库的操作，既快捷又高效。

### 2.3.4 其他

客户端通过axios[13]对服务端发送请求，然后通过.then方法获取到服务端返回的数据并进行处理。

服务端引入了JsonWebToken库，将用户一些信息当作payload（载荷）用来生成用户首次登录时的token，下次用户访问网站时，只需将token携带，交给JsonWebToken来比对token，成功就可以获得传入的载荷。

客户端使用了ElementPlus组件库，其提供了丰富的符合大众审美的强自定义化的组件，可以免去写过多CSS的烦恼，以提升开发效率。

## 2.4开发工具

VS Code：用来编写前后端的代码，其丰富的插件提供了代码提示、自定义关键字补全诸多好用的功能，还可以免命令运行后端和前端项目，十分方便。

Git[14]：用来进行项目的版本控制，查看代码的更改位置，记录每天的项目进度，push到Github后，进行代码的备份。

Postman：后端开发利器，提供了多种请求的发送，可以方便地测试后端代码，还可以将请求分类整理，以项目的形式保存。

Navicat：数据库可视化操作工具，可自动生成数据库实体关系图。

StarUML：用来绘制用例图、流程图、时序图、数据库实体关系图等。

Chrome浏览器：调整CSS样式，控制台打印数据，限制网络速度、查看请求Header等。

# 第三章 需求分析

## 3.1系统概述

本系统的目标是设计出一个在线的音乐网站，让用户能够搜索歌曲、播放和欣赏歌曲，还能在歌曲中评论且能对喜欢的评论点赞。之后能收藏喜欢的歌曲，并在历史记录中查看播放过的歌曲。而作为管理员，应该能够实现对歌曲的管理、对歌手管理、对用户的管理和对歌曲中评论的管理。系统需要实现根据用户不同的权限给予不同的操作界面。而且即使绕过了前端，后端的权限判断也能保证整个系统的安全。

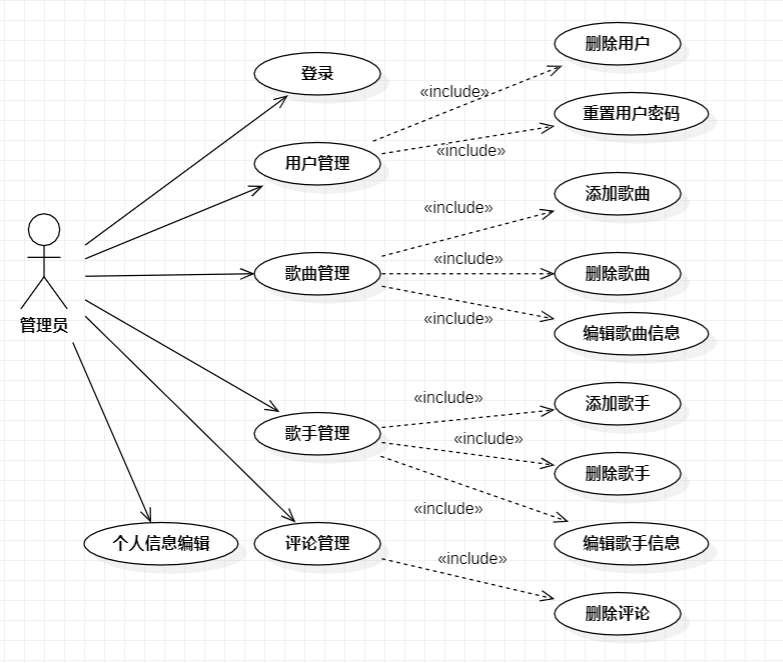
## 3.2可行性分析

本系统使用VS Code作为开发工具，Node.js作为后端的Javascript运行的环境，其有着丰富的库，能够提供给我遇到困难时全面的解决方案。后端采用Koa框架结合koa-body中间件，都是非常成熟的技术，遇到问题也能很快的百度出答案。而Mysql数据库作为我很多次项目的选择，我也已经十分熟悉。前端的Vue框架由国人尤雨溪开发，还有官方的中文文档，旗下的VueRouter和Pinia也有很高的讨论热度。HTML5原生的<audio>标签也完全能够支持我的项目。

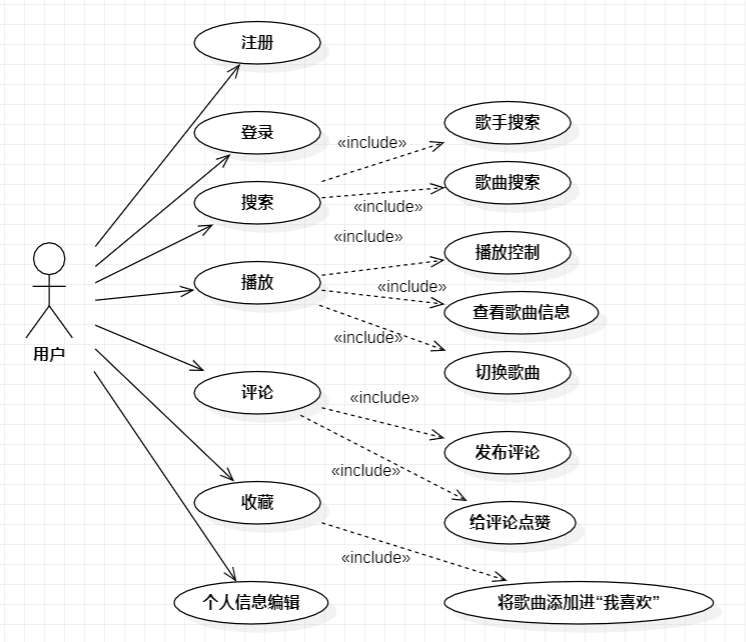
由此可见，本系统从使用技术栈和项目结构两个角度来看，都是有着很高可行性的，是完全可行的。

