



南華大學船山學院

# 毕业设计(论文)

题    目 基于 Springboot+ElasticSearch  
的 vlog 共享平台

专业名称 软件工程

班    级 船本 20 软件 04 班

学    号 20209350401

学生姓名 尹建栋

指导教师 蒋良卫

职    称 讲师

2024 年 5 月 12 日

## 南华大学船山学院学位论文原创性声明

本人声明，所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了论文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南华大学或其他单位的学位或证书而使用过的材料。与我共同工作的同志对本研究所作的贡献均已在论文中作了明确的说明。本人完全意识到本声明的法律结果由本人承担。

作者签名：尹建彬

2024 年 5 月 12 日

## 南华大学船山学院学位论文版权使用授权书

本人同意南华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以编入有关数据库进行检索，可以采用复印、缩印或其它手段保留学位论文；学校可根据国家或湖南省有关部门规定送交学位论文。对于涉密的学位论文，解密后适用该授权。

作者签名：尹建彬

2024 年 5 月 12 日

导师签名：张良包

2024 年 5 月 12 日

---

## 基于 Springboot+ElasticSearch 的 vlog 共享平台的设计

**摘要：**视频分享平台在社交媒体生态中的地位越来越重要，随着数字化时代的不断发展设计实现功能丰富的视频上传平台，以满足多功能需求为出发点。该平台除具备视频上传播放的基本功能外，还包含了一系列功能，如搜索首页，注册登录，弹幕评论，视频审核，视频管理，私信等。这些功能的整合，让用户可以在平台上进行视频的全方位分享，沟通，管理等各个环节的工作。一是通过用户管理、内容管理、交互功能、审核管理等平台基本功能模块的深入分析，确定了用户需求。二是对后续技术实现有清晰的指导和方向可供参考。另外，还采取了包括Minio存储在内的先进的技术手段，保证了视频上传和播放的高效性和稳定性，以及在个性化搜索主页和注册登录系统的帮助下，为用户提供了便捷的体验，并加强了用户信息的安全管理。关于用户交互的问题，通过引入WebSocket实时消息传输技术，为平台增加了弹幕评论以及私信功能，使用户之间的互动性得到了很大的提升。最后，系统测试和用户反馈对平台的稳定性与可用性进行了验证，为平台的进一步改进与扩展奠定了基础。本文所提出的设计与实现方案为视频上传平台的发展提供了有益的借鉴与指导，具有十分重要的指导意义与实用价值。

**关键词：**视频上传；播放；审核；管理；弹幕；互动

---

# Design of Vlog Sharing Platform based on SpringBoot and Elasticsearch

**Abstract:** The position of video sharing platforms in the social media ecosystem is becoming increasingly important. With the continuous development of the digital age, the starting point is to design and implement a video upload platform with rich functions to meet multifunctional needs. In addition to the basic functions of video upload and playback, the platform also includes a series of functions, such as search homepage, registration and login, bullet comments, video comments, video management, private messaging, etc. The integration of these features allows users to fully share, communicate, and manage videos on the platform. Firstly, by conducting in-depth analysis of the basic functional modules of the platform, such as user management, content management, interaction function, and review management, the user requirements were determined. The second is to provide clear guidance and direction for the implementation of subsequent technologies, for reference. In addition, advanced technologies including Minio storage have been adopted to ensure the efficiency and stability of video uploading and playback. With the help of personalized search homepage and registration login system, it provides users with a convenient experience and strengthens the security management of user information. In terms of user interaction, by introducing WebSocket real-time message transmission technology, the platform has added bullet comments and private message functions, greatly improving user interaction. Finally, system testing and user feedback verified the stability and usability of the platform, laying the foundation for further improvement and expansion of the platform. The design and implementation scheme proposed in this article provides useful reference and guidance for the development of video upload platforms, and has important guiding significance and practical value.

**Key words:** Video uploading; Playback; Review; Management; Barrage; Interactio

## 第一章 绪论

### 1.1 研究背景

在这个信息泛滥的时代，视频已经成为人们获取信息、消遣娱乐的首选方式。视频分享平台仿佛是一片广袤的土地，为用户提供了广阔的创作空间和交流平台。然而，传统的视频分享平台却只是在这片土地上生长了一些零星的绿草，远远无法满足用户日益增长的需求。

用户对于视频分享平台的期望越来越高，他们渴望在这个开放的空间里尽情驰骋，发现那些令人心动的视频，与他人分享心得，深入交流。然而，现有的平台往往无法提供足够丰富和多样的内容，也缺乏有效的交流和互动机制。

因此，我们急需设计一个功能丰富、充满活力的视频上传乐园，以满足用户的迫切需求。我们深入了解了用户的需求和期待，从而建立了一套完善的功能体系。用户管理系统可以让用户轻松管理个人信息和喜好，从而获得更加个性化的推荐和服务。内容管理系统确保平台上的内容质量和多样性，为用户提供丰富的选择。互动功能让用户可以与视频创作者和其他观众直接交流，分享心得和体验。审核管理系统则保证了平台内容的合法性和安全性，让用户可以放心使用。

通过这些功能的完善，我们为用户打造了一个开放、创新、安全的视频分享乐园，让他们在其中尽情享受创作和交流的乐趣。

具体地说，基于 SpringBoot+ElasticSearch 的 VLOG 共享平台已经实现了以下功能。

#### 视频上传功能：

视频上传的界面，就像是等待用户创作的一块广袤的画布。在这个空间里，有轻轻的上传视频文件、绚丽绽放的视频封面、随意选择的视频分区、清晰呈现的视频名称和简介。在上传的时候，分片操作就像细密的纹路样，编织出神奇的断点续作。完成上传后，将分片合并，为完整的视频资料保驾护航。在上传过程中闪动，引领用户前进；而上传失败的处理方式，则是为用户解除焦虑的安抚音符。

#### 视频播放功能：

在页面中央的 Video 播放器更是引领用户的眼球。四周如星光点缀般的控制按钮和显示区域，如诗中律动般的播放/暂停，音量控制，进度条时长显示，为观者指

点迷津。播放/暂停按钮:用来控制播放和暂停视频的功能。音量控制键:对视频音量大小进行调节。播放进度条,就像是一条流动的时间轨迹,将视频的进程细细勾勒,让用户想怎么拖就怎么拖,或勾起回忆,或朝着未来的方向快速前进。像变幻戒一样的倍速播放按钮,轻轻一触,就能改变时间的流逝速度,为观者打造出千姿百态的别样观影体验。

#### **视频审核功能:**

你上传得视频不能直接被大家看到,它得先经过一个管理系统。然后管理员会来看这个视频,检查一下。如果管理员觉得没问题,那大家就能看到了。所以,耐心等待一下,审核通过了,你得视频就能和大家见面了。

#### **视频举报功能:**

您希望观看哪一段视频,进行点击就能进入到举报的接口里。从“涉及侵权”到“内容有伤风化”,你会看到一系列的举报理由供你选择。当然,如果欲言又止,也可以把备注资料填进去。那么这些报道的信息,一般都会传递到我们的加工队伍当中去。他们可是超级强大的,对每一个报道都会仔细审核,保证健康积极的平台内容。最后,请一定不要忘记,在完成举报的处理过程之后,将很快收到一条温馨的反馈消息来告知处理的结果——是“已处理”还是“无需处理”也都会有明确的答复的。快速的处理反馈机制是为了防止不良内容困扰的必不可少的一环——让你轻松应对不良内容的问题。

#### **弹幕功能:**

用户沉浸于有趣视频片段中时,会因心潮澎湃而大放厥词,各抒己意的文字从指缝间飞舞而过,如流水般传递着内心深处的思绪,从天南海北传递到每一个正在观看的人的眼前。如果弹幕像雨点一般密集,会扰乱观影的宁静,这时用户只要轻轻一按。就可使这片弹幕的喧嚣与杂扰得到关闭,重归一片静谧的片刻。让观影体验更上一层楼,同时也增加了互动性和趣味性。

#### **点赞关注功能:**

在每个视频得脚下,会有一个点赞按钮,点赞过后会有颜色和大小上的变化。

#### **搜索功能:**

搜索使用了门面模式、享元模式,使用 ES 进行模糊搜索,速度非常快。

私信功能:提供用户可查看并发送私信的私信页面或私信对话框。界面上有一

份私信列表，让大家对能聊到的人一目了然。你想私信给谁，就把他选在列表里，把你想说的话输入进去，点到为止，就可以发送啦！私信内容会在它的私信界面中发送给目标用户并显示出来。平台还具备保证大家上传观看的内容安全优质的视频审核管理功能。这样一来，大家在使用平台的时候就比较放心了。而且我们也做了系统的测试，也做了用户的反馈，证明这个平台是很稳定的，也是很好用的一个平台。这就为以后平台做的更好，做的更多的功能奠定了一个比较扎实的基础。

## 1.2 课题开发意义

基于 SpringBoot 和 Elasticsearch 开发 VLOG 共享平台是重要的举措。这个可以为用户带来视频分享、传播等多元化体验的平台。视频已经成为当今数字化时代人们主要的娱乐方式、学习方式和社交方式。因此,为用户轻松上传、分享、观看 VLOG 内容提供了一个便捷的平台,这将使其视频娱乐选择得到极大的扩展为满足广大用户的愿望增添娱乐乐趣。

同时,该平台还可以通过用户上传的丰富内容,促进社会文化的传播与交流,为其他用户提供学习、娱乐的机会。在这个信息爆炸的时代,视频内容的分享和传播已经成为人们获取各种资讯和知识的迫切需求。因此,一个能够满足用户学习需求、促进社交文化传播与交流、为繁荣社交文化做出积极贡献的内容丰富的 VLOG 分享平台。最终,这个平台通过各种社交功能编织了一张用户与用户沟通的网。让用户的社交体验丰富起来,评论、点赞、私信等功能就像一双翅膀。这些功能在促进用户互动的同时,也为增强用户的归宿感和参与感,构建了更为紧密的社区联系。在虚拟社交时代,人们渴望与他人接触,分享内心所想所感,而这个平台,就是自己内心的港湾。因此,满足用户社交需求,提升用户归属感和参与感,提升用户体验,这样的 VLOG 分享平台社交功能丰富,社会意义和市场前景都是正面的。

## 1.3 课题现状与发展趋势

### 1.3.1 现状

我国内容行业正经历“视频转向”趋势,中视频逐渐崭露头角。自 2020 年,中视频内容趋向多元化、专业化和精品化,成为新得文娱产业热点。随着长短视频红利饱和,互联网平台寻求新突破口,中视频成为热门领域。西瓜视频率先提出中视频概念,与长短视频形成区隔。中视频时长为 1 至 30 分钟,适合讲述完整故事,呈

现丰富视觉信息。知乎、小红书等平台推出扶持政策和专区，降低创作门槛，进入内容混战时代。长、中、短视频互补，创作者数量与质量、持续生产能力、商业化变现能力成为视频平台发展得关键。

### 1.3.2 发展趋势

我国视频行业面临挑战，主要是创作者数量不足导致内容质量参差不齐。各大平台通过优厚条件吸引优质创作者，但短剧普遍存在同质化、粗糙化问题。盗版和侵权问题严重，影响整个视频市场秩序。中视频作为长短视频创作者得新舞台，具有巨大发展潜力。5G、人工智能和大数据技术得发展将降低中视频制作门槛，促进优质内容产出。内容策划能力将成为平台得核心竞争力。采用“社交+算法”双驱并行得分发模式有望优化分发模式，减少对算法推荐得依赖。

## 1.4 本文组织结构

本文一共六章，具体安排如下所示。

第一章：绪论。绪论部分首先分析当前视频平台各方面的问题，对研究背景进行深刻的分析后，提出了当前该课题十分具有发展前景与研究意义。

第二章：相关技术介绍。本章主要介绍了 Vlog 视频系统所用的主要的框架。本系统后端采用 SpringBoot<sup>[1]</sup>。前端采取 Vue 框架，并在后端利用 Java 语言进行开发，利用 MyBatis 框架进行数据交互，数据库系统选取 MySQL 数据库系统。

第三章：系统需求分析与概要设计。本章主要明确系统的用途以及用户的需求，把握系统开发的方向，明确要开发具有哪些功能的系统，其次进行概要设计，确定系统总体设计与总体架构，确定数据库的概念结构与逻辑结构。

第四章：系统详细设计与实现。本章节涉及到系统的详细设计，依据上一章节的模块划分明确各个模块主要实现的功能以及细节加以设计，同时将数据库设计完整，对系统的实现做出整体性描述。

第五章：系统测试。本章首先对系统进行较为全面的功能测试，记录了功能测试各个用例的结果并对测试结果进行分析。

第六章：总结与展望。本章概括全文内容，分析其优缺点，并提出了未来工作的改进与发展方向。



## 第二章 相关技术介绍

### 2.1 开发框架

#### 2.1.1 SpringBoot

SpringBoot 这个开源框架是专门用于简化 Java 应用开发的基于 Spring 的，它帮助开发人员省去了很多配置上的麻烦，以自动配置和“约定优于配置”的原则使开发更加快捷、简便、高效。SpringBoot 拥有自动配置、启动依赖、内嵌式容器等诸多强大功能。它还内置了 Tomcat、Jetty、Undertow 等常用的 Servlet 容器，开发者可以直接将应用打包成可执行的 JAR 文件，然后直接运行在内嵌的 Servlet 容器中，省去了部署的麻烦。

#### 2.1.2 Redis

redis 是个数据存储的高手，就在内存中保管着它的身影。它是一个多面手，既可作为缓存使用，又可以充当消息队列的中枢，又可以对数据进行长期存储在内存中。redis 的运行速度极快而且持久性很强，扩展性也很好<sup>[2]</sup>。经常把它用来缓存经常被访问的数据或实时对数据进行加工分析处理和充当消息队列的中枢等。

#### 2.1.3 Docker

最开始，人们用 LXC 来实现，后来又搞出了 Libcontainer，再后来又进化成了 Container<sup>[3]</sup>。而 Docker 呢，它就像是给容器穿上了一层漂亮的外衣，让容器的创建和维护变得超简单。比起虚拟机，容器真得是轻如鸿毛，快如闪电。

#### 2.1.4 MySQL

MySQL 是一种广泛应用于 Internet 应用领域的开源关系型数据库管理系统。MySQL 是一个与客户端/服务器模式相同的数据库管理系统。它可以运行在多种操作系统和编程语言上，尤其好用，可靠，对各种情况也能做到应有尽有。简单的安装和配置，也支持各种想怎么用就怎么用的存储引擎和数据类型。不仅如此，MySQL 的 web 可视化安全控制在密码保护、数据加密、权限控制等方面也做得尤为出色，使得数据能够做到安全稳定。在处理海量数据时，即使很多人同时访问，也能在分布式数

数据库和集群系统中使用，数据是安全的。总的来说，MySQL 是款性能高，可靠性强，而且特别灵活的数据库管理系统，既好学又好用<sup>[4]</sup>。

### 2.1.5 WebSocket

WebSocket 是一种在 Web 应用程序中实现全双工通信的协议，能够在客户端和服务端之间建立持久连接，实现实时通信和数据传输<sup>[5]</sup>。

### 2.1.6 Elasticsearch

个人通俗的认为，ElasticSearch 是一个已经分词的并且有索引的模糊查询引擎。

### 2.1.7 Rabbitmq

RabbitMQ，迎接消息流动的大门。作为开源消息队列服务软件，它诞生于 LShift，是 AMQP 的开源实现之一<sup>[7]</sup>。以其 Erlang 的高性能、健壮性和可伸缩性而著称，RabbitMQ 如同坚实的堡垒，承载着信息的沟通。

### 2.1.8 Canal

他是对数据库增量变化进行监听和消费的一个开源工具，由阿里巴巴开发。简单点说就是它用来对数据库中的数据变化进行监控和通知其他系统，从而将数据变更同步到目标系统中去。对数据库中的数据变化进行监听和消费支持对 MySQL 等关系型数据库进行使用；并且可以将数据变更同步到消息队列等目标系统中去。对数据库的增量变化进行监控和消费是一项十分实用的功能。

### 2.1.9 Vue

VUE.JS 是为打造优秀的用户界面和单页应用而打造的轻量级 JS 框架 Vue.js 相对于其他框架而言，它的优点是上手容易，学习曲线陡峭，性能好，文档简洁清晰，所以成为了很多开发者的首选框架之一<sup>[8]</sup>。此外，Vue.js 还有一个庞大的社区，开发者可以通过该社区获得与 Vue.js 相关的开发工具、插件、库、教程、实例等信息，从而在大幅降低 Web 应用搭建难度的同时，提高开发效率和质量。总之，Vue.js 的出现对开发 Web 应用有很大的促进作用，提供了更卓越的开发体验，编程效果也更高。

### 2.1.10 Ant Design Vue

这是众多组件库的一种，有很多封装好的组件可以直接拿来用。

## 2.2 开发语言

Java 是一种面向对象编程的语言，在 1995 年 5 月，由 Sun 公司推出的 Java 面向对象程序编程语言以及 Java 平台获得业内人士的一致好评。Java 语言结构严谨、语法简洁、功能强大，Java 可移植性强，还具有支持多线程的特点。

### 2.2.1 MySQL 数据库

MySQL 是关系型数据库的代表之一，其改变了将全部的数据存放在一个“仓库”中的策略，直接把数据存储在不同的数据表中，在数据库中查找时能够明显提高速度，并且使得数据库更加灵活。相较于同类型的其他关系型数据库系统，MySQL 虽然有一些缺陷，但是由于 MySQL 具有可移植性强、成本低、速度快等等特点，一般在开发中通常应用于中小型的 web 系统。

### 2.2.2 IDEA

众所周知，IDEA 是 java 开发的过程中的常用软件，简化了开发人员在编写代码、测试阶段的工作，作为一个拥有许多的插件的自由集成开发环境，通过各种插件可以拓展其功能，其相比于其他的 IDE 更加具有更高的灵活性。

## 2.3 本章小结

本章主要介绍了 Vlog 共享平台所用的主要的框架。本系统后端采用 SpringBoot，前端采取 Vue 框架，并在后端利用 Java 语言进行开发，利用 MyBatis 框架进行数据交互，数据库系统选取 MySQL 数据库系统。

