深圳技术大学
SHENZHEN TECHNOLOGY UNIVERSITY

本科毕业论文（设计）

题目：基于 Springboot 的弹幕视频网站后端系统设计与实现

姓　　名：潘俊霖

学　　院：大数据与互联网学院

专　　业：物联网工程

学　　号：202100201068

指导教师：孙瑞泽

职　　称：助理教授

提交日期：2025 年 5 月 5 日

深圳技术大学本科毕业论文（设计）诚信声明

本人郑重声明：所呈交的毕业论文（设计），题目《基于 Springboot 的弹幕视频网站后端系统设计与实现》是本人在指导教师的指导下，独立进行研究工作所取得的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明。除此之外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。本人完全意识到本声明的法律结果。

毕业论文（设计）作者签名：

日期：　　年　　月　　日

目 录

基于 Springboot 的弹幕视频网站后端系统设计与实现

【摘　要】

【关键词】Springboot；视频在线播放；弹幕系统

Design and implementation of the back-end system of the barrage video website based on Springboot University

【Abstract】

【Key words】 SZTU; LaTeX; Template

1 引言

1.1 研究背景与意义

随着 5G、云计算与边缘计算等新一代网络技术的高速发展与推广，当今的互联网内容生态正经历从图文静态展示向动态视频交互的变革。视频媒介凭借视听一体化的沉浸式体验，大幅降低了用户信息处理成本，契合当下碎片化时代的认知习惯，已成为信息传播与娱乐消费的核心载体。据第 55 次《中国互联网络发展状况统计报告》[1] 显示，截至 2024 年 12 月，我国网民规模达 11.08 亿人，在文旅娱乐类应用的分类中，网络视频用户规模达 10.70 亿人，占网民整体的 96.6%，远高于同分类下的其他应用，充分印证了当前互联网内容生态以视频媒介为主导的现状。

在 Web3.0 与元宇宙技术浪潮推动下，互联网正从信息交互的 1.0 时代向价值交互的 3.0 时代跃迁，用户对视频平台的需求也随之发生深刻变革。相较于早期以内容单向输出为主的视频网站，如今的用户更倾向于在观看过程中获得深度参与感与沉浸式社交体验。这种需求的转变，不仅体现在对高清画质、流畅播放等基础功能的追求上，更聚焦于构建双向互动、实时共享的虚拟空间。

弹幕作为实时互动的典型创新，通过即时评论叠加形成群体共鸣，打破了传统视频观看的时空壁垒。在观看视频时，用户发送的弹幕会以动态文字的形式实时叠加在视频画面上，让不同地域、不同时间的观众得以在同一虚拟空间中交流互动。这种“边看边聊”的模式，将原本孤立的观看行为转化为集体参与的社交活动，创造出独特的虚拟社交场域。用户不再是被动的内容接收者，而是成为内容传播与共创的重要参与者。

这种实时交互模式不仅显著提升了用户参与感，更催生出丰富的衍生文化现象。例如，热门视频中的经典台词、精彩片段往往会通过弹幕传播形成网络热梗，进而引发二次创作热潮。用户基于弹幕内容进行的二次创作，如剪辑、配音、图文解析等，进一步扩大了视频内容的传播范围与影响力。此外，弹幕文化还形成了独特的语言体系与互动规则，如“前方高能”“弹幕护体”等特定用语，成为 Z 世代网络社交的重要符号。弹幕不仅是一种互动方式，更成为 Z 世代表达自我、寻找同好、构建社群的重要载体，深刻影响着当代网络文化的发展走向。因此在用户需求持续迭代与行业竞争加剧的背景下，视频平台不仅需要保障视频播放流

畅，画质清晰等的基础功能，平台还需要拓展以弹幕功能为核心的实时评论、用户互动、社交分享等功能。

随着 5G、云计算技术的普及，互联网视频内容生产呈现多元化格局。其中职业生产内容 (Occupationally Generated Content OGC) 依托专业团队输出精品版权作品，用户生产内容（Professionally Generated Content，UGC）借助移动设备记录生活百态，专业生产内容（Professionally Generated Content，PGC）则由 MCN 机构打造兼具专业性与娱乐性的优质内容。[2] 多元生产模式推动视频内容爆发式增长，主流平台日均新增视频量突破千万级。面对海量内容，传统分类浏览已无法满足用户需求。在此背景下，检索系统成为了视频平台必不可缺的一部分功能。

基于以上分析，本文设计并实现了一个基于 Springboot 的视频弹幕网站，它不仅能实现视频在线播放的基础功能，同时还添加了视频弹幕推送、投币、评论等的互动功能。

1.2 国内外研究现状

当前，全球多数视频网站的核心业务聚焦于视频内容供应，通过会员付费、广告投放等模式，为用户提供视听服务。在这一竞争格局下，优质内容已然成为行业竞争的关键要素。各平台主要通过自制或引进热门视频资源吸引用户，但在视频功能拓展方面投入相对不足，如用户观点交互渠道搭建、视频创作参与度提升等领域的开发仍有待加强。

从用户粘性角度来看，缺乏互动的单向视频观看模式，使得平台过度依赖内容数量与质量维系用户留存。视频内容的生产与版权采购成本高昂，自制内容面临同质化、制作经验匮乏、意识形态把控不足等挑战；版权采购则加剧平台间的竞争，进一步推高运营成本。

在国际视频平台中，YouTube 以专业用户生产内容（PUGC）为特色，内容多元且国际化程度高。平台通过创作者激励机制，营造浓厚的原创氛围，相较于传统内容采购模式，有效降低版权成本。但该平台功能相对单一，缺乏弹幕互动、博客发布等社交功能，用户与订阅者的互动基本局限于视频投稿。[3]Netflix 作为全球最大的付费视频平台，凭借早期 DVD 租赁业务积累的用户付费习惯，成功实现向线上平台的转型。平台始终将优质内容视为发展核心，以会员订阅为主要营收来源，凭借高质量的剧集内容，吸引大量付费用户。然而，对内容品质的过度依

赖，使得版权采购成本居高不下，且需持续产出优质内容以维持用户粘性。

国内视频平台如哔哩哔哩，初期聚焦二次元内容创作与分享，近年来通过拓展内容分区实现多元化发展，平台内容形成了以 UGC 和 PGC 为核心，PGC 为辅的局面。依托兴趣社区的运营优势，哔哩哔哩在弹幕互动、用户社交等方面表现突出，高互动率形成正向循环，持续吸引用户参与。但受发展历程影响，其内容覆盖领域相对较窄，主要集中于年轻用户偏好的领域，难以满足全年龄段用户的观看需求。反观国内其他主流视频网站如爱奇艺、腾讯视频等，多采用专业生产内容（PGC）模式，与 Netflix 的运营逻辑相似，版权竞争激烈。与此同时，普遍存在内容质量欠佳、社区运营薄弱等问题，叠加高成本内容投入、单一盈利模式等因素，导致行业整体长期处于亏损状态，会员付费与广告收入难以覆盖内容制作与版权采购成本。

1.3 主要研究内容

(1) 对国内外视频平台发展现状进行了分析，设计并实现了以视频在线播放为基础功能、弹幕功能为核心的视频弹幕网站后端系统，为用户提供了一个在线看视频同时还能进行弹幕交互的平台。
(2) 为用户提供了包括点赞、收藏、投币、关注等的一系列交互功能。同时还进一步添加了弹幕查询、在线人数查询、全局内容搜索、观看记录等功能。

1.4 论文组织结构

全文内容共四章，具体内容组织如下：

第一章为前言。介绍了本文的研究背景与意义，以及国内外相关研究现状，阐明了主要研究内容。

第二章为相关技术分析。介绍了本文实现视频弹幕网站所用涉及到的相关技术。

第三章为需求分析。对本文所实现的视频弹幕网站后端系统从功能性和非功能性两个角度进行需求分析。

第四章为系统设计。

第五章为系统实现。

第六章总结与展望，总结了本文的主要工作，展望了下一阶段的研究方向。

2 相关技术分析

2.1 SpringBoot

SpirngBoot 是基于 Spring 框架构建的开发框架，旨在简化 Java 应用的开发与配置，帮助使用者快速打造独立、生产级别的应用程序，在当今后端开发领域应用广泛。

它采用“约定优于配置”的原则，通过自动配置机制，能够自动装配数据库连接、Web 服务器等组件，大幅减轻了开发者的配置负担；提供丰富的启动器，可以快速搭建项目，让开发者能够专注业务逻辑；支持将应用打包成为可执行的 JAR 或 WAR 文件独立运行，简化了开发者的部署流程；同时还支持微服务架构，继承众多即插即用的功能，便于微服务的独立部署与拓展，显著提升了开发效率和应用的可用性。

2.2 MySQL 数据库

MySQL 数据库是一个开源的关系型数据库管理系统，它能够为开发者提供高效、可靠的数据存储和管理功能，现在属于 Oracle 公司。它以其中高性能、可靠性和易用性而闻名，被广泛应用于从个人网站到大型企业级应用的各种规模应用。

MySQL 数据库主要具有高性能、可拓展性、安全性、易用性、跨平台、开源和社区支持等特点。[4]

(1) 高性能：MySQL 被设计用于处理高并发和大数据量的应用，支持如 nnoDB、MyISAM 等多种储存引擎，能够提供快速的数据读写能力同时通过索引、查询优化和缓存机制，MySQL 能够高效地处理复杂的查询和事务。
(2) 可拓展性：MySQL 支持主从复制和分片技术，能够轻松拓展数据库的容量和性能，适合大型应用。通过增加更多服务器（水平拓展）和增加单个服务器的资源（垂直拓展），MySQL 能够满足不断增长的数据请求。
(3) 安全性：MySQL 提供多种安全特性，如用户权限管理，SSL 加密、数据加密等，能够确保数据的安全性。同时通过细粒度的权限控制，MySQL 能够保护敏感数据免于遭受未授权的访问。

