# 诚 信 声 明

我声明，所呈交的毕业论文是本人在老师指导下进行的研究工作及取得的研究成果。据我查证，除了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，也不包含为获得其他教育机构的学位或证书而使用过的材料。我承诺，论文中的所有内容均真实、可信。

毕业论文作者签名：　　签名日期：　年　月　日

# 高校本科生论文选题系统设计

**[摘 要]** 本科毕业论文对学生而言至关重要。传统的表格论文选题存在着论文信息展示不全面、论文题目管理混乱的情况。为了让同学能够更方便地选择适合自己的论文、导师更有效率地选择合适的同学。设计了高校本科生论文选题系统。

本文首先分析当前的研究目标和意义，总结传统表格论文选题存在的问题，明确研究目的——致力于让学生更加方便选择适合自己的论文、导师更有效率地选择合适的同学。随后对系统开发相关技术进行介绍，对系统用户需求进行分析。最后根据需求对系统进行整体设计和详细设计，通过标签筛选法和协同过滤推荐算法进行论文推荐提高本系统的设计要求。系统可以正常运转，具有实用意义。

**[关键词]** 论文选题；标签筛选；协同过滤；论文推荐

# Design of the Thesis Topic Selection System for Undergraduate Students in Colleges and Universities

**Abstract：** The undergraduate graduation thesis is of great significance to students. In the traditional thesis topic selection method using forms, there are problems such as incomplete display of thesis information and chaotic management of thesis topics. In order to enable students to select the most suitable thesis more conveniently and enable tutors to select appropriate students more efficiently, a thesis topic selection system for undergraduate students in colleges and universities has been designed.

This paper first analyzes the current research objectives and significance, summarizes the problems existing in the traditional thesis topic selection using forms, and clarifies the research purpose, which is committed to enabling students to select suitable theses more conveniently and enabling tutors to select suitable students more efficiently. Subsequently, the relevant technologies for system development are introduced, and the user requirements of the system are analyzed. Finally, according to the requirements, the overall design and detailed design of the system are carried out. The design requirements of this system are improved by using the tag screening method and the collaborative filtering recommendation algorithm for thesis recommendation. The system can operate normally and has practical significance.

**Keywords： Thesis Topic Selection; Tag Screening; Collaborative Filtering; Thesis Recommendation**

# 目　　录

# 1 绪论

## 1.1 开发背景

随着高校教育信息化的推进，本科生毕业论文管理工作对于数字化平台的需求日益迫切。当前我校采用共享文档进行毕业论文选题管理。该模式存在显著的缺陷。

（1）信息透明度不足，共享表格提供的信息缺乏关键词标签、具体研究内容、选题详细要求等结构化数据，导致学生难以快速定位匹配论文题目。

（2）流程管理低效，依赖人工核对选题冲突，频繁出现多人重复选择同一论文题目的情况。且导师与学生需通过线下会议进行选题确认，效率低下、时间成本高昂导致学生难以选择到心仪的题目。

## 1.2 研究目标和意义

本文的研究目标是为毕业论文选择提供可靠的解决方案。为高校同学和高校导师在论文选题阶段提供便捷性，让同学能够快捷清晰地选择到心仪的论文，导师可以方便地选择到合适的人选。

本文介绍的解决方案是通过 BS 架构搭建一个线上的论文选择平台。在论文选择阶段，教师可以为论文提供自定义标签。学生可以通过自定义标签快速检索定位到自己感兴趣的论文。系统会根据学生行为通过协同过滤推荐算法为学生推荐其可能感兴趣的论文。大大提高了学生选择论文的效率和准确性。

学生选择阶段，论文信息的详细展示能够帮助同学对论文有快速全面的了解，当同学对论文感兴趣时，进入平台沟通界面进行实时通信，可以大大提高学生和教师之间的沟通效率。

通过严格权限管理规范论文和学生的选择关系，避免论文选择混乱的情况。

本文的研究意义主要有以下几点：

（1）提升毕业论文选择的管理效率，系统自动化处理选题匹配和冲突检测问题，可以大大减少人工干预成本。

（2）优化资源配置，基于标签筛选和算法推荐实现学生兴趣和导师研究方向的精准匹配，提高优质课题的利用率及同学和导师的满意度。

## 1.3 论文工作

本文以软件工程方法论为指导，完成以下研究工作。

通过问卷调查与访谈，提炼用户核心需求，明确核心业务与系统的功能边界。

技术选型采用 Springboot+Vue 构建前后端分离架构。

集成 webSocket 通信协议，支持师生在线实时讨论选题。

后续章节将按以下逻辑展开：

第二章阐述关键技术，对使用到的技术做出简要的介绍。

第三章进行系统需求分析，对整体的需求进行分析和设计，搭建开发框架，以及考虑不同用户的功能设计。

第四章说明数据库建模，对系统中涉及到的数据以及数据结构做详细

的介绍和分析。

第五章分析核心模块实现细节，对系统的各个模块给出详细的介绍和解释，并且详细解读运行时的业务逻辑流程。

# 2 相关技术

## 2.1 Springboot

### 2.1.1 Springboot 概述

Spring Boot 是由 Pivotal 团队于 2014 年推出的开源 Java 框架，隶属于 Spring 生态系统。它基于“约定优于配置”（Convention Over Configuration）的设计理念，旨在简化基于 Spring 框架的企业级应用开发流程。通过提供自动化配置和默认预设，Spring Boot 显著降低了传统 Spring 应用中繁琐的 XML 配置和依赖管理复杂度，使开发者能够快速构建独立运行、生产就绪的应用程序。

### 2.1.2 框架核心技术特性

（1）自动配置

根据项目依赖的 JAR 包自动配置 Spring 应用。例如，检测到 spring-boot-start-web 依赖时，自动配置 Tomcat 服务器和 SpringMVC 组件。通过条件化注解实现智能配置。开发者也可以通过配置文件覆盖默认配置。

（2）起步依赖

提供预定义依赖模块解决传统 Maven/Gradle 依赖冲突问题，每个 starter 包含功能相关的完整依赖链。

（3）嵌入式容器

内嵌 Tomcat、jetty、Undertow 服务器，无需部署至外部 Web 容器即可独立运行。通过命令直接启动应用，无需部署至外部 Web 容器。

### 2.1.3 技术选型依据

（1）敏捷开发需求

系统的预期开发时间为两个月，SpringBoot 的快速开发特性可以缩短 30%的编码周期

（2）社区支持

SpringBoot 拥有全球最大的 Java 开发者社区，GitHub Star 数超过 70k，确保技术的可维护性和可行性。

## 2.2 Vue

### 2.2.1 Vue 简介

Vue.js（简称 Vue）是由尤雨溪于 2014 年推出的开源渐进式 JavaScript 前端框架，专注于构建用户界面与单页面应用（SPA）。作为一款轻量级框架，Vue 以“渐进式”为核心理念，允许开发者根据项目需求逐步扩展功能，既可作为轻量视图层嵌入现有项目，也可通过工具链构建复杂企业级应用。截至 2023 年，Vue 在 GitHub 上累计获得超过 28 万星标，NPM 周下载量突破 400 万次，成为全球三大主流前端框架之一（与 React、Angular 并列）。

### 2.2.2 核心技术特性

（1）响应式数据绑定

通过数据劫持结合发布者订阅者实现响应式数据绑定，简化用户输入与状态同步逻辑。当数据更新后，用户不必考虑数据的绑定问题。

（2）组件化开发

以.vue 文件形式封装模板，实现高内聚的模块化开发，支持 socped CSS 避

免样式污染。本系统中的论文展示卡片，搜索框等通用组件均利用了这一特性。

（3）轻量级的框架生态

Vue 拥有一系列的官方工具链和生态系统，Vue CLI/Vite（标准化脚手架工具）、Vue Router（路由解决方案）、Pinia（集中式状态管理）等。

### 2.2.3 技术选型依据

（1）渐进性集成优势

支持从核心库逐步扩展功能模块的特性，与论文推荐系统的迭代开发路线高度契合。初期可快速搭建基础推荐界面，后期还可以逐步集成可视化分析模块（ECharts 整合）

（2）工程化支持

通过 VueCLI 脚手架实现快速开发以及开发/测试/生产环境的独立配置。通过 ESLint 和 Prettier 同意管理代码风格。

## 2.3 MySQL

MySQL 是一种广泛使用的开源关系型数据库管理系统（RDBMS），由瑞典 MySQL AB 公司开发，后被甲骨文（Oracle）公司收购。在当今的软件开发领域，数据的存储和管理至关重要，关系型数据库以其结构化的数据存储方式和强大的查询功能，成为了众多应用系统的首选。MySQL 以其高性能、可靠性和易用性，在 Web 应用开发、企业级应用开发等领域得到了广泛应用。