## 第三章 系统需求分析与概要设计

### 3.1 系统目标

我们的平台旨在成为一个方便和简单的中心，为用户提供一个直观的界面来探索、共享和连接。无论是捕捉日常时刻还是重要里程碑，我们的系统都能让用户毫不费力地记录他们的生活和经历。通过提供易于使用的共享和互动功能，我们旨在培养一个个人可以有意义地参与彼此故事和庆祝活动的社区。从关注朋友的冒险经历到发现与志同道合的人的新联系，我们的平台鼓励积极、鼓励和相互欣赏的文化。通过无缝导航和强大的沟通工具，我们让用户能够围绕共同的兴趣和激情建立自己充满活力的社区，创造一个包容的空间，在这里每个人的声音都受到重视。

### 3.2 可行性分析

#### 3.2.1 技术可行性

Spring Boot 作为后端开发得主要框架，Spring Boot 提供了快速构建和部署得能力，能够有效地提高开发效率和系统稳定性。Redis：作为缓存和消息队列，Redis 可以用于缓存视频数据、可用于视频得分片上传，在弹幕流量大时，可以直接走缓存提高运行效率，在 RabbitMQ 得配合下是大量得弹幕有序运行，从而防止系统崩溃并且增强用户体验。Elasticsearch，模糊查询速度非常快。它为用户提供了快速、准确得搜索和过滤功能。而 WebSocket 可以使用户面对面交流一般，快速地即时通讯。

#### 3.2.2 经济可行性

SpringBoot、Redis、Docker、Rabbitmq、Canal、Vue、Ant Design Vue、ElasticSearch、MySQL、WebSocket，这些技术栈都是免费开源的，这意味着无需支付高昂得许可费用，大大降低了开发成本。本系统为本人独立开发，无需支付额外人力费用，且服务器为本地搭建，无需支付云服务器得费用。

#### 3.2.3 操作可行性

SpringBoot 可以帮你处理文件上传，而 ElasticSearch 则可以帮你对这些多媒体内容进行存储和搜索。两者结合，让您更有效率、更灵活地管理您的视频内容。用户只需通过简单的操作就可以将自己的视频上传，在平台上就可以实现视频的高

效观看。视频审核举报功能能够保证平台内容的质量和安全性，而 SpringBoot 的后端逻辑能够实现审核机制。这些功能在现有得视频分享平台中普遍存在。ElasticSearch 可以实现搜索接口，而 Vue 和 Ant Design Vue 可以实现搜索页面和主页得前端设计。私信功能可以让用户之间进行私密得交流和互动，Spring Boot 可以实现后台逻辑，WebSocket 技术可以实现实时通信，Vue 和 Ant Design Vue 可以实现前端界面，可行性较高。

### 3.3 需求分析

确定了 Vlog 视频平台可行性后，在系统开发之前，我们先不去计较具体实现的方法和细节，而是首先要明确该系统是要拿来做什么，确定系统用户人群对该系统的功能需求、性能需求等等。

基于 Springboot+ElasticSearch 的 vlog 共享平台的最基本的需求概述如下：

(1) 视频上传：功能包括选择要上传得视频文件、填写相关信息（如标题、描述等）以及提交上传请求。上传得视频将被存储在平台得服务器上，并在审核通过后显示在平台上。

(2) 视频播放：用户可以观看其他用户上传得视频内容。

(3) 视频审核：对视频内容的审查是一项十分重要的任务，对上传者是否违反平台规定起着举足所在。那些能够通过审核的视频将会像清晨的阳光一样在平台上熠熠生辉；而不能够通过审核的视频可能会被拒绝在平台之外，或者被标记有不适宜的烙印。所以平台管理员们一定要像一位守护者一样认真审查每一个视频的上传情况。

(4) 视频举报：用户在平台上有权对不良视频内容进行举报，向平台管理员提交有关举报请求并提供必要的证据或描述等资料进行审核后采取相应措施处理违规视频内容。

(5) 弹幕：用户在观看视频的时候可以进行弹幕的发送，实时的和其他用户进行沟通和互动。弹幕功能可以让用户在视频播放时，在视频播放器上方以动画的形式显示文字消息，让用户能够实时了解其他用户的评论以及反馈信息。

(6) 评论：用户可以就视频内容发表自己的观点和看法，在视频页面的下方进行评论。Review 功能可以让使用者输入文字内容后发布到影音页面上，供其它使用者观看及回复。

(7) 点赞收藏:用户可以点赞或者收藏喜欢的视频,方便以后再看,也可以分享。点赞和收藏功能可以让用户在用户的个人资料中,点击相应的按钮,就可以将这些操作记录下来,表达自己对视频的喜爱或者感兴趣。

(8) 私信:可以向作者发送消息进行交流,和喜欢得人进行聊天。

### 3.3.1 功能需求

本视频共享平台主要面向喜欢分享视频内容的用户,目的是让用户能够方便地上传、浏览和交流各种视频作品,系统的使用者大体上可以划分为两大类:用户、系统管理员,每个角色的介绍以及对于系统的使用权限如下表 3.1 所示。

表 3.1 系统使用者角色介绍与权限表

角色	介绍	权限
用户	该角色拥有登录系统的账号,可以使用系统的主要功能,但是对于系统的部分功能使用受限	修改个人信息 浏览系统主页 发布作品 点赞 评论 收藏 关注 弹幕 作品管理 观看记录 作品操作管理 私信 视频举报 评论举报
系统管理员	该角色拥有本系统的全部使用功能,作为系统的管理人员	用户管理 增删改查 视频标签管理 增删改查 视频审核 视频信息管理 增删改查 关注管理 增删改查

		喜欢管理 增删改查
		评论管理 增删改查
		评论举报管理 增删改查
		视频举报管理 增删改查

### ● 用户用例图

用户使用电话号码验证码登陆过之后，除了浏览基本的视频信息之外，还能够上传视频，根据自己的喜好和兴趣见闻，上传属于自己的视频；并在个人信息这里看到自己上传的视频信息状态，是否上传成功、是否通过审核、视频状态是否正常。用户还可以对自己喜欢的视频进行操作，对视频点赞表达自己对视频的喜爱，收藏视频对视频反复观摩，关注作者催更更加优秀的作品，也可以在评论区和弹幕区发表自己对视频的看法。如果观看了视频，突然退了出去，然而还意犹未尽，还可以在观看记录查看自己管看过的视频。如果视频有令人不舒服的地方，也可以对视频进行举报，如果看到嚣张的言论也可以进行举报，保证本平台的干净稳定，积极向上的一面。用户的用例图如下图 3.1 所示。

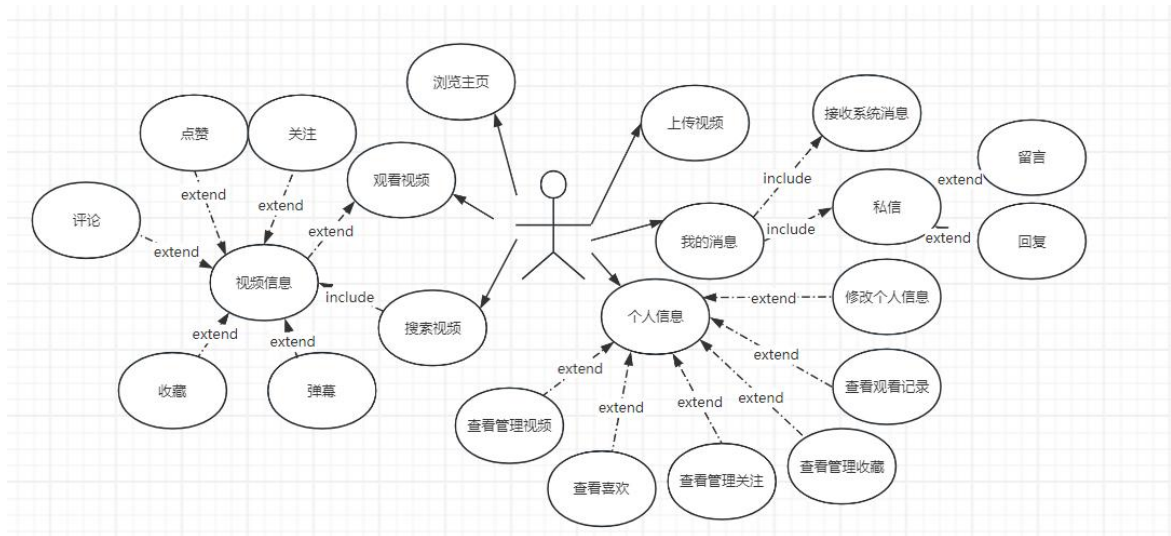


图 3.1 用户用例图

### ● 系统管理员用例图

管理员拥有极高的权限，可以直接管理用户，管理用户视频，用户的操作，对举报进行处理，对视频标签进行动态管理等。除此之外，系统管理员还能够对视频状态进行直接修改，审核视频。管理员的管理功能本质上是对各项信息进行增删查改，部分管理功能详细介绍如下：视频管理：视频标签可添加、删除、编辑视频得类型，动态得去管理视频得分 类；想知道视频得全部信息，没问题，视频信息

功能帮你搞定，你可以查看视频 得所有信息，还能进行相关得操作，简单又方便。

举报管理：视频举报通过标签筛选可以查找出正常视频、被举报视频、违规视频，筛选出违规视频，对违规视频进行观看，然后进行审核，违规得视频会被下 架；如果发现不当得评论，可以举报它。举报后，可以查看举报得内容。如果被确认违规，这些评论就会被删掉本系统的管理员的用例如下图 3.2 所示。

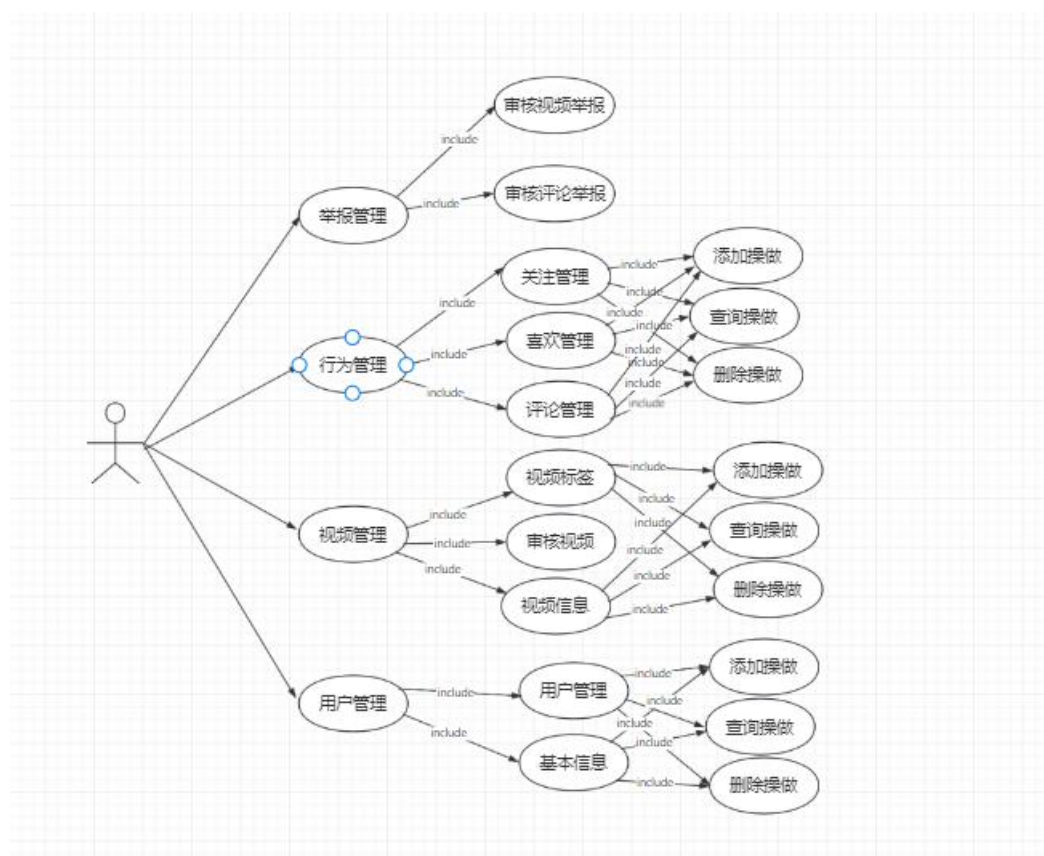


图 3.2 系统管理员用例图

### 3.3.2 非功能需求

一般而言，最后的系统除了要实现所有的功能需求点，系统用户的使用感和体验感直接反映了系统性能的好坏，一般而言，用户对系统性能也会存在一定的要求，如：

(1) 界面简洁，主题明确。系统前端设计时应当以用户为中心，本系统针对人群主要为社区空巢老人，所以应当贯彻绿色健康的理念，色调不应该太浓烈。

(2) 响应时间。对于响应式的交互系统，一般要求系统响应效果不能让用户体会到明显的延迟感，此类普通式响应系统的响应时间一般在 1 秒左右。



(3) 安全性。本系统采集了用户的部分私人信息并且涉及到药物销售的交易行为，所以极有必要保证信息的隐蔽与安全，只有被授权用户即拥有系统账号密码的用户只能看到和修改权限范围内的信息。

(4) 可靠性。由于系统在一定的环境下发生错误的原因数不尽数，为了使得用户能够使用上一个可靠的系统，可靠性需求分析阶段系统的开发人员必须要弄清楚系统出错的原因，前期预防缺陷，后期消除缺陷并能够实现预报缺陷。

(5) 可扩展性。信息迭代周期缩短、功能需求不断增大等等众多的因素，都对系统的可扩展性提出了要求，在系统设计时必须要考虑清楚哪些是不可变的，把不可变的逻辑沉淀到系统的核心逻辑中去，预留出接口作为日后的新增功能点的入口，保证系统扩展的便捷性。

## 3.4 概要设计

### 3.4.1 总体设计与架构

经过了需求分析阶段明确了本系统的功能导向之后，可以确定系统大体的架构。就整个系统而言，可以将系统架构划分为用户层、表示层、控制层、业务逻辑层、数据层。用户请求的完整流程为：用户通过点击系统界面触发事件，将用户的请求发送至系统控制器，接着由业务逻辑层与数据库的交互获取用户请求信息，最后逐步向上层反馈。

### 3.4.2 功能模块

我们不从角色的角度出发，让我们来看一下功能的划分，本平台为用户划分了视频展示模块、视频搜索模块、视频上传模块、点赞等功能模块、弹幕模块、评论区模块、系统通知模块、私信模块、信息管理模块、视频管理模块、关注管理模块、收藏管理模块、视频记录模块、举报模块、举报管理模块、视频审核模块、视频管理模块、用户管理模块、视频操作管理模块、视频标签模块。

本系统的使用角色对于系统功能的使用权限不同，为了明确各类型角色对系统功能模块的使用情况，具体的模块按照用户角色类型进行划分，因此计划按照如下图3.3所示角色划分功能模块。