## 3.3系统用例分析

本系统的设计目的是实现一个在线的音乐网站。而网站内的信息不是本来就存在于数据库中的，于是需要一个管理员的角色，管理员除了要有与用户相同的登录、修改个人信息等功能需求外，在用户管理中，需要实现用户删除和重置用户密码的功能需求；在歌曲管理中，需要实现添加歌曲、删除歌曲、编辑歌曲信息、筛选歌曲的功能需求；在歌手管理中，需要实现添加歌手、删除歌手、编辑歌手信息的功能需求；在评论管理中，需要实现评论删除的功能需求。而对于普通用户来说，他的功能需求应包括：注册、登录、编辑个人信息、搜索歌曲、评论、给评论点赞、删除自己的评论、收藏歌曲等。系统管理员的用例图如图3-1所示，普通用户的用例图如图3-2所示。



**图3-1 管理员用例图**



**图3-2 用户用例图**

## 3.4系统用例规约

使用用例规约是用来进一步说明本系统的需求，是下一阶段系统设计的基础，也是测试用例的重要依据。以下是部分用例的用例规约表。

### 3.4.1 播放用例的用例规约

播放用例的设计是本在线音乐点播系统的重点，所有其他用例都要服务于它，用户需要进入到本系统，选择一首歌曲点击播放按钮进行播放。同时，用户还可以查看歌曲的热度、歌词等其他信息。播放用例的用例规约如表3-3所示。

**表3-3 播放用例用例规约表**

|  |  |
| --- | --- |
| 用例名称 | 播放用例 |
| 用例描述 | 用户播放音乐 |
| 参与者 | 用户 |
| 状态 | 打开在线音乐点播平台 |
| 前置条件 | 无 |
| 后置条件 | 歌曲播放或暂停 |
| 基本操作流程 | 1. 用户进入到平台 2. 用户点击歌曲进行播放 3. 用户暂停歌曲 4. 用户切换歌曲 |
| 异常流 | 用户播放歌曲失败，提示用户网络出现故障 |
| 补充说明 | 如果用户处于登录状态，则将该歌曲存入到该用户的播放历史记录中，并保存到数据库中 |

### 3.4.2 登录用例的用例规约

用户若没有登录，则不能记录用户的播放历史记录，用户也不能收藏歌曲和评论。登录用例的用例规约如表3-4所示。

**表3-4 登录用例用例规约表**

|  |  |
| --- | --- |
| 用例名称 | 登录用例 |
| 用例描述 | 用户登录在线音乐点播系统 |
| 参与者 | 用户 |
| 状态 | 未处于登录状态 |
| 前置条件 | 未处于登录状态 |
| 后置条件 | 用户成功登录进入系统 |
| 基本操作流程 | 1. 用户进入到平台 2. 用户输入账号和密码 3. 用户点击登录按钮 4. 用户成功登入系统 |
| 异常流 | 登录失败，提示用户检查密码或用户网络出现故障 |
| 补充说明 | 登录成功后，服务器将返回token，客户端则将token存入到LocalStorage中，以进行用户的自动登录 |

### 3.4.3 添加歌曲用例的用例规约

管理员需要先创建一名歌手，然后在该歌手名下创建歌曲，注意新创建的歌曲不能与该歌手下已经存在的歌曲同名。添加歌曲用例的用例规约如表3-5所示。

**表3-5 添加歌曲用例用例规约表**

|  |  |
| --- | --- |
| 用例名称 | 添加歌曲用例 |
| 用例描述 | 管理员添加新的歌曲 |
| 参与者 | 管理员 |
| 状态 | 管理员处于登录状态 |
| 前置条件 | 歌曲所属的歌手名下没有同名歌曲 |
| 后置条件 | 一首新的歌曲被添加到该歌手的名下 |
| 基本操作流程 | 1. 管理员进入到平台 2. 管理员输入账号和密码 3. 管理员点击登录按钮 4. 管理员成功登入系统 5. 管理员进入到歌手管理页面 6. 管理员选择一名歌手并点击添加歌曲按钮 7. 管理员在弹出的对话框中输入歌曲名称 8. 管理员选择歌曲的音乐文件 9. 管理员输入歌曲的歌词 10. 管理员点击添加 11. 成功添加了一首新歌曲 |
| 异常流 | 添加失败，提示管理员检查歌曲名称或检查选择的音乐文件，或是管理员网络出现故障 |
| 补充说明 | 无 |