(4) 易用性：依赖于命令行工具和 MySQL Workbench 这一类的图形化管理工具，使得数据库的管理和操作都变得简单。开发者可以轻松地创建、修改和查询数据库，但无需深入了解复杂的数据库知识。

(5) 跨平台：MySQL 可以在包括 Windows、Linux、macOS 等多种操作系统上运行，提供良好的跨平台兼容性。这使得开发者能够在不同开发环境中使用 MySQL 而无需担心兼容性问题。

(6) 开源和社区支持：作为一款开源的数据库管理系统，MySQL 拥有活跃的社区支持，开发者可以从中获取丰富的教程、文档和插件。同时社区驱动的开发模式也使得 MySQL 能够快速响应市场需求，从而不断地改进和更新。

2.3 FastDFS 分布式存储

FastDFS 是一款基于 C 语言开发的轻量级开源高性能分布式文件系统。主要功能包含文件存储、文件同步、文件访问（上传/下载）。FastDFS 解决了在大容量场景下的文件存储和高并发访问的问题，同时是了文件存储时的负载均衡，特别适合以文件为载体的中大型网站的在线服务，如照片共享网站、视频共享网站等（图片、文档、音频、视频等）。[5]

和其他的分布式文件系统相比较，FastDFS 文件系统通过取消了存储元数据的服务器，实现了元数据的可内存化，达到了提高检索效率的目的。如图2.1所示，FastDFS 文件系统总共分为了跟踪服务器（Tracker Server）、存储服务器（Stroage Server）和客户端（client）三部分。[6]

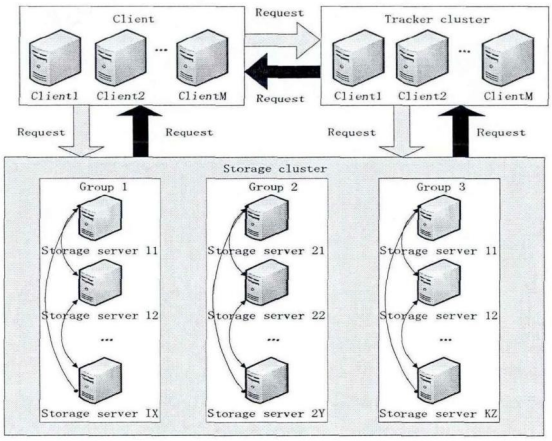

图 2.1　FastDFS 架构图

FastDFS 分布式存储主要有着高性能、高可用和易拓展的特点。

(1) 高性能：从设计之初 FastDFS 就致力于处理高并发的文件上传和下载请求，通过文件分片和负载均衡技术，它能够高效地处理大小文件，支持大量的文件存储和访问，提供快速的文件读写能力。

(2) 高可用性：FastDFS 采用了主从架构，支持文件冗余存储，这确保了在某个节点发生故障时文件仍然可用。并且 FastDFS 还通过心跳检测和自动故障转移这两个机制，保证了系统的稳定性和可靠性。

(3) 易拓展性：为满足不断增长的文件存储需求，FastDFS 支持动态拓展存储节点，因此能轻松增加存储容量。开发者可以通过简单的配置就可以实现快速部署和拓展 FastDFS 集群。

2.4 WebSocket 实时通信

WebSockets 是一种网络通讯协议，被用于在 Web 应用程序中实现实时、双向的通信通道。[7] 与传统的 HTTP 请求的一次请求一次响应不同，WebSocket 允许服务器和客户端之间建立一个持久连接，允许服务器即时向客户端推送数据同时也就可以接收客户端发送的数据，实现实时数据传输。

如2.2所示，WebSocket 建立在普通的 HTTP 协议上，所有的 WebSocket 请求都会通过普通的 HTTP 协议发送出去，WebSokcet 通过 HTTP/1.1 协议中的 Upgrade 头信息来告诉服务器，希望协议从 HTTP/1.1 升级到 WebSocket 协议[8]。服务器会根据 HTTP 协议识别特定头信息 Upgrade，同时也会判断请求信息中的 Upgrade 是否存在。在这个过程中 HTTP 协议是必不可少的。

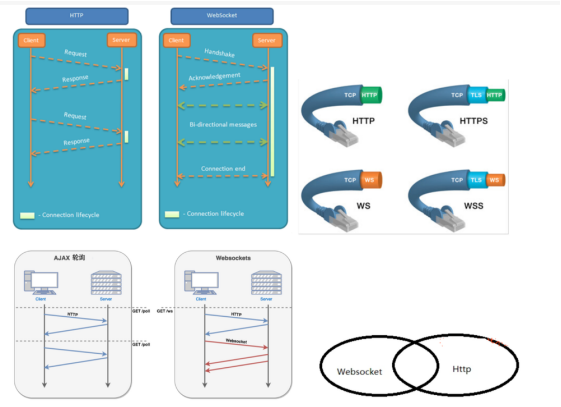

图 2.2 WebSocket 原理

开发者可以通过 WebSocket 的使用构建更加动态和响应迅速的 Web 应用，达到提升用户体验的目的。

2.5 Elasticsearch 搜索引擎

Elasticsearch 是一个基于 Lurcene 库构建的开源搜索引擎。提供了强大的全文搜索、结构化搜索、分析功能，被广泛应用于日志分析、实时监控、数据挖掘等领域。它通过 RESTful 接口提供搜索结果。Elasticsearch 的分布式架构将索引细分为多个分片，这些分片随后可复制到集群内的多个服务器（节点）上。正因如此，Elasticsearch 能够应对高查询负载，平均分配处理负载，并弥补任何服务器故障带来的影响。这种架构支持即使在高负载情况下，搜索结果也能及时反映数据变化[9]。

像亚马逊这样的大型企业提供专门的 Elasticsearch 服务，Github 和 StackOverflow 都成功使用了这款搜索和分析引擎。通过 Elasticsearch 项目能够满足用户对快速、准确搜索的需求。

3 需求分析

3.1 业务概述

在快节奏的现代生活中，视频凭借其生动形象的特点，逐渐成为人们获取信息、放松娱乐的首选方式。相较于单调的文字与静态的图片，视频融合了画面与声音，能为人们带来更加丰富和沉浸式的体验。人们可以借助视频学习各类知识，在忙碌的工作之余通过观看有趣的视频缓解压力，还能在探索不同视频内容的过程中培养新的兴趣爱好。

然而，随着网络技术的迅猛发展，网络上的视频数量呈现出爆炸式增长1.1。面对数以亿计的视频资源，用户想要找到自己真正感兴趣的内容变得愈发困难。而且，不同用户查找视频的习惯大相径庭。部分用户偏好通过浏览自己感兴趣的视频分类来筛选内容；有些用户则更倾向于在搜索框中输入关键词进行查找；还有一些用户不喜欢主动搜索，期望视频网站能依据自身喜好，精准地为其推荐可能感兴趣的视频。由此可见，视频网站只有实现上述功能，才能满足不同用户的多样化需求。

此外，互联网的蓬勃发展不仅改变了人们获取信息的方式，还极大地激发了人们的社交欲望1.1。如今，人们在观看视频时，不再仅仅满足于被动观看，而是迫切希望表达自己对视频内容的独特见解，与其他用户进行互动交流。在这样的趋势下，视频网站若仅提供基础的视频观看功能，显然无法满足用户的需求。因此搭建多样化的视频评价渠道和社交平台，成为视频网站发展的必然趋势。这不仅有助于促进用户之间的思想交流，进一步提升网站视频的质量，还能增强用户对网站的认同感和归属感，有效提高视频网站的用户粘合度，使网站在竞争激烈的网络环境中占据优势地位。

3.2 功能性需求分析

弹幕视频网站后端系统的目标非常明确，即通过建立弹幕视频网站平台，实现以视频在线播放为基础功能、弹幕功能为核心的视频弹幕网站后端系统，为用户提供了一个在线看视频同时还能进行弹幕交互的平台(1)。

根据系统目标，可以明确弹幕视频网站后端系统的功能性需求主要有五个，分

别是用户管理、视频管理、社交功能、权限管理、搜索功能和实时通信功能模块。

3.2.1 用户管理需求

如3.1所示，用户管理功能包括用户注册、用户登录、用户信息管理和用户角色管理这四个功能。

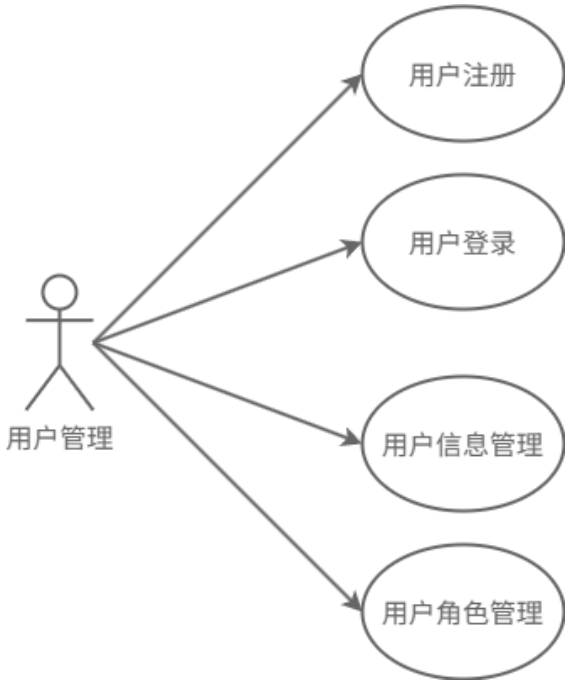

图 3.1　用户管理用例图

用户注册功能旨在为用户创建在本系统内的专属账户，实现用户身份的唯一标识与管理，为用户使用系统各项功能提供前提条件。用户在注册页面输入用户名、密码、邮箱等必要信息，其中用户名需遵循系统规定的命名规范，密码需达到一定强度要求，邮箱必须为有效的邮箱地址。系统对用户输入的信息进行有效性验证，若全部有效，系统将创建新的用户账户，并将相关信息存储至数据库，同时向用户返回“注册成功”的提示信息；若信息存在无效情况，系统将返回相应的错误信息，提示用户重新输入正确信息。