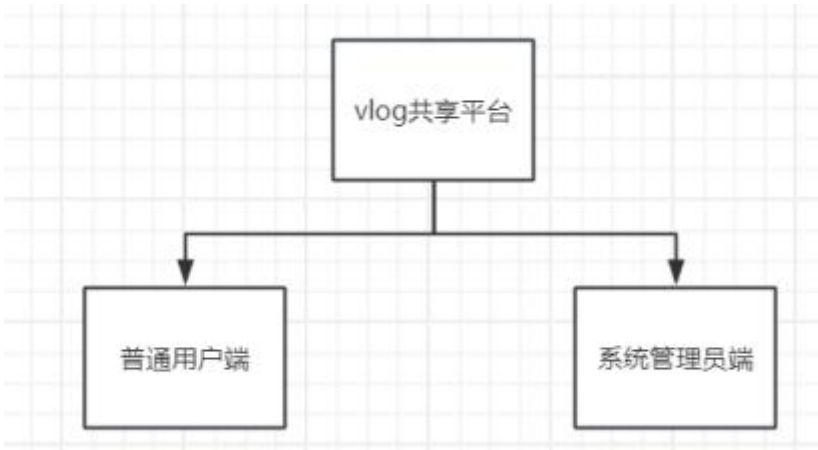


图 3.3 功能模块

(1) 普通用户端

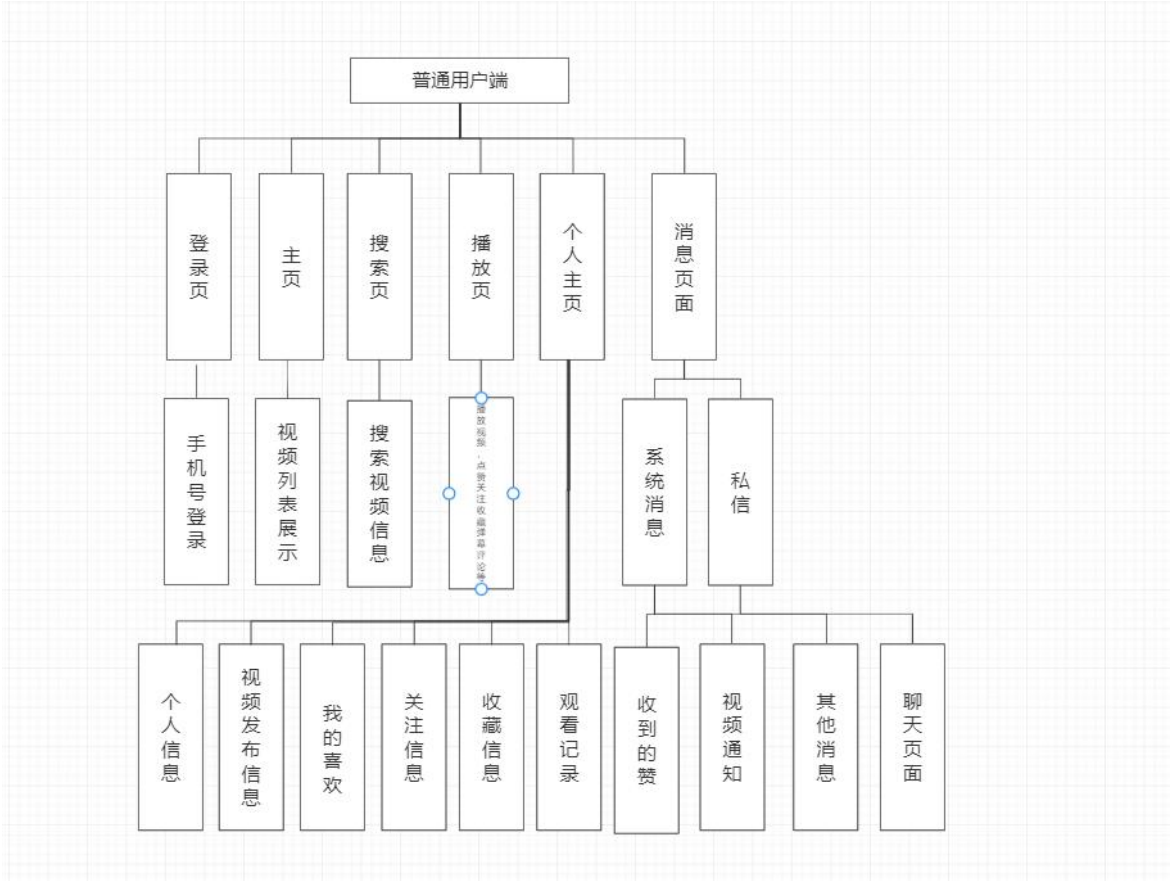


图 3.4 普通用户端功能模块

(2) 系统管理员端

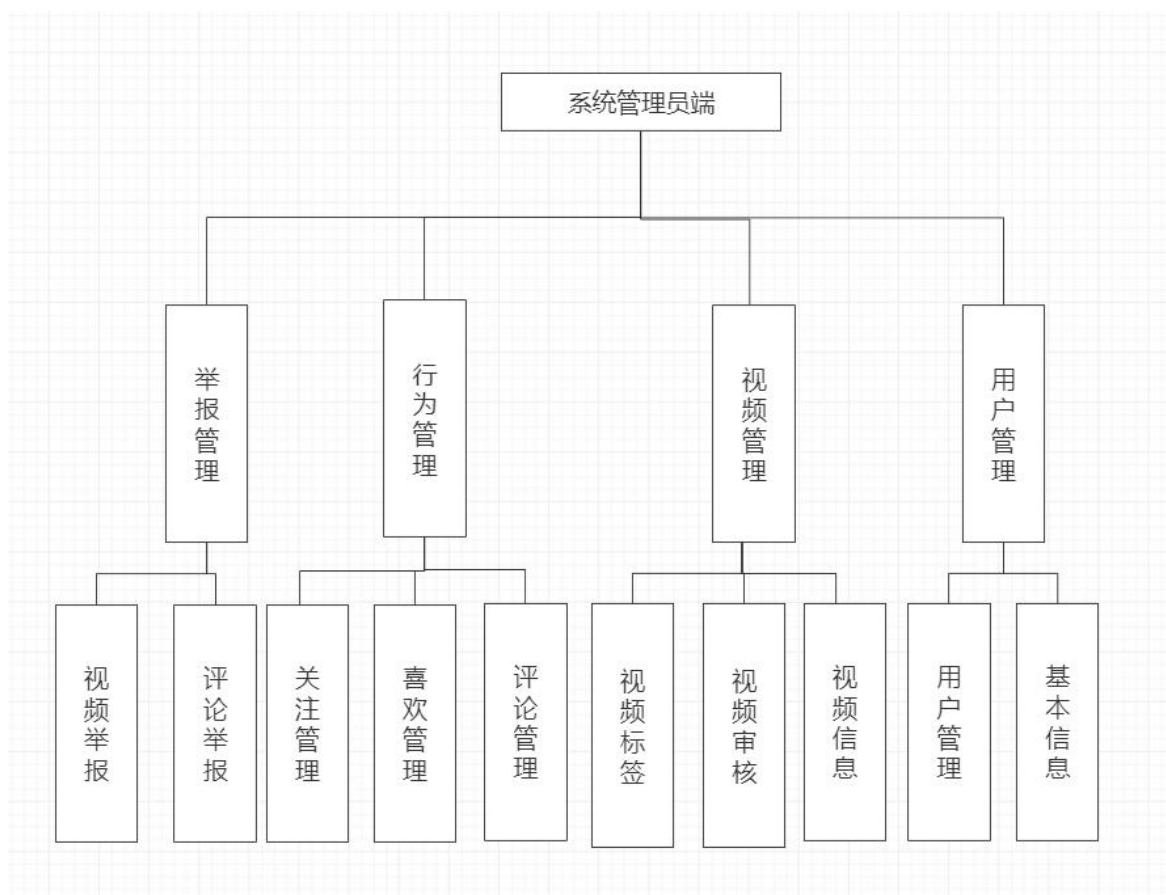


图 3.5 系统管理员端功能模块

### 3.4.3 数据库设计

通过需求分析以及概要设计功能模块分析阶段可以得出本系统具体的数据实体，描述将以模块名加冒号加必要字段的形式列举。

- ✓ 用户界面：手机号、密码、盐、用户名、用户头像、用户签名、用户状态。
- ✓ 视频界面：桶名、视频原名、视频名称、视频封面、视频状态、视频审核。
- ✓ 视频记录：用户 id、视频 id、观看时长。
- ✓ 视频举报：被举报者 id、举报者 id、视频 id、举报状态、举报留言。
- ✓ 视频标签：标签编号、标签名称。
- ✓ 用户喜欢：用户 id，视频 id。
- ✓ 弹幕：用户 id、视频 id、弹幕内容、视频当前时间。
- ✓ 关注分组、收藏分组：用户 id、分组 id、分组名称。
- ✓ 评论：用户 id、视频 id、回复对象 id、评论状态、评论内容。
- ✓ 私信：发送 id、接收 id、信息内容。

### 3.5 本章小结

本章主要对系统进行需求分析，把握系统开发的方向，明确要开发具有哪些功能的系统，其次进行概要设计，确定系统总体设计与总体架构，对系统开发的模块进行明确的划分，最后确定数据库的概念结构与逻辑结构。

## 第四章 系统详细设计与实现

### 4.1 系统开发环境与架构

#### 4.1.1 系统开发环境

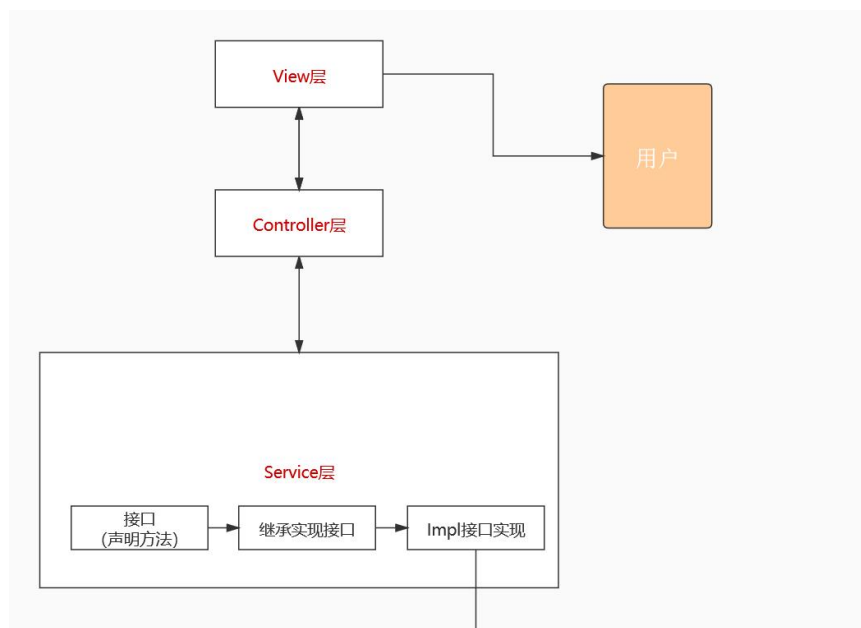
本系统采用了稳定的开发环境，在一定程度上保证 Vlog 视频平台开发工作的平稳进行，系统开发环境如下表 4.1 所示。

表 4.1 系统开发环境表

操作系统	Windows11
开发工具	IDEA
数据库	MySQL
服务器	Tomcat
浏览器	Google Chrome

#### 4.1.2 系统架构

社区空巢老人健康管理系统基于 SpringBoot 框架，一般而言，SpringBoot 可分为 entity (model) 层、dao 层、controller 层、service 层四层结构各层作用以及架构如下图所示。



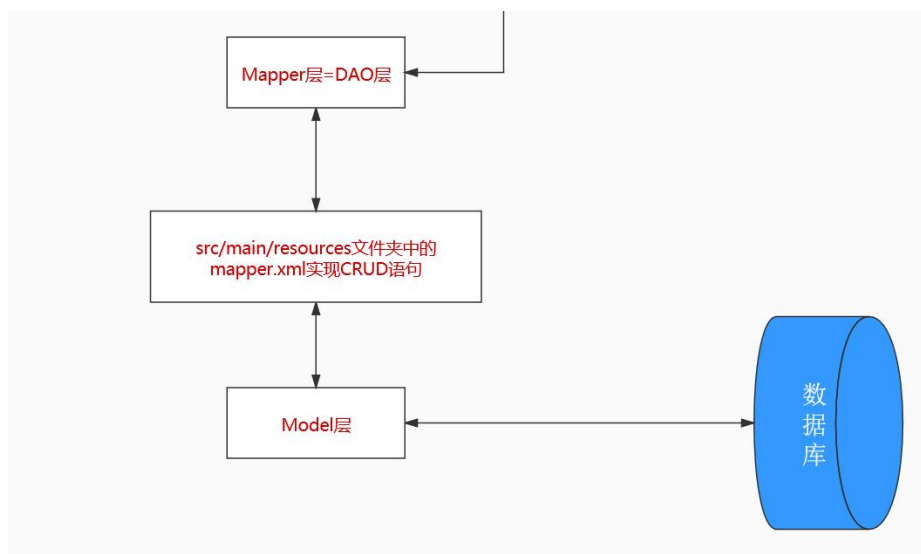


图 4.1 系统架构图

## 4.2 功能模块详细设计

### ● 会员用户

(1) 视频展示模块：只要成功登录，就会进入主页，看到相关视频的展示，用户可以通过分类查看自己想看的视频类型。

(2) 视频搜索模块：对视频信息进行精确的搜索，这里可以保留用户的搜索记录，刷新页面后，搜索记录会自动填充。

(3) 视频上传模块：这是为创作者专门来开发，这里不仅可以上传视频，还可以填写自己对视频的看法，选中相关分区后对视频进行上传。

(4) 点赞等功能模块：这块功能有点赞、关注、收藏、关注分组、收藏分组，看到喜欢的视频和一些想反复观摩的视频可以对视频进行相关操作，如果作者足够优质，也可以对作者进行关注。

(5) 弹幕模块：在弹幕区发送对视频的看法，弹幕就可以从视频上轻轻划过，分享给正在观看和之后观看视频的用户。

(6) 评论区模块：可以在这里对视频进行深入讨论，深入研究，对视频增添更深层的意义。

(7) 系统通知模块：你的视频被别人点赞、举报成功、你发布视频成功、都会在这里获得通知。

(8) 私信模块：这里只针对你关注的用户，只有你关注了该用户，你才能对他进行私聊，在这里可以催更作者哦。

(9) 信息管理模块：对自己的头像、昵称和个性签名不满意的话，可以在这里进行修改。

(10) 视频管理模块：你上传的视频是否上传成功、是否通过审核、在这里都可以看到，如果视频被举报在这里还可以看到视频的状态。

(11) 关注管理模块：对已经关注的作者进行分组管理，创建分组、删除分组、将关注作者放入该分组。

(12) 收藏管理模块：和关注管理模块大同小异，对已经收藏的视频进行分组管理，创建分组、删除分组、将收藏的视频放入该分组。

(13) 视频记录模块：有时候我们观看了视频，一开始不感兴趣，或者刚进去又退了出来，然后又想去看，在这里就可以找到该视频。

(14) 举报模块：用户可以对违规的视频和评论进行举报，填写相关的举报留言，交给管理员去处理。

#### ● 系统管理员端

(1) 举报管理模块：用户举报视频或者用户举报评论，可以在这里进行审核，审核结果会通过系统通知反馈给用户，感谢用户对平台做出的贡献。

(2) 视频审核模块：上传视频后，视频并不会第一时间再平台进行显示，搜索结果中也不会看到该视频的信息，视频只能让作者自己看到，只有通过审核后，视频才会在平台上一一显示。

(3) 视频管理模块：在这里可以查看所有作者上传的所有视频，如果该作者举报次数过多，可以对所有视频进行统一操作。

(4) 用户管理模块：对用户进行管理，比如有人不会更改密码，完全在这里进行修改。

(5) 视频操作管理模块：这个视频平台，用户对视频点赞、对视频收藏、对作者关注，都可以在这里进行查看和管理，比如视频十分优质有深度，但用户可能不太关注这些，管理员就可以对食品进行推波助澜，不能让优质的视频就此埋没，当然一些受欢迎但低俗的视频也可以做相应的操作。

(6) 视频标签模块：视频平台的标签是可以动态管理的，视频主页的分类就是来自于此，可以根据社会潮流，对视频的分区进行添加删除。

## 4.3 数据库及数据表

### 4.3.1 E-R 图

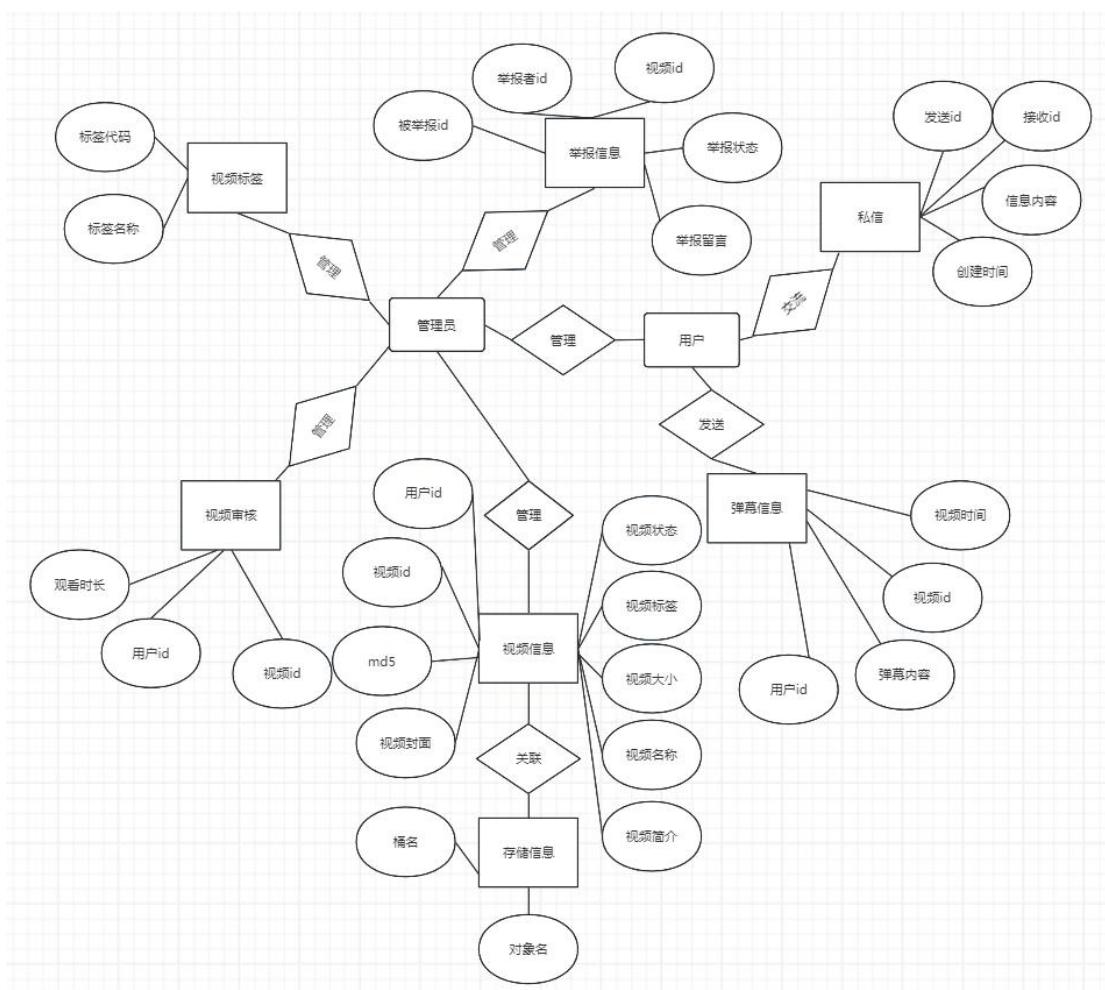


图 4.2 系统 E-R 图

### 4.3.2 数据表的设计

通过上一阶段的梳理，Vlog 视频系统一共包含 14 个数据表：视频表、视频信息表、观看记录表、视频举报表、视频标签表、私信表、视频收藏表、收藏分组表、评论表、关注表、关注分组表、弹幕表、用户表、用户信息表、喜欢表。



表 3.2 视频表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
bucket_name	varchar	255	是			所属桶名
object_key	varchar	500	是			文件的 key
create_date	timestamp		是			创建时间
update_date	timestamp		是			更新时间

表 3.3 视频信息表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
user_id	bigint		是			用户 id
video_id	bigint		是			视频 id
video_name	varchar	255	是			视频名称
video_summary	varchar	4069	是			视频简介
video_size	varchar	255	是			视频大小
video_cover	varchar	512	是			视频封面
works_label_id	int		是			标签 id
video_status	int		是			0-正常 1-异常
video_review	int		是			0-未审核 1-已审核 2-未通过
video_md5	varchar	255	是			md5

is_delete	int	是	删除 0-正常 1-已删
create_date	timestamp	是	创建时间
update_date	timestamp	是	更新时间

表 3.4 观看记录表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
user_id	bigint		是			用户 id
video_id	bigint		是			视频 id
duration	varchar	255	是			观看时长
create_date	timestamp		是			创建时间
update_date	timestamp		是			更新时间

表 3.5 视频举报表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
reported_user_id	bigint		是			被举报 id
reporting_user_id	bigint		是			举报者 id
video_id	bigint		是			视频 id
report_status	int		是			0-正常 1-被举报 2-违规 3-不违规

report_message	varchar	512	是		举报留言
cerete_date	timestamp		是		创建时间
update_date	timestamp		是		更新时间

表 3.6 视频标签表

列名	数据类型	长度	允许空	主键	外键	说明
id	int		否	是		标签 id
label_code	varchar	255	是			标签编号
label_name	varchar	255	是			标签名字

表 3.7 私信表

列名	数据类型	长度	允许空	主键	外键	说明
id	int		否	是		id
send_user_id	int		是			发送者 id
accept_user_id	int		是			接收者 id
message	varchar	2048	是			消息内容
create_time	timestamp		是			创建时间

表 3.8 视频收藏表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id

user_id	bigint	是	用户 id
video_id	bigint	是	视频 id
group_id	int	是	分组 id
create_date	timestamp	是	创建时间
update_date	timestamp	是	更新时间