与其他数据库管理系统相比，MySQL 具有以下显著特点：首先，它是开源的，这意味着开发者可以免费使用和修改其源代码，降低了开发成本。其次，MySQL 支持多种操作系统，如 Windows、Linux、Mac OS 等，具有良好的跨平台性。此外，MySQL 提供了丰富的功能，包括事务处理、存储过程、触发器等，能够满足不同应用场景的需求。

在本次开发的线上论文选择系统中，MySQL 作为数据存储的核心，承担了存储系统中各种数据的重要任务。系统中的数据主要包括学生信息、导师信息、论文题目信息、选题记录等。

为了存储这些数据，设计了多个表，如 students 表存储学生的基本信息，teachers 表存储导师的基本信息，papers 表存储论文题目的详细信息，selections 表存储学生的选题记录。通过合理设计表结构和定义字段类型，确保了数据的完整性和一致性。

在数据查询方面，使用 SQL 语句实现了各种复杂的查询需求。例如，根据学生的兴趣标签查询符合条件的论文题目，根据导师的研究方向查询其指导的学生信息等。同时，利用 MySQL 的索引功能，对经常用于查询条件的字段创建索引，提高了查询效率。

在数据的增删改操作方面，通过编写相应的 SQL 语句，实现了学生信息的注册、导师信息的修改、论文题目的发布和删除等功能。为了保证数据的一致性和完整性，还使用了 MySQL 的事务处理功能，确保在多个操作同时进行时，要么全部成功，要么全部失败。

综上所述，MySQL 凭借其强大的功能和良好的性能，为线上论文选择

系统提供了稳定、高效的数据存储和管理解决方案，保证了系统的正常运行和数据的安全。

### 2.4 MyBatis

MyBatis 是一款轻量级的持久层框架，主要用于实现 Java 应用程序与关系型数据库的交互 [5]。其核心思想是通过 XML 或注解的方式，将 SQL 语句与 Java 对象进行映射，实现数据的持久化操作。与传统 JDBC 相比，MyBatis 大幅减少了样板代码的编写量，同时保留了对 SQL 语句的完全控制权，适用于复杂查询场景。

在本系统的开发中，MyBatis 主要负责处理学生信息、导师信息、论文数据等核心实体的数据库交互。通过定义 Mapper 接口和对应的 XML 映射文件，系统能够便捷地完成数据的增删改查操作。例如，当学生提交选题申请时，MyBatis 会将申请信息封装为实体对象，并通过预编译的 SQL 语句安全高效地写入数据库。

MyBatis 的动态 SQL 功能在论文推荐模块中尤为重要。系统需要根据学生的浏览行为、标签匹配度等多维度条件生成个性化推荐列表。通过 MyBatis 的<if>、<where>等标签，可以灵活组合查询条件，避免硬编码 SQL 语句带来的维护问题。此外，MyBatis 的二级缓存机制还能缓存高频访问的论文数据，有效减轻数据库压力。

考虑到系统采用 Spring Boot 架构，MyBatis 通过 mybatis-spring-boot-starter 实现与 Spring 生态的无缝集成。这种整合方式简化了数据源配置和事务管

理，开发人员只需关注业务逻辑的实现。同时，MyBatis 提供的逆向工程工具（如 MyBatis Generator）能够根据数据库表结构自动生成基础的 Mapper 接口和映射文件，进一步提升开发效率。

相较于 Hibernate 等全自动 ORM 框架，MyBatis 在本项目中更具优势。由于论文选题涉及复杂的关联查询（如导师 - 论文 - 学生的多对多关系），MyBatis 的手动 SQL 编写能力能够更精准地优化查询性能。同时，XML 映射文件的可读性和可维护性，也为团队协作开发提供了便利。

## 2.5 协同过滤推荐算法

### 2.5.1 协同过滤推荐算法概述

协同过滤推荐算法是一种在推荐系统领域广泛应用的技术，其核心思想是基于用户行为数据（如浏览记录、购买记录、评分等）来发现用户之间的相似性或者物品之间的相似性，进而为用户推荐可能感兴趣的物品。在当今信息爆炸的时代，用户面临着海量的信息和选择，如何从这些信息中筛选出符合用户个性化需求的内容成为了一个重要的问题。协同过滤推荐算法通过分析用户的历史行为，能够为用户提供精准的个性化推荐，提高用户获取信息的效率和满意度。

协同过滤推荐算法主要分为基于用户的协同过滤（User-based Collaborative Filtering）和基于物品的协同过滤（Item-based Collaborative Filtering）两种类型。基于用户的协同过滤算法通过计算用户之间的相似度，找到与目标用户兴趣相似的其他用户，然后将这些相似用户感兴趣的物品推荐给目标

用户。基于物品的协同过滤算法则是计算物品之间的相似度，根据目标用户过去喜欢的物品，推荐与之相似的其他物品。

本文主要应用的技术是基于用户的协同推荐算法。

### 2.5.2 基于用户的协同推荐算法原理

基于用户的协同过滤算法的基本步骤如下：。

（1）计算物品相似度

本系统中通过余弦相似度的方法计算物品之间的相似度。假设物品 i 和物品 j 被 m 个用户评分，分别用向量$\vec{r_i} = (r_{1i}, r_{2i}, \dots, r_{mi})$和$\vec{r_j} = (r_{1j}, r_{2j}, \dots, r_{mj})$表示，那么物品 i 和物品 j 的余弦相似度$sim(i,j)$可以通过以下公式计算

$$sim(i,j) = \frac{\vec{r_i} \cdot \vec{r_j}}{|\vec{r_i}||\vec{r_j}|} = \frac{\sum_{u=1}^{m} r_{ui} r_{uj}}{\sqrt{\sum_{u=1}^{m} r_{ui}^2} \sqrt{\sum_{u=1}^{m} r_{uj}^2}}$$

其中$r_{ui}$表示用户 u 对物品 i 的评分，$r_{uj}$表示用户 u 对物品 j 的评分。

（2）找到相似物品

对于目标用户喜欢的物品，选择与之相似度较高的 k 个物品作为相似物品。

（3）生成推荐列表

根据目标用户对喜欢物品的评分和这些物品与其他物品的相似度，预测目标用户对其他物品的评分。预测评分的计算公式如下：

$$P_{u,i} = \frac{\sum_{j \in N(i)} sim(i,j) r_{u,j}}{\sum_{j \in N(i)} |sim(i,j)|}$$

其中，$P_{u,i}$表示目标用户 u 对物品 i 的预测评分，N(i)表示物品 i 的相似物品集合，$r_{u,j}$表示目标用户 u 对物品 j 的评分，最后，根据预测评分对物品进行排序，选择评分较高的物品作为推荐列表。

## 2.6 WebScoket

WebSocket 是一种在单个 TCP 连接上实现客户端与服务器全双工通信的网络协议。它通过在 HTTP 协议基础上升级握手过程，突破了传统 HTTP 协议单向请求响应的限制，允许客户端和服务器在任意时刻主动发起数据传输，显著提升了实时交互场景的性能与效率。

其核心工作流程包括：客户端通过 HTTP 发起升级协议请求（Upgrade: websocket），服务器响应确认后建立持久化连接。通信过程中采用二进制或文本帧格式传输数据，协议头部开销仅为 2-10 字节，远低于 HTTP 请求的冗余数据量。该协议支持跨域通信，并通过 wss:// 实现 TLS/SSL 加密传输，保障数据安全。

WebSocket 的核心优势体现在：

（1）实时性：消除了轮询机制的延迟问题，适用于毫秒级响应场景

（2）双向通信：客户端与服务器可独立主动推送消息

（3）高效性：降低了网络带宽消耗和服务器资源占用

（4）标准化：浏览器原生支持，无需额外插件

在本系统中，WebSocket 技术主要用于构建用户实时交互模块。在学生和导师的交流模块中，学生与导师的聊天消息通过 WebSocket 通道即时推送。通过与 Spring Boot 的 WebSocket 模块集成，系统实现了消息的异步处理与集群化部署，确保高并发下的稳定通信。