用户登录功能是用户访问系统的入口，通过验证用户身份，确保只有授权用户能够使用系统的各项功能，保障系统的安全性和用户数据的保密性。用户在登录页面输入注册时使用的用户名和密码，系统将其与数据库中存储的信息进行匹配验证。若匹配成功，系统允许用户登录，并返回“登录成功”的信息，同时为用户生成会话标识，用户可凭借此标识在会话期间访问系统功能；若不匹配，系

统返回错误提示，阻止非法访问。

用户信息管理功能允许已登录用户查看和修改个人信息，满足用户个性化需求，确保用户信息的准确性和时效性。用户登录系统后，通过导航菜单进入个人信息页面，可查看当前已有的个人信息。如需修改，可在相应编辑区域对用户名、密码、邮箱等信息进行修改，完成修改并提交后，系统保存修改后的信息，并返回“更新成功”的提示，确认信息修改操作已成功执行。

用户角色管理功能由系统管理员负责操作，通过对用户角色的管理和权限分配，实现系统资源的合理访问控制，确保系统的正常运行和数据安全。系统管理员登录系统后，进入用户角色管理页面，可查看当前系统中所有用户的角色列表。管理员根据业务需求，为用户分配新的角色或移除已有的角色，完成操作后，系统保存角色分配信息。

3.2.2 视频管理需求

如3.2所示，视频管理功能包括视频上传、视频播放、弹幕功能这三个功能。

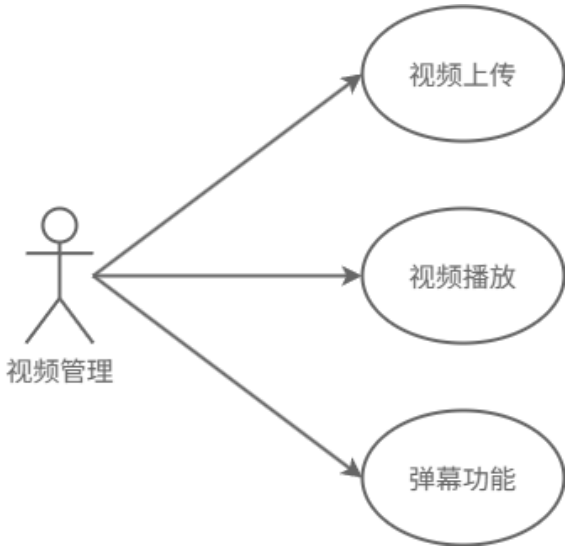

图 3.2　视频管理用例图

视频上传功能赋予用户向系统提交视频文件的权限，丰富系统的视频资源库，满足用户分享内容的需求。用户登录系统后，通过导航菜单进入视频上传页面。在此页面，用户可选择本地要上传的视频文件。系统随即对所选视频文件的格式和大小进行严格验证，确保其符合系统预设的标准。若视频文件符合要求，系统会将其上传至存储系统，并向用户返回“上传成功”的提示信息；若不符合要求，系

统会返回具体的错误信息，如“文件格式不支持，请选择正确格式的视频文件”或“文件大小超出限制，请选择合适大小的视频”，引导用户重新选择文件。

视频播放功能为用户提供了观看已上传视频的服务，满足用户的视听娱乐需求。用户登录系统后，进入视频播放页面。在该页面，用户可从众多视频资源中选择想要播放的视频。系统接收到用户的选择指令后，会快速加载所选视频文件，并自动开始播放。播放过程中，用户可通过播放控制按钮，如播放、暂停、音量调节、进度拖动等，灵活控制视频播放，获得个性化的观看体验。

弹幕功能增强了用户在视频观看过程中的互动体验，促进用户之间的交流与分享。当用户在视频播放页面观看视频时，可在专门的输入区域输入弹幕内容。输入完成后，用户点击发送按钮，系统会将弹幕信息发送到服务器。服务器借助 WebSocket 技术，将弹幕信息实时推送给所有正在观看该视频的用户。这些弹幕会在视频播放界面上以动态形式显示，用户不仅能看到自己发送的弹幕，还能实时浏览其他用户发送的弹幕，实现用户之间的实时互动交流。

3.2.3 社交功能需求

如3.3所示，社交功能包括用户关注、用户动态、用户硬币这三个功能。

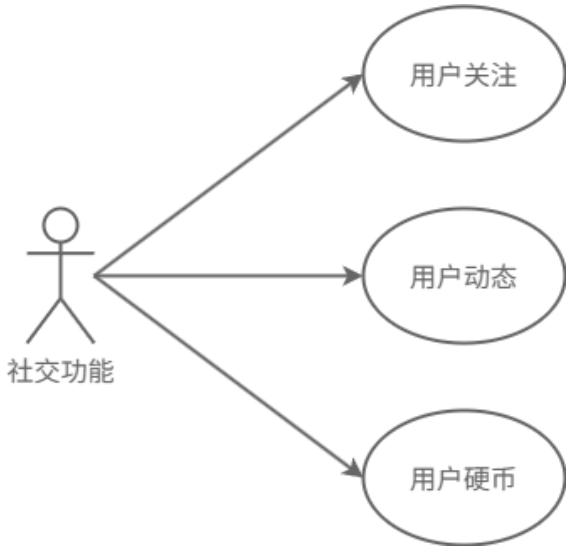

图 3.3　社交功能用例图

用户关注功能为用户搭建起社交关系网络，方便用户与其他用户建立联系，拓展社交圈子。用户登录系统后，通过搜索或浏览等方式进入其他用户的个人主页。在对方的个人主页上，用户点击“关注”按钮，系统会迅速将这一关注请求发送

到服务器。服务器接收到请求后，对其进行处理，更新该用户的关注列表，同时也会更新被关注用户的粉丝列表。完成处理后，系统返回“关注成功”的信息，此时用户在自己的关注列表页面，即可看到刚刚关注的用户，成功建立起与其他用户的关注关系。

用户动态功能给予用户分享自身状态、见解以及各类内容的平台，促进用户间的信息交流与互动。用户登录系统后，找到并进入动态发布页面。在这个页面中，用户可以在输入框内输入想要分享的动态内容，还能根据需求选择添加本地的图片或视频来丰富动态展示效果。当用户完成内容编辑后，点击“发布”按钮，系统会将包含文字、图片或视频等的动态信息发送到服务器。服务器接收到信息后，将其保存至数据库，并向用户返回“发布成功”的提示。随后，用户发布的动态不仅会展示在自己的个人主页上，还会出现在关注者的动态流中，让关注者能够及时看到该用户的分享内容，实现用户间的信息共享与互动。

用户硬币作为一种虚拟资产功能，为用户提供了购买内容或打赏其他用户的途径，激励内容创作与平台互动。用户登录系统后，通过导航栏找到并进入硬币管理页面。在此页面，用户能够直观地看到当前自己拥有的硬币数量。若用户有购买内容或打赏其他用户的需求，可在页面中选择相应的操作选项，并按照提示完成后续步骤。系统接收到用户的硬币使用请求后，会对请求进行处理，从用户的硬币余额中扣除相应数量，并同步更新用户的硬币余额数据，完成操作后返回“使用成功”的信息，告知用户硬币已成功使用。

3.2.4 权限管理需求

如3.4所示，权限管理功能包括角色管理、菜单管理、操作权限管理这三个功能。

角色管理功能赋予系统管理员对系统内角色进行全面管控的能力，通过创建、修改和删除角色，并合理分配权限，实现对不同用户群体访问系统资源和执行操作的精细控制，确保系统的安全性和稳定性。系统管理员登录系统后，通过特定的入口进入角色管理页面。在该页面，管理员能够查看当前系统中已存在的角色列表，了解各个角色的基本情况。若需要创建新角色，管理员可进行相应操作，设置角色名称和描述，明确该角色的功能和适用场景。对于现有角色，管理员可以根据实际业务需求，对角色名称、描述等信息进行修改。当某些角色不再适用于

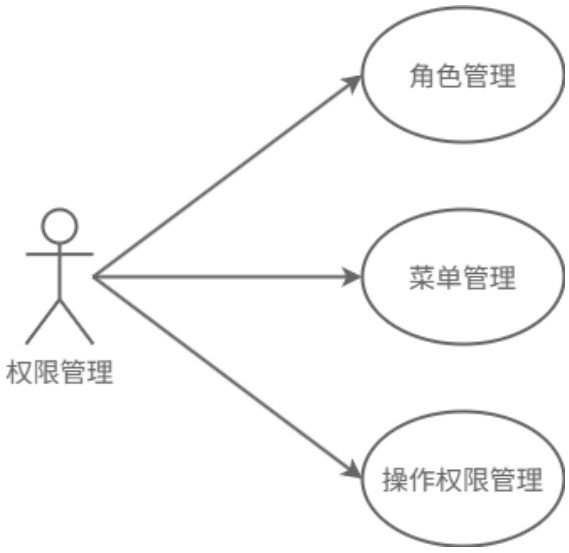

图 3.4　权限管理用例图

系统时，管理员可将其删除。在完成上述任何操作后，系统会保存角色信息，并向管理员返回“更新成功”的提示，确认角色管理操作已生效。

菜单管理功能让系统管理员能够灵活管理系统菜单，通过设置不同角色对菜单项的访问权限，实现系统界面展示的个性化和资源访问的安全性。系统管理员登录系统后，进入菜单管理页面。在此页面，管理员可以查看当前系统的菜单列表，清晰了解菜单的构成和布局。管理员可根据业务需求创建新菜单项，设置菜单名称、路径和图标，使新菜单项能够准确对应相应的功能模块，并在界面上有清晰的展示。对于已有的菜单项，若有调整需求，管理员可以修改其名称、路径等信息。对于不再使用的菜单项，管理员可将其删除。系统在保存这些菜单信息变更后，会返回“更新成功”的信息，告知管理员菜单管理操作已成功完成。

