

南京信息工程大学

滨江学院 本科生毕业论文(设计)



题 目 基于 Django 及 PyQt5 的跨平台音乐管理系统

学生姓名 梁雄

学 号 20202383015

学 院 物联网工程学院

专 业 软件工程

指导教师 李振宏

二〇二四年四月三十日

声 明

本人郑重声明：

- 1、以“求实、创新”的科学精神从事科学研究工作。
- 2、本论文中除引文外，所有测试、数据和相关材料均为真实有效的。
- 3、本论文是我个人在指导教师的指导下进行的研究工作和取得的研究成果，请勿用于非法用途。
- 4、本论文中除引文和致谢的内容外，并未抄袭其他人或其他机构已经发表或撰写过的研究成果。
- 5、关于其他同志对本研究所做的贡献均已在论文中作了声明并表示了谢意。

作者签名： 梁桂

日期：2024.5.10

目 录

1. 绪论	1
1.1 系统开发背景	1
1.2 系统开发意义	1
1.2.1 音乐管理系统的现状	1
1.2.2 解决问题的迫切需求	1
1.2.3 本系统的开发意义	2
2. 可行性分析与需求分析	2
2.1 可行性分析	2
2.1.1 技术可行性	2
2.1.2 经济可行性	4
2.1.3 操作可行性	4
2.2 需求分析	4
2.2.1 业务需求分析	4
2.2.2 系统功能分析	5
3. 系统总体设计	6
3.1 系统设计目标	6
3.2 系统设计原则	6
3.2.1 简洁直观	6
3.2.2 易用性	6
3.2.3 高度可扩展性	6
3.2.4 可维护性	7
3.2.5 跨平台兼容性	7
3.3 系统架构设计	7
3.3.1 用户认证及权限模块	7
3.3.2 歌手歌曲管理模块	8
3.3.3 用户视频管理模块	8
3.3.4 用户动态管理模块	9
3.3.5 用户消息管理模块	9
3.3.6 用户评论管理模块	10
3.3.7 用户日志管理模块	10
3.3.8 管理员日志管理模块	11
3.3.9 标签管理模块	11
3.3.10 图片管理模块	12
3.3.11 数据统计分析模块	12
3.3.12 歌曲推荐模块	13
3.3.13 排行榜模块	13
3.3.14 搜索模块	13
3.3.15 歌曲播放模块	13
3.3.16 视频播放模块	13

4. 系统数据库设计	14
4.1 实体类关系设计	14
4.1.1 用户认证及权限模块实体关系	14
4.1.2 歌手歌曲管理模块实体关系	15
4.1.3 用户视频管理模块实体关系	15
4.1.4 用户动态管理模块实体关系	15
4.1.5 用户消息管理模块实体关系	15
4.1.6 用户评论管理模块实体关系	16
4.1.7 用户日志管理模块实体关系	16
4.1.8 管理员日志管理模块实体关系	16
4.1.9 标签管理模块实体关系	17
4.1.10 图片管理模块实体关系	17
4.2 数据库表设计	17
4.2.1 认证用户表	17
4.2.2 认证用户组表	18
4.2.3 权限表	18
4.2.4 内容类型表	19
4.2.5 管理日志表	19
4.2.6 会话表	19
4.2.7 用户表	19
4.2.8 地区表	20
4.2.9 歌手表	20
4.2.10 专辑表	20
4.2.11 歌曲表	21
4.2.12 歌单表	21
4.2.13 视频表	22
4.2.14 用户动态表	22
4.2.15 消息表	23
4.2.16 评论表	23
4.2.17 用户日志表	24
4.2.18 标签表	25
4.2.19 图片表	26
5. 详细设计及系统实现	26
5.1 用户认证及权限模块	26
5.1.1 用户认证模块	26
5.1.2 用户操作模块	29
5.2 歌手歌曲管理模块	32
5.2.1 歌手管理模块	32
5.2.2 专辑管理模块	35
5.2.3 歌单管理模块	38
5.3 用户视频管理模块	42
5.4 用户动态管理模块	45
5.5 用户消息管理模块	47
5.6 用户评论管理模块	49

5.7 用户日志管理模块	52
5.7.1 歌曲记录模块	52
5.7.2 视频记录模块	53
5.8 管理员日志管理模块	54
5.9 标签管理模块	54
5.10 图片管理模块	55
5.11 数据统计分析模块	56
5.12 歌曲推荐模块	56
5.13 排行榜模块	57
5.14 搜索模块	58
5.15 歌曲播放模块	59
5.16 视频播放模块	61
6. 总结与展望	62
7. 参考文献	63
8. 致谢	64

基于 Django 和 PyQt5 的跨平台音乐管理系统

梁雄

南京信息工程大学滨江学院，江苏 无锡

摘要：本论文设计并实现了一款主要基于 Django 的跨平台音乐管理系统。该系统采用 Django 框架搭建后端服务，分别使用 vue3 和 PyQt5 构建了直观友好的 web 前端及桌面端界面，旨在提供一种便捷、高效的音乐管理和播放体验。系统具备跨平台特性，可在不同操作系统上运行，并支持多种音乐文件格式。通过系统，用户可以管理本地音乐库、创建和编辑播放列表，实现对音乐文件的分类和搜索，并享受个性化的音乐推荐服务。同时，系统还考虑了用户隐私和安全性，采用了合适的身份验证和数据加密机制。实验结果表明，该音乐管理系统具有良好的性能和稳定性，为用户提供了愉悦的音乐管理和播放体验。

关键词：跨平台、音乐管理系统、前后端分离、数据加密、音乐推荐

A Cross-Platform Music Management System based on Django and PyQt5

Liang Xiong

Binjiang College of Nanjing University of Information Science and Technology, Jiangsu Wuxi

Abstract: This paper designs and implements a cross-platform music management system primarily based on Django. The system uses the Django framework to build the backend service and employs vue3 and PyQt5 to construct an intuitive and user-friendly web front end and desktop interface, aiming to provide a convenient and efficient music management and playback experience. The system features cross-platform capabilities, allowing it to run on different operating systems, and supports various music file formats. Through the system, users can manage local music libraries, create and edit playlists, categorize and search for music files, and enjoy personalized music recommendations. Additionally, the system considers user privacy and security by implementing appropriate authentication and data encryption mechanisms. Experimental results show that this music management system has good performance and stability, offering users a pleasant music management and playback experience.

Keywords: Cross-platform, Music Management System, Frontend-backend separation, Data Encryption, Music Recommendation

1. 終論

1.1 系统开发背景

随着信息技术的不断发展，数字化和智能化已经深刻地改变了人们的生活方式。传统的音乐管理方式已经无法满足用户对于便捷、个性化音乐管理的需求。用户不再仅仅满足于简单地购买和下载音乐，更期望能够通过先进的技术手段来管理和享受音乐。

在过去，音乐管理主要依赖于本地文件夹和简单的标签信息。然而，这样的管理方式存在很多问题，比如用户难以快速定位和管理大量音乐文件，无法方便地进行分类和检索。随着音乐资源的增多和多样化，用户对于更智能、高效的音乐管理系统的需求逐渐凸显。

为了解决这些问题，我们决定开发一款基于 Django、vue3、PyQt5 的跨平台音乐管理系统。通过利用 Web 和桌面平台的结合，用户可以在不同设备上统一管理音乐，无论是在电脑、平板还是手机上，都能够获得相似的音乐管理体验。这不仅为用户提供了更灵活的音乐管理方式，同时也使得音乐的管理和享受变得更加智能、便捷。

本系统的开发背景源于对现有音乐管理系统的不足的深刻认识，以及对用户需求变化的敏锐观察。通过引入先进的开发框架和技术手段，我们期望能够为用户提供一种全新的音乐管理体验，使其更好地融入数字时代的音乐文化中。

1.2 系统开发意义

在这个数字时代，音乐不仅是一种重要的娱乐和文化形式，还广泛应用于教育行业^{[1][2]}等众多领域。

本系统的开发旨在解决现有音乐管理系统的不足，实现音乐文件的分类、搜索、播放等功能，为用户提供更好的音乐管理体验。主要采用 Django、vue3、PyQt5 作为开发框架，实现跨平台运行，使用户能够在不同操作系统上使用相同的音乐管理系统。

1.2.1 音乐管理系统的现状

当前，随着数字音乐时代的到来，音乐作为一种重要的娱乐和文化形式，以及个体的审美体验，扮演着重要的角色。然而，传统的音乐管理方式逐渐显露出一系列问题。用户在面对海量音乐文件时，往往难以高效地进行整理、分类和检索。本地存储和简单的文件夹管理已经无法满足用户对于智能化、便捷化音乐管理的迫切需求。

1.2.2 解决问题的迫切需求

用户对于音乐管理系统的期望不仅仅停留在传统的收听和下载，更希望能够通过技术手

段实现更智能、高效的音乐管理。一方面，用户需要能够通过系统轻松管理庞大的音乐库，进行快速的搜索和分类；另一方面，用户渴望能够在不同设备上拥有一致的音乐体验，无论是在桌面 GUI、还是不同设备的浏览器上。

1.2.3 本系统的开发意义

基于 Django、vue3、PyQt5 的跨平台音乐管理系统的开发具有重要的实际意义。首先，系统的开发将为用户提供一种全新的音乐管理体验，通过先进的技术手段解决了传统音乐管理方式的种种不足，使得用户能够更加轻松地管理、查找和享受音乐。

其次，系统的跨平台特性意味着用户可以在不同设备上拥有一致的音乐管理体验。这种一致性不仅仅包括操作界面和功能特性，更包括用户的音乐库和个性化设置，为用户提供了更灵活、便捷的使用方式。

最后，系统的开发采用了 Django、vue3、PyQt5 等开源框架，这有助于降低系统的开发成本，提高系统的可维护性和可扩展性。这种开发方式不仅使得系统更具竞争力，也为未来的系统升级和扩展奠定了良好的基础。

总体而言，基于 Django、vue3、PyQt5 的跨平台音乐管理系统的开发不仅是对传统音乐管理方式的创新，更是对用户需求的深刻理解和积极响应，将为用户带来更为便捷、高效的音乐管理服务。

2. 可行性分析与需求分析

2.1 可行性分析

2.1.1 技术可行性

本系统使用的技术栈划分为 web 前端、桌面端、后端、以及数据库四个部分，为系统的实现提供了强大的技术支持：

- (1) Web 前端技术：以响应式框架 Vue3 为核心，ElementPlus 开发界面（事实上类似的技术组合方式也比较普遍^{[3][4][5]}）
 - **Vue3**：Vue.js 是一款用于构建用户界面的渐进式 JavaScript 框架，具有轻量级、灵活和高效的特点。Vue3 相较于之前的版本引入了更快的响应式系统和更简洁的组合 API，提升了开发效率和性能^{[6][7]}。
 - **Axios**：本系统中使用 Axios 作为前端与后端之间 http 通信的桥梁。Axios 是一款基于 Promise 的 HTTP 库，能够方便地处理请求和响应，支持拦截器、取消请求等功能，简化了前后端交互。而且，经过国内学者分析，Axios 在传输数据量不太大的情况下，其传输效率与浏览器原生的 Fetch 基本相似^[8]。
 - **Vue Router**：作为 Vue.js 官方的路由管理器，Vue Router 提供了简洁易用的路由配置，支

持嵌套路由、动态路由等特性，便于构建单页应用^[9]。

- **Pinia:** Pinia 是 Vue3 的状态管理库，简化了状态管理的流程，增强了代码的可维护性和可扩展性^[10]。
- **Element Plus:** Element UI 是基于 Vue2.0，由开发的桌面端组件库，由于它组件丰富，样式较多，风格统一，容易上手，代码稳定，且提供多种组件属性和触发事件列表，所以对于使用者使用该组件库进行再开发比较友好。Element Plus 则是由 Element UI 迁移到 Vue3 的组件库，从而使得该组件库更好地兼容 Vue 的最新版本^[11]。
- **TailwindCSS:** TailwindCSS 是一个高度可定制的 CSS 框架，支持原子化的 CSS 类名，极大地提高了样式的灵活性和开发效率^[12]。

(2) 桌面端技术

- **PyQt5:** PyQt5 是 Python 与 Qt 结合的 GUI 开发框架，提供了强大的 GUI 组件和跨平台的支持，使得开发高质量的桌面应用变得更加简便和高效^{[13][14]}。
- **PyInstaller:** PyInstaller 用于将 Python 应用打包成独立的可执行文件，简化了部署和分发，使得桌面应用能够在各种操作系统上运行^[15]。

(3) 后端技术

- **Django:** Django 是一个高层次的 Python Web 框架，以“开发迅速、设计优雅”著称。Django 提供了强大的 ORM、自动化管理后台、模板系统和表单处理功能，极大地提高了开发效率和代码可维护性^{[16][17]}。
- **Django Rest Framework(DRF):** DRF 是 Django 的一个强大的扩展，用于构建 Web APIs。它支持丰富的 API 功能，包括认证、序列化、视图和路由，简化了 RESTful API 的开发^[18]。

(4) 数据库技术

- **SQLite3:** SQLite3 是一款轻量级的关系型数据库，具有零配置、易嵌入和高效的特点，适用于中小型应用的数据持久化需求。它的单文件存储方式方便了数据库的管理和迁移^{[19][20]}。

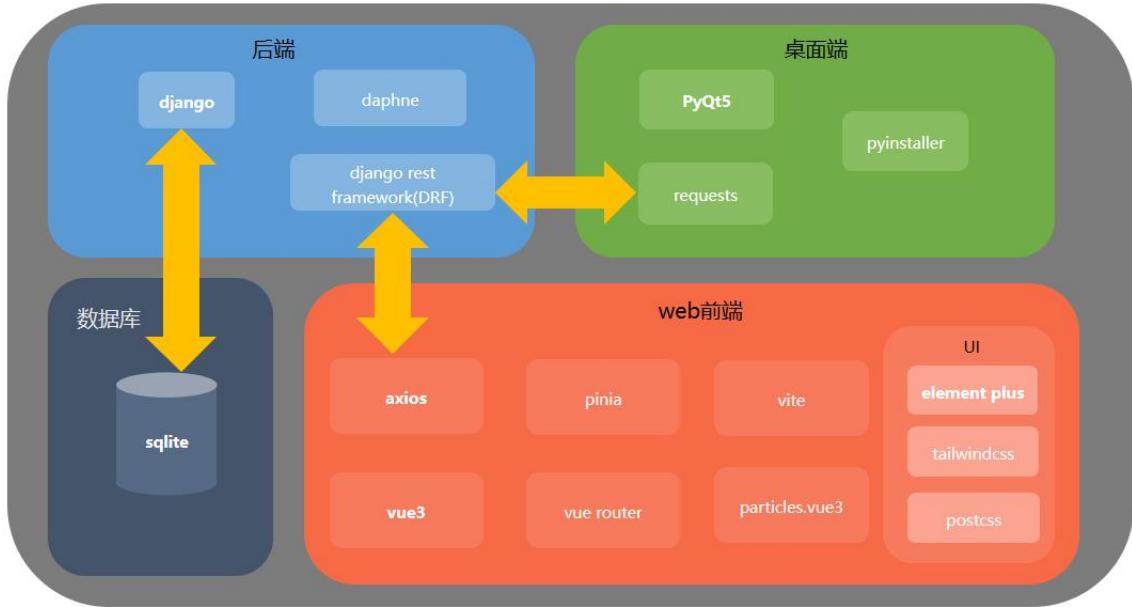


图 2-1 技术栈

经过调研不难发现，这种前后端分离的思想是很常见的^[21]。如图 2-1 所示，本系统虽划分为四个部分，但彼此之间仍然通过某种方式直接或间接地进行通信。后端是整个系统的核心，其中 Django 实现的接口能很方便的为开发这对数据库进行交互，Django Rest Framework 提供的接口则分别负责处理桌面端的 requests 模块和 web 前端的 axios 模块所发出的 http 请求，并返回响应。

2.1.2 经济可行性

相较于其他音乐管理系统，采用开源框架可以降低开发成本，使系统更具竞争力。

2.1.3 操作可行性

采用 Django 和 PyQt5 框架，系统将具备良好的用户操作界面和易用性，保证用户能够轻松上手，提高系统的操作可行性。

2.2 需求分析

2.2.1 业务需求分析

在音乐产业不断发展的今天，传统的音乐管理方式已经显得不够高效和先进。为了更好地满足用户对音乐资源的管理和体验需求，我们提出开发一款基于 Django 和 PyQt5 的跨平台音乐管理系统。该系统将实现数字化、网络化、现代化的管理方式，为用户提供更便捷的音乐资源管理体验。