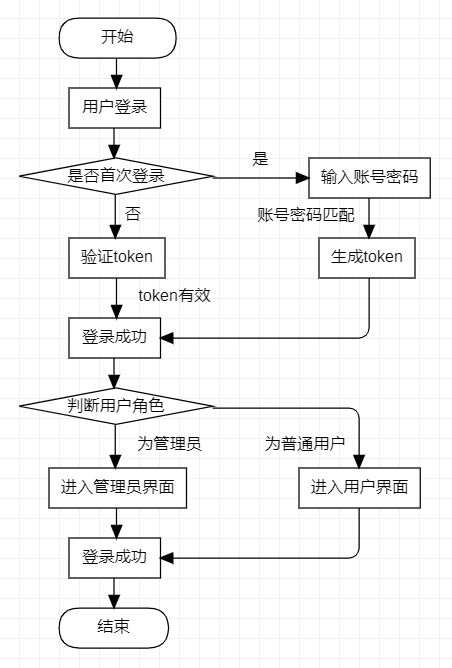
# 第四章 系统设计

## 4.1系统功能模块设计

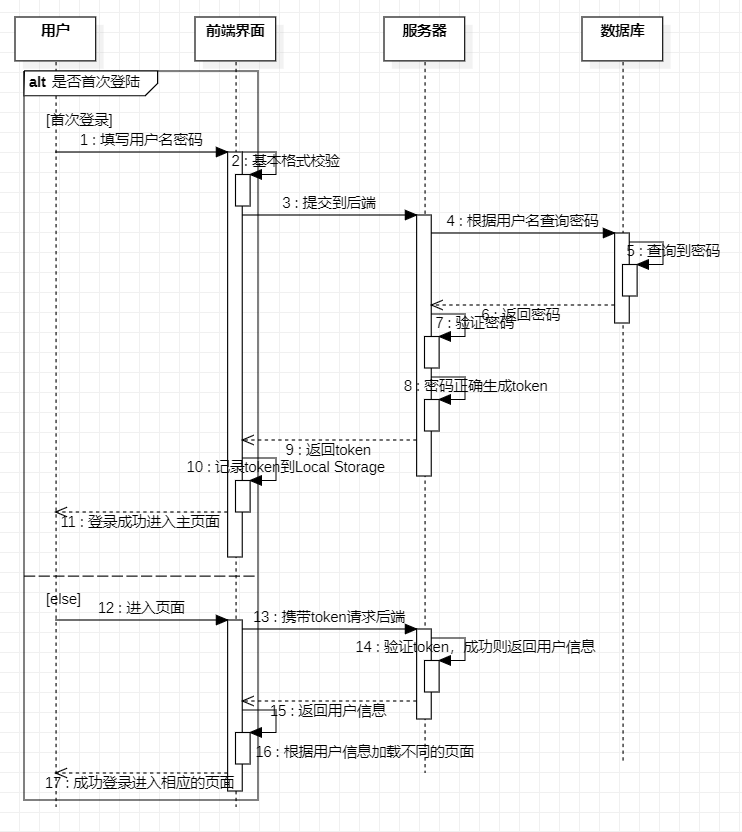
在上一节的需求分析阶段，已经充分了解了本系统的目的和具体任务，还通过系统用例规约了解了各个用例大致的流程。于是便可以通过绘制流程图或是时序图来进一步描绘本系统各个功能的主要执行过程，以便于我们能清晰明了的了解过程中涉及到的操作和发现其中的逻辑漏洞，使得能在编写代码时做出补救措施。以下便是部分具有代表性的模块的说明与时序图。

### 4.1.1 用户登录模块

用户登录向来是各个系统的重中之重，因为其涉及到某些功能的是否开放、身份的验证和权限的授予等。在本系统中，用户也会拥有两种身份：普通用户和管理员，不同的身份拥有不同的权限，登陆后也将展示不同的界面，系统也将提供不同的功能。用户登录模块的流程图如图4-1所示，时序图如图4-2。



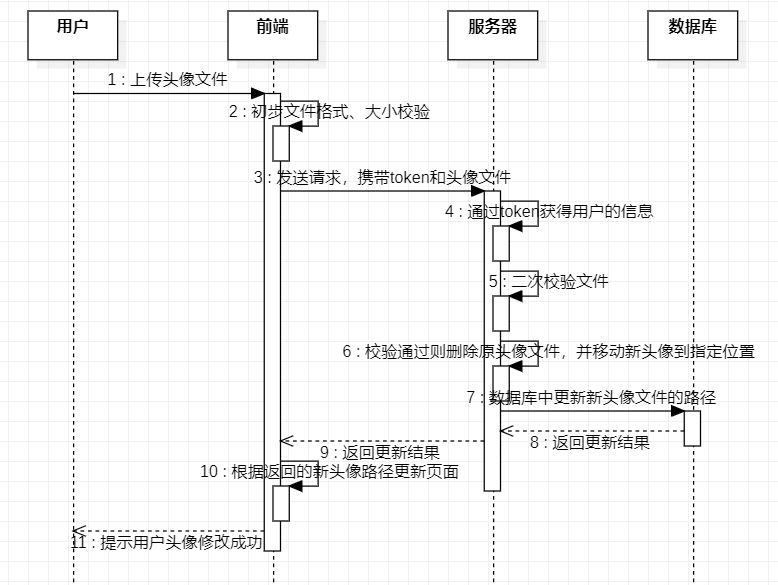
**图4-1 用户登录模块流程图**



**图4-2 用户登录模块时序图**

### 4.1.2 用户修改头像模块

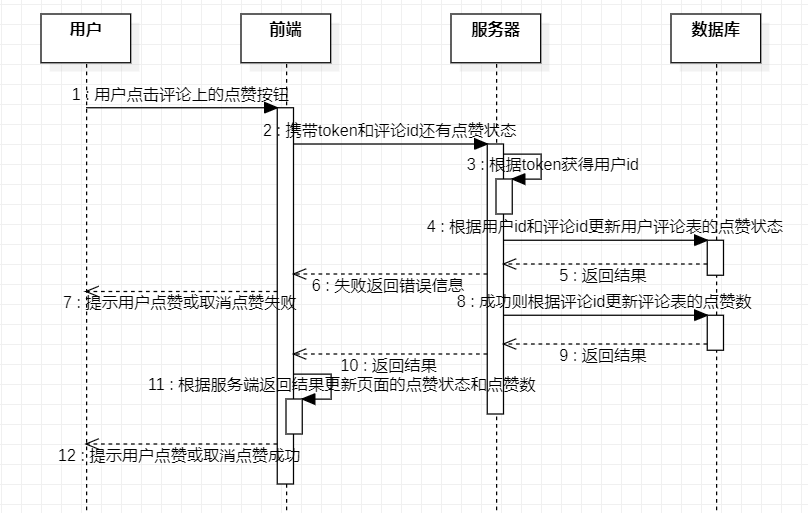
用户在初次注册后会有默认的头像，当用户想要更改自己的头像时，需要调用此模块的接口完成头像的修改。用户修改头像模块时序图如图4-3所示。