操作权限管理功能帮助系统管理员对系统内的操作权限进行有效管理，通过明确不同角色可执行的操作，保障系统的正常运行和数据安全。系统管理员登录系统后，进入操作权限管理页面。在该页面，管理员可以查看当前系统的操作权限列表，掌握各个操作权限的设置情况。当业务需要新的操作权限时，管理员可以创建新操作权限，详细设置权限名称和描述，界定该权限的具体作用。对于已有的操作权限，管理员可以根据实际情况修改其名称、描述等信息。若某些操作权限不再使用，管理员可将其删除。系统在保存操作权限信息的变动后，会返回“更新成功”的信息，表明操作权限管理操作已顺利执行。

3.2.5 搜索功能需求

如3.5所示，搜索功能包括视频搜索和数据分析这两个功能。

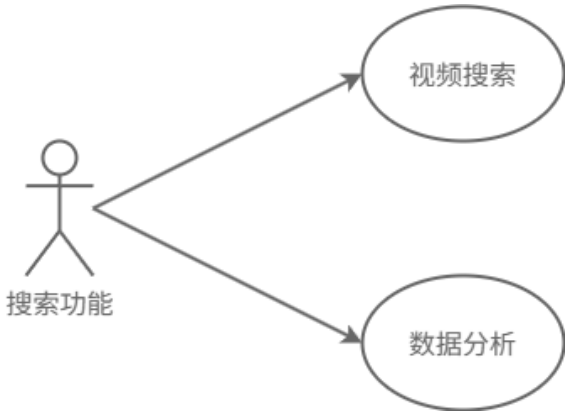

图 3.5 搜索功能用例图

视频搜索功能为用户提供了在海量视频资源中快速定位所需内容的途径，满足用户对特定视频的查找需求。用户登录系统后，通过导航栏或快捷入口进入搜索页面。在搜索页面的搜索框中，用户输入想要查找视频的关键词，这些关键词可以是视频标题中的字词、视频描述里的关键信息等。输入完成后，用户点击“搜索”按钮，系统会将包含关键词的搜索请求发送到服务器。服务器借助 Elasticsearch 强大的全文搜索能力，在视频数据库中进行精准查找，筛选出与关键词相关的视频。随后，系统将搜索到的视频结果展示给用户，用户可以看到视频的标题、描述、上传时间等信息，以便判断该视频是否为自己所需，进而进行选择观看。

数据分析功能帮助系统管理员深入了解用户行为和系统运行状况，为优化系统功能、提升用户体验提供数据支持。系统管理员登录系统后，找到并进入数据分析页面。在该页面，管理员可以根据业务需求选择要分析的数据类型，比如用户行为数据，可用于研究用户的浏览习惯、搜索偏好等；或者选择视频热度数据，以此了解不同视频的受欢迎程度。选定数据类型后，管理员进一步设置分析的时间范围，如近一周、一个月等，以及其他相关条件，如特定地区的用户数据、特定类型视频的数据等。系统利用 Elasticsearch 对符合条件的数据进行分析处理，并生成统计报表。最后，系统将分析结果以图表和报表的形式展示给管理员，管理员通过查看这些图表和报表，能够直观地了解用户需求和系统性能，为后续的系统优化和决策提供有力依据。

3.2.6 实时通信需求

如3.6所示，实时通信功能包括弹幕功能、实时通知和聊天功能这三个功能。

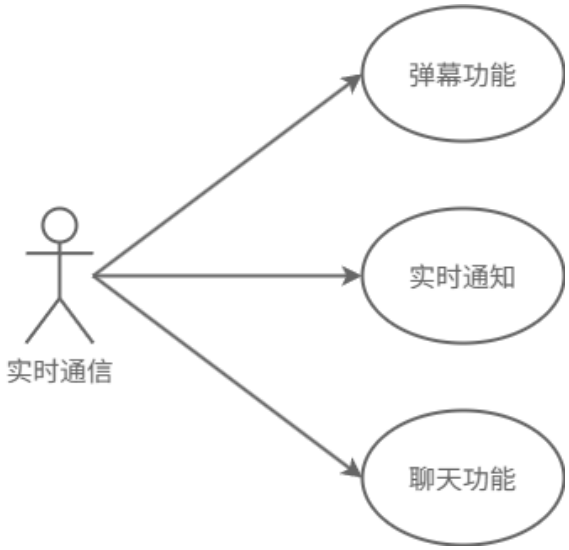

图 3.6 实时通信用例图

弹幕功能极大地丰富了用户在视频观看过程中的互动体验，让用户能够在观看视频时实时分享自己的想法，增强用户之间的交流。当用户登录系统并进入视频播放页面后，可在专门设置的弹幕输入区域输入弹幕内容。完成输入后，用户点击“发送”按钮，系统会迅速将弹幕信息发送到服务器。服务器借助 WebSocket 技术，把该弹幕信息实时推送给所有正在观看同一视频的用户。这些弹幕会以动态形式在视频播放界面上显示出来，这样，每个用户在观看视频时，不仅能看到视频内容，还能实时浏览其他用户发送的弹幕，实现用户之间的即时互动，为视频观看增添乐趣。

实时通知功能确保系统能够及时向用户传达重要信息，如新消息提醒、系统公告等，使用户能够及时知晓与自身相关的事务和系统动态。系统在检测到需要推送的通知，如新消息到达、发布系统公告等情况时，会通过 WebSocket 将通知信息发送到服务器。服务器接收到通知信息后，会根据用户的相关信息，将通知实时推送给对应的用户。用户在登录系统后的界面上会收到通知提示，点击提示即可查看通知的具体内容，从而及时了解系统的最新动态和个人相关信息，保证用户与系统之间的信息同步。

聊天功能为用户搭建了一个实时交流的平台，方便用户与其他用户进行一对

一或群组沟通，满足用户的社交需求。用户登录系统后进入聊天页面，在聊天页面中，用户可以选择想要聊天的对象，既可以是单个用户发起一对一聊天，也可以在群组聊天场景中选择相应群组。选定聊天对象后，用户在聊天框中输入想要发送的消息内容，输入完成后点击“发送”按钮，系统会将消息发送到服务器。服务器利用 WebSocket 技术，把消息实时推送给接收者。接收者在自己的聊天界面上会收到消息提醒，点击即可查看消息内容，实现用户之间高效、便捷的实时交流，促进用户之间的社交互动。

3.3 非功能性需求分析

系统非功能性需求指的是除功能性需求之外的特性，其中包括了软件质量、运行环境、外部接口等要求。非功能性需求从另外一个角度衡量一个系统是否符合用户需求，它要求我们在设计软件的时候不仅仅满足软件可用的要求，还要使软件易用，因此，非功能性需求与功能性需求同样重要[10]。以下是本弹幕视频网站后端系统的非功能性需求：

(1) 性能需求：作为一款视频网站，系统需要强大的高并发处理能力以确保大量用户同时使用的响应速度。除此之外为确保用户在高并发场景下仍然能够快速获取数据，实现流畅的操作，系统还应该优化网络传输机制以降低数据延迟。
(2) 可用性需求：系统应该具备高可用性，通过冗余设计对关键组件进行备份，确保当某个组件发生故障的时候系统能仍然能够持续正常与和运行。同时系统还要有完备的容错机制，能够自动检测和处理各种异常状况，减小异常状况发生时给用户带来的影响，保证服务的连续性。
(3) 安全性需求：据安全至关重要，系统需采用多种加密技术，对用户数据进行加密存储和传输，防止数据被窃取或篡改。在用户认证方面，要支持强身份验证机制，如多因素认证，确保只有授权用户能够访问系统。
(4) 可拓展性需求：系统应该具备良好的可拓展性以应付业务的增长与用户数量的增加，因此在设计之初就应该支持水平拓展和模块化设计。水平拓展通过增加节点提高了系统的存储与查询能力。模块化设计则便于日后的维护和拓展。
(5) 可维护性需求：系统的代码应具有良好的可读性，遵循统一的编程规范和代码风格，添加详细的注释，方便开发人员理解和修改代码。同时，建立完善的日志管理系统，记录系统运行过程中的关键事件、错误信息等。通过分析日志，开发人员可以快速定位问题，进行故障排查和修复，提高系统的维护效率。

4 系统设计

4.1 系统架构设计

4.1.1 系统分层设计

本系统采用基于 SpringBoot 与 SpringCloud 框架构建的微服务架构，系统架构图如4.1所示。微服务架构的特点是将系统拆分成为了多个独立的服务，每个服务负责特定的功能，服务间通过轻量级通信机制（通常为 HTTP REST API）进行交互[11]。这种架构设计使系统具有高可用性、可拓展性和灵活性的特点。