系统的主要业务需求如下：

- (1) 用户需求：用户可以注册账户、登录、修改个人信息、搜索音乐、播放音乐、收藏音乐、评论音乐、查看排行榜等。

- (2) 管理员需求：管理员需要能够管理用户、音乐、评论等信息，例如删除不当的评论或音乐。
- (3) 推荐算法需求：系统需要使用基于用户的协同过滤算法，通过分析用户历史行为和兴趣爱好，为用户推荐相似的音乐。
- (4) 数据存储需求：系统需要能够存储大量的音乐和用户数据，因此需要使用高效的分布式数据库，如 MySQL。
- (5) 性能需求：系统需要具备高性能，包括快速响应用户请求、快速加载音乐和图片等。
- (6) 用户界面需求：系统需要具备良好的用户界面，包括直观易用的界面、美观的设计、可自定义的主题等。
- (7) 安全性需求：系统需要保证用户数据和隐私的安全，包括加密用户密码、安全的数据存储、防止恶意攻击等。

2.2.2 系统功能分析

根据音乐管理系统的业务所涉及到的人员，将用户划分为四类：

- (1) 访客：系统的使用者作为访客对本系统进行访问，此时系统默认该使用者为访客，若不及时进行身份验证，那么系统会限制此人的操作权限，如：只能注册、登录、播放免费音乐、查询歌曲等操作。
- (2) 普通用户：经过注册和身份认证但只具有最低权限的操作者，称为普通用户，这类使用者被允许执行一些基本操作，如：管理用户信息、管理歌单、上传音乐、上传视频、发布评论等。
- (3) 歌手：经过注册和官方认证的操作者，成为会员，这类使用者被允许更高权限的操作，如：发布专辑等。
- (4) 管理员：经过身份认证而操作权限最高的操作者，称为管理员，这类使用者被允许最高权限的操作。

根据本系统的功能，可以类总如下几个管理模块：

- (1) 用户管理模块：用于注册、登录、管理用户信息、密码重置等功能。
- (2) 歌曲管理模块：用于添加、编辑、删除歌曲信息，包括歌曲名称、歌手、专辑、歌词等。
- (3) 播放列表管理模块：允许用户创建、编辑和删除播放列表，将歌曲添加到播放列表中。
- (4) 搜索模块：提供用户通过关键词搜索歌曲、歌手或专辑的功能。
- (5) 推荐模块：基于用户的听歌历史、喜好等信息，提供个性化的音乐推荐。
- (6) 播放控制模块：包括播放、暂停、快进、音量控制等功能。
- (7) 社交分享模块：允许用户分享他们喜欢的歌曲、播放列表等信息到社交媒体。
- (8) 统计与分析模块：用于分析用户行为、歌曲热度等数据，以改进系统和提供更好的服务。
- (9) 权限管理模块：管理不同用户角色的权限，确保合适的用户可以执行相应的操作。

3. 系统总体设计

3.1 系统设计目标

本系统旨在实现跨平台的音乐管理服务，为用户提供便捷、高效的音乐管理体验。

表 3-1 系统设计目标

设计目标	具体细则
高效性	实现音乐资源的快速检索和播放，确保系统运行流畅，减少用户等待时间。
用户友好性	通过直观的界面设计和简单的操作流程，确保用户能够轻松上手，提升用户体验。
跨平台兼容性	利用 Django 和 PyQt5 的跨平台特性，使系统在不同操作系统和设备上都能够无缝运行，为用户提供一致性的使用体验。
全面性	提供包括音乐资源管理、播放列表定制、用户个性化账户管理、高效音乐播放控制、数据统计与分析、系统个性化设置、社交分享等多项功能，满足用户在音乐管理方面的多元化需求。
可扩展性	采用分层设计和模块化结构，使系统具备良好的可扩展性，方便后续引入新功能和模块。
可维护性	遵循简洁直观、易用性、高度可扩展性的设计原则，以确保系统的长期可维护性。

3.2 系统设计原则

3.2.1 简洁直观

系统设计遵循简洁直观的原则，保持界面和功能的简单性，以降低用户的认知负担。清晰而直观的设计有助于用户迅速理解系统的操作流程，提高用户的使用效率。

3.2.2 易用性

系统设计注重易用性，确保用户可以轻松上手，并在系统中执行各项操作。良好的用户体验是系统设计的关键，通过合理的交互设计和用户界面布局，使用户在系统中找到所需功能，并以直观的方式进行操作。

3.2.3 高度可扩展性

为了应对未来可能的功能扩展和业务变化，系统采用高度可扩展的设计。模块化的系统结构和清晰的接口定义使得引入新功能或进行系统升级变得更加简便，同时减少了对系统其他部分的影响。

3.2.4 可维护性

系统设计考虑可维护性，确保系统的稳定性和可维护性。采用模块化、分层的设计结构，便于开发人员定位和解决问题，同时也使系统更易于维护和升级。

3.2.5 跨平台兼容性

考虑到用户可能使用不同的操作系统和设备，系统在设计上注重跨平台兼容性。通过 Django 和 PyQt5 的跨平台特性，确保用户在各种环境下都能够无缝地使用系统，提高系统的可访问性。

3.3 系统架构设计

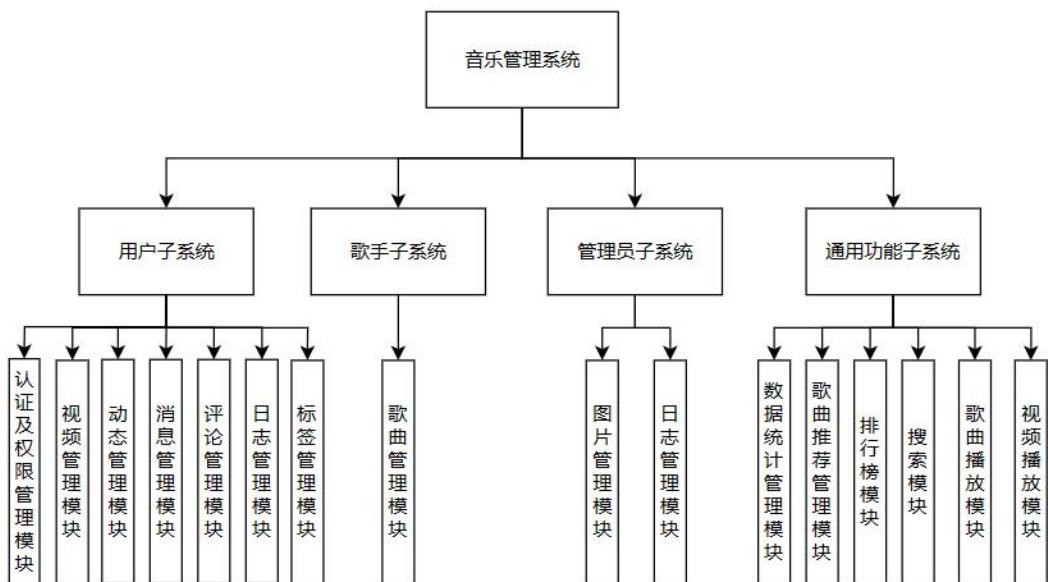


图 3-1 音乐管理系统总架构

3.3.1 用户认证及权限模块

用户认证及权限管理模块分为认证管理模块和操作权限模块：

- (1) 对于所有访客：均可查看公共信息，有权进行账号的注册、登录、注销。
- (2) 对于普通用户：均可查看公共信息、编辑个人信息、修改密码、增删改查操作权限范围内的数据。
- (3) 对于管理员用户：均可增删改查系统中大部分数据。