## 2.7 JWT

JWT（JSON Web Token）是一种基于 JSON 格式的轻量级安全令牌标准，用于在网络应用中传递身份信息，它由头部、载荷和签名三部分组成。用户登录后，服务器会生成 JWT 并发送给客户端。客户端将令牌存储在本地中，后续携带 JWT 向服务器发送请求，服务器通过验证 JWT 确认用户身份。JWT 具有一定的有效期，过期后需要重新获取新的令牌，JWT 的跨域性、安全性、扩展性适合在分布式架构中实现身份验证。

JWT 的请求流程：

1. 用户使用账号和密码发出 post 请求

2. 服务器使用私钥创建 JWT

3. 服务器返回 JWT 给浏览器

4. 浏览器将 JWT 存储在本地缓存中，在请求头携带 Token 向服务器发送求。

5. 服务器验证 JWT

6. 返回响应的资源给浏览器。

JWT 是有三段信息构成的，将这三段信息文本用“.”连接就构成了 JWT 字符串。三段信息分别为

Header 头部（标题包含了令牌的元数据，并包含签名或者加密算法的类型）

Payload 负载

Signature 签名

浏览器和服务器通过 JWT 来进行身份合法性的验证。

# 3 系统分析

本系统的核心目标在于方便快捷地让同学和导师能够获得最合适的选择。要实现这一目标，需要深入的了解到各个用户端的核心需求。本系统共有三种用户身份——学生、导师和管理员。下文将需求分为通用需求、学生需求、导师需求和管理员需求四个方面展开。

## 3.1 通用需求

### 3.1.1 基础登录功能

系统需要保证用户身份的正确性。必须是本系统内的用户才可以使用本系统。没有身份的用户无法进入本系统的界面。

即使知道系统中页面的路由也无法通过直接访问访问，需要通过身份验证确认身份才可以访问系统。

## 3.2 学生需求

### 3.2.1 浏览需求

本系统的核心目标是让学生更方便的选择到更加适合自己的论文。

学生的核心需求在于最方便、最准确的选择最合适的论文。系统中应该能够提供非常方便快捷的浏览论文页面。

学生的心里常常有一些论文的关键词。比如自己更加擅长哪些技术，更适合哪种类型的论文等等。学生希望系统可以有方便快捷的论文筛选功能。

当学生自己的目标比较模糊时，学生希望系统可以根据自己的浏览行为自己进行论文推荐。

学生希望可以清晰明了、层次分明的了解论文的详细信息。

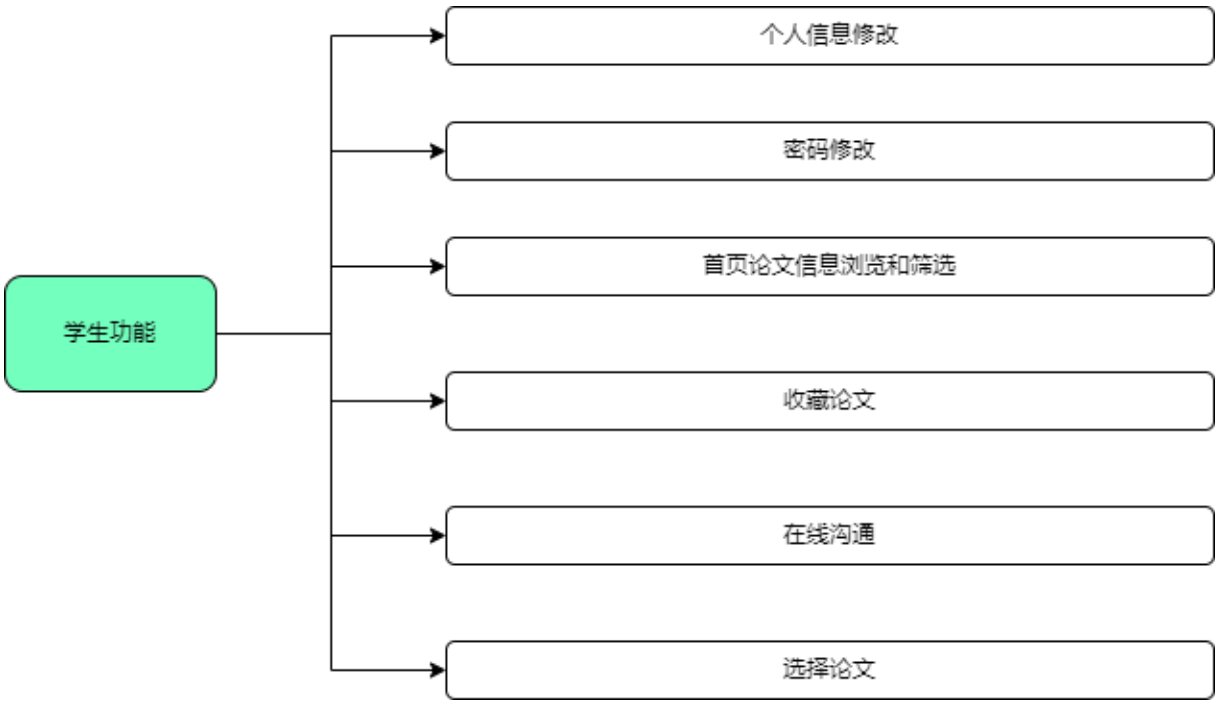
### 3.2.2 选择需求

相比于原来的选择方式、学生希望论文能够有良好的管理。被选择的论文可以直接提醒学生以免产生多个学生选择同一个论文的情况。

同学希望能够浏览多个论文，且对自己感兴趣的论文做标记。如常见的收藏功能。

### 3.2.3 沟通需求

学生希望与导师能够有更方便的沟通方式。而不是还需要通过添加联系方式才可以沟通。




图表 1 学生需求图

（1）个人信息修改，学生可以按照实际需求更改自己的个人信息，导师可以根据基本信息对学生有更直观的了解。

（2）密码修改，为保障账号安全，学生可以修改密码。

（3）首页论文信息浏览和筛选，学生可以根据自己的兴趣爱好，对论文题目进行筛选和选择。

（4）收藏论文，需要心仪的论文时，可以对论文进行收藏，可以在筛选中仅显示收藏。

（5）在线沟通，和论文的指导老师进行沟通，以实现论文的选择。

## 3.3 导师需求

### 3.2.1 论文管理需求

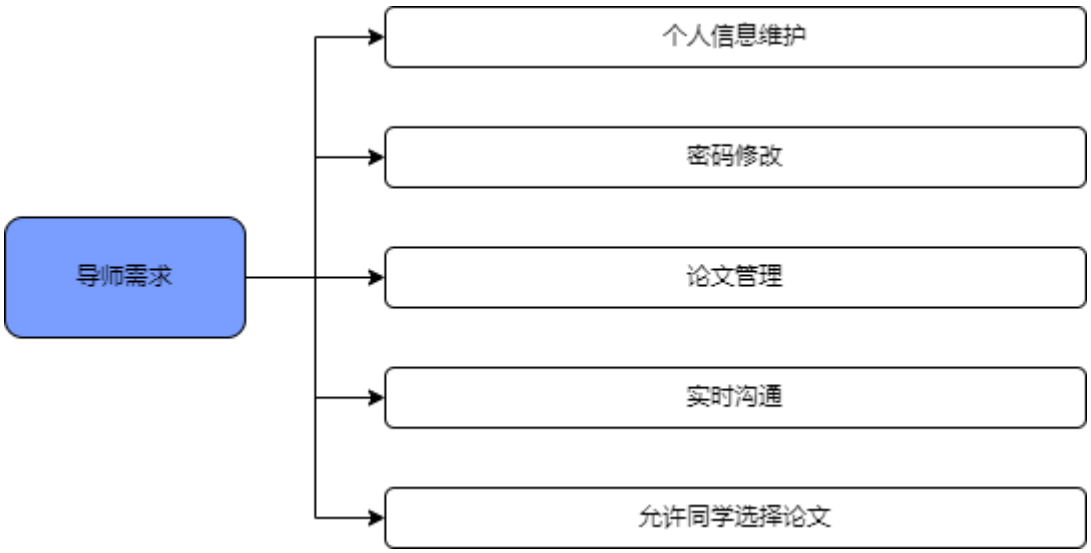
导师希望能够更加方便的管理论文，论文的创建要尽可能简单方便。对于论文的管理的操作要明确易懂，论文展示的信息要简单明了。