表 3.9 收藏分组表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
user_id	bigint		是			用户 id
group_name	varchar	255	是			分组名称
create_date	timestamp		是			创建时间
update_date	timestamp		是			更新时间

表 3.10 评论表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
content	text		是			内容
video_id	bigint		是			视频 id
user_id	bigint		是			用户 id
reply_to_user_id	bigint		是			回复对象用户 id

report_status	int	是	评论举报状态
create_date	timestamp	是	创建时间
update_date	timestamp	是	更新时间

表 3.11 关注表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
user_id	bigint		是			本人 id
up_id	bigint		是			作者 id
group_id	bigint		是			分组 id
create_date	timestamp		是			创建时间
update_date	timestamp		是			更新时间

表 3.12 关注分组表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
user_id	bigint		是			用户 id
group_name	varchar	255	是			分组名称
create_date	timestamp		是			创建时间

update_date	timestamp	是	更新时间
-------------	-----------	---	------

表 3.13 弹幕表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
user_id	bigint		是			用户 id
video_id	bigint		是			视频 id
scrolling_context	varchar	512	是			弹幕内容
relative_time	bigint		是			弹幕相对时间

表 3.14 用户表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
phone_num	varchar	255	是			电话
password	varchar	512	是			密码
salt	varchar	255	是			salt
create_date	timestamp		是			创建时间
update_date	timestamp		是			更新时间

表 3.15 用户信息表

列名	数据类型	长度	允许空	主键	外键	说明
id	int		否	是		id
user_id	bigint		否			用户 id
image	varchar	255	是			头像
role_id	int		是			角色 id
status	int		是			用户状态
nickname	varchar	255	是			昵称
signature	varchar	255	是			个性签名
create_date	timestamp		是			创建时间
update_date	timestamp		是			更新时间

表 3.16 喜欢表

列名	数据类型	长度	允许空	主键	外键	说明
id	bigint		否	是		id
user_id	bigint		是			用户 id
video_id	bigint		是			视频 id
create_date	timestamp		是			创建时间
update_date	timestamp		是			更新时间

### 4.3.3 数据库的连接

本系统使用 MyBatis 连接 MsSql 数据库,需要借助 jdbc 连接池与数据进行绑定。实现数据库的连接的具体实现方法是要建立 application.properties 文件,配

```
#datasource
spring.datasource.url=jdbc:mysql://192.168.100.130:3307/ruoyi?serverTimezone=Asia/Shanghai
spring.datasource.username=root
spring.datasource.password=123456
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#mybatis
mybatis.mapper-locations=classpath:mapper/*.xml
mybatis.configuration.map-underscore-to-camel-case=true
```

置连接数据库相关信息，其中包含用户、密码、MySQL sever 的端口号等信息。

## 4.4 系统实现

### 4.4.1 登录注册界面设计及实现

目前使用得是手机号注册登录，当用户获取到验证码后，系统会读取数据库，判断当前用户是否已注册，如果注册过就直接登录，并且返回 token，其中 token 是一个字符串是通过 JWT，根据自定义得标签、RSA 加密算法、所需要加密得数据、所加密数据过期时间，通过这些参数获取得字符串作为 token 返回给前端。若账号尚未注册，那便是时候开启注册得仪式了。首先，我们会将用户提供得注册信息保存到数据库中。然后，我们将携着这份信息，生成 token 返回给前端。

vlog 共享平台登录注册图如图 4.1 所示，登录成功图如图 4.2 所示；具体流程图如图 4.3 所示。



该图展示了 Vlog 平台的登录注册界面。界面顶部中央有一个蓝色的耳机图标，旁边是“Vlog平台”字样。下方是一个包含输入框和按钮的表单。第一个输入框下方有提示文字“请输入手机号”。第二个输入框下方有提示文字“请输入验证码”，其右侧有一个蓝色的“获取验证码”按钮。在输入框下方是一个宽大的蓝色按钮，上面写着“登录 / 注册”。

图 4.1 登陆注册图



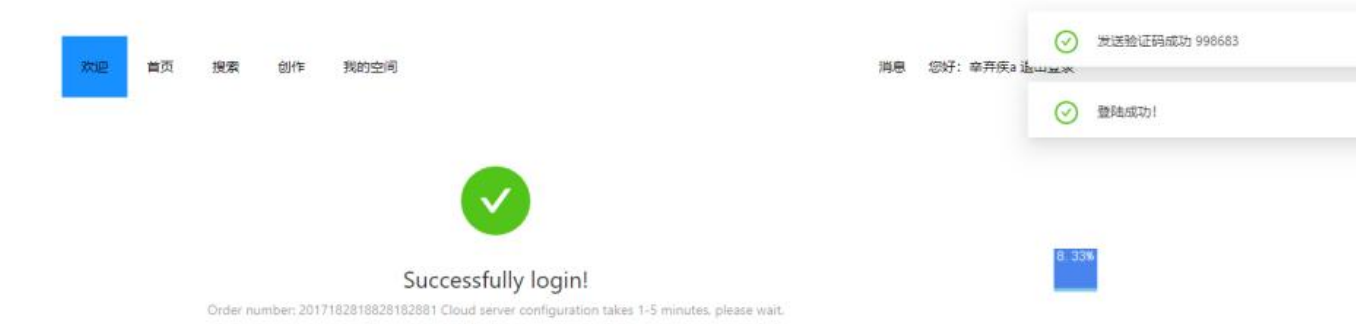


图 4.2 登录注册成功图

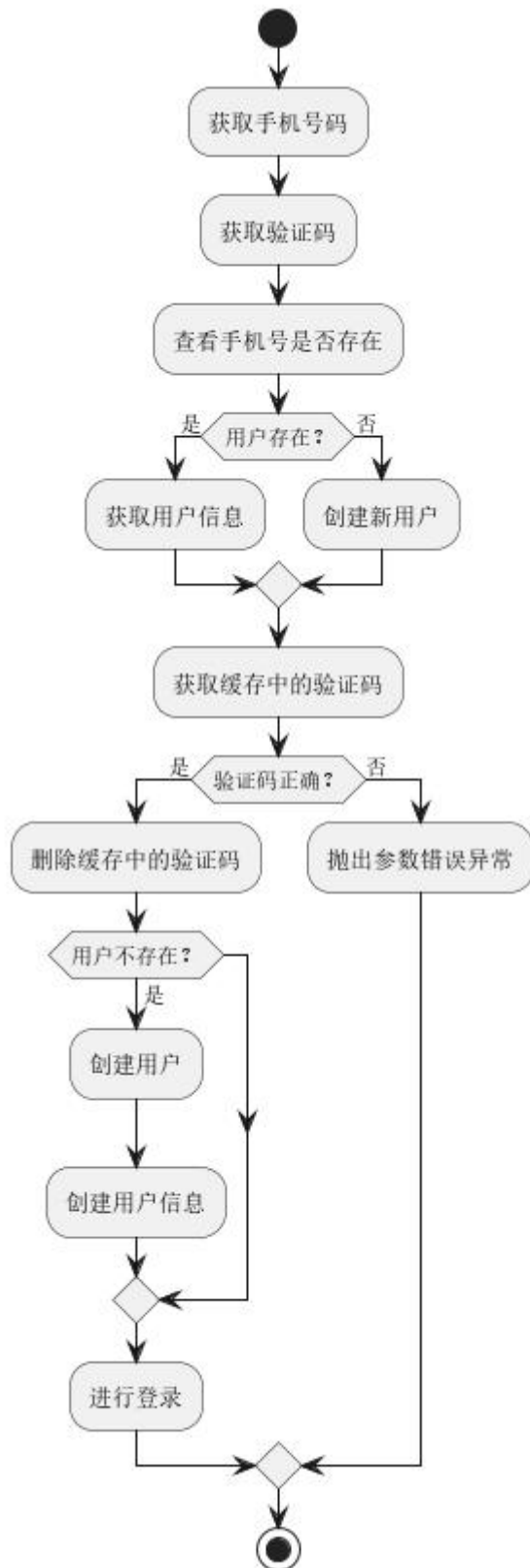


图 4.3 注册登录流程图

## 4. 4. 2 视频上传功能模块设计及实现

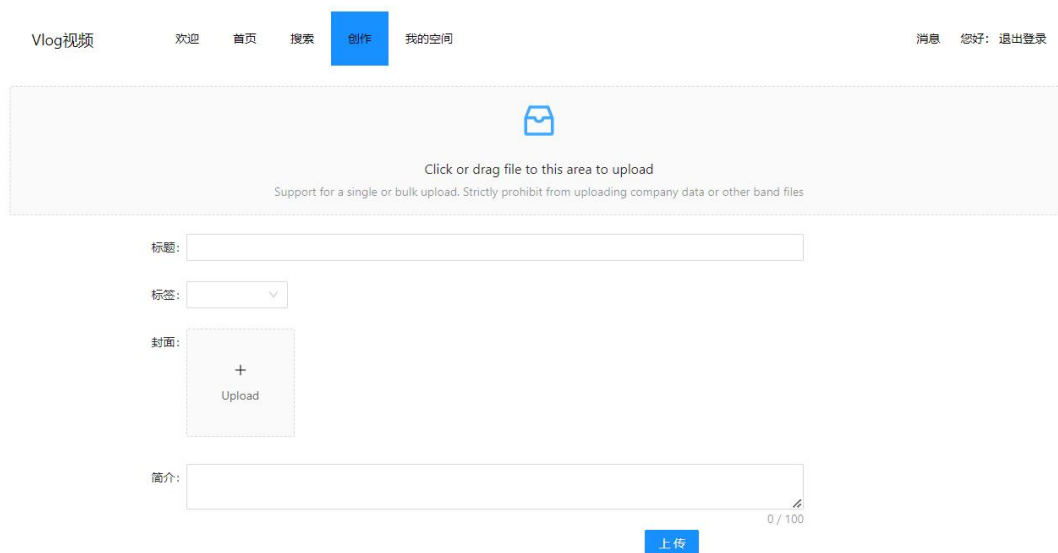
首先，通过 `MultipartHttpServletRequest` 获取上传得文件对象和相关得参数，包括文件分片索引、总片数、文件名、文件总大小和 MD5 值等。

根据文件得 MD5 值创建一个唯一得桶（Bucket）名称，用于在 MinIO 中存储分片文件<sup>[10]</sup>。

检查 Redis 中记录得上传进度，如果当前分片已经上传过，则返回状态码 20001，表示该分片已上传。

如果当前分片不是最后一块，则执行分片上传操作。将当前分片得 `InputStream` 上传到 MinIO 中，并在 Redis 中记录上传进度，设置记录得过期时间为一天。

在视频上传图如图 4.4 所示；具体流程图如图 4.5 所示。



The image shows a web interface for video upload. At the top, there is a navigation bar with links: 'Vlog视频', '欢迎', '首页', '搜索', '创作' (highlighted in blue), and '我的空间'. On the right side of the navigation bar, there are links for '消息', '您好', and '退出登录'. Below the navigation bar, there is a large dashed box containing a blue folder icon and the text: 'Click or drag file to this area to upload' and 'Support for a single or bulk upload. Strictly prohibit from uploading company data or other band files'. Below this box, there are several input fields: '标题:' (Title), '标签:' (Tags) with a dropdown arrow, '封面:' (Cover) with a '+' icon and 'Upload' button, and '简介:' (Description) with a text area. At the bottom right, there is a blue button labeled '上传' (Upload) and a character count '0 / 100'.

图 4.4 视频上传图

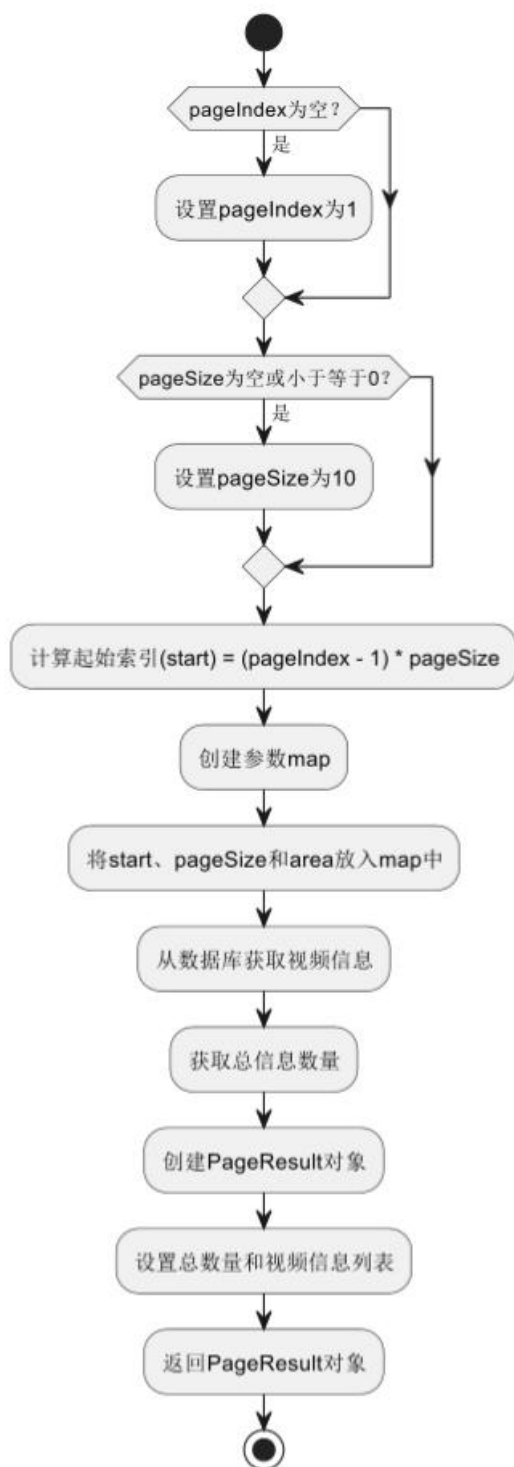


图 4.5 视频上传流程图

#### 4.4.3 视频播放功能模块设计及实现

用户点击后，即将开启一段魔幻之旅。首先，前端会像给后端的门卫递钥匙一样，将视频的 ID 传递到后端。后端则以此 ID 为线索，探索 Minio 中的视频隐藏

之处。同时, 后端也会通过这个 ID, 搜寻与之相关的视频资料, 比如视频名称、简介、封面和创作时间。最终, 在通过 URL 获得视频播放数据的同时, 用户则可以在视频播放中享受视频快感。

视频成功播放图如图 4.6 所示; 具体流程图如图 4.7、图 4.8 所示;

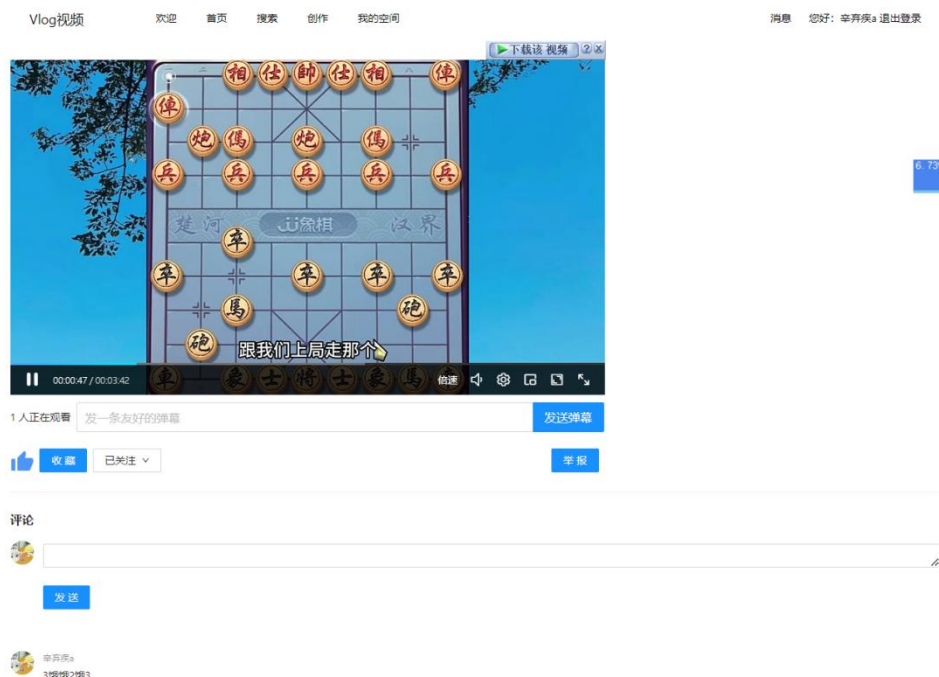


图 4.6 视频成功播放图



图 4.7 获取 URL 流程图

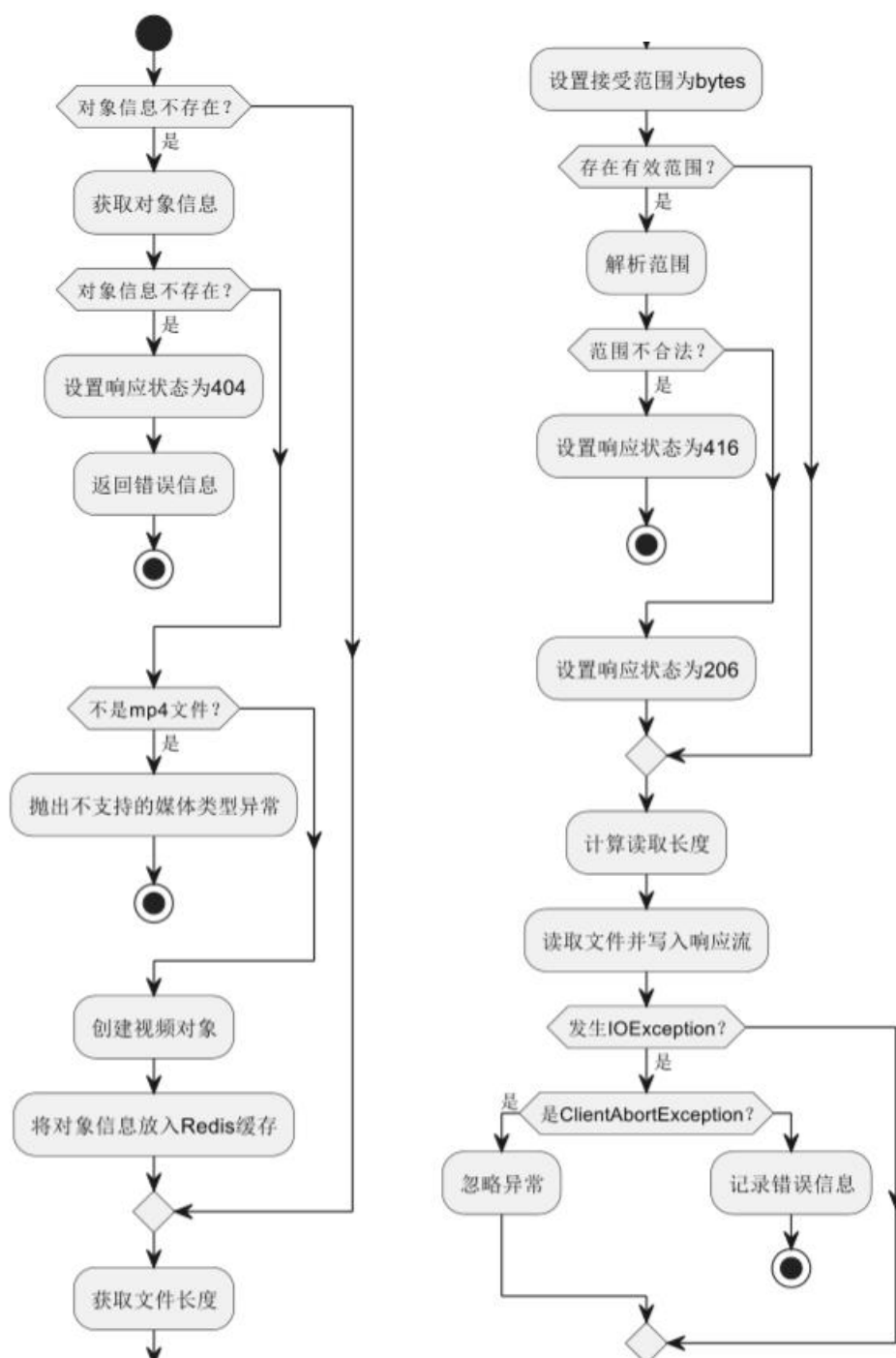


图 4.8 获取视频流流程图

#### 4.4.4 弹幕功能模块设计及实现

当用户播放视频时，弹幕请求也随之发送，弹幕请求到达后端后，因为有可能有大量得人去查询弹幕信息，所以会先查询 Redis，获取到相应视频得相应弹幕，如果该视频得弹幕信息在 Redis 里面查不到，会在去查询数据库，查询到弹幕信息后会第一时间返回到前端，并通过异步保存到 Redis 里面，如果有大量得人同时访问