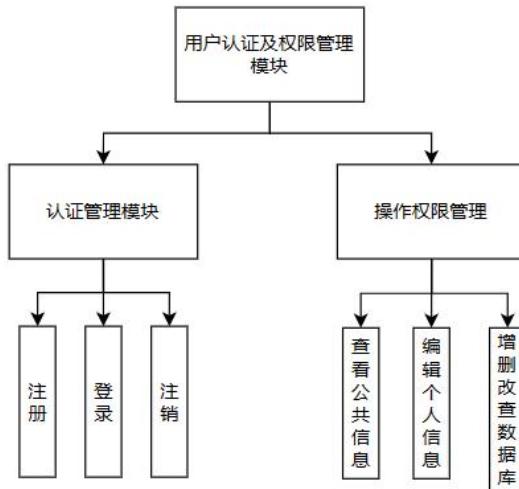


图 3-2 用户认证及权限模块架构图

3.3.2 歌手歌曲管理模块

歌手歌曲管理模块分为歌手管理模块、专辑管理模块、歌单管理模块等子模块。

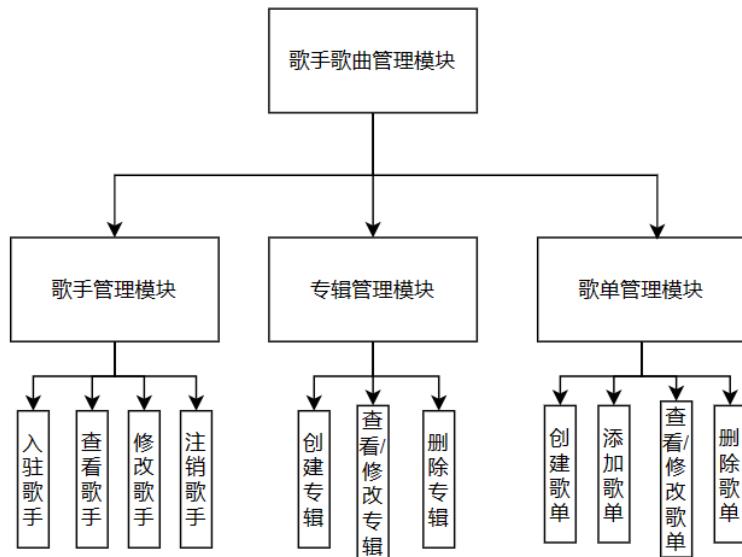


图 3-3 歌手歌曲模块架构图

3.3.3 用户视频管理模块

用户视频管理模块分为视频上传模块、视频信息管理模块等子模块。

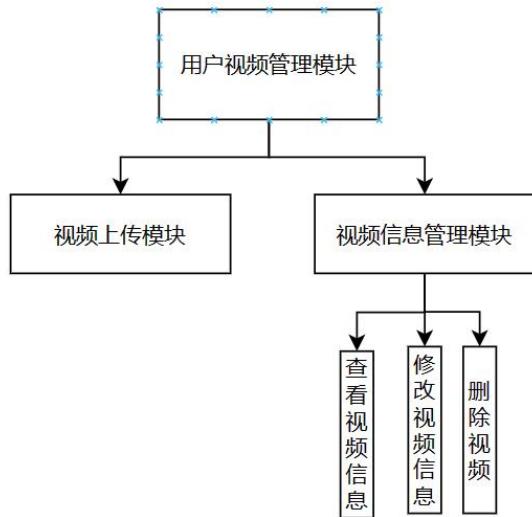


图 3-4 用户视频模块架构图

3.3.4 用户动态管理模块

用户动态管理模块分为动态上传模块、动态信息管理模块等子模块。

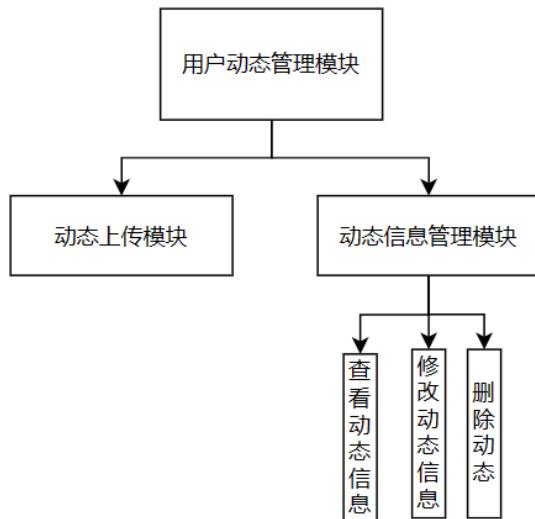


图 3-5 用户动态模块架构图

3.3.5 用户消息管理模块

用户消息管理模块支持用户进行发送消息、接受消息、以及删除消息等三个功能。

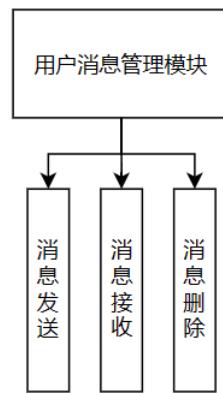


图 3-6 用户消息模块架构图

3.3.6 用户评论管理模块

用户评论模块分为专辑评论模块、歌曲评论模块、歌单评论模块、视频评论模块、动态评论模块等五个模块。

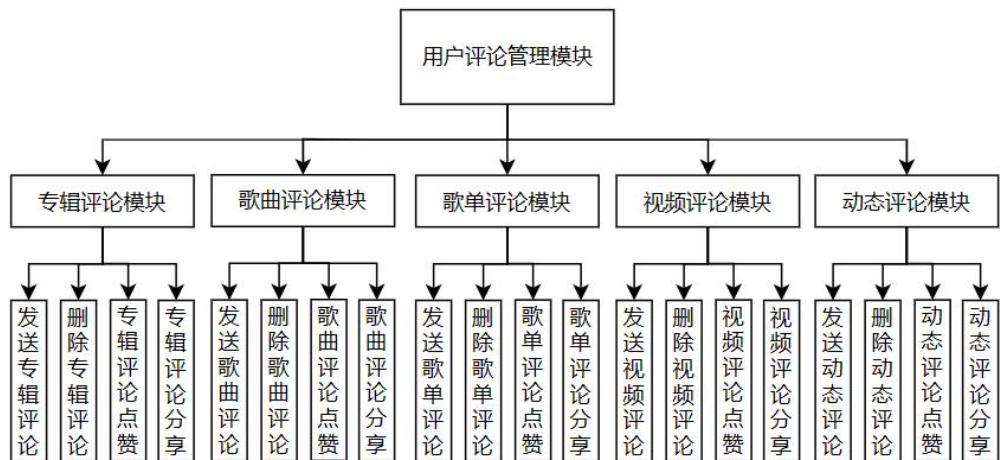


图 3-7 用户评论模块架构图

3.3.7 用户日志管理模块

用户日志管理模块分为歌曲记录模块、视频记录模块等两个模块。

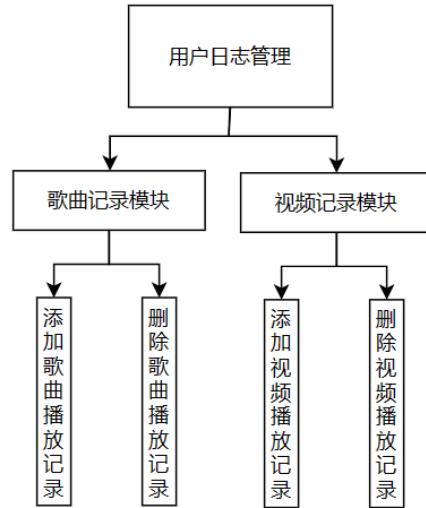


图 3-8 用户日志模块架构图

3.3.8 管理员日志管理模块

管理员日志管理模块支持数据库字段、数据库记录的修改/添加/删除的跟踪。

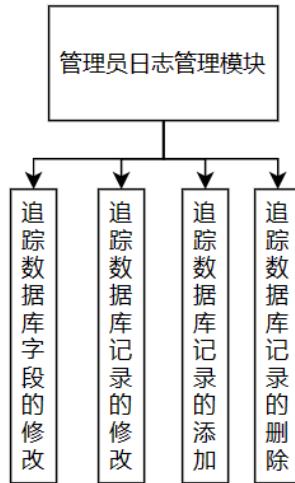


图 3-9 管理员日志模块架构图

3.3.9 标签管理模块

标签管理模块分为歌手标签管理、专辑标签管理、歌曲标签管理、歌单标签管理、视频标签管理等五个子模块。

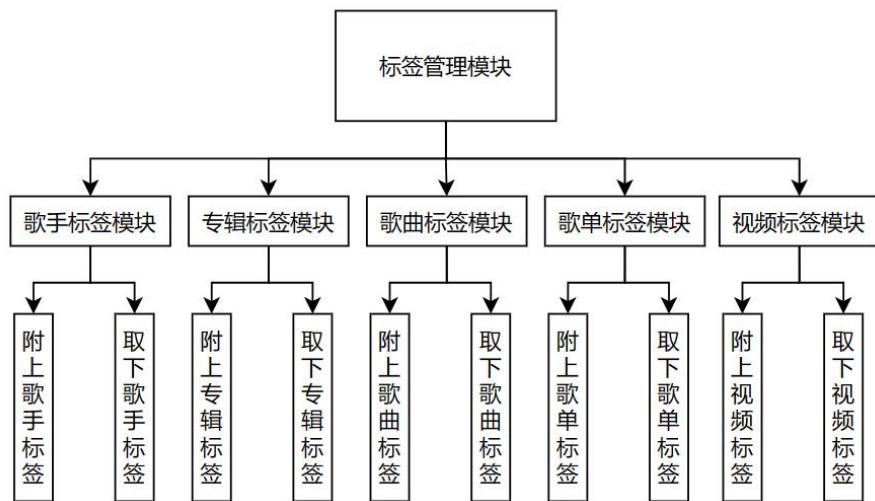


图 3-10 标签模块架构图

3.3.10 图片管理模块

图片管理模块分为图片上传模块、图片删除模块等两个子模块。

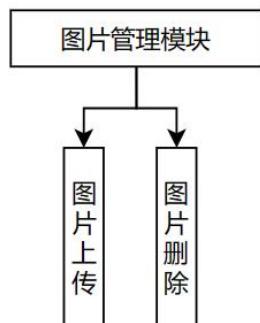


图 3-11 图片模块架构图

3.3.11 数据统计分析模块

数据分析统计模块分为点赞数统计模块、分享数统计模块、热度分析模块等三个子模块。

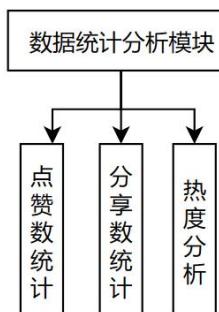


图 3-12 数据分析统计模块架构图

3.3.12 歌曲推荐模块

歌曲推荐模块根据对用户的播放记录进行分析，推断出用户可能感兴趣的歌曲。

3.3.13 排行榜模块

排行榜模块包含热歌排行榜、热门专辑排行榜、热门歌单排行榜等三个功能模块。

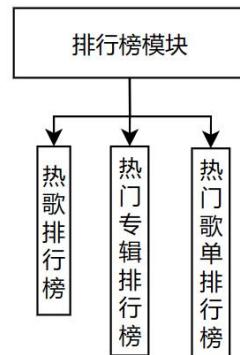


图 3-13 排行榜模块架构图

3.3.14 搜索模块

搜索模块包含歌曲搜索、歌单搜索、专辑搜索、视频搜索、歌手搜索、用户搜索等六个模块。

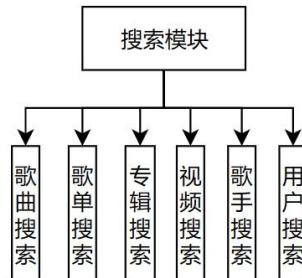


图 3-14 搜索模块架构图

3.3.15 歌曲播放模块

歌曲播放模块作为音乐管理系统中，提高用户体验的一个核心的功能模块，提供了歌曲的播放、暂停等功能。

3.3.16 视频播放模块

视频播放模块作为音乐管理系统中，提高用户体验的另一个核心的功能模块，提供了视频的播放、暂停、控制播放模式等功能。

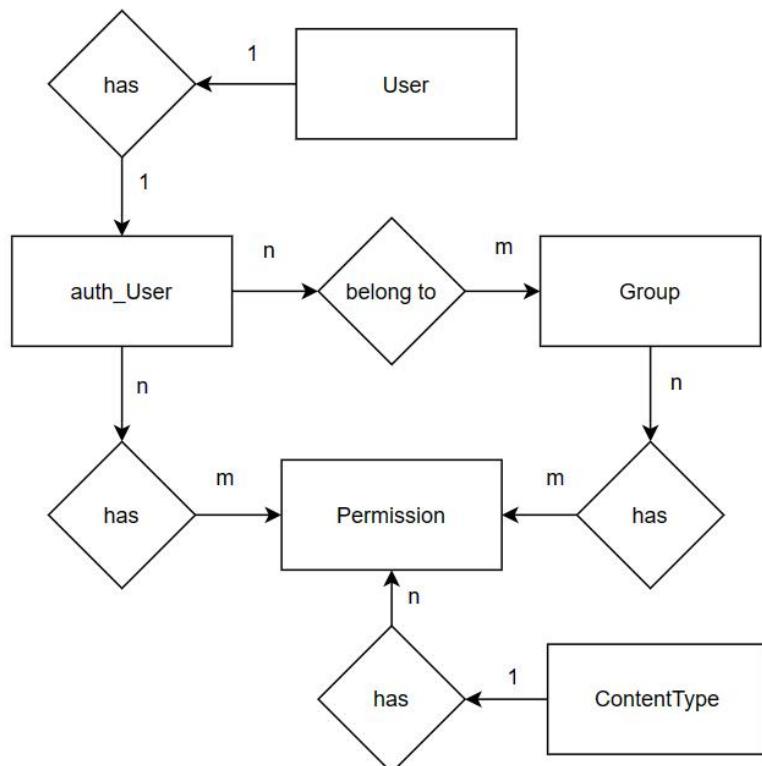
4. 系统数据库设计

4.1 实体类关系设计

E-R 图是一种用于表示实体之间关系的图形化工具，它在数据库设计和信息系统建模中广泛应用。在这个系统中，你可以为每个实体绘制一个框，框内写上实体的名称，然后使用连线表示它们之间的关系。

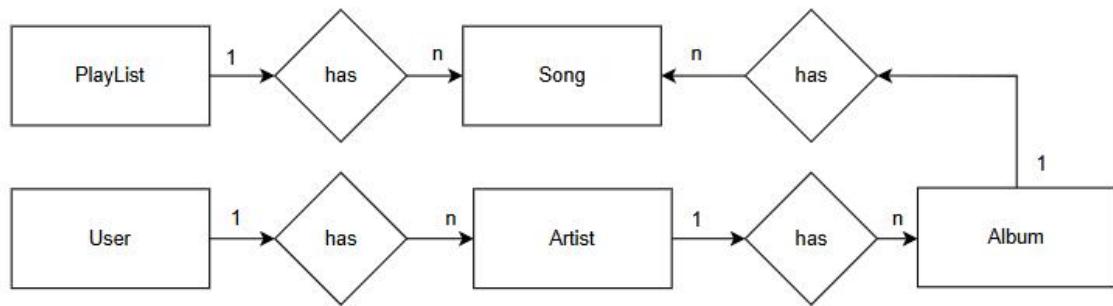
而本系统划分为用户认证、歌手歌曲、用户视频、用户动态、用户消息、用户评论、用户日志、管理员日志、标签、图片等十个管理模块，并且包含数据统计分析、歌曲推荐、排行榜、搜索等四个模块；总共涉及认证用户、认证用户组、权限、内容类型、管理日志、会话、用户、地区、歌手、专辑、歌曲、歌单、视频、用户动态、消息、评论、用户日志、标签、图片等 19 个实体，以下将采用 E-R 图对各类实体进行设计。

4.1.1 用户认证及权限模块实体关系



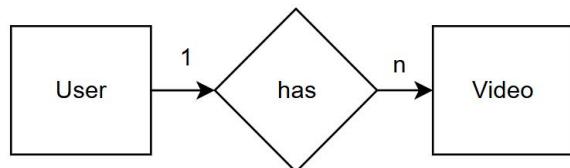
4-1 用户认证及权限模块 ER 图

4.1.2 歌手歌曲管理模块实体关系



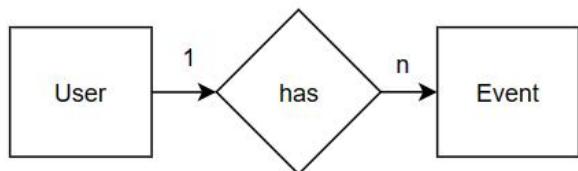
4-2 歌曲管理模块 ER 图

4.1.3 用户视频管理模块实体关系



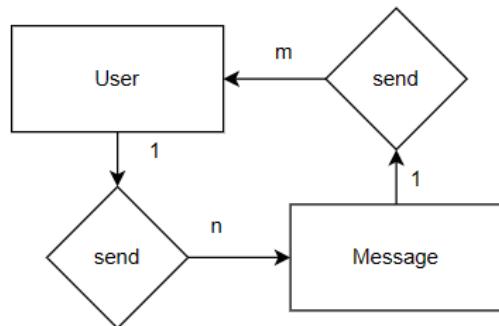
4-3 视频管理模块 ER 图

4.1.4 用户动态管理模块实体关系



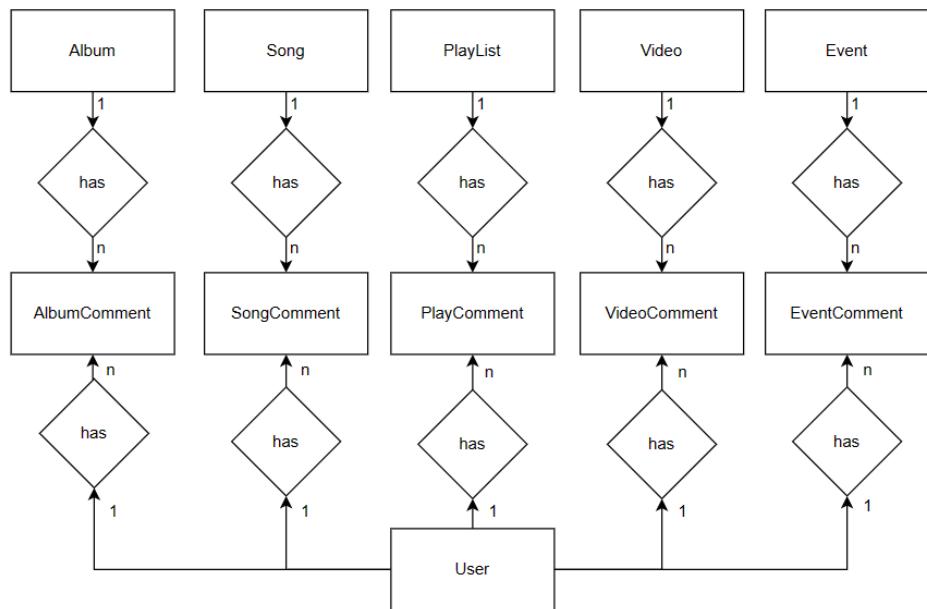
4-4 用户动态管理模块 ER 图

4.1.5 用户消息管理模块实体关系



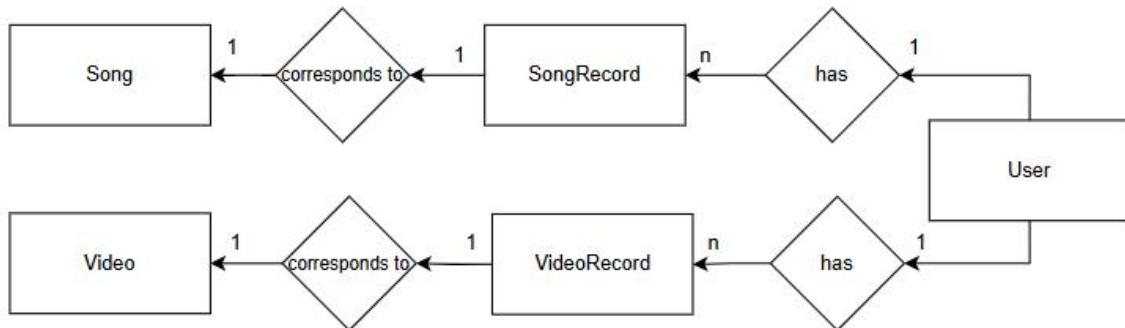
4-5 用户消息管理模块 ER 图

4.1.6 用户评论管理模块实体关系



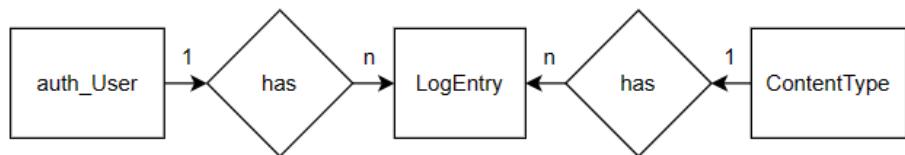
4-6 用户评论管理模块 ER 图

4.1.7 用户日志管理模块实体关系



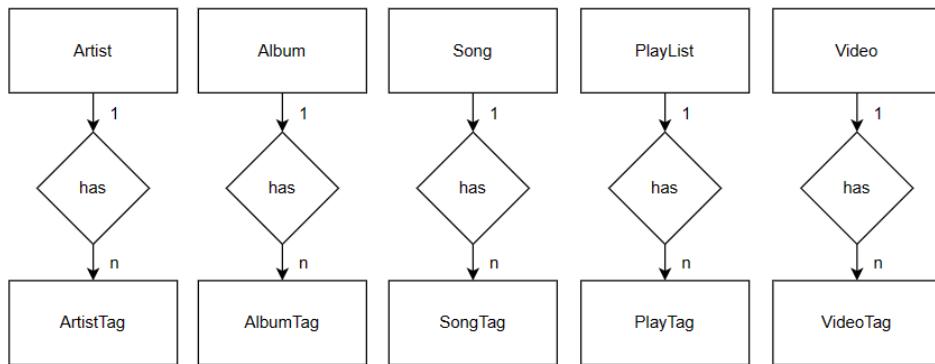
4-7 用户日志管理模块 ER 图

4.1.8 管理员日志管理模块实体关系



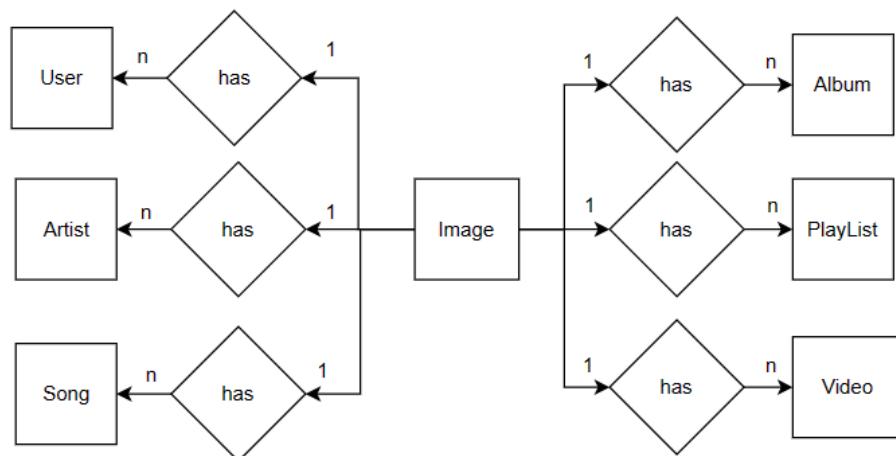
4-8 管理员日志管理模块 ER 图

4.1.9 标签管理模块实体关系



4-9 标签管理模块 ER 图

4.1.10 图片管理模块实体关系



4-10 图片管理模块 ER 图

4.2 数据库表设计

音乐管理系统的数据库包含认证用户表、认证用户组表、权限表、内容类型表、管理日志表、会话表、用户表、地区表、歌手表、专辑表、歌曲表、歌单表、视频表、用户动态表、消息表、评论表、用户日志表、标签表、图片表等十九个数据库表。

4.2.1 认证用户表

表 4-1 auth_user 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	认证用户唯一标识

username	CharField	varchar(20)		认证用户在系统中的用户名
password	CharField	varchar(20)		认证用户密码的 pbkdf2 sha256 加密哈希值
email	EmailField	varchar(20)		认证用户在系统中的邮箱
first_name	CharField	varchar(10)		
last_name	CharField	varchar(10)		
date_joined	DateTimeField	datetime		注册时间
last_login	DateTimeField	datetime		上次登录时间
is_active	BooleanField	bool		是否登录过
is_staff	BooleanField	bool		是否为管理员
is_superuser	BooleanField	bool		是否为超级用户

4.2.2 认证用户组表

表 4-2 auth_group 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	认证用户组唯一标识
name	CharField	varchar(20)		组名

4.2.3 权限表

表 4-3 auth_permission 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	权限唯一标识
content_type_id	ForeignKey	integer	外键	
codename	CharField	varchar(20)		"机器友好"权限名
name	CharField	varchar(20)		"人类友好"权限名

4.2.4 内容类型表

表 4-4 content_type 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	内容类型唯一标识
app_label	CharField	varchar(20)		对应 django app
model	CharField	varchar(20)		对应的模型

4.2.5 管理日志表

表 4-5 admin_log 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	日志唯一标识
content_type_id	ForeignKey	integer	外键	内容类型外键
user_id	ForeignKey	integer	外键	用户外键
action_flag	CharField	varchar(20)		操作标志位
action_time	DateTimeField	datetime		执行操作的时间
object_id	TextField	varchar(20)		操作对象的 id
object_repr	TextField	varchar(20)		操作对象的显示名
change_message	CharField	varchar(100)		操作日志

4.2.6 会话表

表 4-6 session 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
session_key	CharField	varchar(20)	主键	会话唯一标识
expired_date	DateTimeField	datetime		过期日期
session_data	TextField	text(65535)		会话数据

4.2.7 用户表

表 4-7 user 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
-----	-------------	------------	-----	----

id	AutoField	integer	主键	用户唯一标识
avatar_id	ForeignKey	bigint	外键	头像
location_id	ForeignKey	bigint	外键	位置
origin_user_id	OneToOneField	bigint	外键	认证用户
birthday	DateTimeField	datetime		生日
cellphone	CharField	varchar(20)		电话
experience	IntegerField	integer		经验值
gender	IntegerField	integer		性别
github	URLField	varchar(20)		github 链接
nickname	CharField	varchar(50)		昵称
qq	CharField	varchar(20)		qq

4.2.8 地区表

表 4-8 location 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	地区唯一标识
code	SmallIntegerField	integer		区号
location	CharField	varchar(20)		地区名