对于论文的管理操作应该自动化，如论文的选择唯一，论文被选择自动关闭其他选择通道。

### 3.2.2 沟通需求

导师希望能够和最合适的同学进行沟通，尽可能的提高沟通效率。

对自己



图表 2 导师需求图

（1）个人信息维护，导师按照实际情况设置个人信息，学生可以根据这些信息选择老师的论文。

（2）密码修改，为保障账号安全，导师可以修改自己的密码。

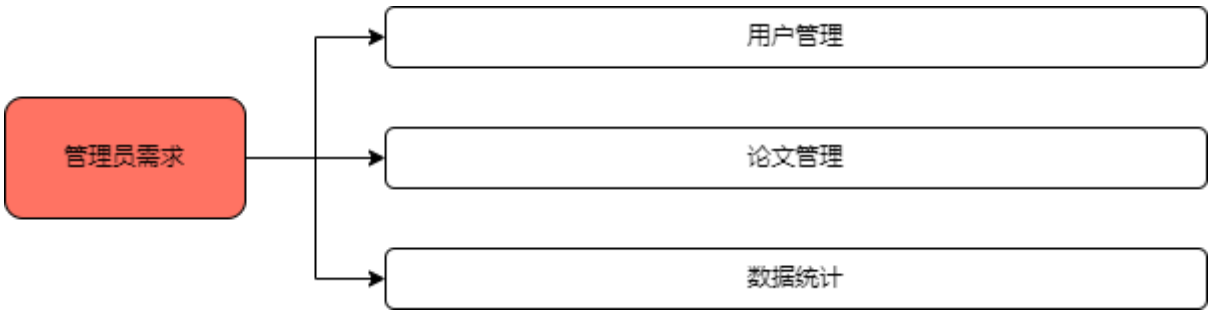
（3）论文管理，导师可以创建、修改、删除论文。创建的论文审核通过后可以被学生看到并选择。

（4）实时沟通，学生可以和导师沟通论文信息，导师也可以在沟通中选择心仪的学生。

（5）允许同学选择论文，学生无法直接选择论文，需要教师开发学生的选择权限学生才可以选择。

### 3.4 管理员需求

管理员希望对于信息的管理和展示要尽可能的清晰明了。操作要尽可能的简单易懂。




图表 3 管理员需求表

（1）用户管理，管理学生、导师等用户，保证用户正常使用系统。

（2）论文管理，导师创建论文后需要审核后才可以给学生选择。

（3）数据统计，后台管理论文和学生的选择情况，方便对学生的论文选题情况进行了解和管理

# 4 数据库设计

## 4.1 数据库简介

MySql 是一款开源的关系型数据库管理系统，由瑞典 MySQL AB 公司于 1995 年开发，现归属于 Oracle 旗下。它支持多种操作系统，采用客户端 / 服务器模式，能够高效、可靠、稳定地存储和管理数据。作为目前最流行的开源关系型数据库之一，MySQL 在 Web 应用程序、企业应用程序、移动应用程序等诸多领域广泛应用。其优势显著，开源免费降低了使用成本，高可靠性确保数据安全与完整，高性能可应对大量数据和高并发访问，简单易用的特性便于上手，可扩展性满足不同场景需求，同时还支持多种编程语言 ，如 Java、PHP、Python 等，具备跨平台能力 。

## 4.2 数据库概念模型设计

管理员、学生、导师三者是系统的用户，username 和 password 是系统的账号密码，用于实现对用户身份的校验。Role 角色是一个枚举类型，分为 STUDENT、TEACHER、ADMIN，用于区分用户身份，其他为基本字段。

表格 1 管理员表

| admin | |
|---|---|
| PK | id 主键id |
| | avatar 头像 |
| | name 用户名 |
| | role 角色 |
| | username 账号 |
| | password 密码 |

表格 2 学生表

| student | |
|---|---|
| PK | id 主键id |
| | username 用户名 |
| | password 密码 |
| | avatar 头像 |
| | name 昵称 |
| | email 邮箱 |
| | phone 电话号码 |
| | wechat 微信号码 |
| | QQ QQ号码 |
| | role 角色 |

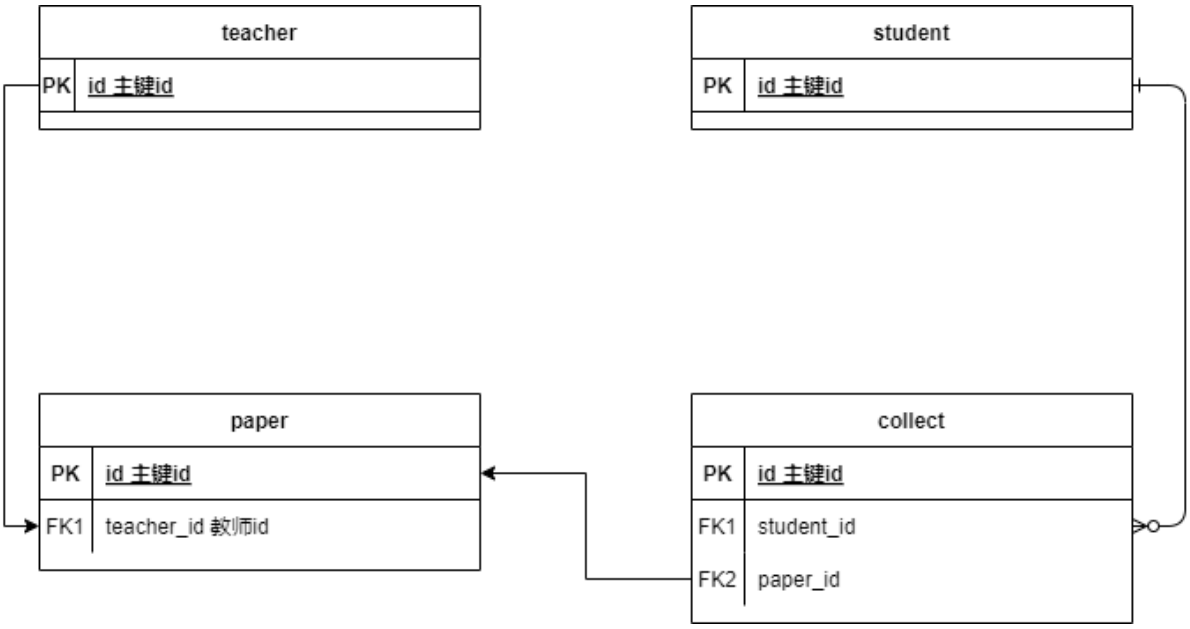
表格 3 导师表

| teacher | |
|---|---|
| PK | id 主键id |
| | username 教师账号 |
| | password 密码 |
| | address 办公地址 |
| | phone 电话号码 |
| | wechat 微信号码 |
| | QQ QQ号码 |
| | email 电子邮箱 |
| | name 教师姓名 |
| | role 角色 |
| | research_direction 研究方向 |
| | avatar 头像 |

收藏表用于实现学生对论文的收藏功能。

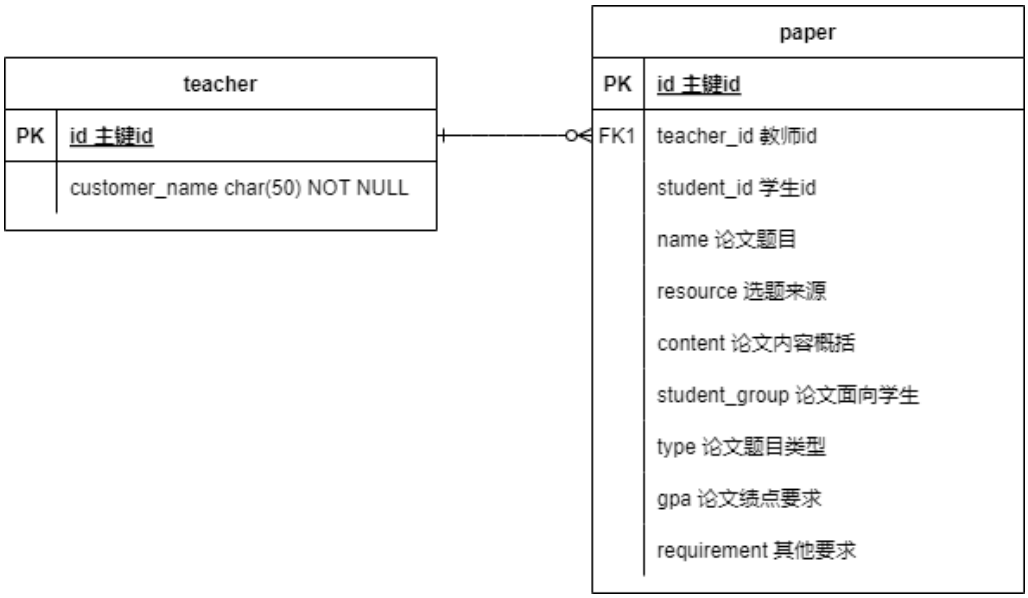
Student_id 用于关联学生、paper_id 用于关联论文。记录学生和论文的收藏关系。