数据库。发过来的弹幕有可能数量庞大，并且每一条都要再发给很多人，所以使用 Rabbitmq 进行排队，开辟线程池一步进行发送，使弹幕有序的发送，并且让程序继续运行；保存弹幕的过程使用异步，避免阻塞，为减轻服务器和数据库的压力，现将弹幕信息保存在 Redis 里面，后面再使用定时任务对弹幕进行统一保存。

弹幕发送成功图如图 4.9 所示；具体流程图如图 4.10 所示。



图 4.9 弹幕发送成功图

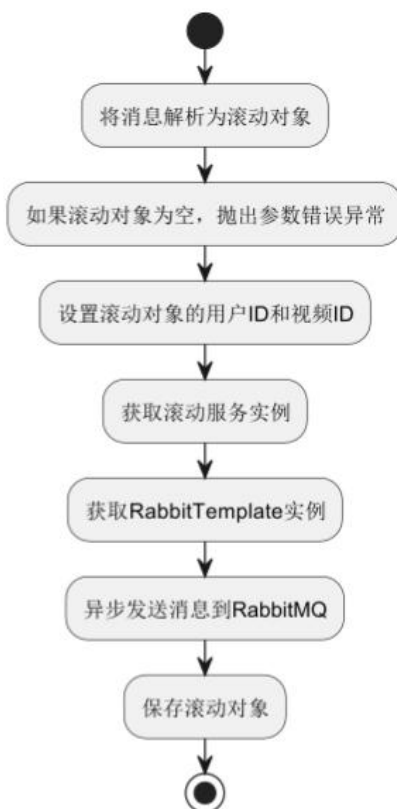


图 4.10 弹幕发送流程图

#### 4.4.5 搜索功能模块设计及实现

搜索请求后确定它是否是视频信息的请求。如果是的话，通过 Java 代码块从 Hashmap 里面获取相应的请求方式。当请求到达后，只需要通过 Type 的指引就可以很方便的找到对应的被询者，使用 Elasticsearch 的 ik 分词器，对内容进行保存的时候对其进行分词理性的分词，这样搜索出来的结果就可以近似的实现百度的效果。SpringDataElasticSearch 查询速度快，模糊查询非常好用<sup>[13]</sup>。多数据源的问题主要使用了门面模式、适配器模式和享元模式来进行解耦和减少重复的代码。

搜索记录图如图 4.11 所示，搜索成功图如图 4.12 所示，具体流程图如图 4.13 所示。





图 4.11 搜索记录图

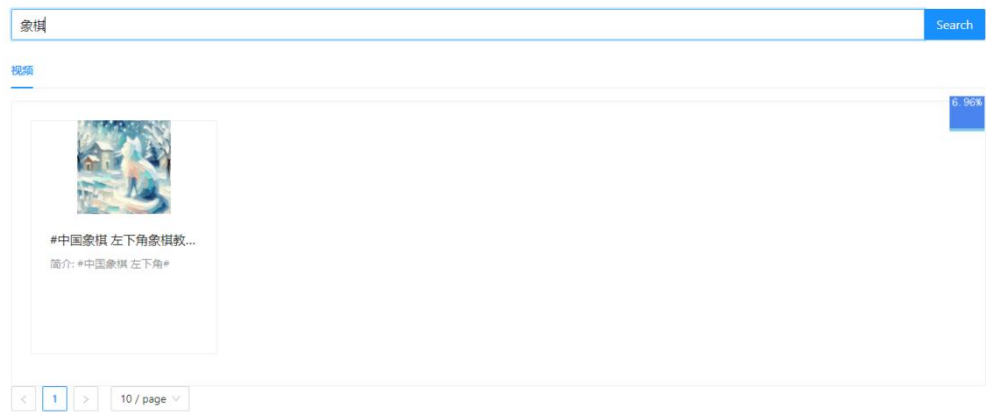


图 4.12 搜索成功图

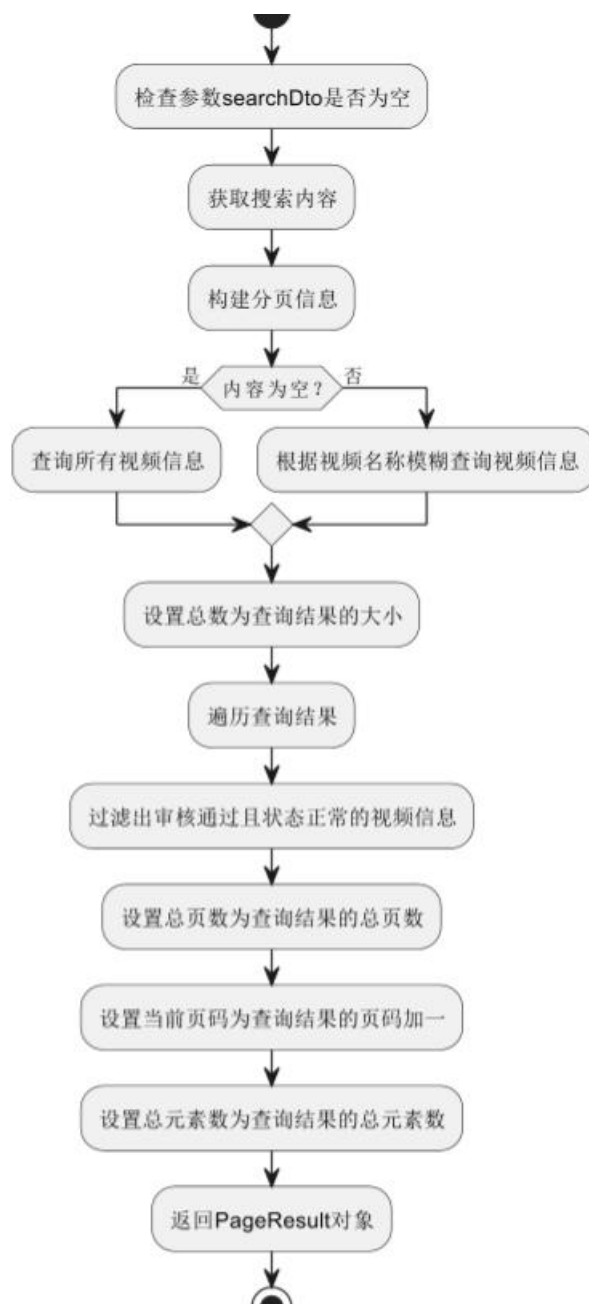


图 4.13 搜索流程图

#### 4.4.6 用户操作模块设计及实现

进入视频播放之后，会对后端进行多个用户操作相关得请求操作，会有以下请求：获取视频是否被点赞、获取视频是否被收藏、获取视频作者是否被关注、如果视频已被收藏则获取收藏分组信息、视频作者已被关注则获取关注分组信息。如果视频没有被收藏，点击收藏过后可以进行分组设置，可以选择一个分组或者多个分

组；关注分组也是同理；举报按钮记录有视频 id 和自定义得举报留言到数据库之后有管理员去处理。

点赞成功图如图 4.14 所示；视频收藏图如图 4.15 所示；收藏分组图如图 4.16 所示；作者关注图如图 4.17 所示；关注分组图如图 4.18 所示；具体流程图如图 4.19 所示。