4.2.9 歌手表

表 4-9 artist 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌手唯一标识
account_id	OneToOneField	bigint	外键	对应的用户
background_id	ForeignKey	integer	外键	背景图
description	TextField	text(50)		描述
joined_time	DateTimeField	datetime		用户成为歌手的时间
stagename	CharField	varchar(20)		艺名

4.2.10 专辑表

表 4-10 album 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	专辑唯一标识

artist_id	ForeignKey	bigint	外键	发布该专辑的歌手
background_id	ForeignKey	bigint	外键	背景图
title	CharField	varchar(20)		专辑名
count	IntegerField	integer		歌曲数量
description	TextField	text		描述
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_company	CharField	varchar(20)		出版公司
pub_date	DateField	date		出版日期

4.2.11 歌曲表

表 4-11 song 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌曲唯一标识
album_id	ForeignKey	bigint	外键	专辑
img_id	ForeignKey	bigint	外键	图片
title	CharField	varchar(20)		歌曲名
data	FileField	varchar(20)		歌曲数据相当于服务器本地的位置
lyrics	TextField	text		歌词
count	IntegerField	integer		播放数
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

4.2.12 歌单表

表 4-12 song_list 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌单唯一标识
user_id	ForeignKey	bigint	外键	发布用户
img_id	ForeignKey	bigint	外键	图片
title	CharField	varchar(20)		歌单名

description	TextField	text		描述
count	IntegerField	integer		播放数
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

4.2.13 视频表

表 4-13 video 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌单唯一标识
img_id	ForeignKey	bigint	外键	图片
user_id	ForeignKey	bigint	外键	发布用户
title	CharField	varchar(20)		歌名单
data	FileField	varchar(20)		视频数据相当于 服务器本地的位 置
description	TextField	text(500)		描述
count	IntegerField	integer		播放数
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

4.2.14 用户动态表

表 4-14 event 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	用户动态唯一标 识
user_id	ForeignKey	bigint	外键	发布用户
content	TextField	text(500)		内容
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

4.2.15 消息表

表 4-15 message 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	消息唯一标识
send_user_id	ForeignKey	bigint	外键	发送者
receive_user_id	ForeignKey	bigint	外键	接收者
content	TextField	text(500)		消息内容
send_time	DateTimeField	datetime		发送时间

4.2.16 评论表

表 4-16 album_comment 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	专辑评论唯一标识
user_id	ForeignKey	bigint	外键	用户
album_id	ForeignKey	bigint	外键	专辑
content	TextField	text(500)		评论内容
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

表 4-17 song_comment 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌曲评论唯一标识
user_id	ForeignKey	bigint	外键	用户
song_id	ForeignKey	bigint	外键	歌曲
content	TextField	text(500)		评论内容
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

表 4-18 play_list_comment 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌单评论唯一标识

user_id	ForeignKey	bigint	外键	用户
playlist_id	ForeignKey	bigint	外键	歌单
content	TextField	text(500)		评论内容
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

表 4-19 video_comment 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	视频评论唯一标识
user_id	ForeignKey	bigint	外键	用户
video_id	ForeignKey	bigint	外键	视频
content	TextField	text(500)		评论内容
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

表 4-20 event_comment 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	用户动态评论唯一标识
user_id	ForeignKey	bigint	外键	用户
event_id	ForeignKey	bigint	外键	用户动态
content	TextField	text(500)		评论内容
liked_count	IntegerField	integer		点赞数
shared_count	IntegerField	integer		分享数
pub_date	DateField	date		发布日期

4.2.17 用户日志表

表 4-21 song_comment 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌曲日志唯一标识
user_id	ForeignKey	bigint	外键	用户
song_id	ForeignKey	bigint	外键	歌曲

count	IntegerField	integer		播放次数
last_time	DateTimeField	datetime		上次播放时间

表 4-22 video_comment 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	视频日志唯一标识
user_id	ForeignKey	bigint	外键	用户
video_id	ForeignKey	bigint	外键	视频
count	IntegerField	integer		播放次数
last_time	DateTimeField	datetime		上次播放时间

4.2.18 标签表

表 4-23 artist_tag 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌手标签唯一标识
tag	CharField	varchar(20)		标签大类
subtag	CharField	varchar(20)		标签

表 4-24 album_tag 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	专辑标签唯一标识
tag	CharField	varchar(20)		标签大类
subtag	CharField	varchar(20)		标签

表 4-25 song_tag 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌曲标签唯一标识
tag	CharField	varchar(20)		标签大类
subtag	CharField	varchar(20)		标签

表 4-26 play_list_tag 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	歌单标签唯一标识

tag	CharField	varchar(20)		标签大类
subtag	CharField	varchar(20)		标签

表 4-23 video_tag 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	视频标签唯一标识
tag	CharField	varchar(20)		标签

4.2.19 图片表

表 4-25 image 表

字段名	django 字段类型	mysql 数据类型	键类型	说明
id	AutoField	integer	主键	图片唯一标识
data	CharField	varchar(20)		图片数据相当于服务器本地的位置

5. 详细设计及系统实现

音乐管理系统包含用户认证及权限管理、歌手歌曲管理、用户视频管理、用户动态管理、用户消息管理、用户评论管理、用户日志管理、管理员日志管理、标签管理、图片管理、数据统计分析、歌曲推荐、排行榜、搜索、歌曲播放、视频播放等 16 个模块，我们将采用程序流程图对这些功能进行详细设计。

5.1 用户认证及权限模块

用户认证及权限管理模块分为认证和权限两个部分：其中用户认证部分分为用户注册、用户登录、用户注销三个功能；用户操作模块包含查看公共信息、修改个人信息、修改密码、增删改查数据库等四个功能。

5.1.1 用户认证模块

用户认证模块包含用户注册、用户登录、用户注销三个功能：

- (1) 用户注册功能：用户需要发送邮箱验证码到用户邮箱进行验证；在验证成功的情况下，服务器将比对两次输入的密码是否相等，并且在数据库中查询是否有相同用户名的账户，其中满足两次密码相等并且昵称不存在的用户可以成功注册。
- (2) 用户登录功能：用户填写账户密码，并登录发送给服务器，若账户密码均不为空并且账

号密码输入正确，则给予登录。

(3) 用户注销功能：已登录用户可以点击注销按钮以请求服务器注销账号。

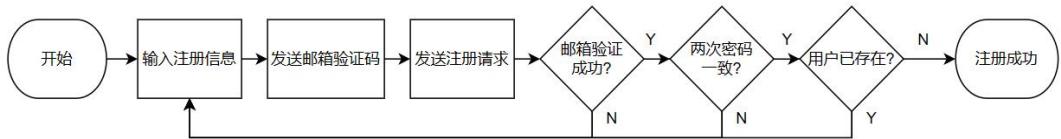


图 5-1 用户注册功能流程图

用户注册功能的关键代码如下：

```
if util == 'register' and req._request.method == 'POST':  
    if not validate_email(req):  
        raise ValidationError("邮箱验证错误")  
  
    username = req.data.get('username')  
    password = req.data.get('password')  
    password_confirm = req.data.get('password_confirm')  
    email = req.data.get('email')  
  
    if password != password_confirm:  
        raise ValidationError("两次密码不一致")  
  
    try:  
        user = m.auth_User.objects.create_user(username=username, password=password,  
email=email)  
  
    except IntegrityError as e:  
        raise ValidationError("用户名已存在")  
  
    except ValueError:  
        raise ValidationError("用户名注册失败")  
  
    # ...  
  
    return Response("注册成功")
```

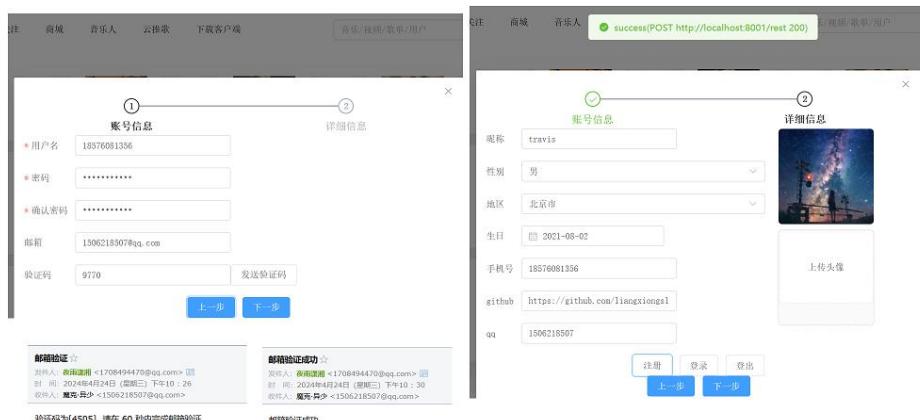


图 5-2 用户注册功能运行截图

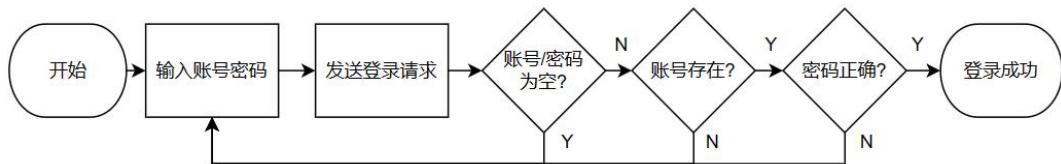


图 5-3 用户登录功能流程图

用户登录功能的关键代码如下：

```

if util == 'login' and req._request.method == 'POST':
    username = req.data.get('username')
    password = req.data.get('password')
    user = None
    try:
        user = authenticate(username=username, password=password)
    except ValidationError:
        raise ValidationError("用户名或密码为空")
    if user is None:
        raise ValidationError("用户不存在或密码错误")
    login(req, user)
    return Response({'msg': "登录成功", 'id': user.user.pk, 'aid': user.user.artist.id})

```

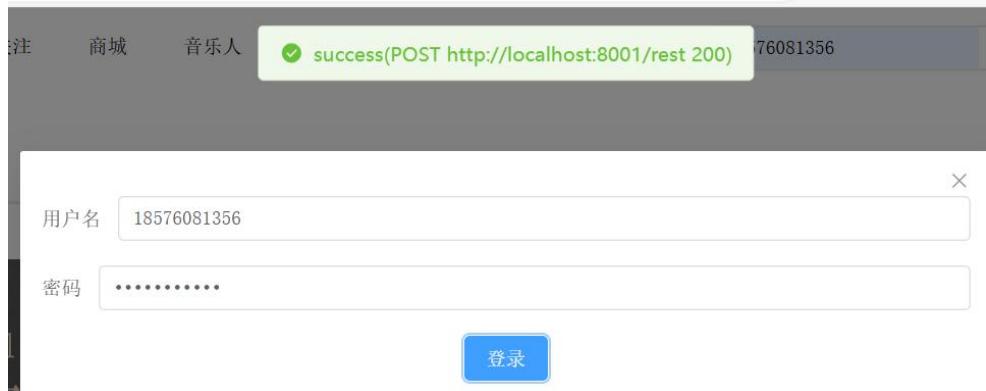


图 5-4 用户登录功能运行截图



图 5-5 用户注销功能流程图

用户注销功能的关键代码如下：

```

if util == 'logout' and req._request.method == 'POST':
    logout(req)
    return Response("注销成功")

```

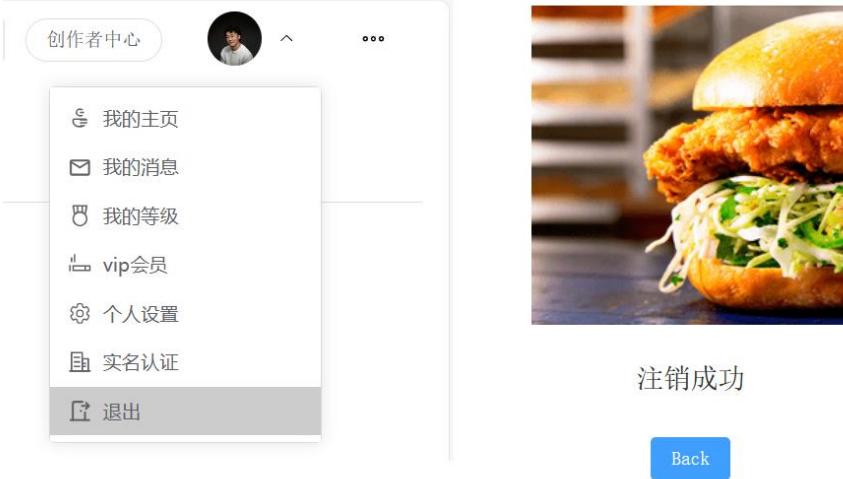


图 5-6 用户注销功能运行截图

5.1.2 用户操作模块

用户操作模块包含查看公共信息、修改个人信息、修改密码、执行增删改查操作等四个功能：

- (1) 查看公共信息：用户可通过访问前端页面直接对公共信息进行查看。
- (2) 修改个人信息：成功登录的用户被允许修改账号信息。
- (3) 修改密码：在已登录的状态下，用户需要邮箱认证通过后方可请求服务器修改密码。
- (4) 执行增删改查操作：已登录的用户需要携有指定的访问权限才能执行特定的操作。

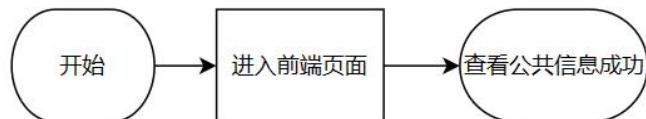


图 5-7 用户查看公共信息流程图



图 5-8 用户查看公共信息运行截图

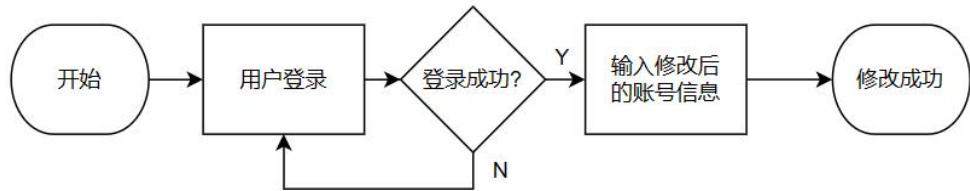


图 5-9 用户修改个人信息流程图

图 5-10 展示了用户修改个人信息的运行截图。界面显示了用户的个人信息编辑表单，包括用户名、性别、生日、电话、QQ、github、邮箱和经验值。所有输入框都已填写，下方有一个“保存”按钮。

travis	性别 男
生日 2021年08月01日	电话 18576081356
qq 1506218507	github https://github.com/liangxiongsl
经验值 0	邮箱 1506218507@qq.com

图 5-10 用户修改个人信息运行截图

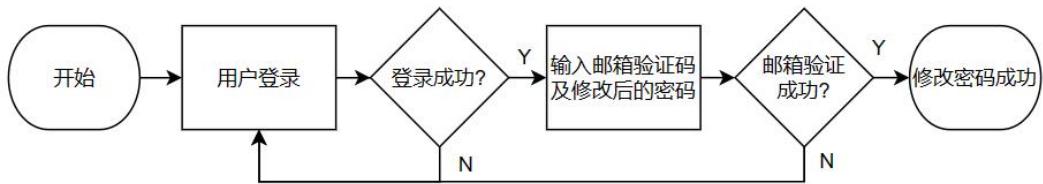


图 5-11 用户修改密码流程图

图 5-12 展示了用户修改密码的运行截图。左侧是用户个人中心的菜单栏，右侧是一个弹出的修改密码对话框。对话框中包含“新密码”、“邮箱”、“验证码”（显示为 9770）和“发送验证码”按钮。下方有一个“修改密码”按钮。

图 5-12 用户修改密码运行截图

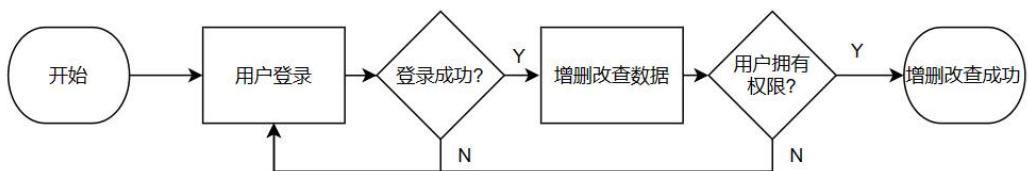


图 5-13 用户执行增删查改操作流程图

用户执行创建或查询操作的关键代码：

```

if req.method == 'GET':
    data = filter(req, Model, ModelSerializer)

```

```

conf = field_to_tag(Model)
return Response({'data': data, 'conf': conf})

elif req.method == 'POST':
    data = JSONParser().parse(req)
    serializer = ModelSerializer(data=data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

用户执行查询、更新、删除操作的关键代码如下：