**图4-3 用户修改头像模块时序图**

### 4.1.3 用户点赞评论模块

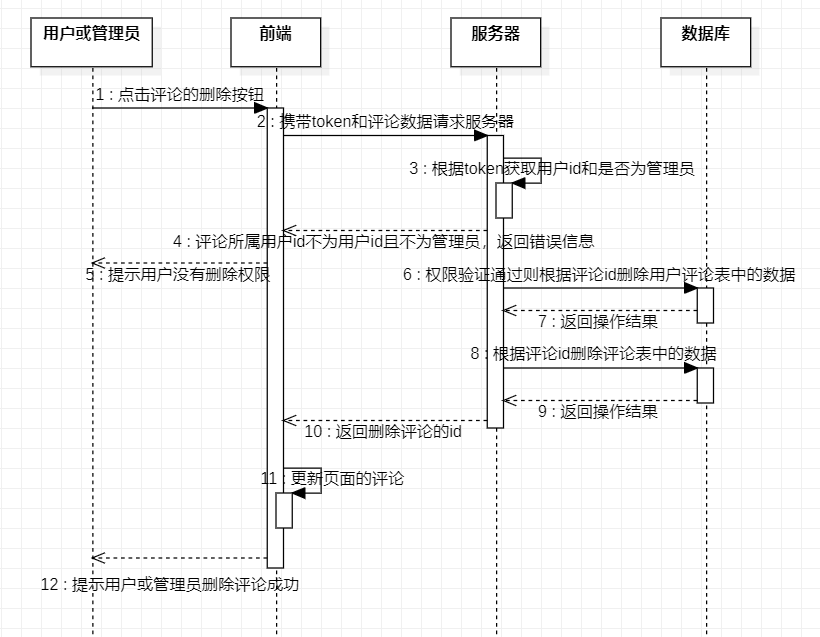
用户须进入到某个歌曲，点击评论时会向后端发送请求，得到这个歌曲id下所有的评论，并显示评论的点赞数和用户针对每个评论的点赞状态，用户对每个评论仅能点赞一次，通过数据库的表进行存储点赞状态。用户点赞评论模块时序图如图4-4所示。



**图4-4 用户点赞评论模块时序图**

### 4.1.4 用户或管理员删除评论模块

用户发现自己的评论不合适、或是管理员发现有违规评论后，都能对歌曲下的评论进行删除操作，但用户只能删除自己的评论，而管理员能删除该歌曲下任意一条评论。虽然会我在普通用户登录状态时，仅在用户自己的评论后生成删除按钮，但在服务端，依然要进行权限的判断。而且因为有着外键约束，用户评论表的comment\_id引用了评论表的主键，所以必须先删除用户评论表中的comment\_id为待删除评论id的数据，方能进行评论表数据的删除。当然也可以使用级联模式，删除评论表数据的时候，自动删除引用了主键作为外键的数据。用户或管理员删除评论模块时序图如图4-5所示。



**图4-5 用户或管理员删除评论模块时序图**

## 4.2系统数据库设计

数据库是按照数据结构来组织、存储和管理数据的仓库。在经过上一节的需求分析和系统功能模块设计后，就已经可以开始着手数据库的设计。数据库中有9张表，分别是：用户表users、歌手表singers、歌曲表songs、评论表comments、收藏歌曲表love\_songs、历史记录表history\_songs、评论点赞状态表user\_comments、歌单表songkus、歌单歌曲表songku\_songs。以下是各个表的详情表。

### 4.2.1 用户表

用户实体包含用户编号、用户名、密码、头像路径、是否为管理员等5个字段名。其详情表如表4-6所示。

**表4-6 用户表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| user\_name | varchar | 用户名 | 不能为空，唯一 |
| password | varchar | 密码 | 不能为空 |
| avatar\_path | varchar | 头像文件名 | 不能为空 |
| is\_admin | tinyint | 是否为管理员 | 不能为空 |

### 4.2.2 歌手表

歌手实体包含歌手编号、歌手名、歌手生日、性别、歌手描述等5个字段名。其详情表如表4-7所示。

**表4-7 歌手表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| singer\_name | varchar | 歌手名 | 不能为空，唯一 |
| birthday | date | 歌手生日 | 可以为空 |
| gender | tinyint | 性别 | 可以为空 |
| description | varchar | 歌手简述 | 可以为空 |

### 4.2.3 歌曲表

歌曲实体包含歌曲编号、所属歌手id、歌曲名、发布时间、歌曲文件名、歌词、访客量等7个字段名。其详情表如表4-8所示。

**表4-8 歌曲表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| song\_name | varchar | 歌曲名 | 不能为空 |
| singer\_id | int | 所属歌手编号 | 外键，不能为空 |
| publish\_time | datetime | 发布时间 | 不能为空 |
| file\_name | varchar | 歌曲文件名 | 不能为空 |
| visitors | int | 访客量 | 不能为空 |
| lyric | text | 歌词 | 不能为空 |

### 4.2.4 评论表

评论实体包含评论编号、所属歌曲id、所属用户id、评论内容、获赞数、发布时间等6个字段名。其详情表如表4-9 所示。

**表4-9 评论表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| song\_id | int | 所属歌曲编号 | 外键，不能为空 |
| user\_id | int | 所属用户编号 | 外键，不能为空 |
| content | text | 评论内容 | 不能为空 |
| publish\_time | datetime | 发布时间 | 不能为空 |
| favour | int | 获赞数 | 不能为空 |

### 4.2.5 收藏歌曲表

收藏歌曲实体包含收藏歌曲编号、歌曲id、所属用户id、听歌次数等4个字段名。其详情表如表4-10所示。

**表4-10 收藏歌曲表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| song\_id | int | 歌曲编号 | 外键，不能为空 |
| user\_id | int | 所属用户编号 | 外键，不能为空 |
| listen\_num | int | 听歌次数 | 不能为空 |

### 4.2.6 历史记录表

历史记录实体包含历史记录编号、歌曲id、所属用户id、最后听歌时间等4个字段名。其详情表如表4-11所示。

**表4-11 历史记录表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| song\_id | int | 歌曲编号 | 外键，不能为空 |
| user\_id | int | 所属用户编号 | 外键，不能为空 |
| listen\_time | datetime | 最后听歌时间 | 不能为空 |

### 4.2.7 评论点赞状态表

评论点赞状态实体包含评论点赞状态编号、所属评论id、所属用户id、点赞状态等4个字段名。其详情表如表4-12 所示。

**表4-12 评论点赞状态表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| comment\_id | int | 所属评论编号 | 外键，不能为空 |
| user\_id | int | 所属用户编号 | 外键，不能为空 |
| has\_zan | tinyint | 点赞状态 | 不能为空 |

### 4.2.8 歌单表

歌单表包含歌单编号、歌单名、歌单封面对应的文件名、歌单简介等4个字段名。其详情表如表4-13所示。

**表4-13 歌单表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| ku\_name | varchar | 歌单名 | 不能为空 |
| file\_path | varchar | 歌单图片文件名 | 不能为空 |
| description | varchar | 歌单简介 | 可以为空 |