表格 4 收藏表




论文表用于表示论文实体，teacher_id 外键对应着论文和导师的归属关系。Student_id 用于对应论文最终被哪个学生选择。其他为信息字段。

表格 5 论文表



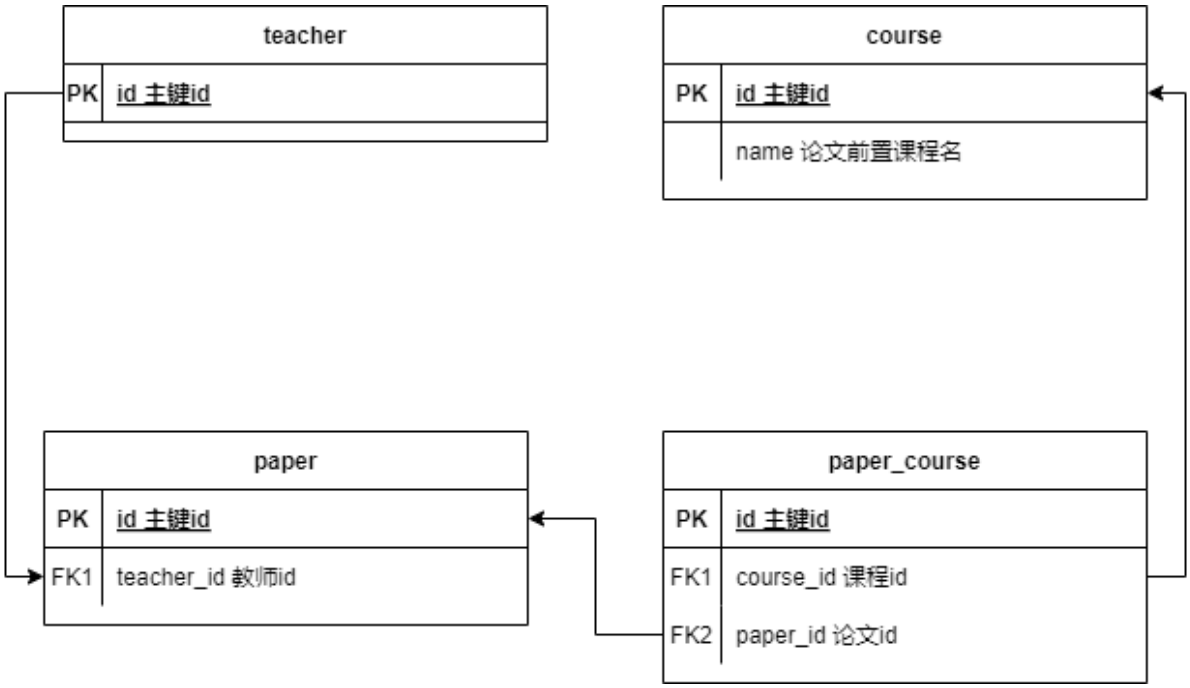
论文还有三个属性，course（选择论文所需前置课程）、language（论文所涉及的编程语言）、technology（论文所需技术）。

由于这三种的实现完全一致，下文仅介绍 course 的实现。Language 和 technology 不做重复赘述。

一个 paper 会对应多个 course，外键无法实现这种对应关系。

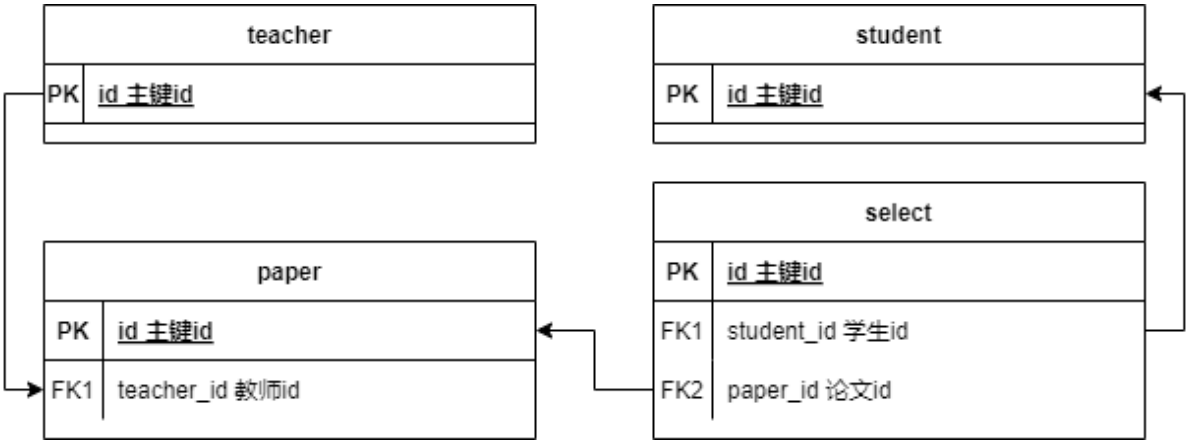
本文的实现通过中间表 paper_course 来记录所有 paper 和 course 的关系。Paper_course 关联着 paper 表和 course 表。Paper_language 和 paper_technology 表同理。关联着 paper 表和 language 表和 paper 表和 technology 表。