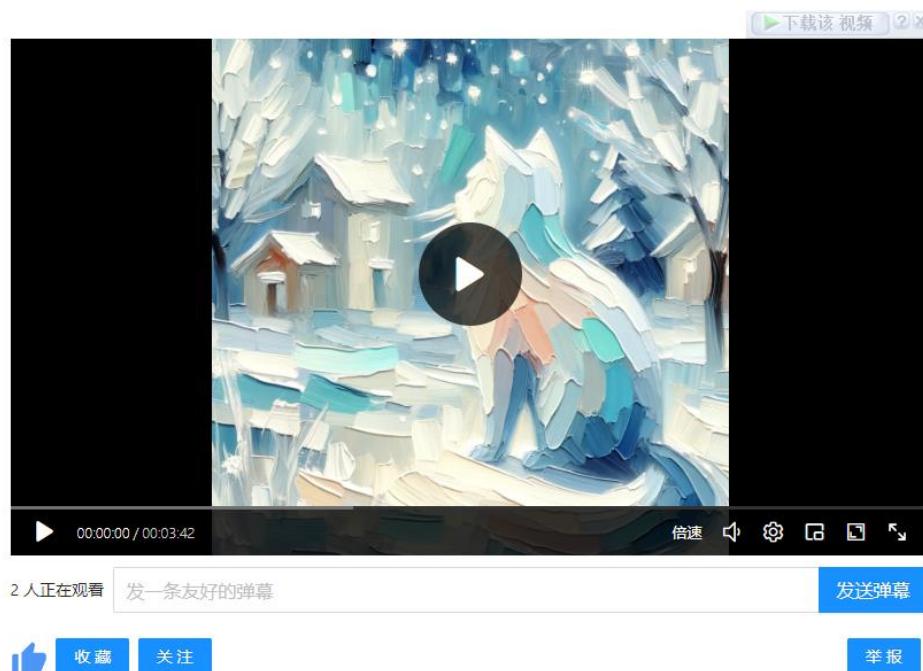


图 4.14 点赞成功图

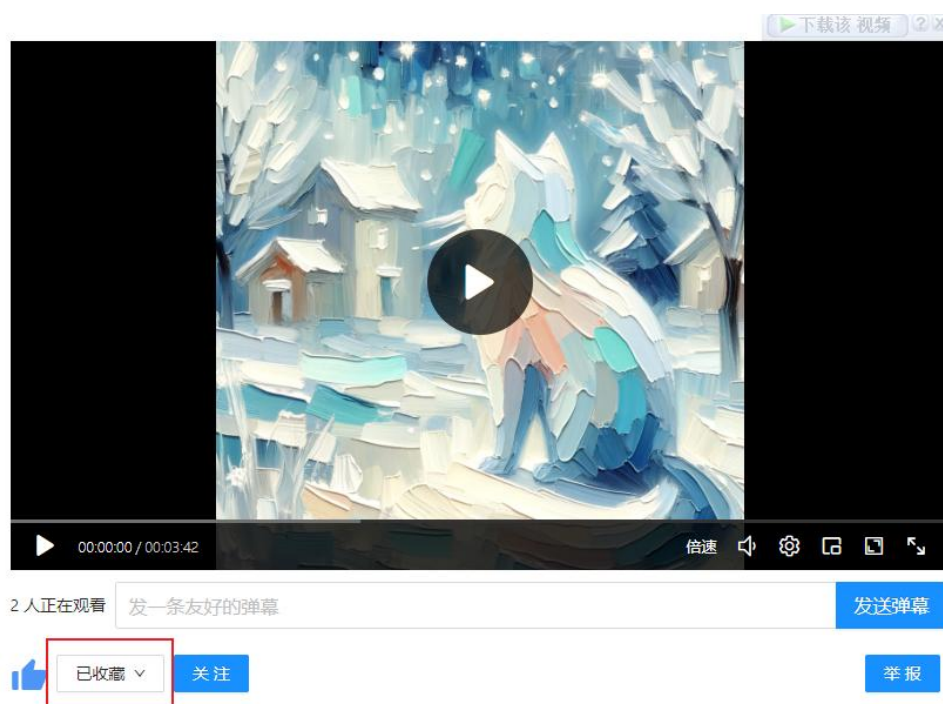


图 4.15 视频收藏图

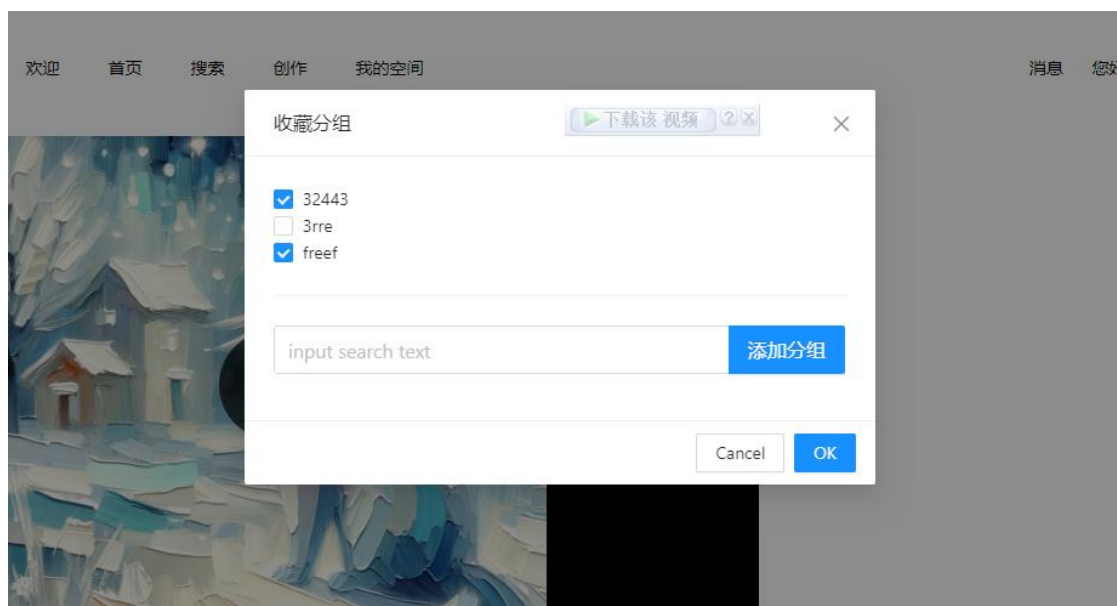


图 4.16 收藏分组图

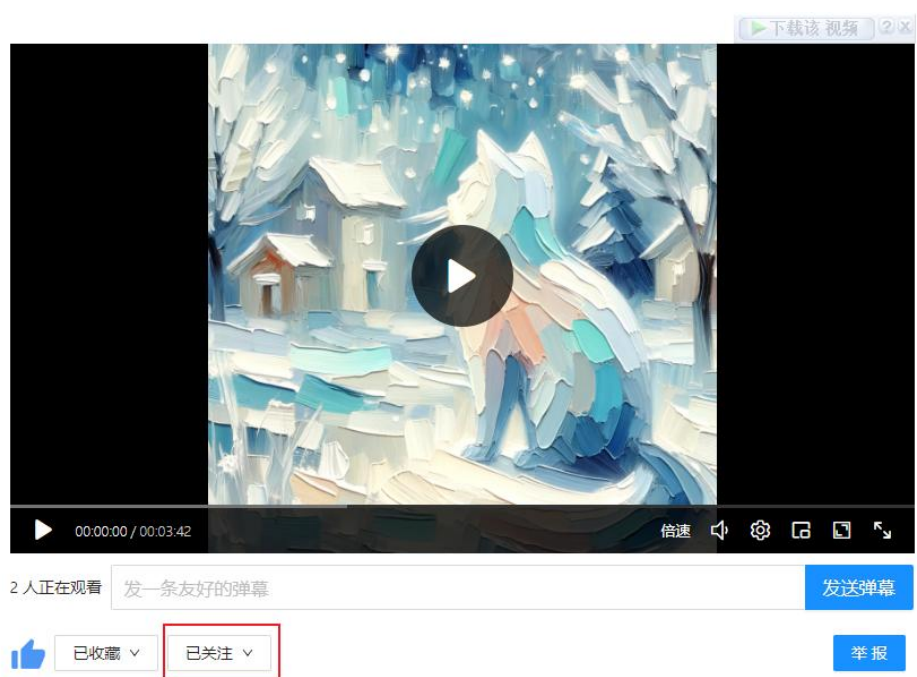


图 4.17 作者关注图

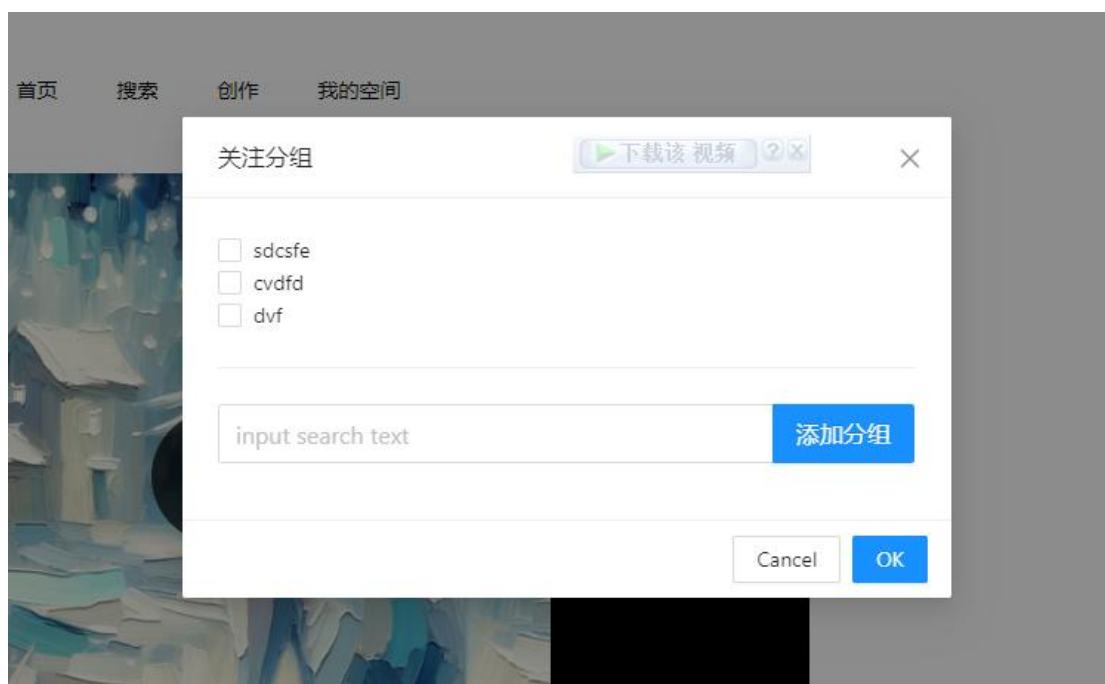


图 4.18 关注分组图

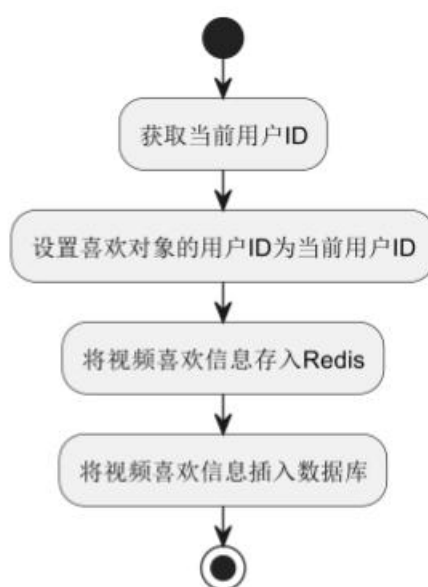


图 4.19 喜欢流程图

#### 4.4.7 消息功能模块设计及实现

点开私信后，手机就会跟服务器索取数据。然后，服务器就会把收到得消息都列出来。如果有消息或者是你发过消息得人，他们得名字都会出现在这个列表里。当你想要发消息时，手机会先检查一下对方是否在线。如果在线，它就会马上把消息发过去，并且把聊天记录保存起来，这样你以后想看得时候就能找到。如果对方

不在线，就会稍微等一下，等对方上线了再发，同时把消息保存到数据库里，这样对方上线后就能看到你的消息了<sup>[11]</sup>。

私信接收消息图图 4.20 所示；消息发送图图 4.21 所示；具体流程图如图 4.22、图 4.23 所示。

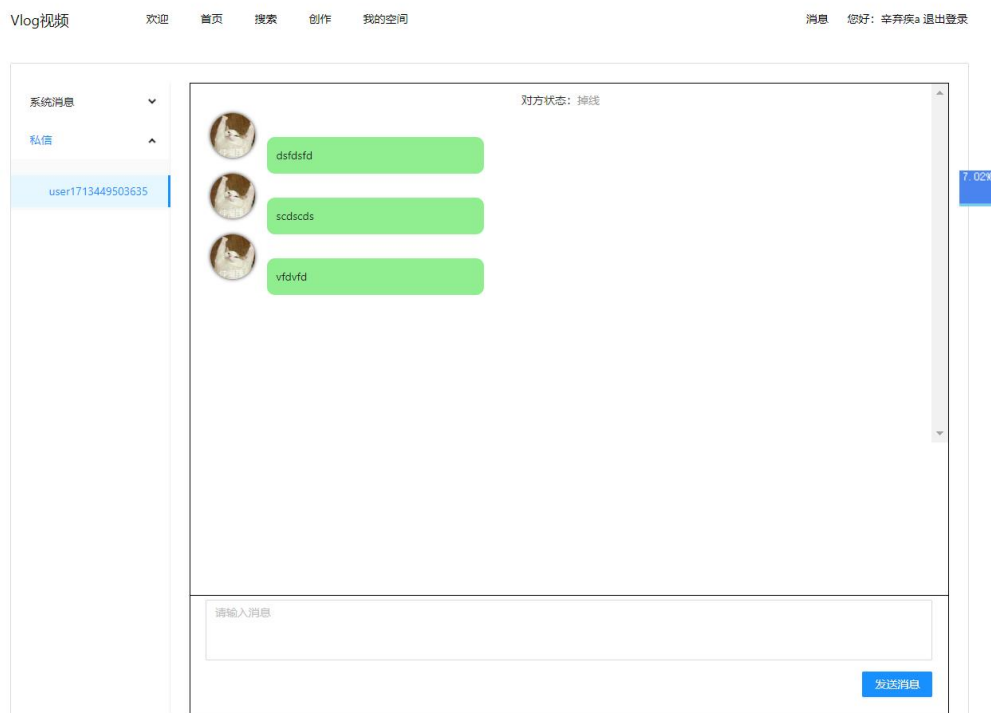


图 4.20 私信接收消息图

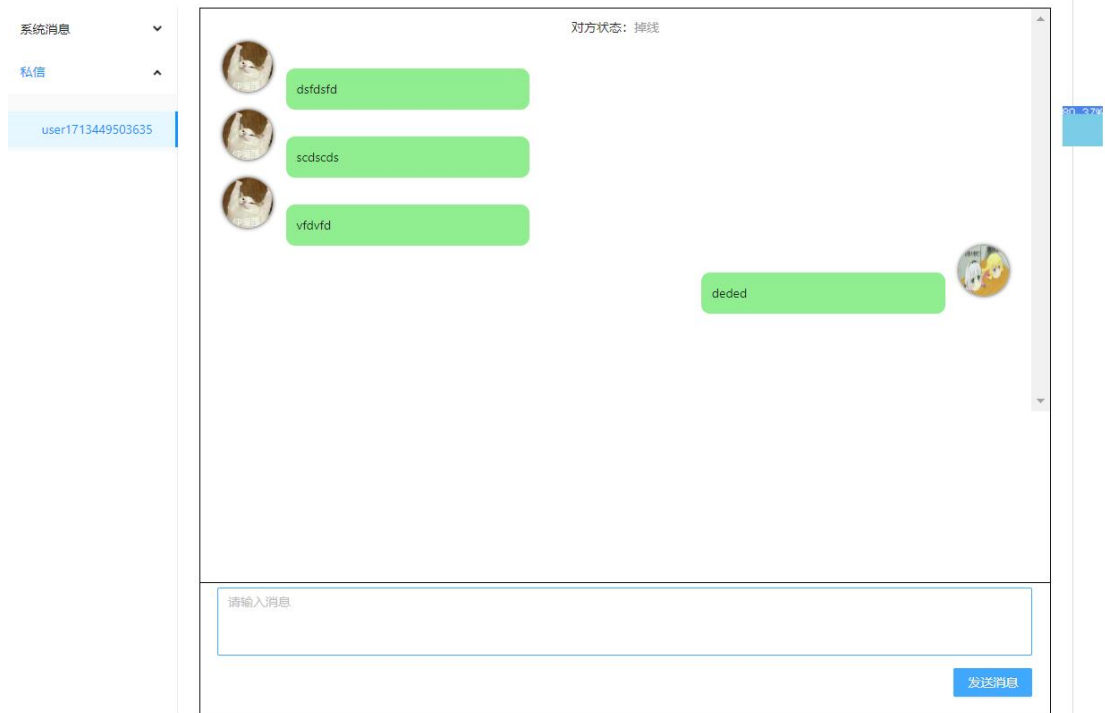


图 4.21 消息发送图

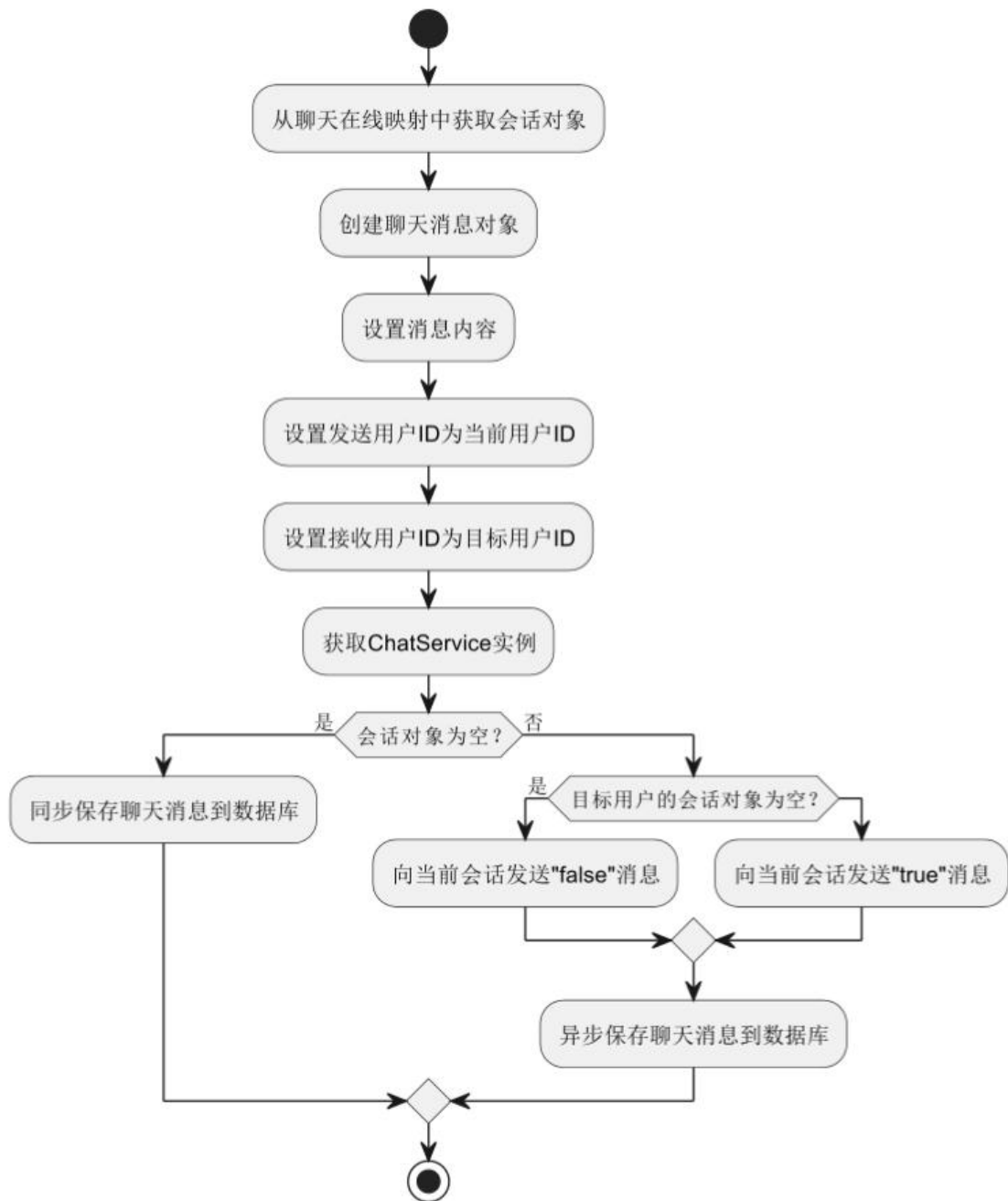


图 4.22 发送消息流程图



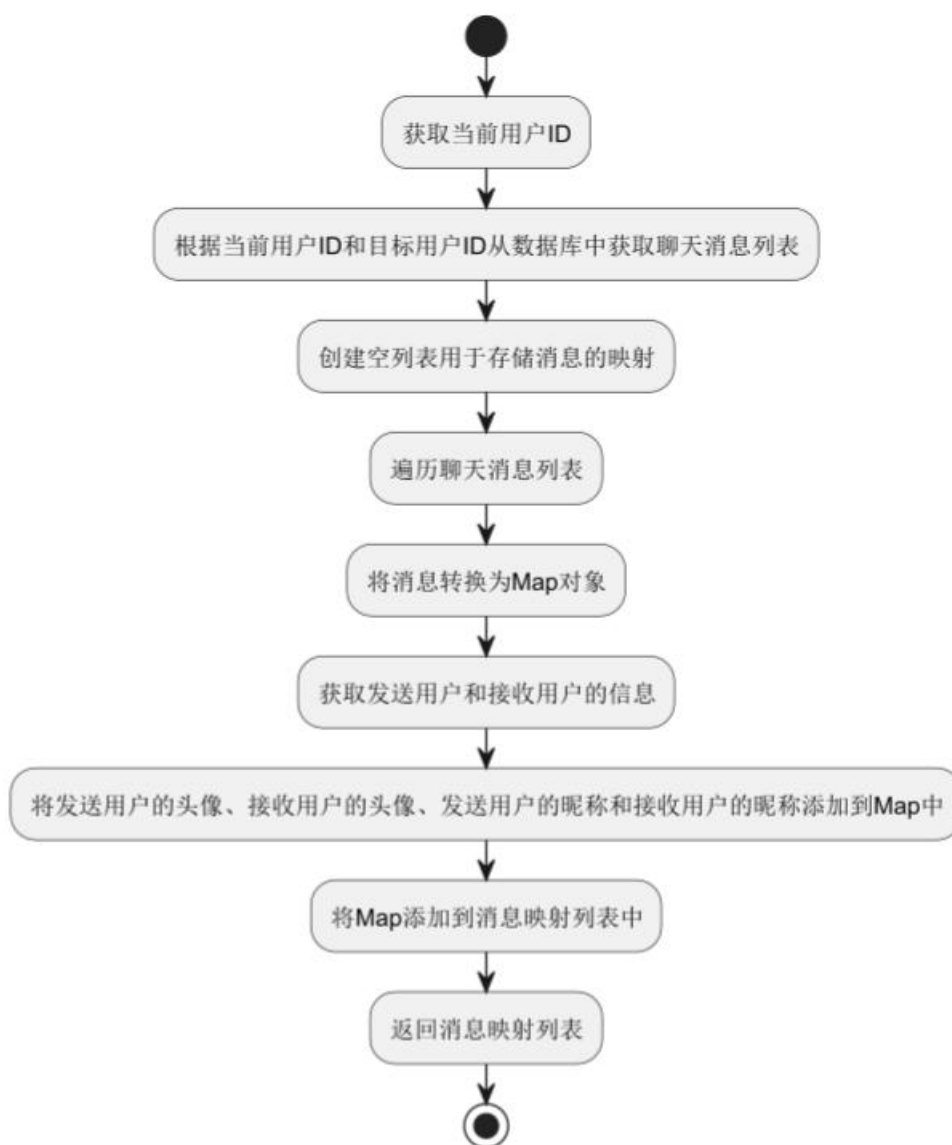


图 4.23 获取消息流程图

#### 4.4.8 视频审核模块设计及实现

进入视频审核界面后，对于还没有审核通过的视频信息，需要通过标签进行筛选。这时页面上已经出现了视频播放的网址，用户点击就能看到视频。需要从 Minio 存储系统中获取视频资料，并对这些资料进行详细的审核，才能保证审核的准确性。

首先，要初步看录像，看录像内容是不是合乎规定。接下来，对录像的各个部分，包括画面、音频、字幕等，都要进行细致的检查。在这一过程中，需要借助人工智能、机器学习等先进的技术手段，对潜在问题的识别进行辅助。

视频审核图如图 4.24 所示；审核界面图如图 4.25 所示；具体流程图如图 4.26 所示。

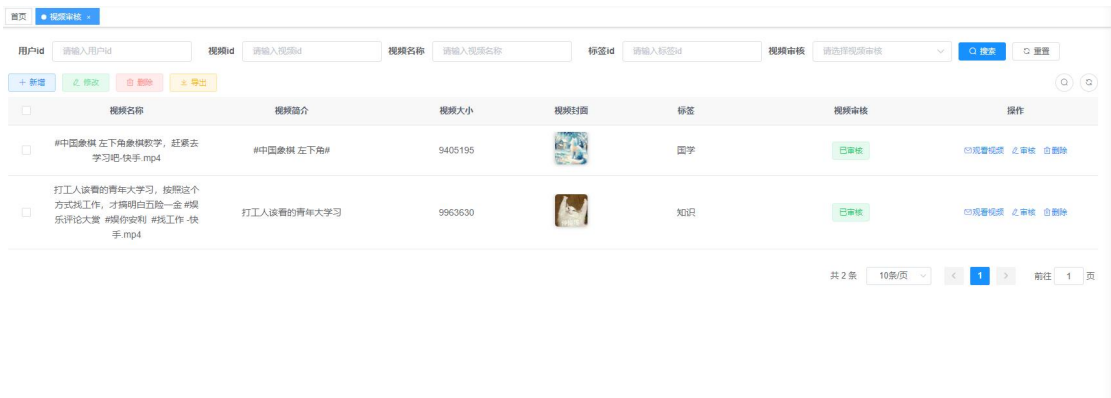


图 4.24 视频审核图

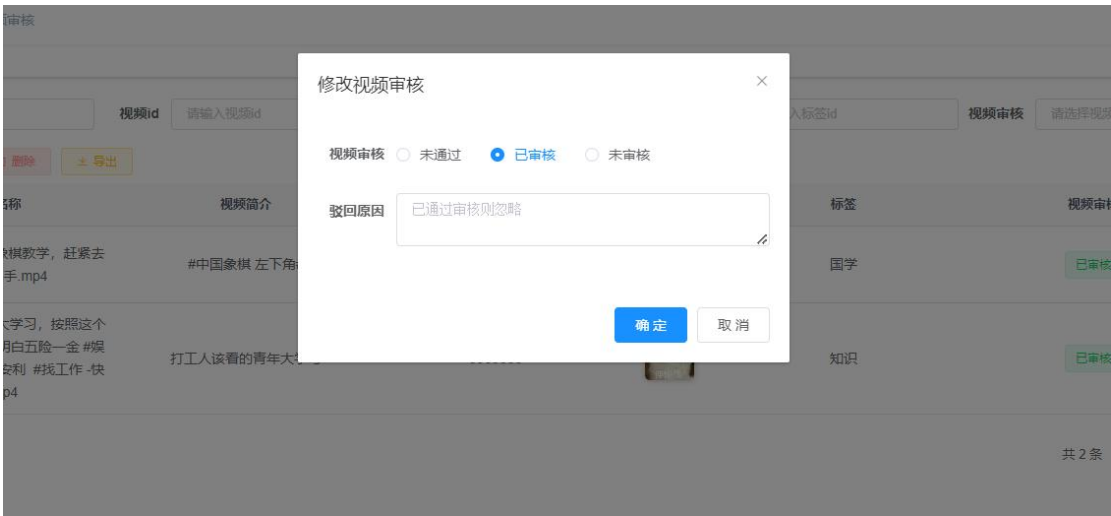


图 4.25 审核界面图

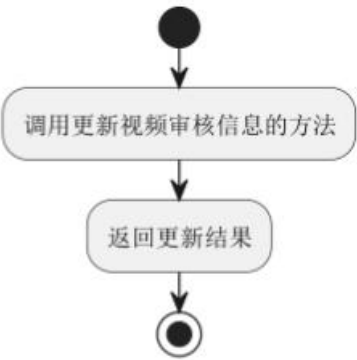


图 4.26 视频审核流程图

4.4.9 观看记录模块设计及实现

点开观看记录页面，后端会获取观看记录消息，用户观看视频时会同步视频得观看位置，所以观看记录请求得非常频繁，减轻数据库压力，先将记录保存到 Redis，

后面在通过定时任务同步到数据库里面<sup>[15]</sup>。所以在获取观看记录时会先获取 Redis 里面得数据,然后在获取数据库里面得数据,为防止重复获取,会对记录通过 HashSet 进行去重。