### 4.2.9 歌单歌曲表

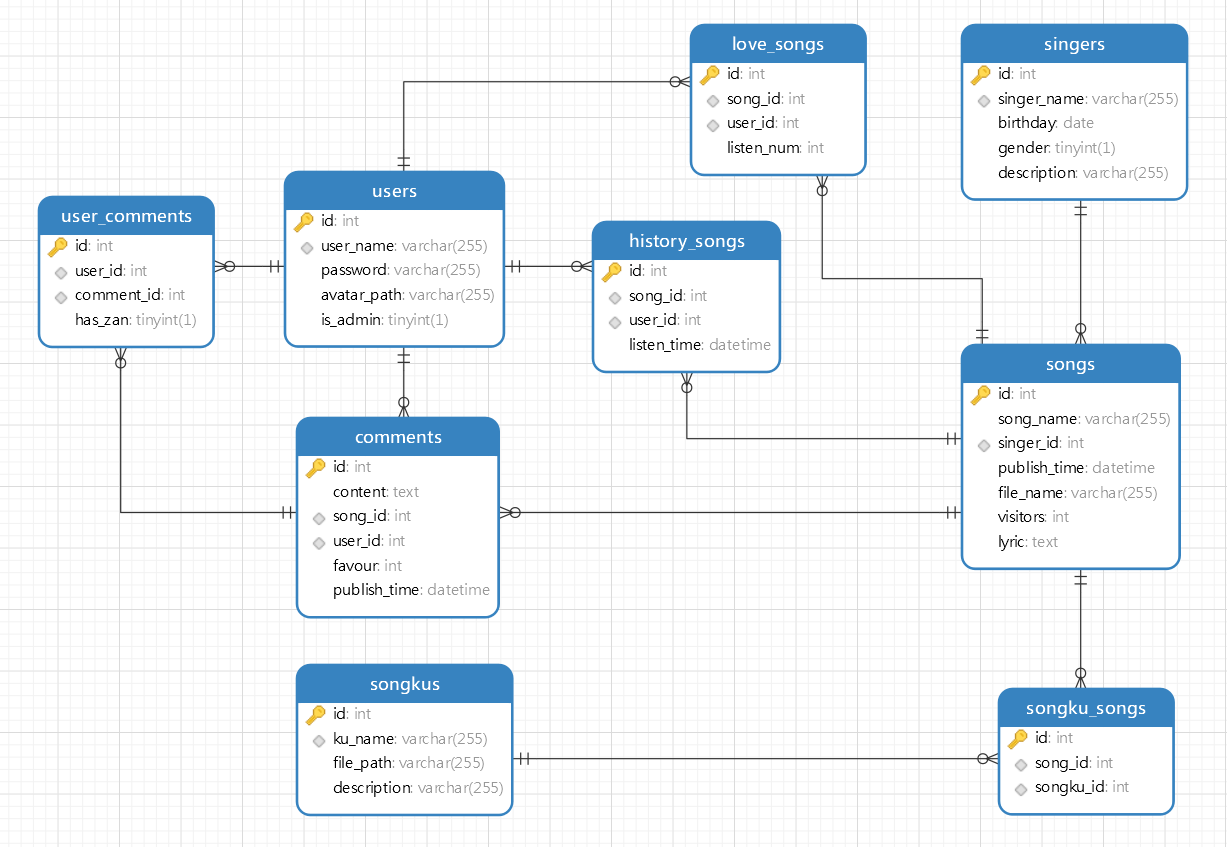
歌单歌曲表包含编号、歌单id、歌曲id等3个字段名。其详情表如表4-14所示。

**表4-14 歌单歌曲表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| id | int | 唯一标识符 | 主键，不能为空 |
| song\_id | int | 歌曲id | 外键，不能为空 |
| songku\_id | int | 歌单id | 外键，不能为空 |

### 4.2.10 数据库实体关系图

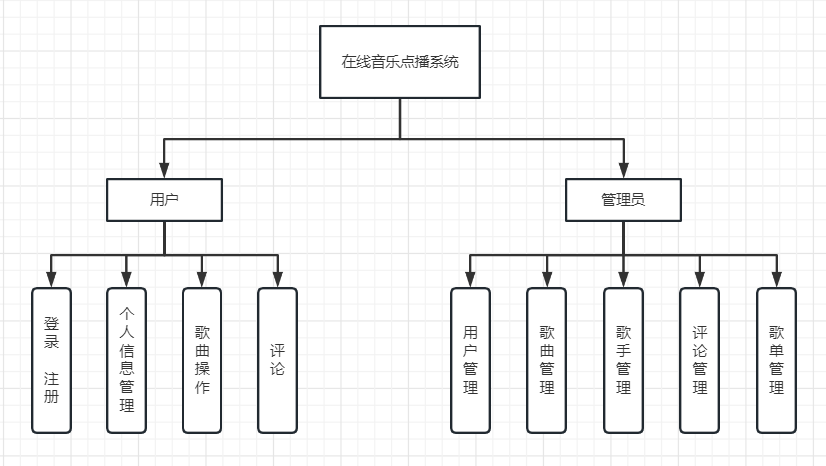
本系统所有实体的关系如图4-15所示，数据库是由Sequelize的模型生成，图是由Navicat导入模型后自动绘制。



**图4-15 数据库实体关系图**

# 第五章 系统实现

在线音乐点播系统包含了两种使用角色。其中，用户包含了登录注册模块、个人信息管理模块、歌曲操作模块和评论模块。管理员包含了用户管理模块、歌曲管理模块、歌手管理模块、歌单管理模块和评论管理模块。我将选取几个具有代表性的模块进行解释说明。在线音乐点播系统的功能模块图如图5-1所示。



**图5-1 系统功能模块图**

## 5.1登录模块

在基本任何一种系统中，为了给各种功能提供支持，账号已经成为了不可或缺的功能，渺小如搜索历史记录，都需要与用户的账户进行关联，所以说登录模块仍然是系统中举足轻重的模块。