表格 6 论文课程中间表



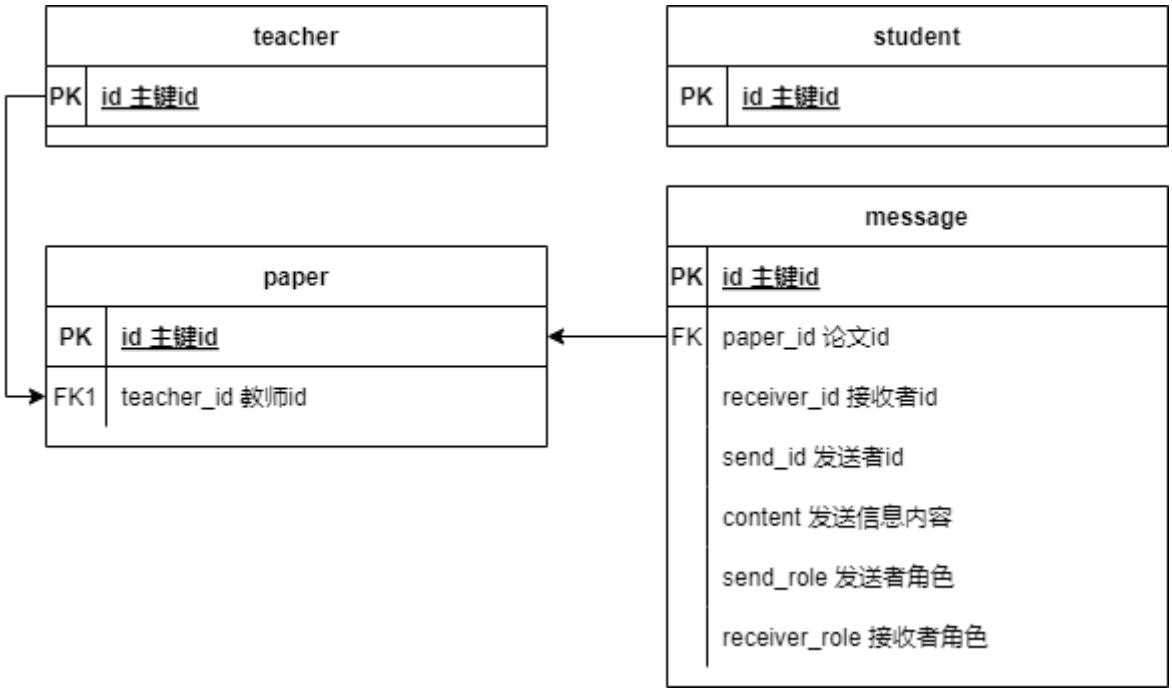
Select 表记录着学生和论文的选择关系。

表格 7 论文选择表



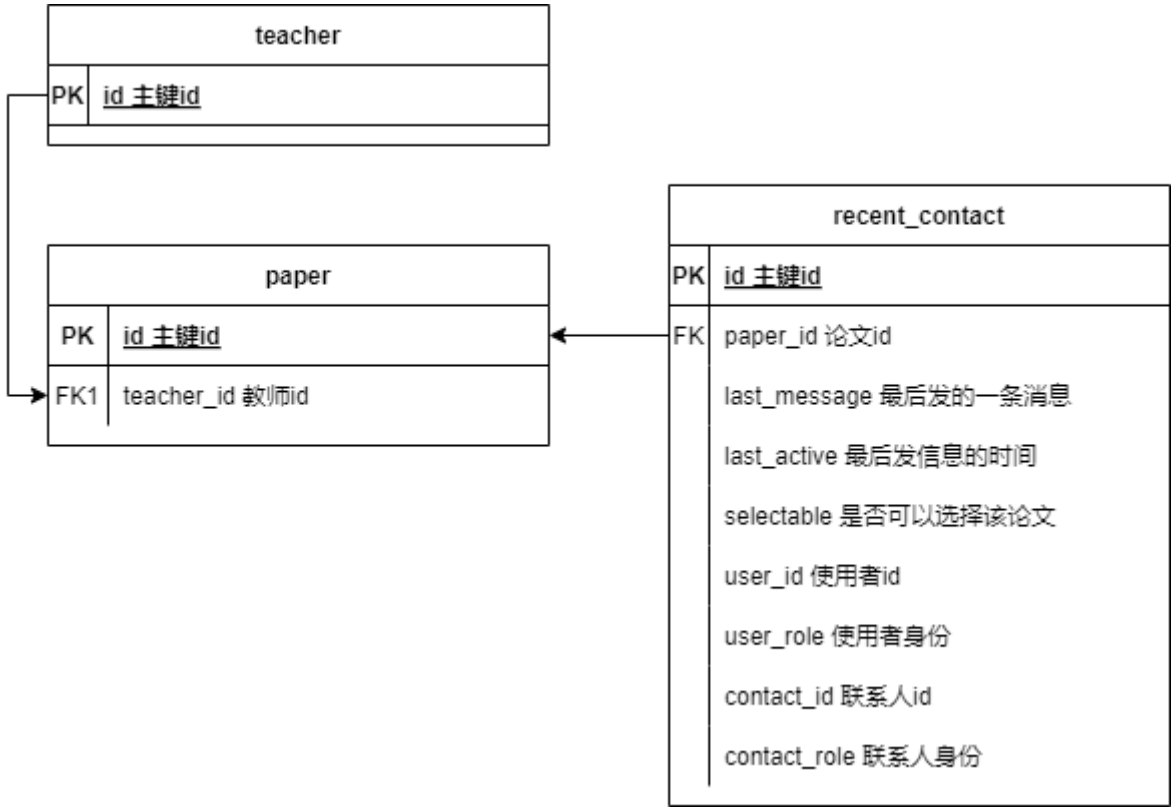

Message 表记录着学生和导师之间的聊天记录。send_id 为发送者 id，send_role 为发送者身份,receiver_id 为接收者 id，receiver_role 为接收者身份。Paper_id 为论文 id。这五个字段确定了一个 message。其余为信息字段。

表格 8 信息表



Recent_contact 表存储着最近会话信息。用于实现导师和学生的沟通功能。user_id、user_role、contact_id、contact_role、paper_id 五个字段确定了一个 recent_contact。其余为信息字段。

表格 9 最近会话表




## 4.3 数据库表结构设计

下面将详细介绍数据库表字段。

表格 10 管理员表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| id | int | not null | 主键 id |
| avatar | varchar(255) | | 头像 |
| name | varchar(255) | | 用户名 |
| role | varchar(256) | | 角色 |
| username | varchar(257) | | 账号 |
| password | varchar(258) | | 密码 |

表格 11 导师表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| id | int | not null | 主键 id |
| username | varchar(255) | | 教师账号 |
| password | varchar(255) | | 密码 |
| address | varchar(255) | | 办公地址 |
| phone | varchar(255) | | 电话号码 |
| wechat | varchar(255) | | 微信号 |
| QQ | varchar(255) | | QQ 号 |
| email | varchar(255) | | 电子邮箱 |
| name | varchar(255) | | 昵称 |
| role | varchar(255) | | 角色 |
| research_direction | varchar(255) | | 研究方向 |
| status | varchar(255) | | 审核状态 |
| avatar | varchar(255) | | 头像 |

表格 12 学生表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| username | varchar(255) | | 用户名 |
| password | varchar(255) | | 密码 |
| avatar | varchar(255) | | 头像 |
| name | varchar(255) | | 昵称 |
| email | varchar(255) | | 邮箱 |
| phone | varchar(255) | | 电话号码 |
| wechat | varchar(255) | | 微信号 |
| QQ | varchar(255) | | QQ |
| role | varchar(255) | | 角色 |
| id | int | not null | 主键 Id |

表格 13 收藏表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| student_id | int | | 学生 id |
| paper_id | int | | 论文 id |
| id | int | not null | 主键 id |

Course 表和 language、technology 表结构完全一致，只展示 course 表。

表格 14 前置课程表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| name | varchar(255) | | 课程名称 |
| id | int | not null | 主键 id |

表格 15 消息表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| send_id | int | | 发送者 id |
| receiver_id | int | | 接收者 id |
| paper_id | int | | 关联论文 id |
| content | longtext | | 消息内容 |
| create_time | datetime | | 创建时间 |
| send_role | enum('student', 'teacher') | | 发送者身份 |
| receiver_role | enum('student', 'teacher') | | 接受者身份 |
| id | int | not null | 主键 id |

表格 16 论文表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| teacher_id | int | | 老师 id |
| student_id | int | | 学生 id |
| name | varchar(255) | | 论文题目 |
| resource | varchar(255) | | 选题来源 |
| content | longtext | | 论文内容简要概括 |
| student_group | varchar(255) | | 面向学生 |
| type | varchar(255) | | 论文题目类型 |
| gpa | float | | 学生绩点要求 |
| requirement | longtext | | 其他要求 |
| id | int | not null | 主键 id |

Paper_course 和 paper_language、paper_technology 的表结构也完全一致，这里只展示 paper_course。

表格 17 论文和课程中间表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| paper_id | int | | paper 主键 id |
| course_id | int | | course 主键 id |
| id | int | not null | 主键 id |

表格 18 论文选择表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|
| paper_id | int | | 论文 id |
| student_id | int | | 学生 id |
| id | int | not null | 主键 id |

表格 19 最近联系人表

| 字段名称 | 字段类型 | 是否为空 | 注释 |
|---|---|---|---|

| | | |
|---|---|---|
| last_message | text | 最后发的一条信息 |
| last_active | datetime | 最后发信息时间 |
| paper_id | int | 关联论文 id |
| selectable | tinyint(1) | 是否可以选择该论文 |
| user_id | int | 使用者 id |
| user_role | enum('teacher','student') | 使用者身份 |
| contact_id | int | 最近联系人 id |
| contact_role | enum('teacher','student') | 最近联系人身份 |