视频观看记录图如图 4.27 所示具体流程图如图 4.28、图 4.29 所示。



图 4.27 视频观看记录图

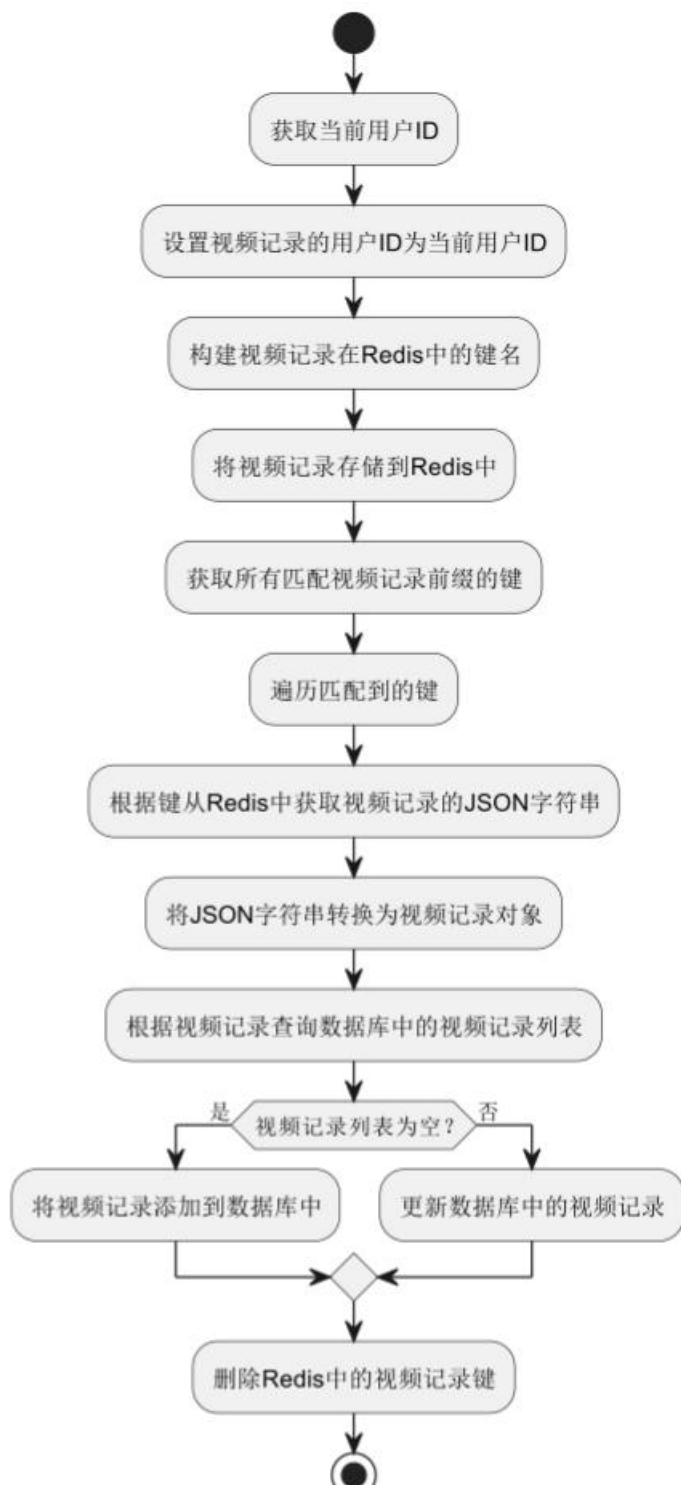


图 4.28 记录流程图

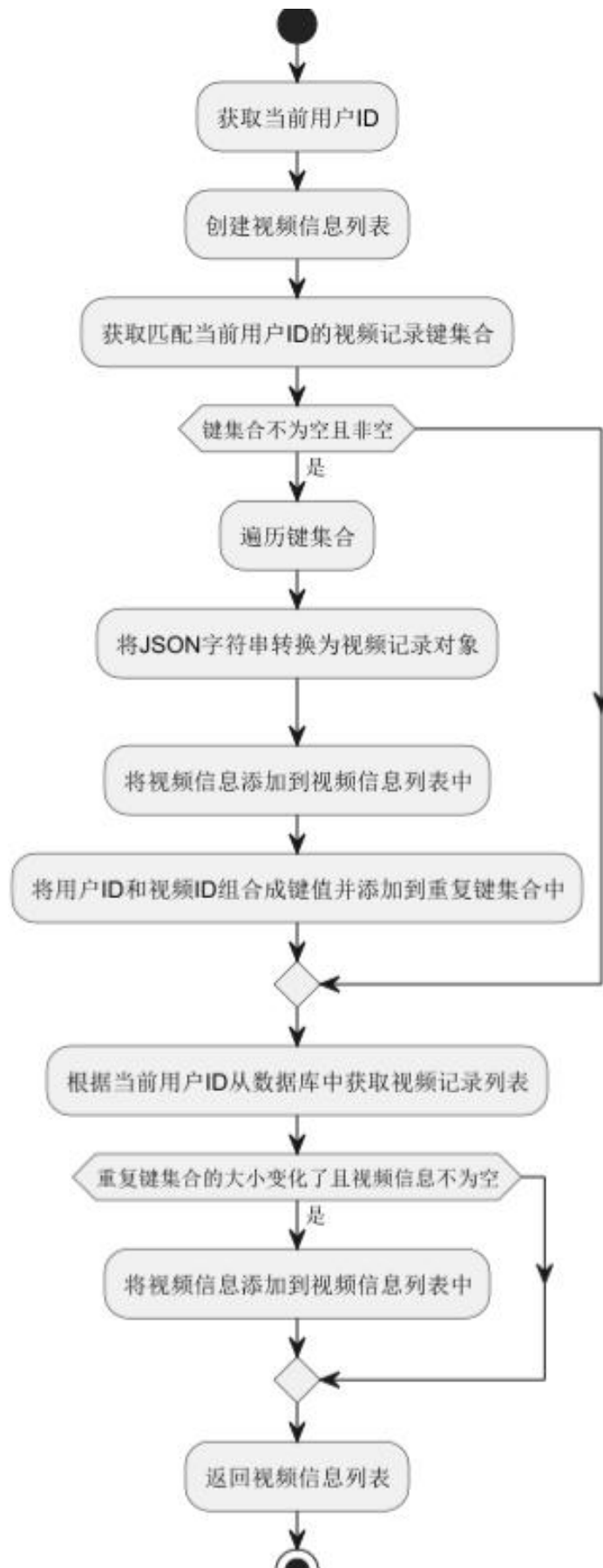


图 4.29 获取记录流程图

#### 4.4.10 关注列表模块设计及实现

获取关注得分组信息：用户在进入关注页面后，系统会首先获取用户已创建得关注分组信息。查看关注分组：用户可以浏览他们已创建得关注分组列表。每个分组通常显示分组名称以及该分组内已关注作者得数量。

选择分组：用户可以点击特定得关注分组，以查看该分组内已关注得作者列表。

这些列表通常显示作者得头像、用户名和简要介绍等信息，以帮助用户快速识别和浏览感兴趣得作者。

关注作者：通常，用户可以通过点击作者得头像或用户名来查看作者得个人主页，并在个人主页上执行关注操作。

作者得个人主页通常包含作者得详细信息、发布得视频列表、粉丝数量等内容，用户可以在个人主页上进一步了解作者得活动和作品。

视频关注列表图如图 4.30 所示；具体流程图如图 4.31、图 4.32 所示。

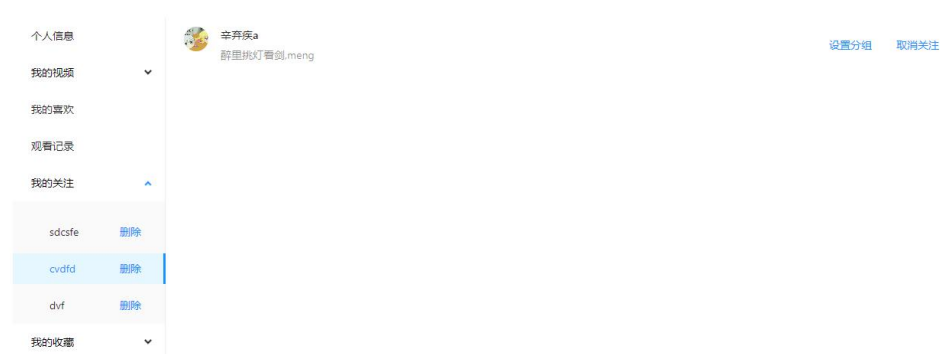


图 4.30 视频关注列表图

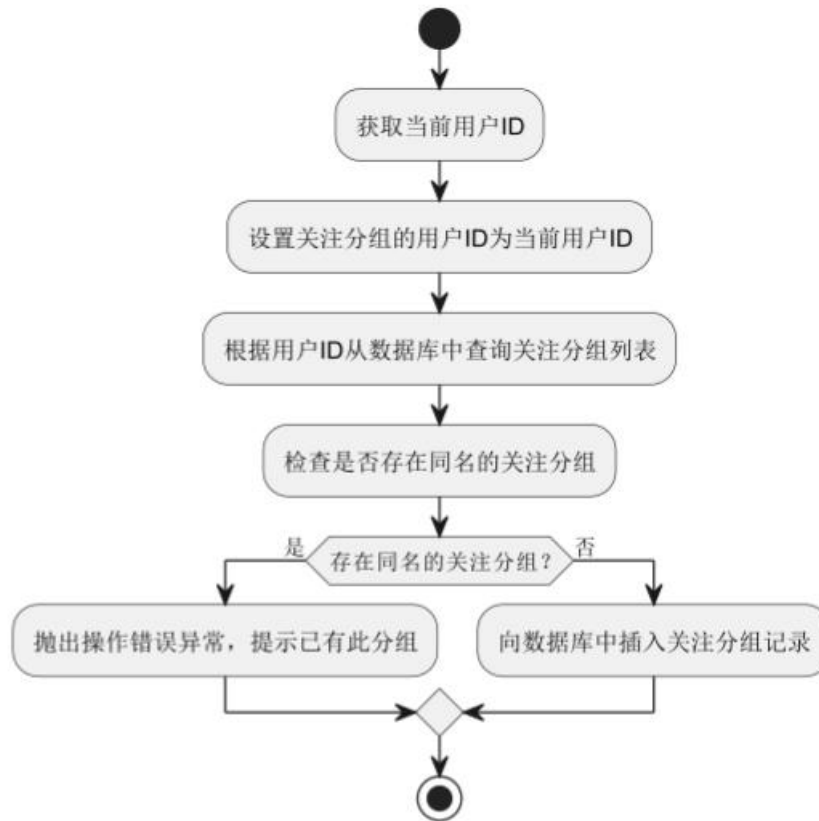


图 4.31 关注分组流程图

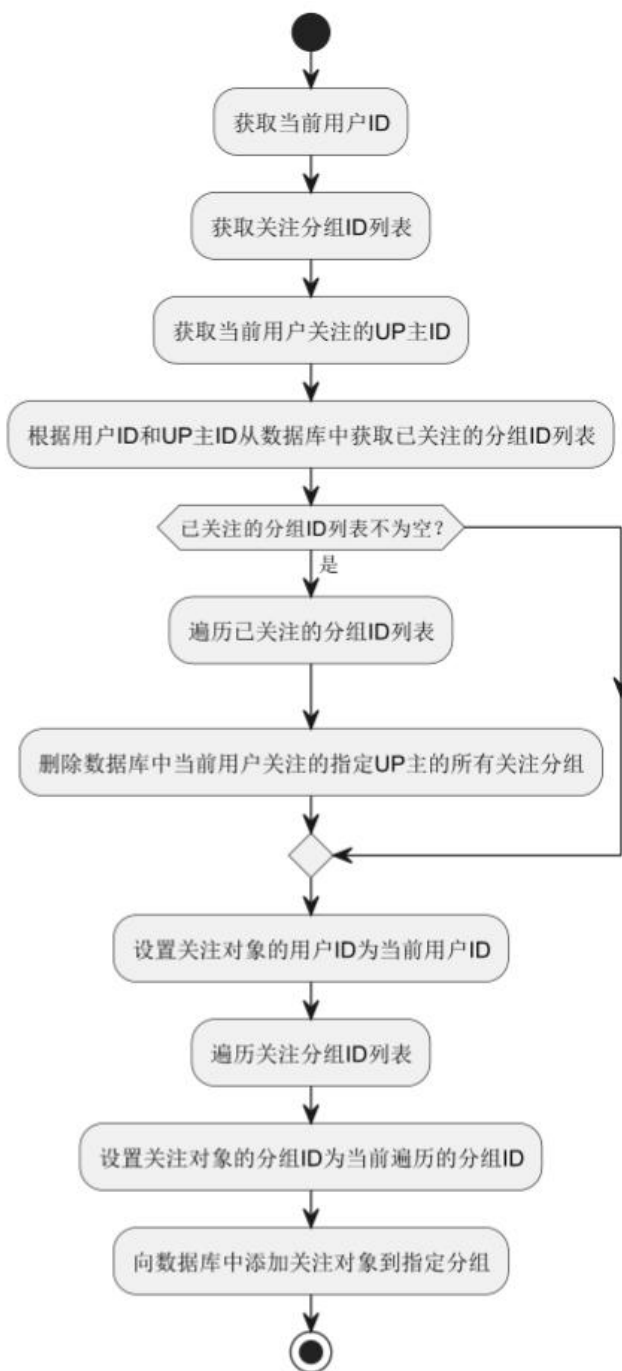


图 4.32 添加关注流程图

## 4.5 本章小结

本章节主要对系统进行详细设计,依据上一章节的模块划分明确各个模块主要实现的功能以及细节设计,同时将数据库设计完整。确定了系统开发环境与架构后,在系统实现部分完全满足了最初的需求且人机交互友好。



## 第五章 系统测试

### 5.1 系统功能测试

系统的各个模块都有各自特有的功能作用，而测试用例是保证其能正常运作的关键。对本平台的各个主要功能和模块进行仔细的测试用例编写，从用户的角度出发，考虑了各种可能的使用情景和操作步骤。在测试过程中，对测试用例进行了不断的调整和改进，保证覆盖到系统的各个方面，做到测试的全面性和有效性。同时也提高了产品的品质，保证了用户的使用体验。

测试完成后，将直接呈现给用户最终页面效果，以直观的方式展示系统的功能是否达到预期，以及用户体验是否流畅，从而为系统的进一步优化和改进提供重要的参考依据，也就是最终目的。

#### 5.1.1 登录注册页面测试用例

表 5.1 登录注册页面测试用例

功能名称		登录注册		
测试目得		能够正常登录注册		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	输入未注册账号	无	自动注册反登陆成功	与期望结果一致
1.2	输入已注册账号	无	登录成功	与期望结果一致

#### 5.1.2 视频上传测试用例

表 5.10 关注列表测试用例

功能名称		视频上传		
测试目得		将视频上传到平台		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	获取分区信息	无	获取到分区信息	与期望结果一致
1.2	上传视频	无	完整上传	与期望结果一致

1.3	上传封面	无	完整上传	与期望结果一致
1.4	上传其他信息	无	上传成功	与期望结果一致

### 5.1.3 视频播放测试用例

表 5.2 视频播放功能测试用例

功能名称		视频播放功能		
测试目的		不同得情况能否正常播放活不播放		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	在首页和搜索时能否播放	status=0;review=1	可以播放	与期望结果一致
1.2	在首页和搜索时能否播放	status=1;review=1	不能播放	与期望结果一致
1.3	在首页和搜索时能否播放	status=0;review!=1	不能播放	与期望结果一致
1.4	在首页和搜索时能否播放	status=1;review!=1	不能播放	与期望结果一致
1.5	在个人主页能否播放	status=0;review=1	可以播放	与期望结果一致
1.6	在个人主页能否播放	status=1;review=1	可以播放	与期望结果一致
1.7	在个人主页能否播放	status=0;review!=1	可以播放	与期望结果一致
1.8	在个人主页能否播放	status=1;review!=1	可以播放	与期望结果一致

### 5.1.4 弹幕功能测试用例

表 5.3 弹幕功能测试用例

功能名称	弹幕功能
测试目的	人数多得情况下能否正常发送接收弹幕
预置条件	无

用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	两个人观看同一条视频是都可以看到对方弹幕	无	可以看到	与期望结果一致
1.2	多人同时观看同一视频，一个人发弹幕，其他人是否都能看到	无	都可以看到	与期望结果一致