在本系统的登录模块中，在已经注册了账号的情况下，若是首次登录，用户需要填写用户名和密码，将填写的数据传递到后端后，登录模块查询用户是否存在并验证密码的正确性。若正确，登录模块使用JsonWebToken库根据用户的一些信息生成token并将其返回到前端页面中，前端将token保存并在下次进入页面后携带token进行自动登录，以免用户进行重复的登录操作。若错误，则提示用户密码不正确，请重新输入。相应代码如表5-2所示。

**表5-2 生成token与自动登录的代码**

|  |
| --- |
| async login(ctx) {  // 从上个中间件取数据, 所有基本不变的字段均被当成payload(载荷)  const { id, user\_name, is\_admin, avatar\_path } = ctx.state.userInfo  ctx.body = {  code: '0',  message: '登录成功',  // 通过私钥（自定义）生成token, 过期时间一天  result: {  token: jwt.sign({ id, user\_name, is\_admin }, 'BINJUR', { expiresIn: '10d' }),  userInfo: { id, user\_name, is\_admin, avatar\_path }  }  }    }  // 通过token的自动登录  async autoLogin(ctx) {  const { id, user\_name, is\_admin } = ctx.state.userInfo  const { avatar\_path } = await getUserInfoByName(user\_name)  ctx.body = {  code: '0',  message: '登录成功',  result: {  userInfo: { id, user\_name, is\_admin, avatar\_path }  }  }  } |

## 5.2个人信息管理模块

在本系统中，用户若处于登录状态，其可以更改自己的密码和头像，两个功能实现的方式大同小异，这里挑出密码修改具体说明。

在修改密码界面，用户须填写自己的旧密码，填写两次新密码，前端会对密码的格式进行基础的校验，若两次密码不相同，也会提示用户重新输入。然后前端会把新密码与旧密码传递给后端，有后端再次进行密码格式的检验。然后通过前端传来的token获取用户id，查询此id用户的密码，与旧密码进行比对。比对成功则修改数据库中对应用户的密码为新密码，返回成功信息，提示用户密码修改成功。失败则返回前端错误原因，前端提示用户旧密码输入有误。注意，在比对旧密码和修改新密码前，都需要使用密码加密中间件。bcrypt是一个用于加密的库，相应代码如表5-3所示。

**表5-3 使用bcrypt进行加密的中间件**

|  |
| --- |
| const cryptPassword = async (ctx, next) => {  const { password } = ctx.request.body  const salt = bcrypt.genSaltSync(10)  const hash = bcrypt.hashSync(password, salt)  ctx.request.body.password = hash  await next()  } |

## 5.3歌曲管理模块

歌曲的管理与其他是有一些区别的，因为其数据不仅存储在数据库中，还有歌曲文件，我将以歌曲信息修改为例阐述出歌曲管理的具体步骤。

在管理员修改歌曲信息时，对携带的token不仅需要进行登录状态验证，还需要进行权限的校验。校验权限的代码如表5-4所示。当权限校验通过后，用户就可以修改歌曲的歌名、歌词、歌曲文件等属性。在本系统中，专门针对歌曲名与歌手名编写了方法用来判定合法性，如表5-5所示。如果发生了文件的修改，则需要删除原本的歌曲文件，并将新歌曲文件移动到合适的位置。如表5-6所示。最后若文件操作成功，则更新数据库中的内容，service层的数据库操作方法如表5-7所示。

**表5-4 使用中间件进行权限的校验**

|  |
| --- |
| const hasAdminPermission = async (ctx, next) => {  const { is\_admin } = ctx.state.userInfo  if(!is\_admin) {  ctx.body = {  code: '10011',  message: '没有管理员权限',  result: ''  }  return  }  await next()  } |

**表5-5 对歌手名与歌曲名的校验**

|  |
| --- |
| isSongNameValid(name = '') {  if(!name) return false  if(name.trim() !== name) return false // 头尾不能包含空格  let arr = name.match(/[^\u4e00-\u9fa5A-z\s]|\_/g) // 找到所有非汉字英文空格的字符  if(arr) return false  arr = name.match(/\s{2,}/g) // 找到所有连续两次及以上的空格  if(arr) return false  return true  }, |

**表5-6 删除旧文件并移动新文件**

|  |
| --- |
| if(songFile) {  newValue.file\_name = songFile.newFilename  oldSongPath = path.join(\_\_dirname, '../../upload/song', old\_file\_name)  newSongPath = path.join(\_\_dirname, '../../upload/song', songFile.newFilename)  fs.existsSync(oldSongPath) && fs.unlinkSync(oldSongPath)  fs.renameSync(songFile.filepath, newSongPath)  } |

**表5-7 更新歌曲信息的service层方法**

|  |
| --- |
| async updateSingerById({ id, singer\_name, birthday, gender, description }) {  let newValue = {}  const whereOpt = { id }  newValue.singer\_name = singer\_name  newValue.birthday = birthday != undefined ? birthday : null  newValue.gender = gender != undefined ? gender : null  newValue.description = description != undefined ? description : null  const res = await Singer.update(newValue, { where: whereOpt })  return !!res[0]  } |

## 5.4界面截图展示



**图5-8 登录界面**



**图5-9 注册自动登录**



**图5-10 用户管理界面**

**图5-11 歌手管理界面**

**图5-12 歌手信息修改界面**



**图5-13 个人中心界面**

# 第六章 系统测试

## 6.1引言

系统测试是项目开发的最后一个环节，项目的上线需要进行有规划的测试，构造测试用例对系统进行测试，去发现项目中的bug并交由开发者去修改。

黑盒测试[15] 是我们平常所说的“功能测试”，它主要测试的是产品功能是否可以实现，不在乎产品内部的结构，只针对产品的功能做出检测评价。测试过程中，通常会将程序看作一个不能打开的黑盒子，在测试人员无法从外面看见产品内部结构的情况下，对软件界面和软件功能进行测试。在本次测试中，我将使用黑盒测试的方法完成。