```

if req.method == 'GET':
    serializer = ModelSerializer(model)
    conf = field_to_tag(Model)
    return Response({'data': serializer.data, 'conf': conf})

elif req.method == 'PUT':
    data = req.data
    serializer = ModelSerializer(model, data=data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

elif req.method == 'DELETE':
    model.delete()
    return Response(status=status.HTTP_204_NO_CONTENT)

```

The figure consists of four screenshots of a web application interface:

- 增加 album comment:** A form for adding a comment to an album. It includes fields for User (薛之谦), Album (Jazz), Content (example), Shared count (0), Liked count (0), and buttons for '保存并返回' (Save and return) and '保存并继续编辑' (Save and continue editing).
- 修改 album comment:** A form for modifying an existing comment. It shows a comment by User (赵雷) titled 'hello world' with Content 'hello world'. It also shows Shared count (9018) and Liked count (7415). Buttons include '保存并返回' and '保存并继续编辑'.
- 选择 album comment 来修改:** A search interface where 'hello world' is selected from a list of comments. The list includes various entries like 'test', '薛之谦', '赵雷', etc.
- 站点管理:** A sidebar navigation menu with sections like MUSIC, ARTISTS, EVENTS, IMAGES, LOCATIONS, MESSAGES, PLAY COMMENTS, PLAY LISTS, SONG COMMENTS, SONG RECORDS, SONG TAGS, and USERS. It also displays recent actions such as '添加' (Add) and '修改' (Modify) operations.

图 5-14 用户执行增删查改操作运行截图

5.2 歌手歌曲管理模块

歌手歌曲管理模块分为歌手管理、专辑管理、歌单管理等三个子模块。

5.2.1 歌手管理模块

歌手管理模块包含歌手入驻、查看歌手信息、修改歌手信息、注销歌手等四个功能：

- (1) 歌手入驻：在已登录的情况下，用户可以通过填写歌手信息表格以请求歌手入驻。
- (2) 查看歌手信息：任何用户均可访问前端对歌手进行查询。
- (3) 修改歌手信息：在已登录的情况下，用户输入修改后的歌手信息后，发送修改请求，即可修改歌手信息。
- (4) 注销歌手：已登录并且已认证的歌手支持注销歌手。

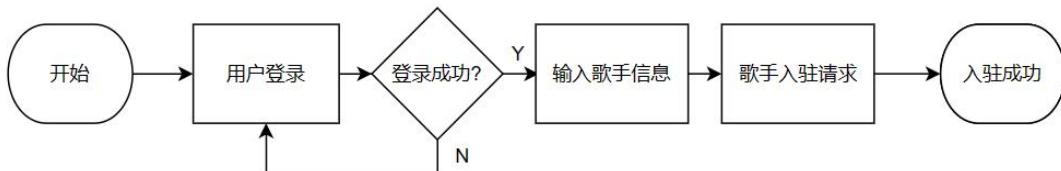


图 5-15 歌手入驻流程图

歌手入驻功能的关键代码：

```

if util == 'join' and req._request.method == 'POST':
    background = req.data.get('background')
    background_id = m.Image.objects.create(data=background).id
  
```

```

stagename = req.data.get('stagename')
description = req.data.get('description')
try:
    artist = m.Artist.objects.create(account=req.user, stagename=stagename,
                                     description=description,
                                     background_id=background_id)
except:
    raise ValidationError("歌手入驻失败")
return Response(artist)

```



图 5-16 歌手入驻运行截图



图 5-17 查看歌手信息流程图

查看歌手信息关键代码如下：

```

if util == 'g-Artist' and req._request.method == 'GET':
    try:
        id = req.query_params['id']
    except Exception:
        raise ValidationError(f"GET {req._request.path} 缺少查询参数 id")
    artist = m.Artist.objects.get(id=id)
    artist_ = s.S_Artist(artist).data
    artist_['albums'] = s.S_Album(artist.albums, many=True).data
    artist_['background'] = fhttp://localhost:8001/rest/img/?id={artist_['background']}
    artist_['songs'] = []
    for v in artist.albums.all():

```

```

artist_['songs'] += s.S_Song(v.songs, many=True).data
artist_['videos'] = s.S_Video(artist.account.videos, many=True).data
# ...
return Response(artist_)

```

[← 返回](#) | index / g / Artist

陈奕迅



热门歌曲 所有专辑 相关MV 艺人介绍



爱情转移
认了吧



淘汰
认了吧

图 5-18 查看歌手信息运行截图

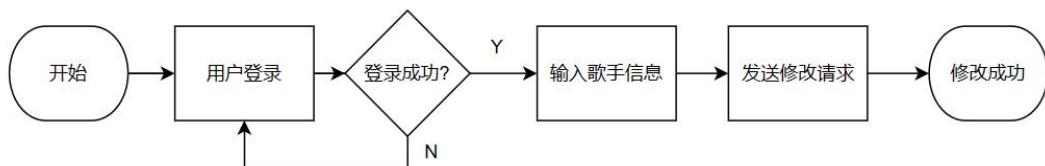


图 5-19 修改歌手信息流程图

修改 artist

薛之谦

历史

Account: 薛之谦 保存 + 新增 编辑

Background: Image object (147) 背景设置

Stagename: 薛之谦

Description:

薛之谦 (Joker Xue), 1983年1月1日出生于上海，中国内地流行乐男歌手、影视演员、音乐制作人，毕业于格里昂酒店管理学院。
2005年，因参加选秀节目《我型我秀》而正式出道。2006年，发行首张同名专辑《薛之谦》，随后凭借歌曲《认真的雪》获得广泛关注。2007年，凭借专辑《你过得好吗》获得MusicRadio中国TOP排行榜内地年度最受欢迎男歌手奖。2008年，发行专辑《深深爱过你》，同年在上海举行个人首场演唱会“谦年传说”。2013年，专辑《八卦薛之谦》获得MusicRadio中国TOP排行榜内地推荐唱片。2014年，凭借专辑《意外》获得第21届东方风云榜颁奖最佳制作人；同年他还获得音乐V榜年度盛典年度创作歌手奖。2015年6月，薛之谦首度担当制作人并发行原创EP《绅士》；同年，他还主演都市励志剧《妈妈像花儿一样》。2016年，凭借EP《绅士》、《一半》获得第16届全球华语歌曲排行榜最受欢迎男歌手、五大最受欢迎男歌手奖、上海地区杰出歌手奖及最受欢迎创作歌手奖，而歌曲《初学者》则获得年度二十大金曲奖。

保存 保存并增加另一个 保存并继续编辑 删除

图 5-20 修改歌手信息运行截图

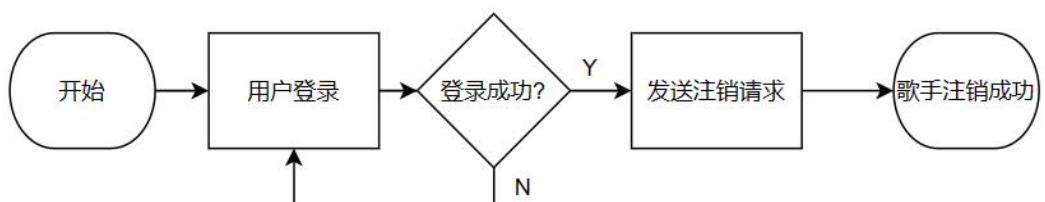


图 5-21 注销歌手流程图

你确定吗？

你确认想要删除 artist "travis"？ 下列所有相关的项目都将被删除：

概览

- Artists: 1

对象

- Artist: travis

是的，我确定

不，返回

图 5-22 注销歌手运行截图

5.2.2 专辑管理模块

专辑管理模块包含创建专辑、查看专辑、修改专辑、删除专辑等四个功能：

- (1) 创建专辑：在已登录的情况下，已入驻歌手被允许创建专辑。
- (2) 查看专辑：所有人均可访问前端页面以查看专辑信息。
- (3) 修改专辑：在已登录的情况下，已入驻歌手被允许修改专辑。
- (4) 删除专辑：在已登录的情况下，已入驻歌手被允许删除专辑。

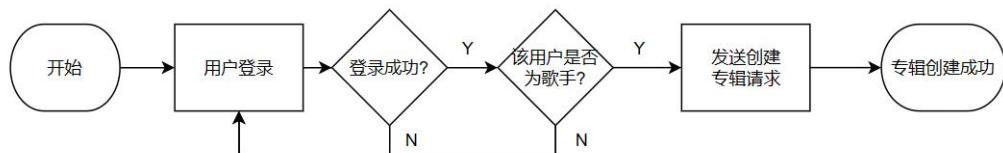


图 5-23 创建专辑流程图

增加 album

Artist:	<input type="text"/>								
Background:	<input type="text"/>								
Collected users:	<table border="1"> <tr><td>林俊杰</td></tr> <tr><td>陈奕迅</td></tr> <tr><td>薛之谦</td></tr> <tr><td>毛不易</td></tr> <tr><td>G.E.M.邓紫棋</td></tr> <tr><td>李荣浩</td></tr> <tr><td>汪苏泷</td></tr> <tr><td>周杰伦</td></tr> </table> <p>按住 Control 键或 Mac 上的 Command 键来选择多项。</p>	林俊杰	陈奕迅	薛之谦	毛不易	G.E.M.邓紫棋	李荣浩	汪苏泷	周杰伦
林俊杰									
陈奕迅									
薛之谦									
毛不易									
G.E.M.邓紫棋									
李荣浩									
汪苏泷									
周杰伦									
Title:	<input type="text"/>								
Count:	<input type="text"/> 0								
Shared count:	<input type="text"/> 0								
Liked count:	<input type="text"/> 0								
Description:	<input type="text"/>								
Pub company:	<input type="text"/>								

图 5-24 创建专辑运行截图



图 5-25 查看专辑流程图

查看专辑相关代码如下：

```

if util == 'g-Album' and req._request.method == 'GET':
    try:
        id = req.query_params['id']
    except Exception:
        raise ValidationError(f"GET {req._request.path} 缺少查询参数 id")
    album_ = m.Album.objects.get(id=id)
    album = s.S_Album(album_).data
    album['background'] = f"http://localhost:8001/rest/img/?id={album['background']}"
    album['artist'] = s.S_Artist(m.Artist.objects.get(id=album['artist'])).data
    album['songs'] = s.S_Song(album_.songs, many=True).data
    for i, v in enumerate(album['songs']):
        album['songs'][i]['img'] = f"http://localhost:8001/rest/img/?id={album['songs'][i]['img']}"
        album['songs'][i]['data'] = f"http://localhost:8001/media{v['data']}"
        album['songs'][i]['artist'] = s.S_Artist(m.Song.objects.get(id=v['id']).album.artist).data

```

```
return Response(album)
```



图 5-26 查看专辑运行截图

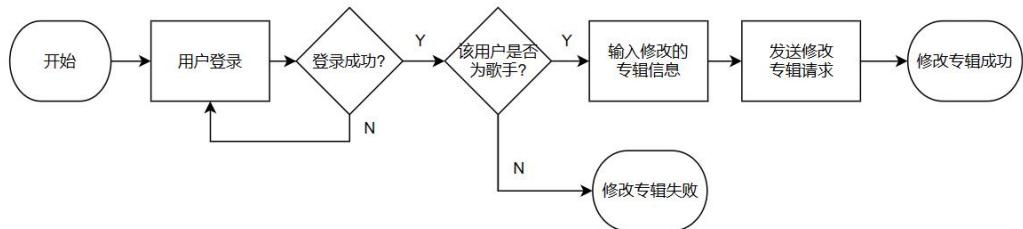


图 5-27 修改专辑流程图

修改 album

无穷

Artist: per se

Background: Image object (282)

Collected users:

- 林俊杰
- 陈奕迅
- 孙之谦
- 毛不易
- G.E.M. 邓紫棋
- 李荣浩
- 汪苏泷
- 周深

Title: 无穷

Count: 12435

Shared count: 4640

Liked count: 3460

Description:

人类一直不理会大自然的启示，顺着贪念，自私地活在暴殄天物的生活中。
天灾连连，他们不只一直找不到快乐，就连容身之地都不再适合居住了。
后来，也许已经没有后来...
per se 的新计划以多个童话故事作题材，用不同角色来探讨人性；而《无穷》这首歌的灵感来自《渔夫和金鱼的故事》。

Pub company:

图 5-28 修改专辑运行截图

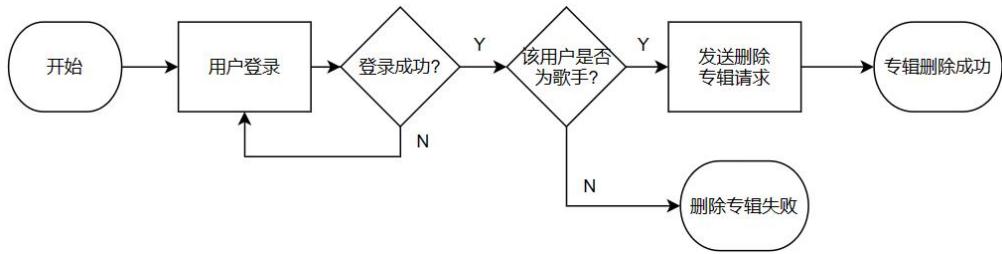


图 5-29 删除专辑流程图

你确定吗?

你确认想要删除 album "如果呢"? 下列所有相关的项目都将被删除:

概览

- Albums: 1
- Songs: 1
- Playlist-song关系: 1

对象

- Album: 如果呢
- Song: 如果呢
- Playlist-song关系: Playlist_join_songs object (69)

是的, 我确定 不, 返回

图 5-30 删除专辑运行截图

5.2.3 歌单管理模块

歌单管理模块包含创建歌单、添加单曲、查看歌单、修改歌单、删除歌单等五个功能:

- (1) 创建歌单: 在已登录的情况下, 用户被允许创建歌单。
- (2) 添加单曲: 在已登录的情况下, 用户被允许添加单曲到个人歌单中。
- (3) 查看歌单: 所有人均可访问前端页面以查看歌单信息。
- (4) 修改歌单: 在已登录的情况下, 用户被允许修改歌单。
- (5) 删除歌单: 在已登录的情况下, 用户被允许删除歌单。

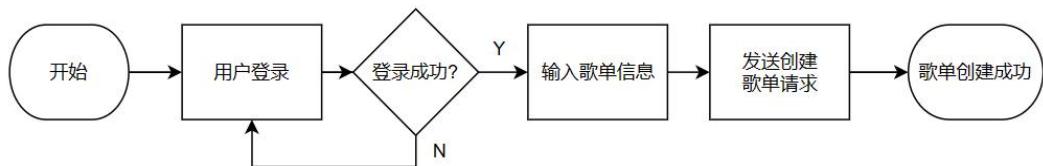


图 5-31 创建歌单流程图



图 5-32 创建歌单运行截图

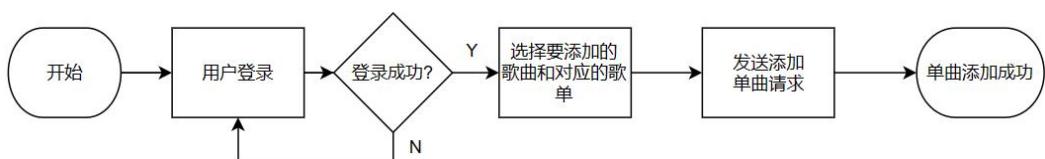


图 5-33 添加单曲

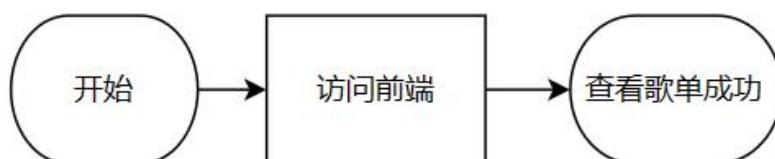


图 5-34 查看歌单流程图

查看歌单关键代码如下：