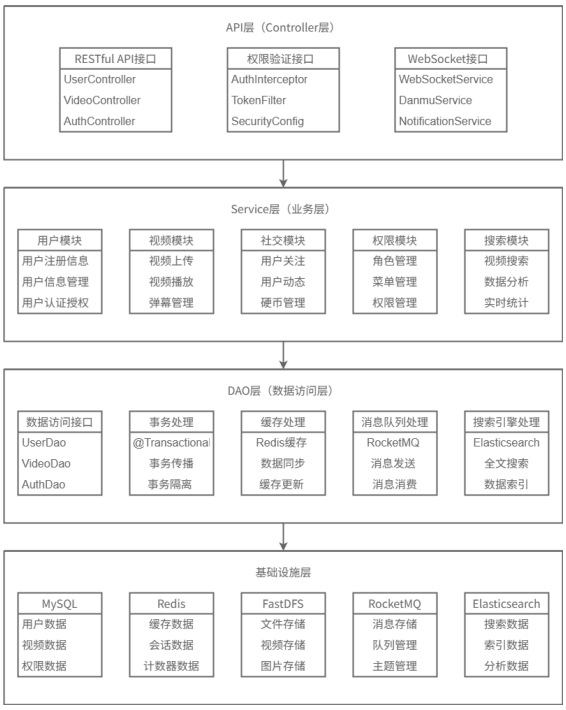

图 4.1 系统架构图

在本系统的微服务下，请求在该架构下的传递过程如图4.2所示。

首先由客户端向本系统发送包含用户操作意图（如获取数据、执行某个功能）的 HTTP 或 WebSocket 请求，这个请求将被发送至 API 层。在 API 层对从接受的请求进行初步的验证和处理后，API 层将会将请求转发给 Service 层，这一步处理是为了让接下来的 Service 层能够进行具体的业务处理。Service 层在接受来自 API 层的请求后会根据业务需要进行逻辑处理，在处理完成后 Service 将会将数据操作的请求发送给 DAO 层。DAO 层在接受数据操作请求后，会使用特定的数据

访问技术和接口来操作基础设施层的数据存储。基础设施层在执行具体数据存储或读取操作后，再将结果返回给 DAO 层。DAO 层在获取到数据操作结果后，结果将会被逐层向上处理包装，最终返还给客户端，完成了整个请求流程。

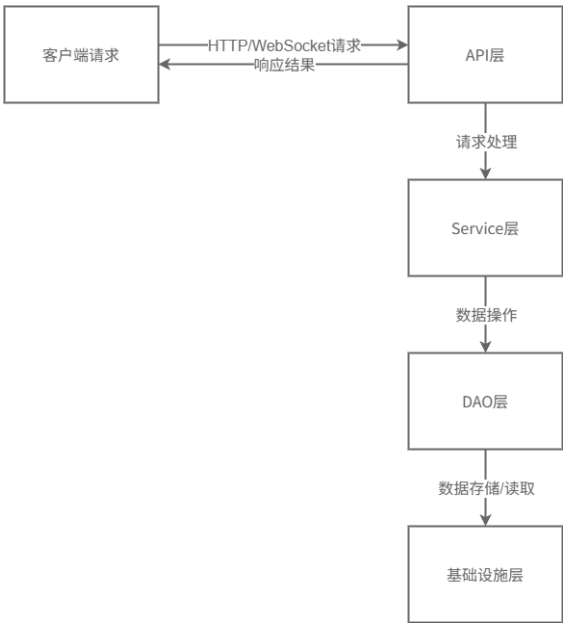

图 4.2　请求传递过程

4.1.2　系统功能模块划分

如图4.3所示，本系统共分为用户管理、视频管理、社交互动、权限管理、搜索服务、消息通知、文件存储、系统监控共八大模块。

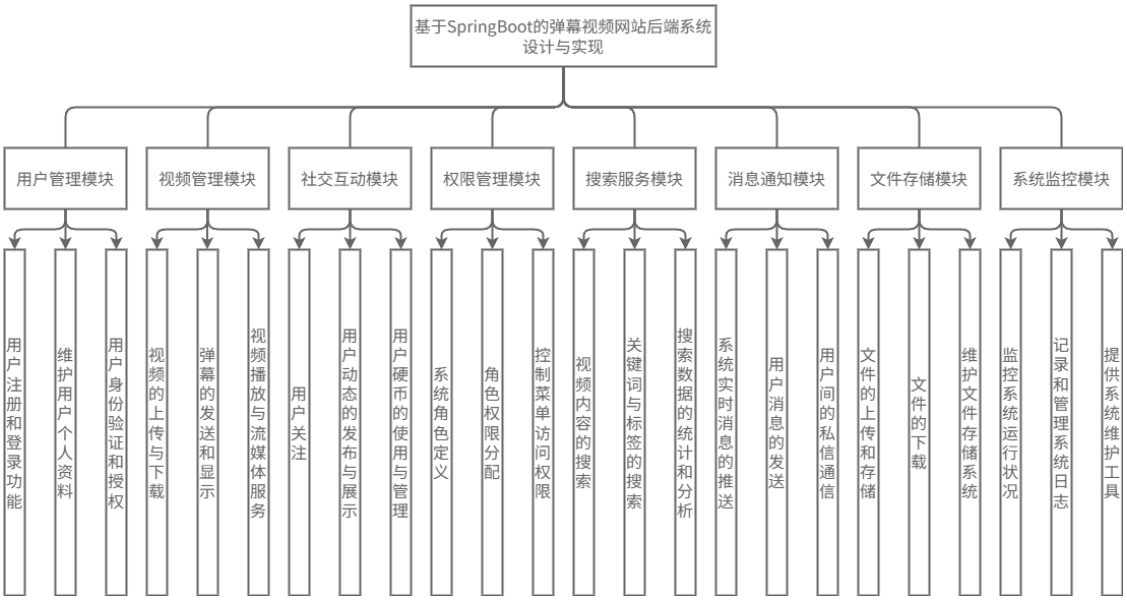

图 4.3　系统功能模块图

垂直关系上，每个模块都遵循分层架构，实现架构层级由上而下的垂直调用。水平关系上，模块共享基础设施服务，模块间通过接口进行通信，模块解耦通过事件驱动实现。依赖关系上，权限模块为其他模块提供认证授权服务，文件存储模块为视频模块提供存储服务，消息模块为其他模块提供通知服务。

4.2 系统数据库设计

4.2.1 数据库概念结构设计

开发者在充分掌握用户需求的基础上，边可以开始展开对数据库的概念结构设计，即将用户相对抽象的需求进行一定的具象化，使之成为能够指引后续设计过程的概念模型[12]。本弹幕视频网站系统的数据库所使用的数据库类型是 MySQL 数据库，数据库实体关系图如4.4。

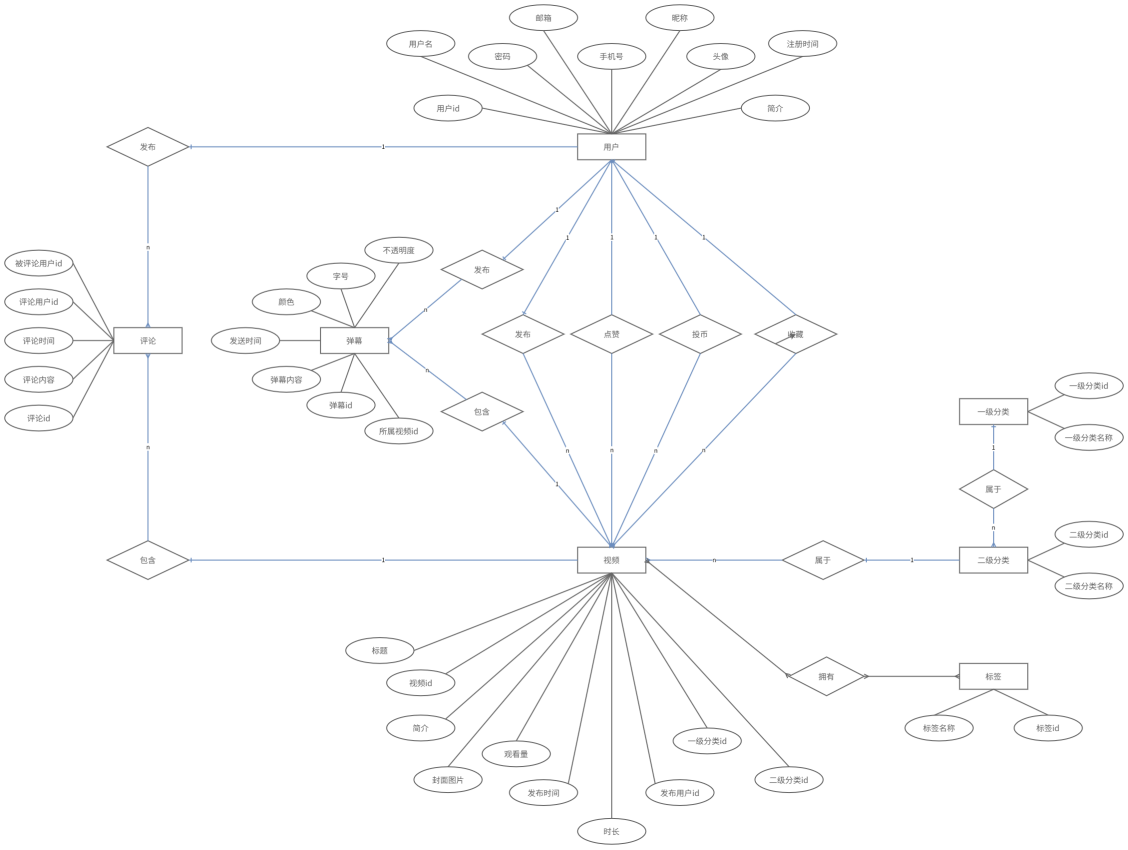

图 4.4　实体关系图

根据实体关系图4.4可以知道，数据库中的实体主要包括用户 User、视频 Video、弹幕 Danmu、评论 Comment、一级分类 Category1、二级分类 Category2 以及标签 Tag。对于本数据库中的用户、视频、弹幕、评论四个主要实体的具体描述属性如

下：

(1) 用户实体负责储存系统中的所有用户信息，包括用户 id、用户名、昵称、头像、邮箱、手机号、密码、注册时间、简介。用户实体图如图4.5所示。

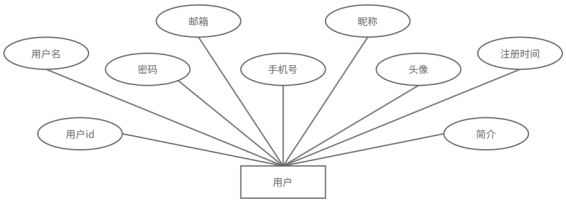

图 4.5　用户实体图

(2) 视频实体负责存储系统中的所有的视频信息，包括标题、视频 id、简介、封面图片、观看量、发布时间、市场、发布用户 id、一级分类 id、二级分类 id 。视频实体图如图4.6所示。