## 6.2测试环境

本次测试在本地环境中测试，具体环境如表6-1所示。

**表6-1 系统测试环境表**

|  |  |  |
| --- | --- | --- |
| 编号 | 名称 | 环境及版本 |
| 1 | 操作系统 | Windows10 家庭版 |
| 2 | 内存 | 8GB |
| 3 | 数据库 | Mysql8.0.32 |
| 4 | 浏览器 | Chrome |

## 6.3测试结果

这一小节是对本系统中各个模块的测试，以及测试结果的记录，并汇总为图表，管理员的模块具有很高的重合度，于是便只选取两大模块罗列。

### 6.3.1 登录注册模块测试表

**表6-2 登录注册模块测试表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 测试项目 | 测试方法 | 预期效果 | 结果 |
| 1-1 | 注册 | 输入合法的用户名和密码 | 提示用户注册成功 | 通过 |
| 1-2 | 重复注册 | 输入已经注册的用户名 | 提示用户名已被注册 | 通过 |
| 1-3 | 用户名为空 | 不填写用户名点击注册 | 按钮无法点击 | 通过 |
| 1-4 | 密码为空 | 不填写密码 | 按钮无法点击 | 通过 |
| 1-5 | 用户名或密码输入空格 | 往用户名或密码框输入空格 | 无法输入 | 通过 |
| 1-6 | 用户名过长 | 往用户名框一直输入 | 输入到7位后不能再输入 | 通过 |
| 1-7 | 密码过长 | 往密码框一直输入 | 输入到12位后不能再输入 | 通过 |
| 1-8 | 登录 | 输入正确用户名和密码 | 登录成功 | 通过 |
| 1-9 | 自动登录 | 登陆后刷新页面 | 登录成功 | 通过 |
| 1-10 | 用户名不存在 | 输入不存在的用户名 | 提示用户名不存在 | 通过 |
| 1-11 | 密码错误 | 输入错误的密码 | 提示密码错误 | 通过 |
| 1-12 | token过期 | 后端设置短的过期时间然后刷新页面 | 提示token已失效 | 通过 |
| 1-13 | token无效 | 更改LocalStorage中的token | 提示无效的token | 通过 |
| 1-14 | 无token | 删除LocalStorage中的token，然后刷新页面 | 页面变为未登录状态 | 通过 |

### 6.3.2 个人信息管理模块测试表

**表6-3 个人信息管理模块测试表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 测试项目 | 测试方法 | 预期效果 | 结果 |
| 2-1 | 修改密码 | 输入原密码，并两次输入新密码，点击修改密码 | 提示用户修改密码成功 | 通过 |
| 2-2 | 修改密码两次密码不同 | 输入新密码时，两次输入不同的密码 | 提示用户两次输入的密码不同 | 通过 |
| 2-3 | 修改密码原密码错误 | 输入错误的原密码 | 提示用户原密码错误 | 通过 |
| 2-4 | 更改头像 | 选择头像文件。点击修改 | 提示用户修改头像成功 | 通过 |
| 2-5 | 更改头像头像格式错误 | 选择非JPG/PNG文件，点击修改 | 提示文件格式错误 | 通过 |
| 2-6 | 更改头像选择过大文件 | 选择一张10M以上的图片文件，点击修改 | 提示文件体积过大 | 通过 |
| 2-7 | 不选择头像图片 | 不选择头像图片 | 修改头像按钮不能点击 | 通过 |

### 6.3.3 播放模块测试表

**表6-4 播放模块测试表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 测试项目 | 测试方法 | 预期效果 | 结果 |
| 3-1 | 播放 | 选择歌曲，双击歌曲或点击歌曲上的播放按钮 | 歌曲正常播放，被播放的歌曲有选中效果 | 通过 |
| 3-2 | 暂停 | 点击暂停按钮 | 歌曲暂停 | 通过 |
| 3-3 | 播放另一首歌 | 选择其他歌曲播放，点击播放按钮 | 原歌曲切换为另一首歌开始播放 | 通过 |
| 3-4 | 拖动进度条 | 点击正在播放歌曲的进度条任意位置 | 歌曲跳转到该位置继续播放 | 通过 |
| 3-5 | 查看歌词 | 点击正在播放歌曲的详情页面 | 出现歌词预览，如果为没有歌词，则显示“纯音乐请欣赏” | 通过 |
| 3-6 | 下一曲 | 点击播放界面的下一曲按钮 | 切换到播放列表的下一曲播放，若没有下一曲，则无事发生 | 通过 |
| 3-7 | 收藏歌曲 | 点击歌曲的收藏按钮 | 提示用户收藏成功，可在“我喜欢”中查看到该歌曲 | 通过 |
| 3-8 | 查看历史记录 | 登录状态点击历史记录按钮 | 显示用户播放过的歌曲，按时间排序 | 通过 |

### 6.3.4 评论模块测试表

**表6-5 评论模块测试表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 测试项目 | 测试方法 | 预期效果 | 结果 |
| 4-1 | 评论 | 点击进入一首歌的评论模块，输入一条评论，点击发送 | 评论发布成功 | 通过 |
| 4-2 | 删除评论 | 点击刚刚发布的评论后的删除按钮 | 评论删除成功 | 通过 |
| 4-3 | 点赞 | 点击一条发布的评论后的点赞按钮 | 评论的点赞按钮样式变化，且数字加一 | 通过 |
| 4-4 | 点赞的状态 | 刷新刚刚点赞的页面，查看刚刚评论的点赞状态 | 评论获赞数和用户的点赞状态依然一样 | 通过 |
| 4-5 | 取消点赞 | 点击一条已经点赞过的评论后的点赞按钮 | 评论的点赞按钮样式变化，且数字减一 | 通过 |

### 6.3.5 用户管理模块测试表

**表6-6 用户管理模块测试表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 测试项目 | 测试方法 | 预期效果 | 结果 |
| 5-1 | 删除用户 | 管理员登录，点击用户后的删除按钮 | 提示删除用户成功，与用户相关的数据库中的内容均被删除 | 通过 |
| 5-2 | 重置密码 | 管理员登录，点击用户后的重置密码按钮 | 提示重置密码成功 | 通过 |