```
if util == 'g-PlayList' and req._request.method == 'GET':
```

```
try:
```

```
    id = req.query_params['id']
```

```
except Exception:
```

```

raise ValidationError(f"GET {req._request.path} 缺少查询参数 id")

playlist_ = m.PlayList.objects.get(id=id)
playlist = s.S_PlayList(playlist_).data
for i, v in enumerate(playlist['join_songs']):
    playlist['join_songs'][i] = s.S_Song(m.Song.objects.get(id=v)).data
for i, v in enumerate(playlist['collect_users']):
    playlist['collect_users'][i] = s.S_User(m.User.objects.get(id=v)).data
playlist['img'] = f"http://localhost:8001/rest/img/?id={playlist['img']}"
playlist['user'] = s.S_User(m.User.objects.get(id=playlist['user'])).data
# ...
return Response(playlist)

```



歌曲列表



达尔文
JJ的咖啡调调, Vol. 2



修炼爱情
因你而在

图 5-35 查看歌单运行截图

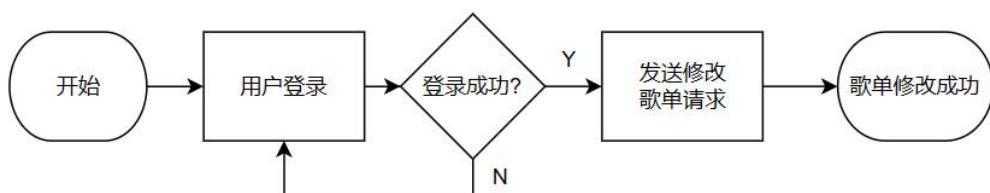


图 5-36 修改歌单流程图



图 5-37 修改歌单运行截图

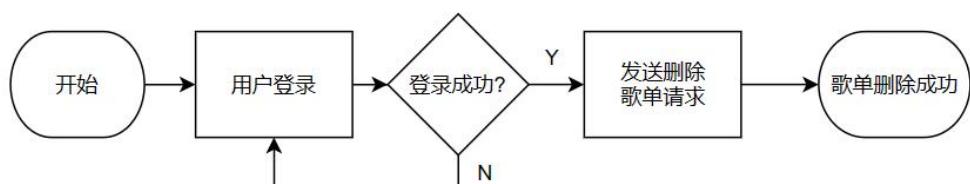


图 5-38 删 除歌单流程图

你确定吗?

你确认想要删除 play list "二十岁后的人生像是按了加快键"? 下列所有相关的项目都将被删除:

概览

- Play lists: 1
- Playlist-song关系: 4
- Playlist-user关系: 2

对象

- Play list: 二十岁后的人生像是按了加快键
 - Playlist-song关系: PlayList_join_songs object (120)
 - Playlist-song关系: PlayList_join_songs object (121)
 - Playlist-song关系: PlayList_join_songs object (122)
 - Playlist-song关系: PlayList_join_songs object (123)
 - Playlist-user关系: PlayList_collect_users object (63)
 - Playlist-user关系: PlayList_collect_users object (64)

是的, 我确定

不, 返回

图 5-39 删除歌单运行截图

5.3 用户视频管理模块

用户视频管理模块包含上传视频、查看视频信息、修改视频、删除视频等四个功能：

- (1) 创建视频：在已登录的情况下，用户被允许上传视频。
- (2) 查看视频：所有人均可访问前端页面以播放视频，并查看视频信息。
- (3) 修改视频：在已登录的情况下，用户被允许修改视频。
- (4) 删除视频：在已登录的情况下，用户被允许删除视频。

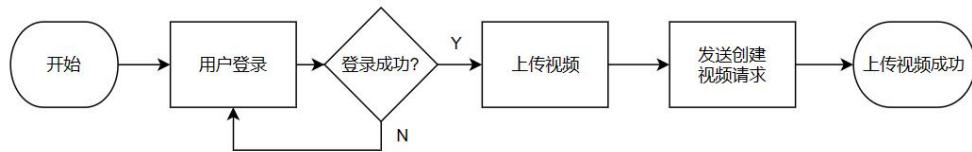


图 5-40 上传视频流程图

增加 video

User: 陈奕迅 编辑 新建 眼睛

Img: Image object (15) 编辑 新建 眼睛

Collect users: +
林俊杰
陈奕迅
薛之谦
毛不易
G.E.M.邓紫棋
李荣浩
汪苏泷
周杰伦

Title: 葡萄成熟时

Count: 5231

Shared count: 7123

Liked count: 1244

Description:

Data: 选择文件 未选择文件

保存 保存并增加另一个 保存并继续编辑

图 5-41 上传视频运行截图



图 5-42 查看视频信息流程图

查看视频信息关键代码如下：

```

if util == 'g-Video' and req._request.method == 'GET':
    try:
        id = req.query_params['id']
    except Exception:
        raise ValidationError(f"GET {req._request.path} 缺少查询参数 id")
    video = s.S_Video(m.Video.objects.get(id=id)).data
    video['data'] = f"http://localhost:8001/media{video['data']}"
    video['img'] = f"http://localhost:8001/rest/img/?id={video['img']}"
    return Response(video)

```

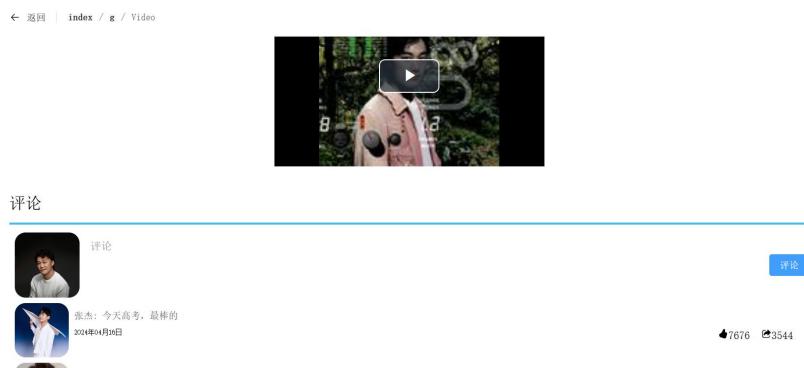


图 5-43 查看视频信息运行截图

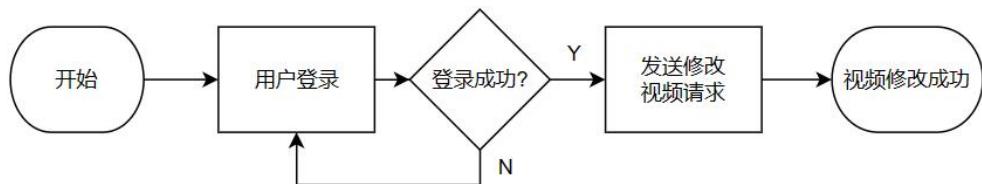


图 5-44 修改视频流程图

修改 video

淘汰

User: 陈奕迅 陈奕迅

Img: Image object (289) Image object (289)

Collect users: 林俊杰 陈奕迅 薛之谦 毛不易 G.E.M.邓紫棋 李荣浩 汪苏泷 周杰伦 +

按住 Control 键或 Mac 上的 Command 键来选择多项。

Title: 淘汰

Count: 32533

Shared count: 45342

Liked count: 23531

Description:

Data: 目前: uploads/video/5reY5rGw.mp4
修改: 未选择文件

保存 保存并增加另一个 保存并继续编辑

图 5-45 修改视频运行截图

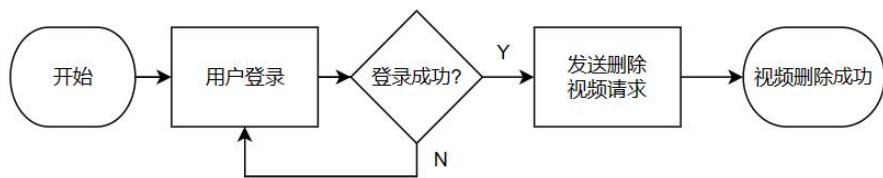


图 5-46 删除视频流程图

你确定吗?

你确认想要删除 video "淘汰"? 下列所有相关的项目都将被删除:

概览

- Videos: 1
- Video-user关系: 1

对象

- Video: 淘汰
- Video-user关系: Video_collect_users object (84)

是的, 我确定

不, 返回

图 5-47 删除视频运行截图

5.4 用户动态管理模块

用户动态管理模块包含发布用户动态、查看用户动态、修改用户动态、删除用户动态等四个功能：

- (1) 发布用户动态：在已登录的情况下，用户被允许上传动态。
- (2) 查看用户动态：所有人均可访问前端页面以查看所有用户的动态。
- (3) 修改用户动态：在已登录的情况下，用户被允许修改个人用户动态。
- (4) 删除用户动态：在已登录的情况下，用户被允许删除个人用户动态。

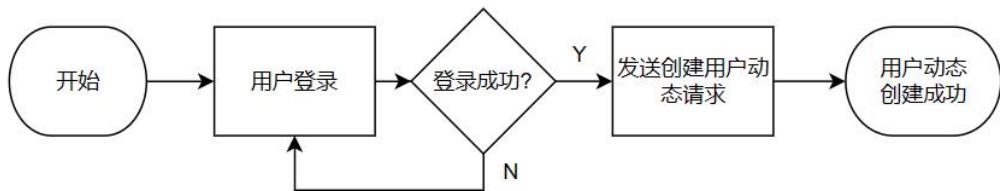


图 5-48 发布用户动态流程图

增加 event

该截图展示了发布用户动态的前端界面。顶部显示了用户名“陈奕迅”和一些操作图标（编辑、新增、查看）。下方有三个输入框：Shared count: 1234, Liked count: 3523, Content: 故乡的樱花树开了。底部有三个按钮：保存、保存并增加另一个、保存并继续编辑。

图 5-49 发布用户动态运行截图



图 5-50 查看用户动态流程图

查看用户动态关键代码如下：

```
if util == 'o-Friends' and req._request.method == 'GET':  
    user = req.user.user  
    friends_ = s.S_User(user.fans, many=True).data + s.S_User(user.concerns, many=True).data
```

```

friends = []
events = []
for v in friends_:
    if v not in friends:
        friends.append(v)
    events += s.S_Event(m.User.objects.get(id=v['id']).events.order_by('-pub_date'),
many=True).data
for v in events:
    v['user'] = s.S_User(m.User.objects.get(id=v['user'])).data
    v['user']['avatar'] = f'http://localhost:8001/rest/img/?id={v["user"]["avatar"]}'
    print(v['user'])
return Response(events)

```



图 5-51 查看用户动态运行截图

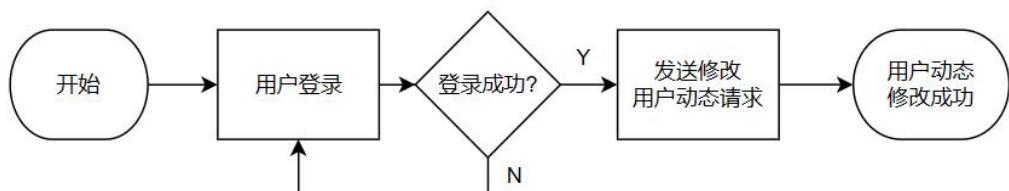


图 5-52 修改用户动态流程图

修改 event

Event object (1)

User:	陈奕迅			
Shared count:	1234			
Liked count:	1245			
Content:	故乡的樱花树开了			

操作按钮: 保存 | 保存并增加另一个 | 保存并继续编辑

图 5-53 修改用户动态运行截图

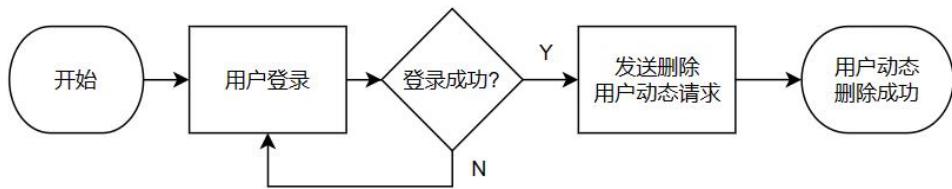


图 5-54 删除用户动态流程图

你确定吗?

你确认想要删除 event "Event object (1)"? 下列所有相关的项目都将被删除:

概览

- Events: 1

对象

- Event: Event object (1)

是的, 我确定

不, 返回

图 5-55 删除用户动态运行截图

5.5 用户消息管理模块

用户消息管理模块包含发送消息、查看消息、删除消息等三个功能:

- (1) 发送消息: 已登录的用户可以向任何用户发送消息或私信。
- (2) 查看消息: 已登录的用户可以查看其接受到的消息或私信。
- (3) 删除消息: 已登录的用户可以单方面删除本地收到的消息。

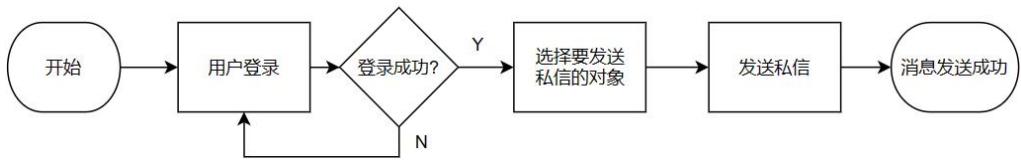


图 5-56 发送消息或私信流程图

发送私信功能的关键代码如下：

```

if util == 'g-Message' and req._request.method == 'GET':
    try:
        id1 = req.query_params['id1']
        id2 = req.query_params['id2']
    except Exception:
        raise ValidationError(f"GET {req._request.path} 缺少查询参数 id1 或 id2")
    try:
        m.User.objects.get(id=id1)
        m.User.objects.get(id=id2)
    except:
        raise ValidationError(f"user1 或 user2 不存在")
    # ...
    return Response(messages)

```



图 5-57 发送消息或私信运行截图

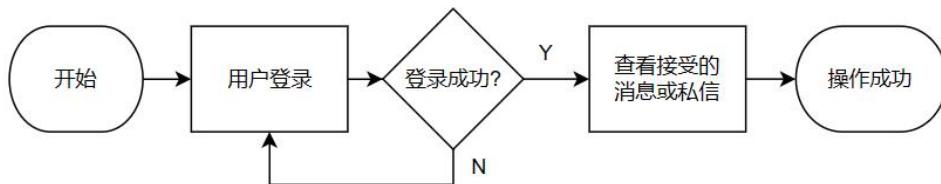


图 5-58 查看消息或私信流程图

查看私信功能的关键代码如下：

```

import {watchEffect, ref} from 'vue'
let data = ref([])

```

```

const id = route.query.id ? route.query.id : localStorage.getItem('id')
let close_handle = watchEffect(async () => {
  get(`my-Messages/?id=${id}`).then(res => data.value = res)
})

```

← 返回 | index / g / Message



图 5-59 查看消息或私信运行截图

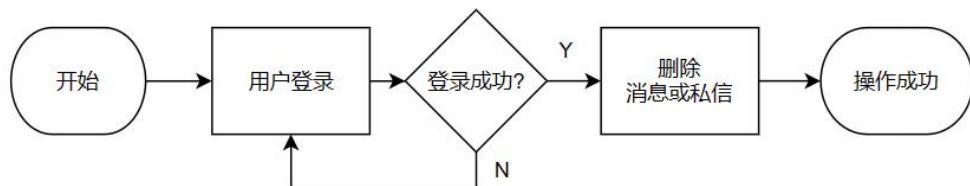


图 5-60 删除消息或私信流程图

你确定吗?

你确认想要删除 message "Message object (14)"? 下列所有相关的项目都将被删除:

概览

- Messages: 1

对象

- Message: Message object (14)

是的, 我确定

不, 返回

图 5-61 删除消息或私信运行代码

5.6 用户评论管理模块

用户评论管理模块包含发表评论、删除评论、点赞评论、分享评论等四个功能:

- (1) 发表评论: 已登录用户可以对任何歌曲/专辑/歌单/视频/用户登台进行评论。
- (2) 删除评论: 已登录用户可以删除自己发表过的任何评论。
- (3) 点赞评论: 已登录用户可以对任何歌曲/专辑/歌单/视频/用户登台进行点赞。
- (4) 分享评论: 已登录用户可以对任何歌曲/专辑/歌单/视频/用户登台进行分享。

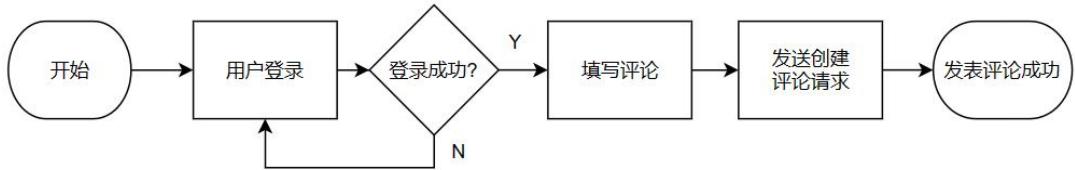


图 5-62 发表评论流程图

发表评论功能的关键代码如下：

```

if util == 'g-Comment' and req._request.method == 'GET':
    try:
        type = req.query_params['type']
    except Exception:
        raise ValidationError(f"GET {req._request.path} 缺少查询参数 type")
    try:
        id = req.query_params['id']
    except Exception:
        raise ValidationError(f"GET {req._request.path} 缺少查询参数 id")
    comments = []
    if type.lower() == 'album':
        comments = s.S_AlbumComment(m.AlbumComment.objects.filter(album_id=id),
many=True).data
    if type.lower() == 'song':
        comments = s.S_SongComment(m.SongComment.objects.filter(song_id=id),
many=True).data
    # ...
    return Response(comments)

```

评论

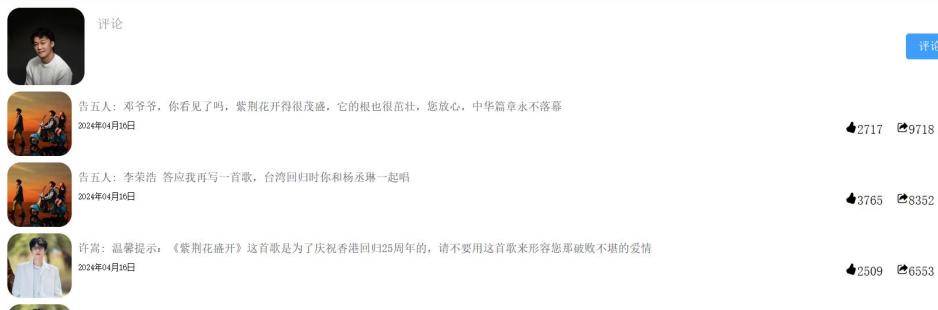


图 5-63 发表评论运行截图

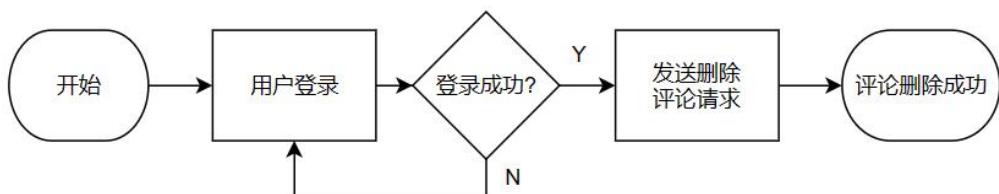


图 5-64 删 除评论流程图

你确定吗？

你确认想要删除 song comment "这张专辑最棒的就是月球上的了人。。。"？下列所有相关的项目都将被删除：

概览

- Song comments: 1

对象

- Song comment: 这张专辑最棒的就是月球上的了人。。。

是的，我确定

不，返回

图 5-65 删 除评论运行截图

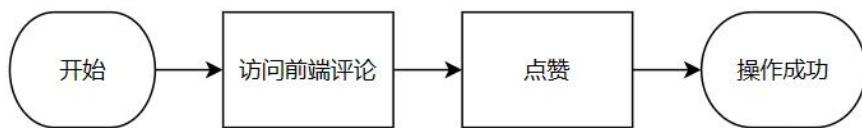


图 5-66 点 赞评论流程图

评论点赞功能的关键代码如下：