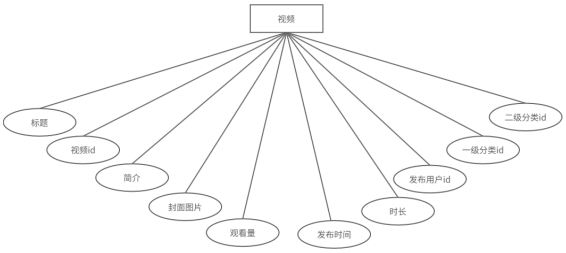

图 4.6　视频实体图

(3) 弹幕实体负责存储系统中的弹幕信息，包括弹幕内容、不透明度、颜色、字号、发送时间、弹幕 id、所属视频 id。弹幕实体图如图4.7所示。

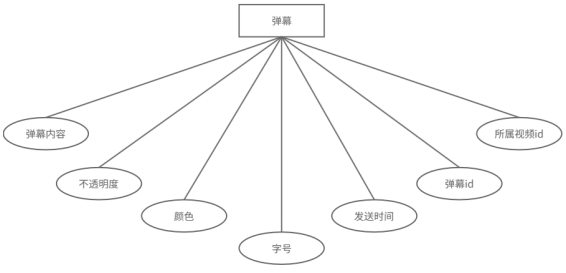

图 4.7　弹幕实体图

(4) 评论实体负责储存系统中的评论信息，包括评论用户 id、评论时间、被评论用户 id、评论内容、评论 id。评论实体图如图4.8所示

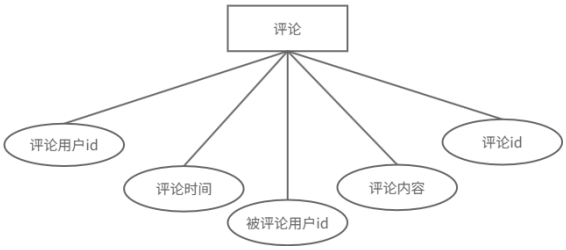

图 4.8　评论实体图

其中用户与视频、用户与评论、用户与弹幕、视频与评论、视频与弹幕、一级分类与二级分类均属于一对多的关系。视频与二级分类属于多对一的关系。视频与标签属于多对多的关系。

4.2.2　数据库逻辑结构设计

基本表是数据的存储过程建立的基础，因此在数据库设计时应首先建好表[13]。根据弹幕视频网站后端系统的业务需求分析以及数据库设计原则，将基本实体图转换成 MySQL 支持的数据模型相符合的逻辑结构，采用合适的数据类型，完成系统数据表的设计。

本数据库名称叫做 imooc-bilibili，数据库中包括 4 张数据表：用户相关表、视频相关表、权限相关表以及其他功能表。每个表都包含主键 id、创建时间 creatTime、更新时间 updatetime（部分表）和相关外键关联字段。

以下列出主要的数据表：

(1) 用户相关表

用户相关表主要存储用户相关信息，如下表4.1所示。

(2) 视频相关表

视频相关表主要存储视频相关信息，如下表4.2所示。

(3) 权限相关表

权限相关表主要储存权限控制的相关信息，如下表4.3所示。

(4) 其他功能表

其他功能表主要存储剩余功能的相关信息，如下表4.4所示。

表 4.1　用户相关表

表名	说明	主要字段
t_user	用户表	id, phone, email, password, salt, createTime, updateTime
t_user_info	用户基本信息表	id, userId, nick, avatar, sign, gender, birth, createTime, updateTime
t_user_following	用户关注表	id, userId, followingId, groupId, createTime
t_user_moments	用户动态表	id, userId, type, contentId, createTime, updateTime
t_user_coin	用户硬币表	id, userId, amount, createTime, updateTime
t_user_role	用户角色关联表	id, userId, roleId, createTime
t_refresh_token	刷新令牌记录表	id, userId, refreshToken, createTime

表 4.2　视频相关表

表名	说明	主要字段
t_video	视频投稿记录表	id, userId, url, thumbnail, title, type, duration, area, description, createTime, updateTime
t_video_comment	视频评论表	id, videoId, userId, comment, replyUserId, rootId, createTime, updateTime
t_video_coin	视频硬币表	id, userId, videoId, amount, createTime, updateTime
t_video_collection	视频收藏记录表	id, videoId, userId, groupId, createTime
t_video_like	视频点赞记录表	id, userId, videoId, createTime
t_video_tag	视频标签关联表	id, videoId, tagId, createTime
t_video_view	视频观看记录表	id, videoId, userId, clientId, ip, createTime
t_video_binary_picture	视频二值图记录表	id, videoId, frameNo, url, videoTimestamp, createTime
t_video_operation	视频操作表	id, userId, videoId, operationType, createTime

表 4.3　权限相关表

表名	说明	主要字段
t_auth_role	权限控制-角色表	id, name, code, createTime, updateTime
t_auth_menu	权限控制- 页面访问表	id, name, code, createTime, updateTime
t_auth_ element_operation	权限控制- 页面元素操作表	id, elementName, elementCode, operationType, createTime, updateTime
t_auth_role_menu	权限控制- 角色页面菜单关联表	id, roleId, menuId, createTime
t_auth_role_ element_operation	权限控制- 角色与元素操作关联表	id, roleId, elementOperationId, createTime

表 4.4　其他功能表

表名	说明	主要字段
t_danmu	弹幕记录表	id, userId, videoId, content, danmuTime, createTime
t_file	文件表	id, url, type, md5, createTime
t_following_group	用户关注分组表	id, userId, name, type, createTime, updateTime
t_collection_group	收藏分组表	id, userId, name, type, createTime, updateTime
t_tag	标签表	id, name, createTime
t_content	动态内容表	id, contentDetail, createTime

5 系统实现

5.1 技术栈、开发环境与开发工具介绍

5.1.1 开发环境

该项目的开发环境整合了多种系统工具与中间件服务。系统环境上，以 Windows 10（版本 10.0.22631）为操作系统，搭配 Java8 作为开发语言环境，利用 Maven 进行项目构建，通过 Git 实现版本控制。中间件方面，MySQL8.0 负责数据库服务，用于存储结构化数据；Redis 提供缓存服务，加速数据读取；RocketMQ 作为消息队列服务，处理异步任务；Elasticsearch 实现搜索服务，满足数据检索需求；FastDFS 承担文件存储服务，管理非结构化文件。这些系统工具和中间件协同工作，为项目开发、运行提供了稳定且高效的环境基础。

5.1.2 技术栈

在后端框架层面，因 SpringBoot 快速开发、自动配置等的特性**??**能够减少大量繁琐的配置工作，极大提升了开发效率，所以选定 SpringBoot2.5.1 为基础框架。在此之上，引入 SpringCloud 微服务框架，在该框架下，本系统借助 Netflix Eureka 实现服务注册与发现，实现了各个微服务之间能够相互识别与通信，保障系统的动态管理与扩展；OpenFeign 用于服务间通信，提供了声明式的 Web 服务客户端，简化了服务调用过程；Hystrix 实现服务熔断机制，当某个服务出现故障时，能及时切断请求，防止故障扩散，确保系统整体的稳定性。

数据存储方面，选用 MySQL8.0 作为关系型数据库，负责存储结构化数据，如用户信息、视频元数据等，其强大的数据处理和事务管理能力为系统提供了可靠的数据支持。Redis 作为缓存数据库，将频繁访问的数据存储在内存中，加速数据读取，减轻数据库压力。Elasticsearch 作为搜索引擎，实现高效的视频搜索功能，支持复杂的全文检索等操作。FastDFS 分布式文件存储则用于存储视频文件等大量非结构化数据，保证文件存储的高可靠性和可扩展性。

在消息队列方面选用了 RocketMQ 它承担了异步处理的任务，在系统发起处理视频上传、弹幕发送等异步任务时，它能够起到削峰填谷、解耦应用等作用。能够提升系统的整体性能和响应速度。

其他技术中，MyBatis 作为 ORM 框架，方便地实现了 Java 对象与数据库表之间的映射，简化了数据持久化操作。WebSocket 实现实时通信，为弹幕功能提供技术支撑，使用户能实时发送和接收弹幕消息。JWT 用于用户认证，通过生成和验证 JSON Web Token，确保只有合法用户能够访问系统资源，保障系统安全。Mahout 作为推荐引擎，分析用户行为数据，为用户推荐个性化的视频内容。JavaCV 则专注于视频处理，例如对上传的视频进行格式转换、剪辑等操作，满足多样化的视频处理需求。

5.1.3 开发工具

本次系统开发选择 Intelli IDEA 作为主要开发工具，它集成了众多实用功能。通过搭配以下多个关键插件使用，能够实现简化代码编写、提升开发效率的目的，进一步增强了开发能力。

(1) Lombok：通过注解自动生成常用代码，减少冗余代码量，简化了 Java 对象的代码编写。
(2) MyBatisX：顾名思义为 MyBatis 框架开发提供便利，支持 XML 与注解两种开发方式，能够快速生成 SQL 语句、映射文件等。
(3) SrpingBoot Assistant：专门针对 SpringBoot 项目，提供快速创建项目、自动配置等的功能，加速应用开发进程。
(4) Maven Helper：优化了 Maven 项目管理，方便管理项目依赖、解决依赖冲突等问题。