# 5 系统设计

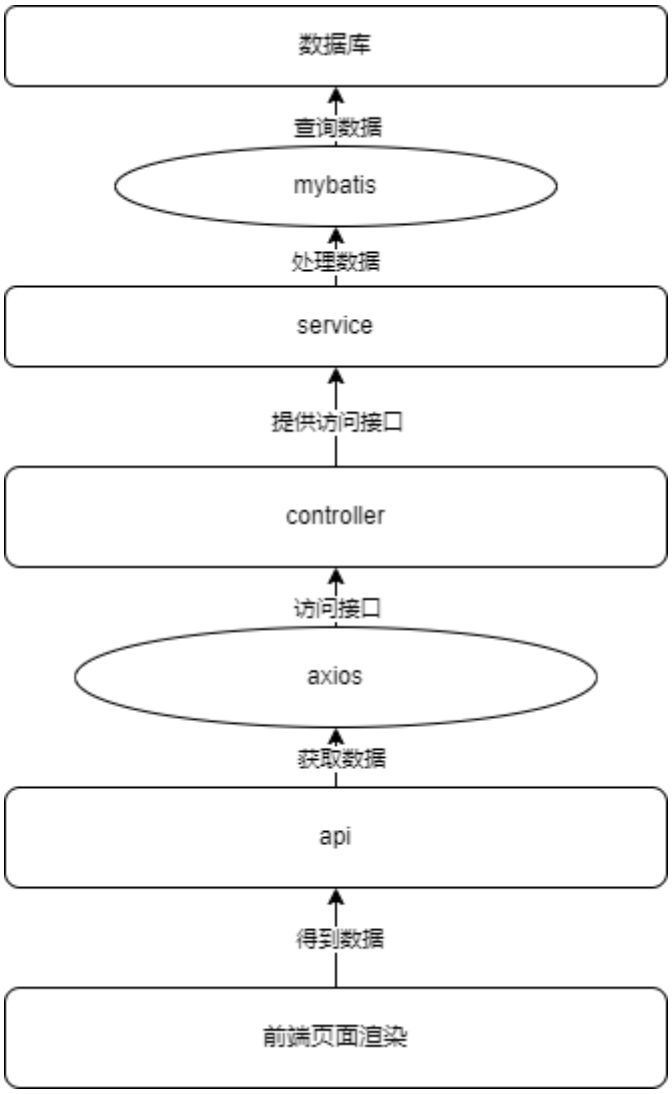
## 5.1 整体设计

### 5.1.1 数据流转与交互设计

该部分主要介绍本项目中信息是如何流动和交互的。如何实现信息从数据库到前端页面的处理。

#### 5.1.1.1 信息传递路径

图表 4 信息传递路径图




（1）数据库管理

本系统中的数据源来自 MySQL 数据库。用户操作生成的数据最终都将存入数据库中。前端的显示的信息也都是从数据库中查询得到。

（2）Mybatis 持久层框架访问数据库

本项目通过 mybatis 持久层框架来访问数据库。

通过在 pom.xml 文件中引入坐标来实现 mybatis 的配置。

其次需要再.yml 文件中对数据库访问接口、用户名、密码等信息进行配置。

创建 mapper 的 java 接口。在 xml 文件中写 sql 语句，与接口中的方法对应。则实现了对数据库的查询。

常见的 mapper 中包含 add（插入数据）、delete（删除数据）、update（更新）、selectByPage（分页查询）、selectById（根据 id 单个查询）函数。

图表 5 mapper 模板

```
@Mapper
public interface AdminMapper {

    void add(Admin admin);

    void delete(Integer id);

    void update(Admin admin);

    List<Admin> selectByPage(Admin admin);

    Admin selectById(Integer id);

    Admin selectByUsername(String username);
}
```

以 adminMapper 为例，每个 mapper 中都有增删改查的方法。对于特殊的查询需要特殊的设计。如 adminMapper 需要通过 username 查询 Admin，则多添加 selectByUsername 在.xml 文件中实现具体的查询语句。

（3）service 层数据处理

在 mapper 层已经实现了对数据的查询。在 service 层将实现数据的处理。

通过注解依赖注入的方式在 service 中使用 mapper 层查询到的数据。

在 service 层进行数据的处理。

（4）controller 层提供接口

通过依赖注入 service 层处理好的信息，在 controller 层已经拿到了前端所需要的数据。Controller 层将提供接口给前端访问。将目标数据封装为 Result。返回给前端。

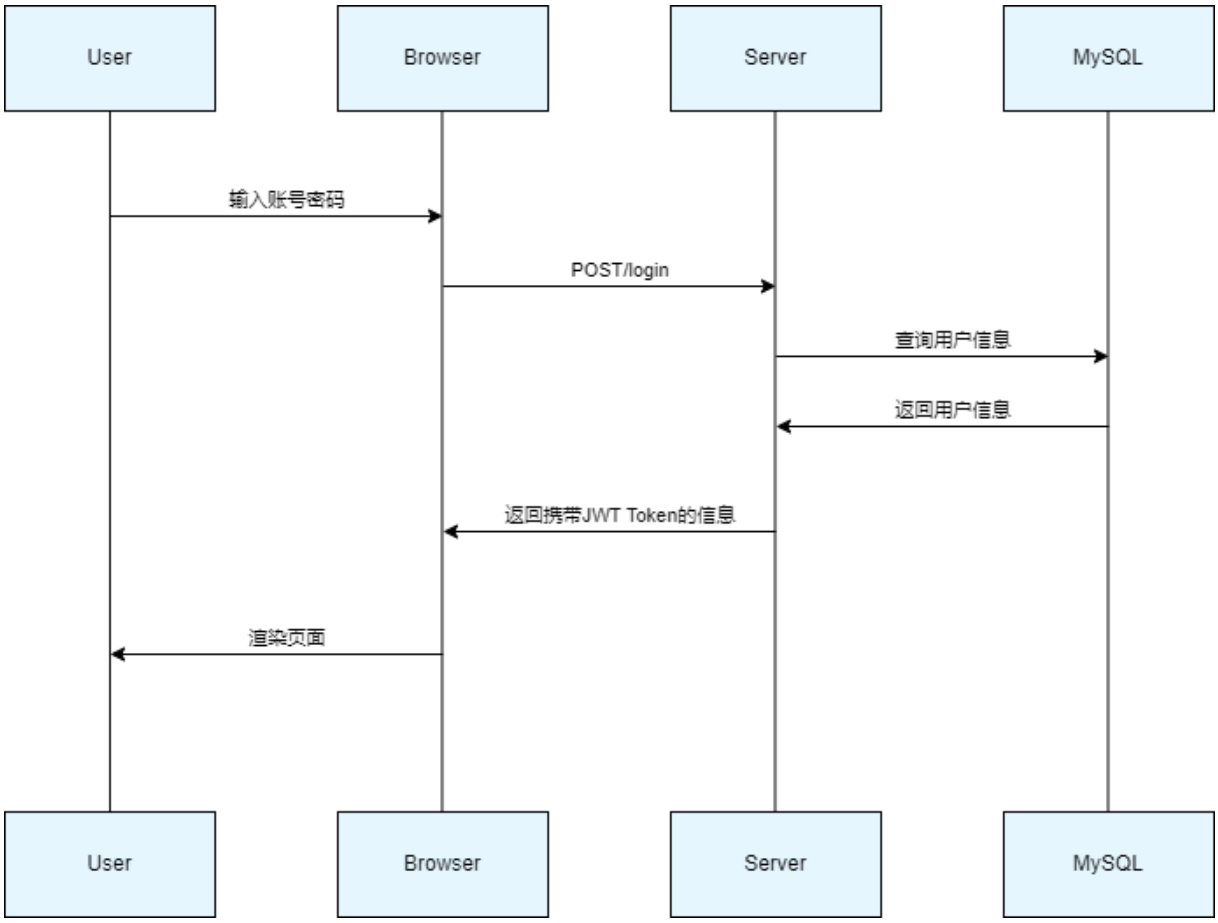
（5）前端数据访问

前端通过访问后端接口请求数据，请求数据后在页面渲染。

### 5.1.2 登录功能整体设计

用户的数据存储在数据库之中。

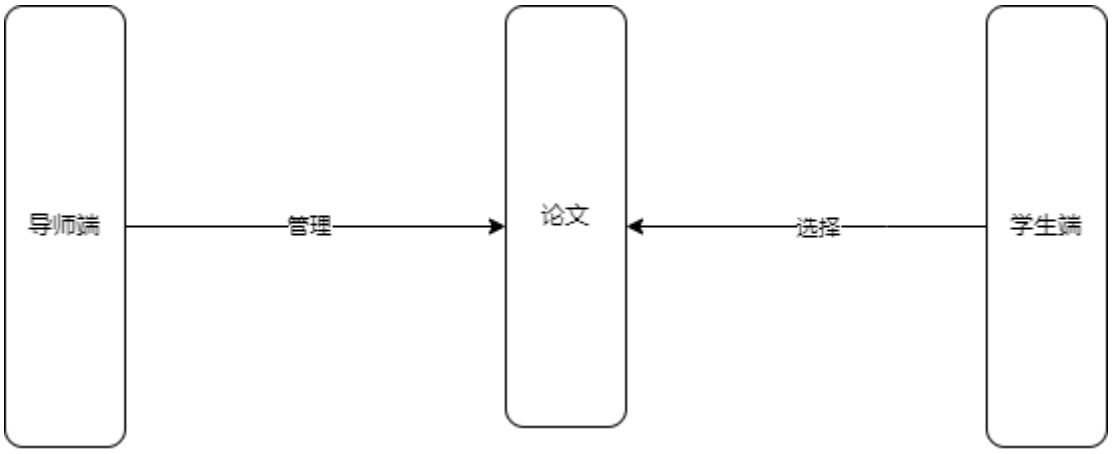
前端通过数据账号密码向后端发送请求。后端根据前端提供的信息进行账号密码的校验。通过校验结果决定是否对用户放行。