```

<span @click="v.liked_count++" :class="c_hover">
    <el-icon><ThumbLike16Filled/></el-icon> {{v.liked_count}}
</span>

```



告五人：邓爷爷，你看见了吗，紫荆花开得很茂盛，它的根也很茁壮，您放心，中华篇章永不落幕

2024年04月10日

12718 9718

图 5-67 点 赞评论运行截图

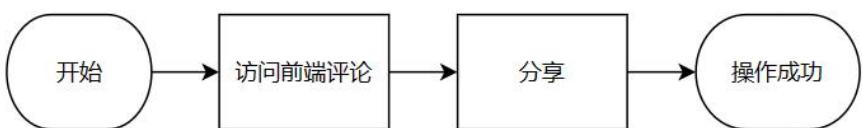


图 5-68 分 享评论流程图

评论分享功能的关键代码如下：

```

<span @click="v.shared_count++" :class="c_hover">
    <el-icon><Share24Filled/></el-icon> {{v.shared_count}}
</span>

```



图 5-69 分享评论运行截图

5.7 用户日志管理模块

用户日志管理模块分为歌曲记录、视频记录两个子模块。

5.7.1 歌曲记录模块

歌曲记录模块包含添加/修改歌曲播放记录、删除歌曲播放记录等两个功能：

- (1) 添加歌曲播放记录：已登录的用户每次请求播放歌曲，都会产生歌曲播放记录
- (2) 删除歌曲播放记录：已登录的用户被允许删除歌曲播放记录。

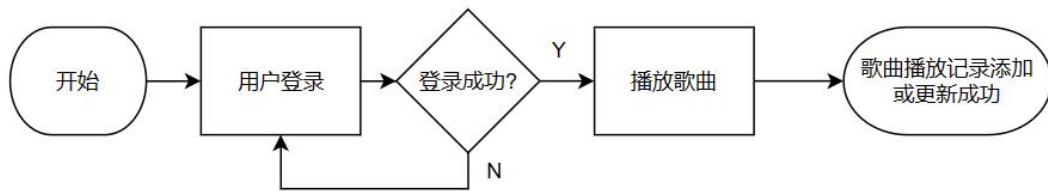


图 5-70 添加/修改歌曲播放记录

修改 song record

SongRecord object (1)

Song:	<input type="text" value="我还想她"/>			
User:	<input type="text" value="薛之谦"/>			
Count:	<input type="text" value="1"/>			
<button>保存</button> <button>保存并增加另一个</button> <button>保存并继续编辑</button>				

图 5-71 添加/修改歌曲播放记录

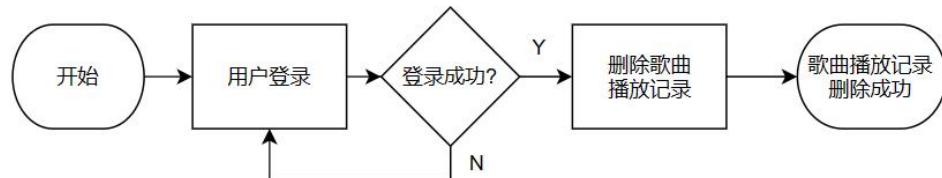


图 5-72 删 除歌曲播放记录流程图

你确定吗?

你确认想要删除 song record "SongRecord object (1)"? 下列所有相关的项目都将被删除:

概览

- Song records: 1

对象

- Song record: SongRecord object (1)

是的, 我确定 **不, 返回**

图 5-73 删除歌曲播放记录运行截图

5.7.2 视频记录模块

视频记录模块包含添加/修改视频播放记录、删除视频播放记录等两个功能:

- (1) 添加视频播放记录: 已登录的用户每次请求播放视频, 都会产生视频播放记录。
- (2) 删除视频播放记录: 已登录的用户被允许删除视频播放记录。

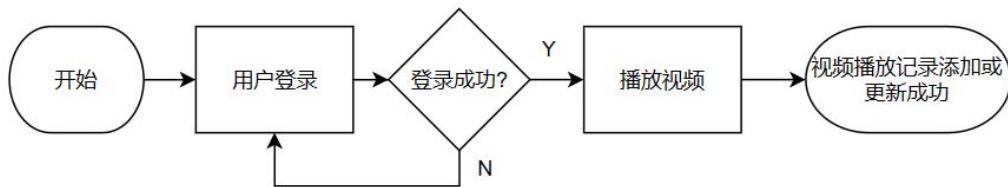


图 5-74 添加/修改视频播放记录流程图

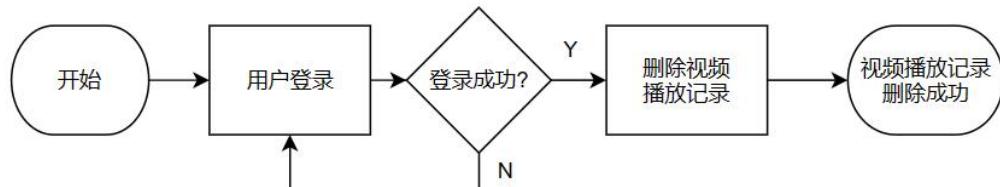


图 5-75 删除视频播放记录流程图

你确定吗?

你确认想要删除 video record "VideoRecord object (1)"? 下列所有相关的项目都将被删除:

概览

- Video records: 1

对象

- Video record: VideoRecord object (1)

是的, 我确定 **不, 返回**

图 5-76 删除视频播放记录运行截图

5.8 管理员日志管理模块

管理员日志管理模块允许具有管理员权限的用户访问 django 后端提供的后台，对系统中的数据库进行管理（即进行增删改查操作），其中管理员每次操作都会记录在 admin_log 表中。

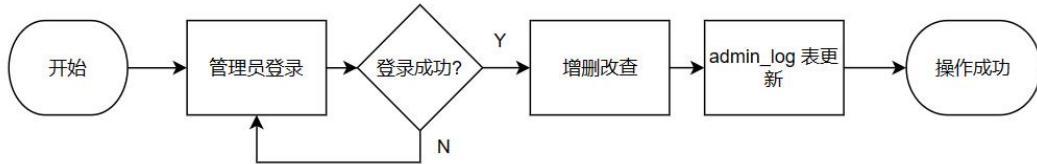


图 5-77 管理员日志管理流程图

<u>id</u>	<u>object_id</u>	<u>object_repr</u>	<u>action_flag</u>	<u>change_message</u>	<u>content_type_id</u>	<u>user_id</u>	<u>action_time</u>
1 33	陈奕迅		2	[{"changed": {"fields": ["Fans", "Birthday"]}}]	10	36	2024-04-15 08:4
2 37	李荣浩		2	[{"changed": {"fields": ["Fans", "Birthday"]}}]	10	36	2024-04-15 08:4
3 33	陈奕迅		2	[{"changed": {"fields": ["Fans"]}}]	10	36	2024-04-15 10:1
4 32	林俊杰		2	[{"changed": {"fields": ["Fans", "Birthday"]}}]	10	36	2024-04-15 12:3
5 35	毛不易		2	[{"changed": {"fields": ["Fans", "Birthday"]}}]	10	36	2024-04-15 12:3
6 49	许嵩		2	[{"changed": {"fields": ["Fans", "Birthday"]}}]	10	36	2024-04-15 12:3
7 52	per se		2	[{"changed": {"fields": ["Fans"]}}]	10	36	2024-04-15 13:5
8 282	Image object (26)		1	[{"added": {}}]	2	36	2024-04-15 13:5
9 52	per se		2	[{"changed": {"fields": ["Avatar"]}}]	10	36	2024-04-15 13:5
10 50	per se		1	[{"added": {}}]	5	36	2024-04-15 14:1
11 1	Video object (1)		1	[{"added": {}}]	14	36	2024-04-15 14:1
12 141	无穷		1	[{"added": {}}]	6	36	2024-04-15 14:2
13 149	Song object (14)		1	[{"added": {}}]	9	36	2024-04-15 14:2
14 283	Image object (26)		1	[{"added": {}}]	2	36	2024-04-15 14:4
15 284	Image object (26)		1	[{"added": {}}]	2	36	2024-04-15 14:4
16 1	PlayList object (1)		1	[{"added": {}}]	7	36	2024-04-15 14:4
17 285	Image object (26)		1	[{"added": {}}]	2	36	2024-04-15 14:5
18 2	Video object (2)		1	[{"added": {}}]	14	36	2024-04-15 14:5
19 286	Image object (26)		1	[{"added": {}}]	2	36	2024-04-15 14:5
20 142	U87		1	[{"added": {}}]	6	36	2024-04-15 14:5
21 150	葡萄成熟时		1	[{"added": {}}]	9	36	2024-04-15 14:5
22 2	Video object (2)		2	[{"changed": {"fields": ["Description"]}}]	14	36	2024-04-15 15:0
23 2	PlayList object (2)		1	[{"added": {}}]	7	36	2024-04-15 15:1
24 2	PlayList object (2)		2	[{"changed": {"fields": ["User"]}}]	7	36	2024-04-15 15:1

图 5-78 管理员日志运行截图

5.9 标签管理模块

标签管理模块规定两种登录角色可进行的操作：

- (1) 已登录的用户：允许对其上传的歌曲/视频/歌单进行附上标签或取消标签的操作。
- (2) 已登录的歌手：允许对其上传的歌曲/视频/歌单/专辑进行附上标签或取消标签的操作，同时还支持对自己的歌手账户进行附上标签或取消标签的操作。

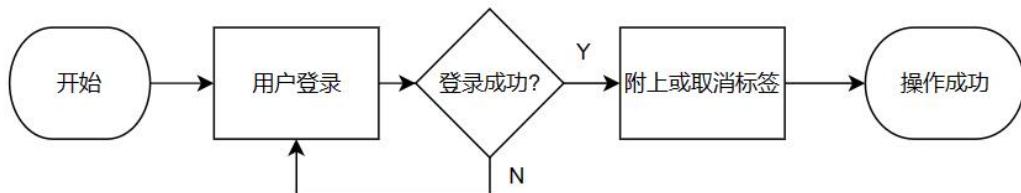


图 5-79 标签管理流程图



图 5-80 标签管理部分截图

5.10 图片管理模块

图片管理模块包含添加片、删除图片等两个功能：

- (1) 添加图片：一切可以携带创建图片副作用的请求（如：注册用户、入驻歌手）系统都会自动处理图片并将其存储于后端本地。
- (2) 删除图片：管理员账户被允许访问 django 后台管理，进行删除图片的操作。

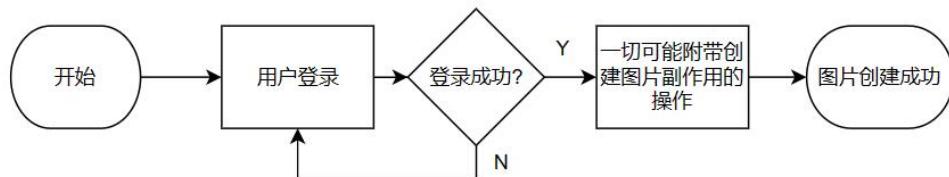


图 5-81 添加图片流程图

增加 image

图 5-82 添加图片运行代码

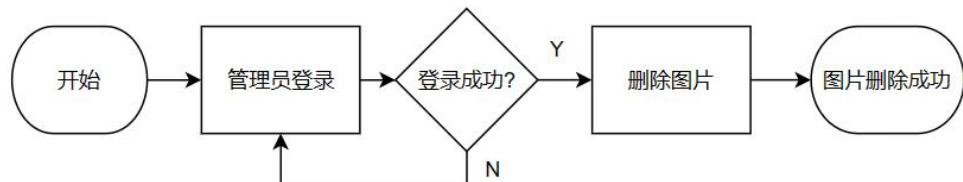


图 5-83 删 除图片流程图

你确定吗?

你确认想要删除 image "Image object (292)"? 下列所有相关的项目都将被删除:

概览

- Images: 1
- Users: 1

对象

- Image: Image object (292)
- User: travis

是的, 我确定

不, 返回

图 5-84 删除图片运行截图

5.11 数据统计分析模块

数据统计分析模块收集数据库中与歌曲/歌单/专辑/视频/评论相关的点赞数或分享数, 对其进行分析处理, 最后返回分析结果到前端。



图 5-85 数据统计分析

5.12 歌曲推荐模块

歌曲推荐模块主要根据已登录用户的偏好对用户可能想听的音乐进行预测。

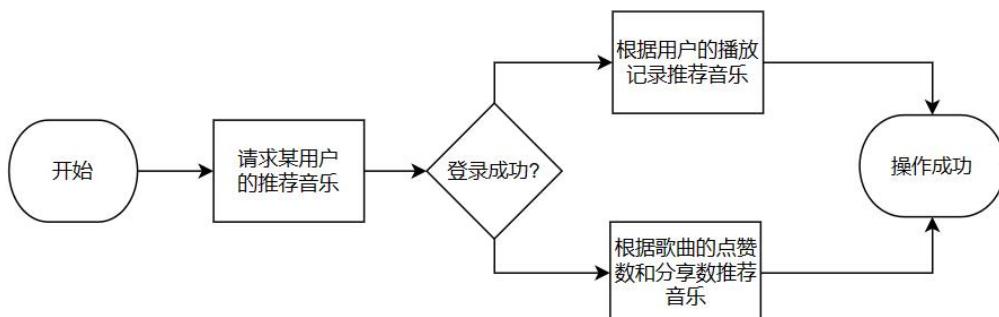


图 5-86 歌曲推荐流程图

歌曲推荐功能相关代码;

```
if util == 'e-Recommends' and req._request.method == 'GET':  
    albums = s.S_Album(m.Album.objects.all().order_by('-pub_date'), many=True).data  
    album_tag = m.AlbumTag.objects.values_list('tag', flat=True).distinct()  
    playlists = s.S_PlayList(m.PlayList.objects.all().order_by('-pub_date'), many=True).data  
    play_tag = m.PlayTag.objects.values_list('tag', flat=True).distinct()
```

```

artists = s.S_Artist(m.Artist.objects.all().order_by('-joined_time'), many=True).data
artist_tag = m.ArtistTag.objects.values_list('tag', flat=True).distinct()
# ...
return Response({
    'albums': albums[:12],
    'album_tag': album_tag,
    'playlists': playlists[:12],
    'play_tag': play_tag,
    'artists': artists[:12],
    'artist_tag': artist_tag,
})

```



图 5-87 歌曲推荐运行截图

5.13 排行榜模块

排行榜模块调用了数据统计分析模块的分析结果，返回歌曲/专辑/歌单以热门程度排序的数据集。



图 5-88 排行榜模块

排行榜模块的关键代码如下：

```

if util == 'e-Ranks' and req._request.method == 'GET':
    songs = m.Song.objects.order_by('-shared_count').order_by('-liked_count')[:10]

```

```

songs = s.S_Song(songs, many=True).data
albums = m.Album.objects.order_by('-shared_count').order_by('-liked_count')[:10]
albums = s.S_Album(albums, many=True).data
playlists = m.PlayList.objects.order_by('-shared_count').order_by('-liked_count')[:10]
playlists = s.S_PlayList(playlists, many=True).data
# ...
return Response({
    'songs': songs,
    'albums': albums,
    'playlists': playlists,
})

```



图 5-89 排行榜模块运行截图

5.14 搜索模块

搜索模块支持对歌曲、歌单、专辑、视频、歌手、用户等内容进行搜索。

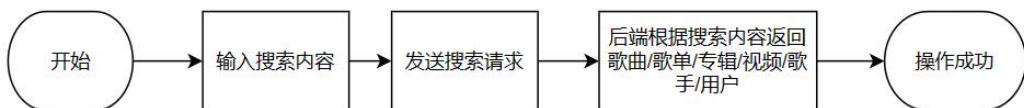


图 5-90 索模块

搜索模块的关键代码如下：