### 5.1.5 搜索功能测试用例

表 5.4 搜索功能测试用例

功能名称		搜索功能		
测试目得		是否能正常检索		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	搜索后进行刷新页面	无	搜索记录依然存在	与期望结果一致
1.2	输入条件进行搜索	无	能够模糊搜索	与期望结果一致
1.3	输入搜索条件	无	能够区分视频状态	与期望结果一致

### 5.1.6 用户操作测试用例

表 5.5 用户操作测试用例

功能名称		用户操作		
测试目得		是否能正常反应		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	点击喜欢	无	喜欢记录保存数据库，前端喜欢按钮变成实心	与期望结果一致
1.2	点击收藏	无	收藏记录保存到数据库，收藏按钮	与期望结果一致

			变成实心	
1.3	点击关注	无	关注记录保存数据库，关注按钮变成实心	与期望结果一致
1.4	发表评论	无	发表评论成功	与期望结果一致

### 5.1.7 消息功能测试用例

表 5.6 消息功能测试用例

功能名称		消息功能		
测试目的		是否能正常反应		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	点击私信	无	能够跳转并私信界面	与期望结果一致
1.2	发送消息	无	成功发送	与期望结果一致
1.3	点击进入聊天界面	无	接收到他人消息	与期望结果一致
1.4	给别人发送过消息后退出在进入	无	能够查询到自己给别人发得消息	与期望结果一致

### 5.1.8 视频审核测试用例

表 5.7 视频审核测试用例

功能名称		消息功能		
测试目的		是否能正常反应		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	获取所有视频审核信息	无	获取到审核信息	与期望结果一致
1.2	审核视频	无	观看视频内容	与期望结果一致

1.3	修改审核状态	无	影响视频播放	与期望结果一致
-----	--------	---	--------	---------

### 5.1.9 观看记录测试用例

表 5.8 观看记录测试用例

功能名称		观看记录		
测试目的		观看视频是否被记录		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	点击一个视频观看	无	有一条视频记录	与期望结果一致
1.2	不停地点击同一个视频观看	无	有一条视频记录	与期望结果一致
1.3	点击多条视频观看	无	有多条视频记录	与期望结果一致

### 5.1.10 关注列表测试用例

表 5.9 关注列表测试用例

功能名称		关注列表		
测试目的		对关注作者进行管理		
预置条件		无		
用例编号	操作步骤	输入数据	期望结果	执行结果
1.1	点击关注列表	无	获取关注分组	与期望结果一致
1.2	点击相应得分组	无	获取相应分组得关注用户	与期望结果一致
1.3	点击相应作者	无	进入作者主页	与期望结果一致
1.4	取消关注	无	取消作者得关注	与期望结果一致
1.5	调整关注分组	无	作者被调到不同得分组	与期望结果一致

## 5.2 性能测试

在性能测试阶段主要针对平均响应时间、处理时间两项指标评判性能优劣。利用 JMeter 对性能进行测试<sup>[19]</sup>。登录注册阶段响应时间和处理时间都小于 1s，其中弹幕系统压力比较大，为保证每条弹幕都能得到处理，使用 RabbitMQ 进行削峰处理，添加死信队列，解决消息以为一些不可知的情况不能被消费引起的报错，对死信队列消息添加过期时间，对消息消费做最后处理。为提高弹幕系统反应效率，使用 ThreadPoolExecutor 线程池进行异步处理，推送弹幕和存储弹幕同时进行；JMeter 压测响应时间由 67ms 降到 19ms，吞吐量由 16/s 增加到 111/s。总体而言，系统相对稳定且响应率较高。

## 5.3 本章小结

系统测试作为必不可少的一个环节，主要涉及功能与性能方面，根据功能测试用例记录情况以及对测试结果的分析，可以确定 Vlog 视频平台功能实现以及性能都达到了预期。

## 第六章 总结及展望

设计并实现基于 SpringBoot 和 Elasticsearch 的 VLOG 共享平台,包括注册登录,视频上传,视频播放,视频审核,视频报料,弹幕,评论,点赞收藏,搜索,主页,私信等功能。通过对用户需求的深入分析,结合先进的技术手段和功能模块的设计,成功打造了视频分享平台,功能丰富,运行稳定,运行高效。用户可以轻松上传、分享、观看平台上的视频,实时与其他用户互动沟通,让自己的影音娱乐体验得到极大的丰富。

未来将继续致力于满足不断增长的用户需求和市场环境变化的平台优化升级。一是加强包括用户体验提升、搜索算法优化、视频内容审核机制强化等在内的平台功能的进一步完著,确保平台持续提供优质服务。其次,将积极拓展平台的社交功能,包括增加用户间的社交、引入个性化推荐功能、开展线上活动等进一步加强用户之间的联系,增强平台的粘性,通过互联网的方式,让用户的社交功能更加紧密,此外,为应对市场竞争和满足用户多样化需求,我们将注重技术创新和业务拓展,积极探索新的技术手段和商业模式,力争把平台打造成行业领先、用户首选的平台。

## 参考文献

- [1] 孙铁强. 基于 SpringBoot 框架得在线监测和专家系统得研究. 工程科技 II 辑; 信息科技. (2021).
- [2] 顾少伟, 井波. Redis 在软件项目中的应用[J]. 电脑编程技巧与维护, 2023(11):16-19+32
- [3] 张玉冰. 计算机软件开发中 Docker 技术应用探讨[J]. 电脑编程技巧与维护, 2023(12).
- [4] 李瑞祥, 王晓东, 王乐乐. 使用 MySQL 对工单数据进行分类汇总统计[J]. 网络安全和信息化, 2023(09):165-166.
- [5] 吴绍卫. WebSocket 在实时消息推送中的应用设计与实现[J]. 福建电脑, 2021, 37(11):80-83. DOI:10.16707/j.cnki.fjpc.2021.11.019.
- [6] 张月. Elasticsearch 得分布式搜索引擎得设计与实现. 信息科技 2020 年第 01 期
- [7] 李明. RabbitMQ 消息队列在分布式系统中得应用研究. 北京: 机械工业出版社. (2020)
- [8] 王龙军, 王晶, 许靖唯. 基于 Spring Boot 与 Vue.js 的问卷模块在数字阅读推广系统中的设计与实现[J]. 内蒙古科技与经济, 2023(18):115-118.
- [9] 高鹏, 高宇, 高梦祎. 弹幕视频网站盈利模式分析——以 B 站为例[J]. 中国商论, 2024(01):41-44. DOI:10.19699/j.cnki.issn2096-0298.2024.01.041.
- [10] 邱宇. 基于 MinIO 分布式存储的微服务模块开发应用[J]. 互联网周刊, 2023(22):38-40.
- [11] 赵元超. 微信与 QQ 功能区别和用户差异分析[J]. 传播力研究, 2018, 2(30):90-91.
- [12] 白茹鑫. 基于 SpringBoot+SSM 框架得企业安全培训管理系统设计与实现. 信息科技. 分类号: TP311.52 (2024).
- [13] 田丽娜. 基于 Elasticsearch 得搜索引擎设计与实现. 信息科技 分类号: TP391.3 (2023)
- [14] 李轲. 原生 Redis 集群得优化与实现. 信息科技, 2019 年第 03 期.
- [15] 罗文辉. 基于 Redis 得实时数据库并发控制算法设计与实现. 信息科技, 2019 年第 02 期. (2019).
- [16] 王磊. Spring Boot 集成 RabbitMQ 实现异步消息处理. 上海: 上海科学技术出版社. (2021).
- [17] 刘文举. 人员定位与门禁管理系统设计 计算机软件及计算机应用 TP311.56 2023.22.05
- [18] 李明华. RabbitMQ 消息队列在分布式系统中得应用研究. 北京: 清华大学出版社. (2020)
- [19] 王靖超. 基于 JMeter 的应用软件性能测试研究[D]. 华北电力大学(北京), 2024. DOI:10.27140/d.cnki.ghbbu.2023.000820.



- [20] Yang Y.Design and Implementation of Student Information Management System Based on Springboot [J]. Advances in Computer, Signals and Systems, 2022, 6
- [21] Yuqing Zhu.Transaction Support over Redis: An Overview.Computer Vision and Pattern Recognition.2017
- [22] 霍育福,金蓓弘,廖肇翊.多模态信息增强得短视频推荐模型[J/OL].浙江大学学报(工学版):1-11[2024-05-01].
- [23] Bastian Greshake Tzovaras Bastian Greshake Tzovaras.Co-Designing a wiki-based community knowledge management system for personal science. Human-Computer Interaction. Thu, 15 Feb 2024 08:54:18 UTC

## 第七章 谢 辞

在此，我要对所有参与这个基于 Spring Boot 和 Elasticsearch 的 vlog 共享平台设计与实现过程的人员表示由衷得感谢。这个项目得成功离不开大家得共同努力和智慧。

在这特殊的时刻，我要把最崇高的敬意献给我敬爱的教官们。在整个工程征途中，老师始终像一盏明灯一样，为我照亮前行之路，是我坚实的支柱。治学态度严谨、悉心指导、耐心教导，使我受益匪浅。在老师的悉心指导下，我对相关的技术和方法能有全面深入的了解和掌握，从而使工程得以顺利开展。在这个过程中，老师教给我的不只是解决问题的方法，更重要的是让我懂得怎样想问题，怎样把理论知识和实际项目结合起来,这对我来说是一笔宝贵的财富,受益终生在此，我深深地向各位教师表示感谢，并致以高的敬意。感恩恩师的教诲，让我受益良多。

感谢所有为这个项目付出过得同学和朋友们！在做得过程中，你们真得帮了我好多，提出了那么多有用得建议和意见。没有你们，这个项目肯定没现在这么完美。非常感谢大家得支持和帮助。

最终，我还要向母校以及所有参与项目得老师和同学们表达最诚挚得感谢。他们是我成长道路上得重要导师和伙伴。我们共同度过了项目实施得艰辛和快乐，共同分享了项目成功的喜悦。感谢这段宝贵经历，让我们共同成长。

## 附录

### 附录 1 视频上传核心代码

```

public Integer upload(HttpServletRequest req) {

    MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest) req;

    // 获得文件分片数据
    // ((MultipartHttpServletRequest) req).getFile()
    MultipartFile file = multipartRequest.getFile("file");

    // 上传过程中出现异常，状态码设置为 50000
    ThrowUtils.throwIf(file == null, ErrorCode.OPERATION_ERROR);

    // 分片第几片
    Integer index = Integer.parseInt(multipartRequest.getParameter("index"));
    // 总片数
    Integer total = Integer.parseInt(multipartRequest.getParameter("total"));
    // 获取文件名
    String fileName = multipartRequest.getParameter("name");
    String totalSize = multipartRequest.getParameter("totalSize");
    String md5 = multipartRequest.getParameter("md5");

    // 创建文件桶
    String bucketName = "md5"+md5;
    minioTemplate.makeBucket(bucketName);
    String objectName = String.valueOf(index);

    log.info("index: {}, total:{}, fileName:{}, md5:{}, objectName:{}, index, total, fileName, md5, objectName);

    Integer processIndex = (Integer) redisTemplate.opsForValue().get(bucketName+objectName);

    // 查看redia 里面的上传进度
    if (index.equals(processIndex)) {
        return 20001;
    }

    // 当不是最后一块时，上传返回的状态码为 20001
    if (index + 1 < total) {

```

```

        try {
            // 上传文件
            OssFile ossFile = minioTemplate.putChunkObject(file.getInputStream(), bucketName, objectName);
            log.info("{} upload success {}", objectName, ossFile);
            // 在 redis 里面记录进度 保存一天时间
            // todo 超过时长没有继续上传，这里的记录删除之后，删除 minio 上传中断，没有继续上传的临时桶
            redisTemplate.opsForValue().set(bucketName, index, 1, TimeUnit.DAYS);

            return 20001;

        } catch (Exception e) {
            e.printStackTrace();
        }
    } else {
        // 为最后一块时状态码为 20002
        try {

            // 上传文件
            minioTemplate.putChunkObject(file.getInputStream(), bucketName, objectName);

            // 上传完成，删除进度
            redisTemplate.delete(bucketName);

            merge(total, bucketName, totalSize, md5);
            return 20002;

        } catch (Exception e) {
            e.printStackTrace();

            return ErrorCode.OPERATION_ERROR.getCode();
        }
    }

    return ResultUtils.success().getCode();
}

```

## 附录 2 私信核心代码

```

public void onMessage(String message) {

```

```

        Session s = chatOnlineMap.get(reUserId);
        ChatMsg chatMsg = new ChatMsg();
        chatMsg.setMessage(message);
        chatMsg.setSendUserId(this.myUserId);
        chatMsg.setAcceptUserId(this.reUserId);
        ChatServiceImpl chatService= (ChatServiceImpl)ScrollingWebsocketController
        r.APPLICATION_CONTEXT.getBean("chatServiceImpl");
        if(s==null){
            chatService.syncSaveChat(chatMsg);
        }else {
            try {
                s.getBasicRemote().sendText(message);
                chatService.asyncSaveChat(chatMsg);
            } catch (Exception e){
                System.out.println(e);
            }
        }
    }
}

@Override
public void syncSaveChat(ChatMsg chatMsg) {
    chatMapper.saveMsg(chatMsg);
}

@Override
public void asyncSaveChat(ChatMsg chatMsg) {
    threadPoolExecutor.submit()->{
        chatMapper.saveMsg(chatMsg);
    });
}

@Override
public List<Map> getChatMsg(Long userId) {
    Long currentUserId = userSupport.getCurrentUserId();
    List<ChatMsg> msgs = chatMapper.getChatMsg(currentUserId,userId);
    List<Map> msgMaps = new LinkedList<>();
    for (ChatMsg msg : msgs) {
        Map<String, Object> stringObjectMap = BeanUtil.beanToMap(msg);
        Long sendUserId = msg.getSendUserId();
        Long acceptUserId = msg.getAcceptUserId();
        UserInfo userInfoBySendUserId = userMapper.getUserInfoByUserId(send
        dUserId);
        UserInfo userInfoByAcceptUserId= userMapper.getUserInfoByUserId(ac
        ceptUserId);
        stringObjectMap.put("sendUserImage",userInfoBySendUserId.getImage());
    }
}

```

```

        stringObjectMap.put("acceptUserImage",userInfoByAcceptUserId.getImag
e());
        stringObjectMap.put("sendUserNickname",userInfoBySendUserId.getNick
name());
        stringObjectMap.put("acceptUserNickname",userInfoByAcceptUserId.get
Nickname());
        msgMaps.add(stringObjectMap);
    }
    return msgMaps;
}

```

### 附录 3 弹幕核心代码

```

@OnMessage
public void onMessage(String message) {
    // ThrowUtils.throwIf(this.userId==null, ErrorCode.NOT_LOGIN_ERROR);
    log.info("sessionId {} 发来消息{}", session.getId(), message);
    Scrolling scrolling = JSONObject.parseObject(message, Scrolling.class);
    ThrowUtils.throwIf(scrolling == null, ErrorCode.PARAMS_ERROR);
    scrolling.setUserId(this.userId);
    scrolling.setVideoId(Long.valueOf(this.videoId));
    ScrollingService scrollingService = (ScrollingService) APPLICATION_CON
TEXT.getBean("scrollingServiceImpl");
    RabbitTemplate rabbitTemplate = (RabbitTemplate) APPLICATION_CONT
EXT.getBean("rabbitTemplate");

    RabbitMQUtil.asyncSendMessage(scrolling, rabbitTemplate);

    scrollingService.saveScroller(scrolling);
}

public static void asyncSendMessage(Scrolling message, RabbitTemplate rabbit) {
    executorService.submit(() -> {
        RabbitTemplate rabbitTemplate=rabbit;
        rabbitTemplate.convertAndSend(RabbitMQConfig.SCROLLING_EXCHA
NGE,
        RabbitMQConfig.SCROLLING_EXCHANGE, JSONUtil.toJson
Str(message));
    });
}

```

## 附录 4 视频记录核心代码

```

@Override
public void addVideoRecord(VideoRecord videoRecord) {

    Long currentUserId = userSupport.getCurrentUserId();
    videoRecord.setUserId(currentUserId);
    String videoRecordKey = videoRecordPrefix + currentUserId + "-" + videoRecord.getId();
    redisTemplate.opsForValue().set(videoRecordKey, JSONUtil.toJsonStr(videoRecord));

}

@Scheduled(fixedRate = 1000)
public void addVideoRecordToDB() {

    Set<String> keys = redisTemplate.keys(videoRecordPrefix + "*");
    for (String key : keys) {
        String s = redisTemplate.opsForValue().get(key);
        VideoRecord videoRecord = JSONUtil.toBean(s, VideoRecord.class);
        List<VideoRecord> videoRecords = videoMapper.getVideoRecord(videoRecord);

        if (videoRecords==null||videoRecords.isEmpty()) {

            videoMapper.addVideoRecordToDB(videoRecord);
        } else {
            videoMapper.updateVideoRecordToDB(videoRecord);
        }
        redisTemplate.delete(key);
    }
}

@Override
public List<VideoInfo> getVideoRecord() {
    Long currentUserId = userSupport.getCurrentUserId();
    LinkedList<VideoInfo> videoInfos = new LinkedList<>();
    //重复
    HashSet<String> repeatKey = new HashSet<>();
    Set<String> keys = redisTemplate.keys(videoRecordPrefix+ currentUserId +
    "*"");

```

```

        if (keys!=null&&!keys.isEmpty()){
            for (String key : keys) {
                String s = redisTemplate.opsForValue().get(key);
                VideoRecord videoRecord = JSONUtil.toBean(s,VideoRecord.class);

                VideoInfo videoInfoByVideoId = videoMapper.getVideoInfoByVideoId(videoRecord.getVideoId());
                videoInfos.add(videoInfoByVideoId);
                repeatKey.add(videoRecord.getUserId()+videoRecord.getVideoId()+"");
            }
        }

        List<VideoRecord> videoRecord = videoMapper.getVideoRecordByUserId(currentUserId);
        for (VideoRecord record : videoRecord) {
            VideoInfo videoInfoByVideoId = videoMapper.getVideoInfoByVideoId(record.getVideoId());
            int size = repeatKey.size();
            repeatKey.add(record.getUserId()+record.getVideoId()+"");
            if (size!=repeatKey.size()&&videoInfoByVideoId!=null){
                videoInfos.add(videoInfoByVideoId);
            }
        }
        return videoInfos;
    }
}

```