除 IDEA 外，本系统还使用了 Cursor 编辑器作为辅助编辑器。它是一款具备 AI 辅助编程功能的编辑器，其 AI 代码补全功能能够根据上下文推测并补全代码，提高代码编写错误；代码重构的功能可以对现有代码优化，提升代码的可读性和可维护性。同时，智能提示功能可以实时为开发者提供代码建议，代码分析功能则能够检测代码中像语法错误这样的潜在问题，助力开发者编写出高质量的代码。

此外，我们还有几种不同的工具搭配使用来保障项目的顺利开发。Postman 用于 API 测试，能方便地发送各种 HTTP 请求，对开发的 API 接口进行功能测试、性能测试等，确保 API 的正确性和稳定性。Navicat 作为专业的数据库管理工具，方便管理 MySQL 等数据库，可进行数据库设计、数据操作、备份恢复等工作。Redis Desktop Manager 专门用于 Redis 数据库管理，以图形化界面展示 Redis 中的数据，

方便进行数据查看、修改、删除等操作。Git 作为版本控制工具，结合 Maven 的项目依赖管理功能，实现对项目代码和依赖的有效管理，方便团队协作开发，追踪代码变更历史，管理项目版本。Github Desktop 作为 GitHub 为 Git 开发的图形化工具，它大大简化了 Git 与 Github 的使用流程，尤其适合不熟悉命令行操作的开发者。

5.2 核心架构搭建

本系统主要使用基于 SpringBoot 框架的微服务架构搭建而成的 Maven 项目工程，从主架构图5.1 可知，本系统在由 API 模块、服务模块和数据访问模块三个子模块构成，这三个子模块的详细结构如图5.2（以 API 模块为例）。

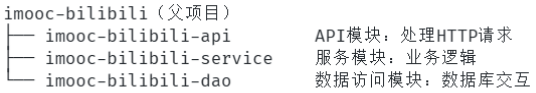

图 5.1　主架构图

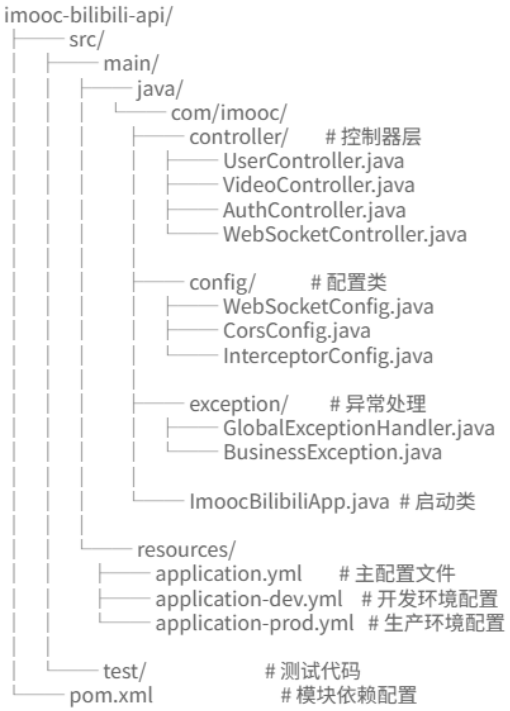

图 5.2　子模块结构

在父项目的目录下的 pom.xml 文件，它是 Maven 项目的核心配置文件。首先它包含着项目的基本信息，这些信息用作唯一标识一个 Maven 项目；其次 POM

文件还可以管理项目的依赖关系，通过 <dependencies> 标签定义项目所需要的库，例如图5.3

```
<!-- 依赖管理 -->
<dependencyManagement>
    <dependencies>
        <!-- Spring Cloud依赖 -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>${spring.cloud.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<!-- 公共依赖 -->
<dependencies>
    <!-- Spring Boot Web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- MySQL驱动 -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>${mysql.version}</version>
    </dependency>

    <!-- MyBatis -->
    <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
        <version>${mybatis.version}</version>
    </dependency>

    <!-- Redis -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-redis</artifactId>
    </dependency>
```

图 5.3　POM 依赖管理

5.3 关键模块实现

5.3.1 用户模块

在系统的用户服务模块图5.4中，UserServiceImpl 类承担着核心的业务逻辑处理工作，它实现了 UserService 接口，借助 @Service 注解被 Spring 容器识别为服务组件。

```java
@Service
public class UserServiceImpl implements UserService {
    @Autowired
    private UserDao userDao;
    @Autowired
    private RedisTemplate redisTemplate;

    // 用户注册
    public void register(UserRegisterDTO registerDTO) {
        // 1. 参数校验
        validateRegisterParams(registerDTO);

        // 2. 检查用户是否存在
        if(userDao.getUserByPhone(registerDTO.getPhone()) != null) {
            throw new BusinessException("用户已存在");
        }

        // 3. 密码加密
        String salt = RandomUtil.generateSalt();
        String encryptedPassword = MD5Util.encrypt(registerDTO.getPassword(), salt);

        // 4. 创建用户
        User user = new User();
        user.setPhone(registerDTO.getPhone());
        user.setPassword(encryptedPassword);
        user.setSalt(salt);
        userDao.addUser(user);

        // 5. 创建用户信息
        UserInfo userInfo = new UserInfo();
        userInfo.setUserId(user.getId());
        userInfo.setNick(registerDTO.getNick());
        userDao.addUserInfo(userInfo);
    }

    // 用户登录
    public String login(UserLoginDTO loginDTO) {
        // 1. 参数校验
        validateLoginParams(loginDTO);

        // 2. 获取用户信息
        User user = userDao.getUserByPhone(loginDTO.getPhone());
        if(user == null) {
            throw new BusinessException("用户不存在");
        }

        // 3. 密码验证
        String encryptedPassword = MD5Util.encrypt(loginDTO.getPassword(), user.getSalt());
        if(!encryptedPassword.equals(user.getPassword())) {
            throw new BusinessException("密码错误");
        }

        // 4. 生成token
        String token = JwtUtil.generateToken(user.getId());

        // 5. 保存登录状态
        redisTemplate.opsForValue().set(
            "user:token:" + user.getId(),
            token,
            24,
            TimeUnit.HOURS
        );

        return token;
    }
}
```

图 5.4　用户模块伪代码

用户注册功能当用户发起注册请求时，系统会调用 register 方法。具体处理流程如下：

(1) 参数校验：对用户提交的注册信息进行初步的合法性检查，确保数据的完整性和准确性。
(2) 用户存在性检查：通过 UserDao 访问数据库，依据用户提供的手机号码查询是否已有该用户存在。若存在，则抛出 BusinessException 异常，提示“用户已存在”。
(3) 密码加密处理：为保证用户密码的安全性，系统会生成一个随机的盐值，然后使用 MD5 加密算法结合盐值对用户输入的密码进行加密。
(4) 用户记录创建：创建一个 User 对象，将加密后的密码和盐值存储其中，并通过 UserDao 将用户信息持久化到数据库。
(5) 用户信息创建：接着创建一个 UserInfo 对象，将用户的昵称等信息存储其中，并与用户 ID 关联，最后同样通过 UserDao 将用户信息保存到数据库。

用户登录功能当用户尝试登录系统时，系统会调用 login 方法。具体处理步骤如下：

(1) 参数校验：对用户提交的登录信息进行基本的格式和完整性检查。
(2) 用户信息获取：通过 UserDao 根据用户输入的手机号码从数据库中获取对应的用户信息。若未找到该用户，则抛出 BusinessException 异常，提示“用户不存在”。
(3) 密码验证：使用相同的 MD5 加密算法和数据库中存储的盐值对用户输入的密码进行加密，然后将加密后的密码与数据库中存储的密码进行比对。若不一致，则抛出 BusinessException 异常，提示“密码错误”。
(4) Token 生成：若密码验证通过，系统会使用 JWT（JSON Web Token）工具为用户生成一个唯一的令牌，用于标识用户的身份。
(5) 登录状态保存：将生成的令牌存储到 Redis 缓存中，并设置 24 小时的有效期，以此来保存用户的登录状态。最后，将生成的令牌返回给用户，用于后续的身份验证和访问授权。

通过上述功能的实现，该用户服务模块为系统提供了安全、可靠的用户注册和登录服务，有效地保障了用户信息的安全和系统的正常运行。

5.3.2 视频模块

如图5.5在视频模块的服务实现中，VideoServiceImpl 类发挥着关键作用。该类通过 @Service 注解被 Spring 容器识别为服务组件，实现了 VideoService 接口，为视频的上传和播放功能提供了具体的业务逻辑处理。

视频上传功能，当用户发起视频上传请求时，系统会调用 uploadVideo 方法，具体的处理流程如下：

(1) 参数校验：对用户提交的视频上传信息进行初步的合法性检查，确保上传的文件和相关参数完整且准确。
(2) 借助 FastDFSClient 将用户上传的视频文件存储到分布式文件系统中，并获取视频文件的存储路径。
(3) 封面图上传：同样使用 FastDFSClient 上传视频的封面图，并获取封面图的存储路径。
(4) 视频信息保存：创建一个 Video 对象，将用户 ID、视频路径、封面图路径、视频标题和描述等信息存储其中，然后通过 VideoDao 将视频信息持久化到数据库。
(5) 视频索引创建：调用 ElasticsearchService 在 Elasticsearch 中为该视频创建索引，以便后续的搜索和查询操作。

视频播放功能，当用户请求播放视频时，系统会调用 getVideoById 方法，具体处理步骤如下：

(1) 视频信息获取：通过 VideoDao 根据视频 ID 从数据库中获取对应的视频信息。若未找到该视频，则抛出 BusinessException 异常，提示“视频不存在”。
(2) 视频统计信息获取：使用 VideoDao 获取该视频的统计信息，如播放量、点赞数等。
(3) 视频标签获取：通过 VideoDao 获取该视频的相关标签信息。
(4) 返回数据组装：创建一个 VideoDTO 对象，将视频信息、统计信息和标签信息进行整合，然后返回给用户，以便用户能够完整地获取视频相关信息进行播放。