```

if util == 'g-Search' and req._request.method == 'GET':
    search = req.query_params.get('s')
    if not search:
        search = '!@#$%^&*()_'
    songs = s.S_Song(m.Song.objects.filter(Q(title__contains=search) |
    Q(album__artist__stagename__contains=search)),
    many=True).data

```

```

videos = s.S_Video(m.Video.objects.filter(Q(title__contains=search) |
Q(user__nickname__contains=search)),
many=True).data

albums = s.S_Album(m.Album.objects.filter(Q(title__contains=search) |
Q(artist__stagename__contains=search)),
many=True).data

playlists = s.S_PlayList(m.PlayList.objects.filter(Q(title__contains=search) |
Q(user__nickname__contains=search)),
many=True).data

users = s.S_User(m.User.objects.filter(nickname__contains=search), many=True).data

artists = s.S_Artist(m.Artist.objects.filter(stagename__contains=search), many=True).data

# ...

return Response({ 'songs': songs, 'videos': videos,'albums': albums,'playlists': playlists,'users':
users,'artists': artists})

```

The screenshot shows a search results page for the artist '陈奕迅'. At the top, there is a navigation bar with links like '我的', '关注', '商城', '音乐人', '云推歌', '下载客户端', a search input field containing '陈奕迅', and a '创作者中心' button. Below the navigation is a breadcrumb trail: 'index / Search'. The main content area has tabs for '歌曲', '视频', '专辑', '歌单', '歌手', and '用户'. The '歌曲' tab is selected. A table lists seven songs by Chen Yixun, showing columns for '歌名', '歌手', '点赞数' (Likes), and '分享数' (Shares). The songs are: '爱情转移' (83 likes, 66 shares), '富士山下' (10 likes, 60 shares), '淘汰' (2 likes, 21 shares), '最佳损友' (56 likes, 30 shares), '十年' (19 likes, 79 shares), and '葡萄成熟时' (42531 likes, 45531 shares).

歌名	歌手	点赞数	分享数
爱情转移	陈奕迅	83	66
富士山下	陈奕迅	10	60
淘汰	陈奕迅	2	21
最佳损友	陈奕迅	56	30
十年	陈奕迅	19	79
葡萄成熟时	陈奕迅	42531	45531

图 5-91 搜索模块运行截图

5.15 歌曲播放模块

根据系统结构设计中对歌曲播放模块的描述，其对应的流程图如下：

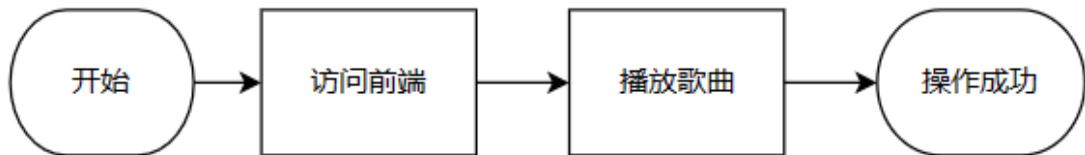


图 5-92 歌曲播放模块流程图

音乐播放模块在 web 前端的核心实现如下：

```

<div :class="[c_row, 'bg-gray-300 text-black h-full rounded-3xl justify-center place-items-center]">
<el-button round :icon="PlaySkipBackOutline"></el-button>
<el-button round :icon="icon" @click="p.flip()"></el-button>

```

```

<el-button round :icon="PlaySkipForwardOutline"></el-button>

<p class="w-3/12 ml-2">
<el-progress :percentage="p.percentage" :format="()=>(`${{p.time}}/${{p.total}}`)"></el-progress>
<p v-if="p.song" :class="[c_hover, 'mx-2']" @click="router.push('/g/Song?id=${{p.song.id}}')">
{{p.song.title}}</p>
<p v-if="p.song.artist" :class="[c_hover, 'mx-2']"
@click="router.push('/g/Artist?id=${{p.song.artist.id}}')">{{p.song.artist.stagename}}</p>
</div>

```

音乐模块模块在 PyQt 前端的核心实现如下：

```

if state != QMediaPlayer.PlayingState:
    if state == QMediaPlayer.StoppedState:
        try:
            self.player.setMedia(
                QMediaContent(QUrl.fromLocalFile(self.files_filtered[self.fileindex_prepared])))
        except Exception as e:
            return
        self.player.play()
        self.view_playbutton('pause')
        _, file = os.path.split(self.files_filtered[self.fileindex_prepared])
        self.fileindex_playing = self.fileindex_prepared
        self.view_label(file)
    else:
        self.player.pause()
        self.view_playbutton('play')

```

音乐模块模块在 django 后端的核心代码如下：

```

if util == 'g-Song' and req._request.method == 'GET':
    try:
        id = req.query_params['id']
    except Exception:
        raise ValidationError(f"GET {req._request.path} 缺少查询参数 id")
    song_ = m.Song.objects.get(id=id)
    song = s.S_Song(song_).data
    song['data'] = fhttp://localhost:8001/media{{song["data"]}}

```

```

song['img'] = f'http://localhost:8001/rest/img/?id={song["img"]}'
song['artist'] = s.S_Artist(song_.album.artist).data
song['album'] = s.S_Album(song_.album).data
song['lyrics'] = song['lyrics'].replace('\n', '<br>')
return Response(song)

```

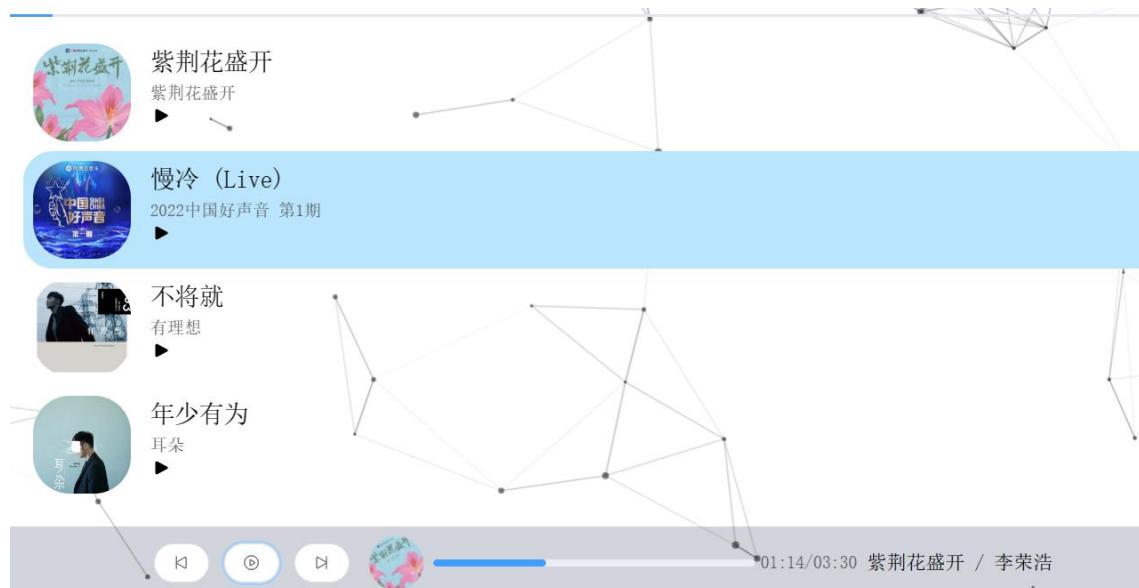


图 9-93 歌曲播放功能演示（web 前端）

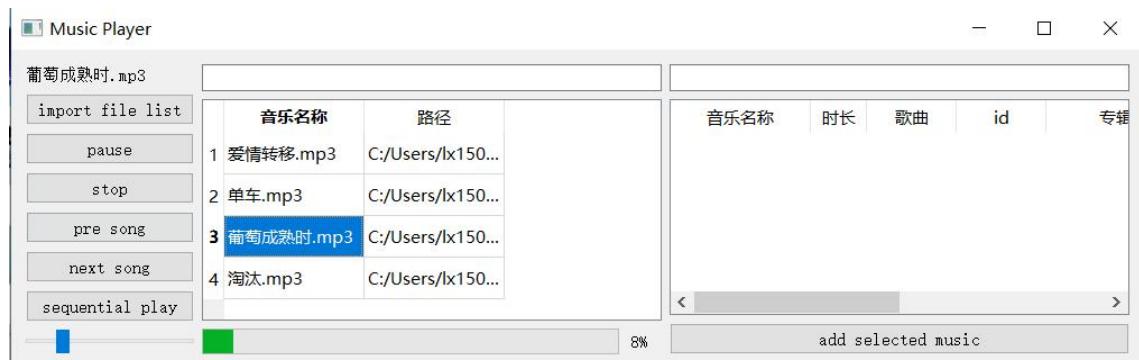


图 9-94 歌曲播放功能演示（pyqt5 前端）

5.16 视频播放模块

根据系统结构设计中对歌曲播放模块的描述，其对应的流程图如下：

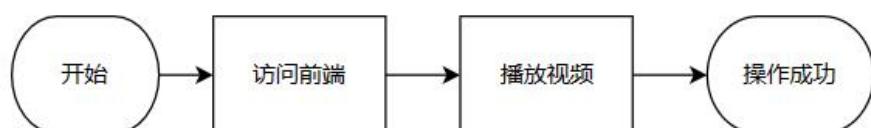


图 5-93 视频播放模块流程图

视频播放模块 web 前端的核心代码如下：

```

let data = ref([]), data_comment = ref([])
watchEffect(async () => {
  const id = route.query.id
  get('g-Video/?id=${id}').then(res => data.value = res)
  get('g-Comment/?type=video&id=${id}').then(res => data_comment.value = res)
})

```

视频播放模块 django 后端的核心代码如下：

```

if util == 'g-Video' and req._request.method == 'GET':
    try:
        id = req.query_params['id']
    except Exception:
        raise ValidationError(f"GET {req._request.path} 缺少查询参数 id")
    video = s.S_Video(m.Video.objects.get(id=id)).data
    video['data'] = f"http://localhost:8001/media{video['data']}"
    video['img'] = f"http://localhost:8001/rest/img/?id={video['img']}"
    return Response(video)

```



图 5-94 视频播放模块演示

6. 总结与展望

在本论文中，我们介绍了一种主要基于 Django、PyQt5、vue3 的跨平台音乐管理系统的
设计与实现。通过该系统，用户可以方便地管理音乐、歌手、专辑、歌单、视频等信息，实现

了用户管理、歌曲管理、播放列表管理等核心功能。

系统的 Web 前端实现中，我们主要使用 vue3、element-plus、tailwindcss 等技术设计网页端的用户界面，使用 vue3 以及 pinia 实现前端的大部分逻辑，然后使用 axios 对后端的 restful 接口的进行调用。

系统的桌面端实现中，主要采用 pyqt5 设计用户界面，使用 python 的 requests 库调用后端的 restful 接口实现相关逻辑业务的调用，最后使用 pyinstaller 对 pyqt5 的代码进行编译打包。

系统的后端实现中，主要使用了 Django 框架来实现业务逻辑功能以及对数据库的存储和交互，并且为 web 前端和桌面端开放了灵活方便且方便使用的 api 接口，以提供获取系统数据库数据、搜索歌曲、获取桌面端可执行文件等功能。

在以 django、pyqt5、vue3 为强大的技术框架支持下，本音乐管理系统才得以顺利开发完成。

7. 参考文献

- [1] 李响. 电子音乐在中西方学院教育体系中的建设与发展[D]. 北京: 中央音乐学院, 2015: 5 -95.
- [2] 杨克冉. 计算机多媒体技术应用于高中音乐创作教学的理论与实践研究[D]. 曲阜师范大学, 2024: 14-17.
- [3] 王志文. Vue+Elementui+Echarts 在项目管理平台中的应用[J]. 山西科技, 2020, 27(6): 45-4 7.
- [4] 章跃庭.Vue-Element-Admin 在广播电视台员工线上培训系统中的应用[J]. 电视技术, 2020, 42(1 2):1-3.
- [5] 王金滔.校园闲置商品交易平台的设计与实现[J].电脑知识与技术,2018,36(25):74-76.
- [6] 朱二华. 基于 Vue.js 的 Web 前端应用研究[J]. 科技与创新, 2017, 4(20): 119-121.
- [7] 麦冬, 陈涛, 梁宗湾. 轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版), 2017, 2 4(7): 58-59.
- [8] 何向繁. Axios 和 Fetch 数据传输效率分析[J]. 江西通信科技, 2023, 30(2): 27-28.
- [9] 江家龙. 基于 Vue.js 框架的餐饮 Web APP 设计与实现[J]. 科技创新与应用, 2023, 12(36): 128-129.
- [10] Pablo David Garaguso. Vue.js 3 Design Patterns and Best Practices: Develop scalable a nd robust applications with Vite, Pinia, and Vue Router[M]. Packt Publishing Limited, 2 023: 143-161.
- [11] 曾晓钰, 唐莹, 温丰蔚, 罗斌, 韦通明. 一种基于 ElementUI 的表格查询组件开发方案 [J]. 现代工业经济和信息化, 2021, 11(12): 50-51+56.
- [12] Matthew Tyson. Tailwind CSS: Learn the joys of functional, responsive CSS[J]. InfoWo

rld.com, 2021, 14(1).

- [13] Yunkai Yang, Qijia Yang, Weifeng Liu, Baodi Liu. Design of integrated interactive system for pre-diagnosis of breast cancer pathological images based on CNN and PyQt5[J]. *Multimedia Systems*, 2024, 30(2): 2-3.
- [14] 陶文玲, 侯冬青. PyQt5 与 Qt 设计师在 GUI 开发中的应用[J]. 湖南邮电职业技术学院学报, 2020, 19(1): 19-21.
- [15] Serdar Yegulalp. How to use PyInstaller to create Python executables[J]. *InfoWorld.com*, 2023, 16(1).
- [16] Xiya Yu, Xianhe Li, Changping Wu, Changping Wu, Gongyou Xu. Design and Deployment of Django-based Housing Information Management System[J]. *Journal of Physics: Conference Series*, 2023, 2425(1): 1-10.
- [17] 余斌. 基于 Django 的用户信息管理系统[J]. 电脑知识与技术, 2021, 28(5): 89-90.
- [18] Kruthika Alnavar, R Uday Kumar, C Narendra Babu. Document Parsing Tool for Language Translation and Web Crawling using Django REST Framework[J]. *Journal of Physics: Conference Series*, 2021, 1962(1): 1-9.
- [19] 徐红梅. 解析 SQLite 在 Python 中的应用[J]. 内江科技, 2023, 30(4): 60-61.
- [20] 于斌, 陆旭, 田聪, 等. 面向 SQLite3 数据库 API 调用序列的并行运行时验证方法[J]. 软件学报, 2022, 33(8): 2755-2768.
- [21] J G Arévalo, L Viecco, L Arévalo. Methodology to define an integration process between frameworks SCRUM, Django REST framework y Vue.js, implemented for software development, from quality management approach and agility[J]. *IOP Conference Series: Materials Science and Engineering*, 2020, 844(1): 1-10.

8. 致谢

想起整个写作的过程，我深感其中的辛苦与快乐，以及与众多的共同努力和支持。在此，我想向他们致以最诚挚的谢意。首先，我要感谢我的指导老师李振宏副教授。在整个研究过程中，他给予了我无私的指导和支持，不仅给予我宝贵的建议和意见，还悉心为我解答了无数的疑惑。他的严谨治学和对学术的追求，激励、促进了我的成长，让我在学术道路上更加坚定。我将一直铭记在心，对他的教诲会使我受益终生。

在研究过程中，我借鉴和参考了众多前辈的研究成果和学术文献。在此向这些学术先知们表示由衷的敬意，感谢他们的智慧和无私分享，正是他们的研究为我提供了了解问题背景和思考的思路。

其次，我要感谢实验室的所有同学和朋友们，他们在我论文写作的过程中提供了很多宝贵的帮助和支持。感谢他们在研究中的讨论、互相分享和激励，让我思维更加开阔，受益匪

浅；感谢他们在我写作过程中的指正和修改，使我的论文更加完善。

此外，我要感谢我的家人和朋友们，在这篇论文的背后他们一直默默地支持和鼓励着我。他们给予了我无尽的关爱和理解，无论在何时何地，他们的支持都是我坚持不懈、追求卓越的动力所在。感谢他们的陪伴和支持，没有他们，我无法顺利地完成这篇论文。在未来，我将继续努力学习，不断提升自己的能力。我将以这篇论文为起点，坚持不懈地追求学术的进步和创新，为科学事业的发展贡献自己的力量。谨以此文，致以对所有支持和帮助过我的人们的深深谢意。谢谢！

最后，感谢提供 VCS 托管服务的开源社区 github 的支持，以及 github 热门开源框架（django/vue/pyqt5/pinia/axios/tailwindcss...）的技术支持，我才得以顺利进行设计并完成音乐管理系统的开发。