### 6.3.6 歌手管理模块测试表

**表6-7 歌手管理模块测试表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6-1 | 添加歌手 | 管理员登录，点击歌手管理标签，添加歌手，输入歌手名，出生日期、性别、简介等信息 | 添加成功，数据库中相关歌手信息被创建，歌手管理标签中该歌手能正常显示 | 通过 |
| 6-2 | 添加重复歌手 | 按以上流程，但是歌手名输入已有歌手 | 提示已有该歌手 | 通过 |
| 6-3 | 编辑歌手信息 | 管理员登录，点击歌手管理标签，点击歌手后的编辑按钮，更改歌手名，出生日期、性别、简介等信息 | 提示修改歌手信息成功 | 通过 |
| 6-4 | 删除歌手 | 管理员登录，点击歌手管理标签，点击歌手后的删除按钮 | 提示删除成功，该歌手下的歌曲也被删除，歌曲下的评论、评论的用户点赞状态、该歌曲的播放历史、歌曲收藏等数据库信息均被清除 | 通过 |

# 第七章 总结与展望

## 7.1总结

随着社会的发展，人们的娱乐需求也越发增加，喜好歌曲的发烧友也不再满足于单一的听歌，也开始寻找一种情感上的共鸣，于是音乐类应用的社交属性也就慢慢受到了重视，我便计划开发一款携带一丝交流属性的音乐播放web应用。

本系统的完成经历了需求分析、系统设计、系统实现、系统测试等阶段。在需求分析阶段，我代入到用户角度，解构出了一个本系统的用户的需求，然后绘制出了用例图。在系统设计阶段，我将系统分为了几大功能模块，并展示了部分模块功能的时序图，这位数据库的设计铺平了道路。在数据库设计上，采用了关系型数据库Mysql，并构建出了用户表、歌手表、歌曲表、评论表、收藏歌曲表、历史记录表、用户点赞状态表等几张表，用于存储和管理系统中产生的数据。在系统实现阶段，我根据剥离出的模块一步一步实现了系统的功能。最后的测试阶段，我按照模块对每个功能进行了测试，发现问题并着手修改。

虽然项目已然接近尾声，但是依然会有不尽如人意的地方，还望各位老师们不吝赐教，让学生能够积极的改正错误。

## 7.2展望

本系统实现了在线音乐点播系统最基础的功能，但由于能力不足，依然有些不够完善的地方，如不能展示逐字的歌词，没有消息通知功能等，在今后的学习中，当我学习完相关知识后，会回来继续完善我的系统，争取能将其上线到服务器上，让广大互联网的朋友做出指导，留下宝贵意见。

# 参考文献

1. 林陈佑康,王风硕.基于Web的音乐播放器前端的设计与实现[J].电脑知识与技术,2021,17(36):89-91.DOI:10.14004/j.cnki.ckt.2021.3526.
2. 马杰瑞,王风硕.基于web的音乐播放器后端的设计与实现[J].电脑知识与技术,2021,17(35):72-74.DOI:10.14004/j.cnki.ckt.2021.3491.
3. 邓海文.基于web的音乐播放器的设计与实现[J].电脑知识与技术,2018,14(29):98-99.DOI:10.14004/j.cnki.ckt.2018.3403.
4. 周超.音乐APP发展探究与建议——基于对网易云音乐发展的研究[J].价值工程,2018,37(19):193-194.DOI:10.14018/j.cnki.cn13-1085/n.2018.19.077.
5. 万湛湛.数字时代我国网络音乐产业的形成与发展[J].艺术评鉴,2019,No.594(19):55-56.
6. 李丹丹. 个性化音乐推荐系统的设计和研究[D].武汉纺织大学,2018.
7. 刘玉红,李健.网络游戏中的社交属性分析与设计研究[J].包装与设计,2022,No.235(06):134-135.
8. 张贵强,王美玲.基于NodeJS的企业网站的设计与实现[J].信息技术与信息化,2019,No.237(12):58-60.
9. 白爱.在线音乐管理系统设计与实现[D].华中科技大学,2019.DOI:10.27157/d.cnki.ghzku.2019.003843.
10. 张浩洋,顾丹鹏,陈肖勇.基于Vue的数据管理平台实践与应用[J].计算机时代,2022,No.361(07):66-67+72.DOI:10.16644/j.cnki.cn33-1094/tp.2022.07.017.
11. 陈月燕,邹庆,边雁等.基于Koa2的可视化页面编辑系统服务端设计[J].电子技术与软件工程,2021,No.214(20):198-200.
12. 陈兰. 基于SaaS的信息服务系统的设计与实现[D].南京大学,2020.DOI:10.27235/d.cnki.gnjiu.2020.002929.
13. 赵军.基于Vue.js的疫情统计图表实现[J].电脑编程技巧与维护,2020,No.417(03):144-147.DOI:10.16184/j.cnki.comprg.2020.03.053.
14. 石庆冬.版本管理工具Git的主要特点[J].电子技术与软件工程,2022,No.225(07):72-75.
15. 妥泽花.基于黑盒测试与白盒测试的比较探究[J].电子世界,2021,No.617(11):55-56.DOI:10.19353/j.cnki.dzsj.2021.11.027.

# 致谢

转眼间四年匆匆就从指尖溜走，大学生活也迎来了谢幕。还记得刚进校园时的青涩与不安，也怀抱着对未来的憧憬，细细打量着周围的一草一木，遇见新的舍友，新的同学，新的老师和辅导员，遇见你们三生有幸。

这四年间，也有过懈怠与自我怀疑，但我的舍友始终有一种迎难而上的气魄，在你的督促和鼓励下，我缓慢的前进，好像不论什么问题，你都能不紧不慢的处理妥当，再次感谢你这四年来的关照。

还有万辅导员和熊辅导员，一直默默关心我这个山村里的孩子，在我外出回家时，关心我的去向和生命安全。让我能够健康的度过这四年。感谢你们。

最后是我的导师刘海，按时提醒我完成毕业设计，给我的项目提供了很多的建议和帮助，感谢您。