通过上述功能的实现，该视频服务模块为系统提供了完善的视频上传和播放

```java
@Service
public class VideoServiceImpl implements VideoService {
    @Autowired
    private VideoDao videoDao;
    @Autowired
    private FastDFSClient fastDFSClient;
    @Autowired
    private ElasticsearchService elasticsearchService;

    // 视频上传
    public void uploadVideo(VideoUploadDTO uploadDTO) {
        // 1. 参数校验
        validateUploadParams(uploadDTO);

        // 2. 上传视频文件
        String videoUrl = fastDFSClient.uploadFile(
            uploadDTO.getVideoFile(),
            uploadDTO.getVideoFile().getOriginalFilename()
        );

        // 3. 上传封面图
        String thumbnailUrl = fastDFSClient.uploadFile(
            uploadDTO.getThumbnailFile(),
            uploadDTO.getThumbnailFile().getOriginalFilename()
        );

        // 4. 保存视频信息
        Video video = new Video();
        video.setUserId(uploadDTO.getUserId());
        video.setUrl(videoUrl);
        video.setThumbnail(thumbnailUrl);
        video.setTitle(uploadDTO.getTitle());
        video.setDescription(uploadDTO.getDescription());
        videoDao.addVideo(video);

        // 5. 创建视频索引
        elasticsearchService.createVideoIndex(video);
    }

    // 视频播放
    public VideoDTO getVideoById(Long videoId) {
        // 1. 获取视频信息
        Video video = videoDao.getVideoById(videoId);
        if(video == null) {
            throw new BusinessException("视频不存在");
        }

        // 2. 获取视频统计信息
        VideoStats stats = videoDao.getVideoStats(videoId);

        // 3. 获取视频标签
        List<Tag> tags = videoDao.getVideoTags(videoId);

        // 4. 组装返回数据
        VideoDTO videoDTO = new VideoDTO();
        BeanUtils.copyProperties(video, videoDTO);
        videoDTO.setStats(stats);
        videoDTO.setTags(tags);

        return videoDTO;
    }
}
```

图 5.5　视频模块伪代码

服务，确保了视频数据的有效存储和展示。

5.3.3 弹幕模块

在弹幕服务模块中图5.6，DanmuServiceImpl 类是核心实现，它通过 @Service 注解被 Spring 容器识别为服务组件，实现了 DanmuService 接口，为弹幕的发送和获取功能提供了具体业务逻辑。

```java
@Service
public class DanmuServiceImpl implements DanmuService {
    @Autowired
    private DanmuDao danmuDao;
    @Autowired
    private WebSocketServer webSocketServer;

    // 发送弹幕
    public void sendDanmu(DanmuDTO danmuDTO) {
        // 1. 参数校验
        validateDanmuParams(danmuDTO);

        // 2. 保存弹幕
        Danmu danmu = new Danmu();
        danmu.setUserId(danmuDTO.getUserId());
        danmu.setVideoId(danmuDTO.getVideoId());
        danmu.setContent(danmuDTO.getContent());
        danmu.setDanmuTime(danmuDTO.getDanmuTime());
        danmuDao.addDanmu(danmu);

        // 3. 实时推送
        webSocketServer.sendDanmu(danmu);
    }

    // 获取弹幕列表
    public List<DanmuDTO> getDanmuList(Long videoId) {
        // 1. 获取弹幕列表
        List<Danmu> danmuList = danmuDao.getDanmuList(videoId);

        // 2. 转换为DTO
        return danmuList.stream()
            .map(this::convertToDTO)
            .collect(Collectors.toList());
    }
}
```

图 5.6　弹幕模块伪代码

弹幕发送功能，当用户发送弹幕时，系统会调用 sendDanmu 方法，详细处理步骤如下：

(1) 参数校验：对用户发送的弹幕数据进行初步检查，确保数据的完整性和合法性，避免无效数据进入后续处理流程。

(2) 弹幕保存：创建一个 Danmu 对象，将用户 ID、视频 ID、弹幕内容以及弹幕发送时间等信息存入其中。接着，通过 DanmuDao 将该弹幕信息持久化到数据库，以便后续查询和分析。

(3) 实时推送：利用 WebSocketServer 将新发送的弹幕实时推送给正在观看该视频

的其他用户，保证弹幕的即时性和交互性。

弹幕列表获取功能，当需要获取某个视频的弹幕列表时，系统会调用 getDanmuList 方法，具体处理如下：

(1) 弹幕数据查询：通过 DanmuDao 根据视频 ID 从数据库中查询该视频对应的所有弹幕信息。
(2) 数据转换：将查询到的 Danmu 对象列表转换为 DanmuDTO 列表。通过 Java 8 的 Stream API 对每个 Danmu 对象进行映射转换，最终收集为一个新的 DanmuDTO 列表并返回，以满足前端展示需求。

通过上述功能实现，该弹幕服务模块为视频系统提供了高效的弹幕交互能力，保证了弹幕的存储、推送和展示功能的稳定运行。

5.3.4 搜索模块

如图5.7在搜索服务模块中，SearchServiceImpl 类起着关键作用，它通过 @Service 注解被 Spring 容器识别为服务组件，实现了 SearchService 接口，为视频搜索和视频索引创建功能提供了具体的业务逻辑实现。

视频搜索功能，当用户发起视频搜索请求时，系统会调用 searchVideos 方法，具体处理流程如下：

(1) 搜索条件构建：根据用户传入的 SearchDTO 对象，调用 buildSearchRequest 方法构建符合 Elasticsearch 要求的搜索请求对象，该对象包含了用户的搜索条件。
(2) 执行搜索操作：将构建好的搜索请求对象传递给 ElasticsearchService 的 search 方法，在 Elasticsearch 中执行搜索操作，并获取搜索响应结果。
(3) 搜索结果解析：对 Elasticsearch 返回的搜索响应结果进行解析，调用 parseSearchResponse 方法将搜索结果转换为 VideoDTO 列表，方便后续处理和展示。
(4) 分页处理：将解析后的视频列表、用户请求的页码、每页数量以及搜索结果的总数量封装到 PageResult 对象中，实现搜索结果的分页展示，提高用户体验。

视频索引创建功能，当需要为新视频创建索引时，系统会调用 createVideoIndex 方法，具体步骤如下：

```java
@Service
public class SearchServiceImpl implements SearchService {
    @Autowired
    private ElasticsearchService elasticsearchService;
    @Autowired
    private VideoDao videoDao;

    // 视频搜索
    public PageResult<VideoDTO> searchVideos(SearchDTO searchDTO) {
        // 1. 构建搜索条件
        SearchRequest request = buildSearchRequest(searchDTO);

        // 2. 执行搜索
        SearchResponse response = elasticsearchService.search(request);

        // 3. 解析结果
        List<VideoDTO> videoList = parseSearchResponse(response);

        // 4. 分页处理
        return new PageResult<>(
            videoList,
            searchDTO.getPageNum(),
            searchDTO.getPageSize(),
            response.getHits().getTotalHits().value
        );
    }

    // 创建视频索引
    public void createVideoIndex(Video video) {
        // 1. 构建索引文档
        IndexRequest request = buildIndexRequest(video);

        // 2. 创建索引
        elasticsearchService.index(request);
    }
}
```

图 5.7　搜索模块伪代码

(1) 索引文档构建：根据传入的 Video 对象，调用 buildIndexRequest 方法构建 Elasticsearch 的索引请求对象，该对象包含了视频的相关信息。

(2) 索引创建操作：将构建好的索引请求对象传递给 ElasticsearchService 的 index 方法，在 Elasticsearch 中为该视频创建索引，以便后续能够对该视频进行搜索。

通过上述功能的实现，该搜索服务模块为系统提供了高效的视频搜索和索引管理能力，确保用户能够快速准确地找到所需视频。

6 总结

参考文献

[1] CNNIC. 第五十五次中国互联网络发展报告[R]. 中国互联网络信息中心, 2025.

[2] 钟潇萌. 基于个性化推荐的视频网站的设计与实现[D]. 华中科技大学, 2022.

[3] 陈积银, 曾涛. 国外智能推荐型视频媒体的产业链研究[J/OL]. 西安交通大学学报 (社会科学版), 2019, 39(05): 121-131. DOI: 10.15896/j.xjtuskxb.201905013.

[4] 兰旭辉邓刚. 基于 MySQL 的应用程序设计[J/OL]. 计算机工程与设计, 2004(03): 442-443+468. DOI: 10.16208/j.issn1000-7024.2004.03.037.

[5] 哈喽沃德先生. FastDFS 分布式文件系统详解[Z]. https://www.cnblogs.com/mrhelloworld/p/fastdfs.html, 2020.

[6] 韩增曦. 分布式文件系统 FastDFS 的研究与应用[D]. 大连理工大学, 2014.

[7] PIMENTEL V, NICKERSON B G. Communicating and displaying real-time data with web-socket[J]. IEEE Internet Computing, 2012, 16(4): 45-53.

[8] 李代立, 陈榕. WebSocket 在 Web 实时通信领域的研究[J]. 电脑知识与技术, 2010, 6(28): 7923-7925+7935.

[9] ELASTICSEARCH B. Elasticsearch[J]. software], version, 2018, 6(1).

[10] 张仁良. 软件架构中的非功能需求[J]. 微型电脑应用, 2009, 25(01): 61-64+75.

[11] DRAGONI N, GIALLORENZO S, LAFUENTE A L, et al. Microservices: yesterday, today, and tomorrow[J]. Present and ulterior software engineering, 2017: 195-216.

[12] 贺适. 软件开发中数据库设计理论实践研究[J/OL]. 电子测试, 2020(08): 65-66. DOI: 10.16520/j.cnki.1000-8519.2020.08.025.

[13] 李艳杰. MySQL 数据库下存储过程的设计与应用[J]. 信息技术与信息化, 2021(01): 96-97.

致谢