图表 6 登录时序图

### 5.1.3 论文管理和选择整体设计

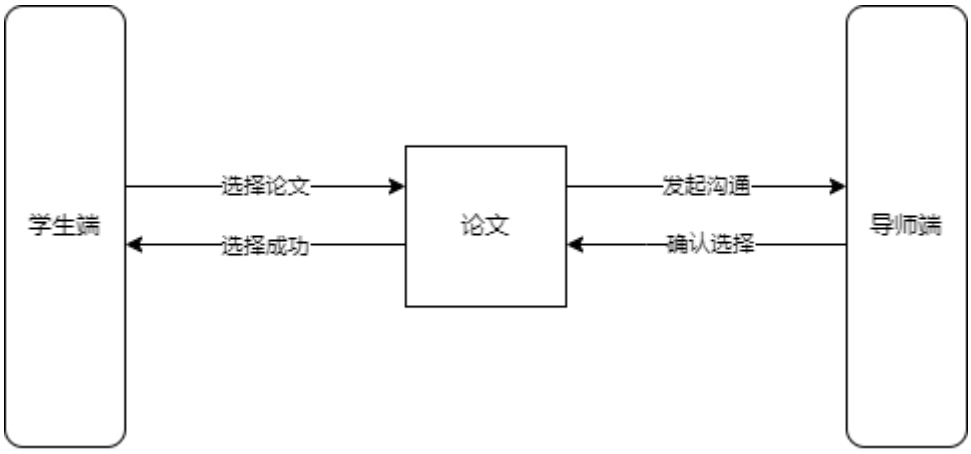
#### 5.1.3.1 论文管理整体设计




导师端对论文进行创建、修改、编辑、删除等管理操作。数据进入数据库后在学生端进行展示。学生端对论文进行筛选、查看、沟通、收藏、选择

操作。

#### 5.1.3.2 论文选择整体设计




学生端进行论文选择，选择论文后才拥有权限和老师沟通。这样设计可以避免大量同学低成本和导师进行沟通造成导师的选择效率底下问题。对学生的选择进行限制，最多只能同时选择和发起沟通五个论文。

#### 5.1.3.3 论文选择权限同步

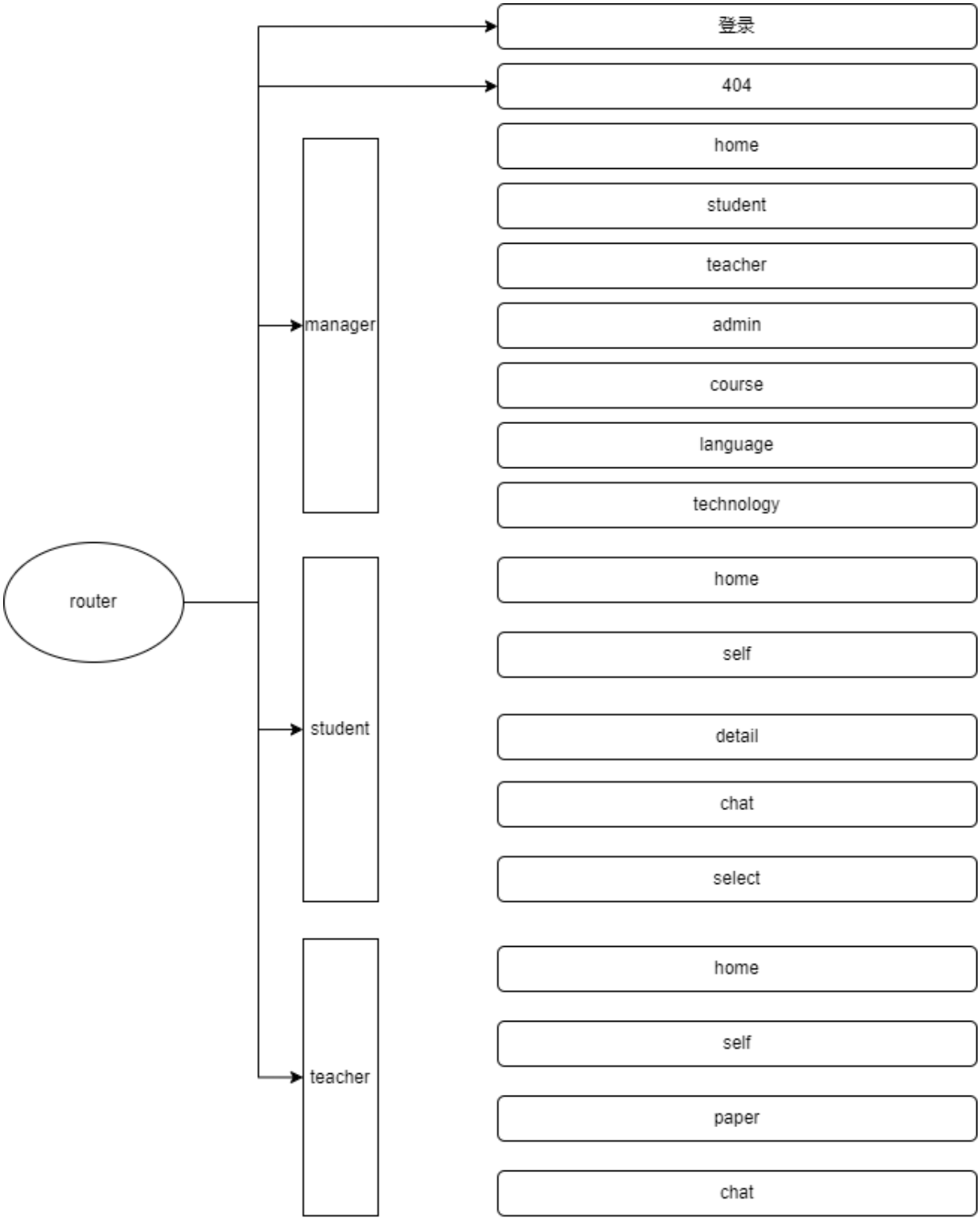
当导师选择论文人选后，其他同学则不能再选择改论文。所以需要在选择论文时进行权限同步控制。

通过为 paper 实体类设置 studentId 字段。当论文选择后，studentId 字段则被赋值。在前端展示和后端查询时，通过对 studentId 字段的校验进行权限控制。

前端判断，当 studentId 不为 null 时，对操作界面进行禁用控制。

后端判断，当 studentId 不为 null 是，通过 studentId 进行字段筛选。保证得到的 paper 是未被选择的。

### 5.1.4 路由整体设计




由于所有的角色使用系统都需要登录功能，因此将登录板块设计为通用模块。

考虑到用户输入未配置的路由，设置 404 路由拦截不法访问，在 404 页面

中点击跳转跳转到首页。

管理员路由模块按功能分为home、student、teacher、admin、course、language、technology，下面将一一介绍每一个路由的作用。

home 管理员路由首页展示系统的总览信息，如发布论文数、论文已选择数量、学生数量、未选择的学生数量。

student 学生路由用于展示和管理学生信息。

teacher 导师路由用于展示和管理导师信息。

admin 管理员路由用户管理管理员信息。

course 课程路由用于管理论文所需要完成的前置课程。

language 编程语言路由用于管理论文涉及到的编程语言。

technology 技术路由用于管理论文涉及到的技术选项。

导师路由模块按功能分为 home、self、paper、chat，下面将一一介绍每一个路由的作用。

Home 导师路由首页展示自己创建了多少论文。

Self 个人信息路由用于导师修改个人展示信息。

Paper 论文管理路由用于导师管理自己的论文。

Chat 聊天界面用于导师与学生进行沟通和商榷。

学生路由模块按功能分为 home、self、detail、chat、select，下面将一一介绍每一个路由的作用。

Home 学生路由首页展示着导师创建且审核通过的论文，学生可以在首页筛选进行论文选择。

Self 个人信息路由用于学生修改个人展示信息

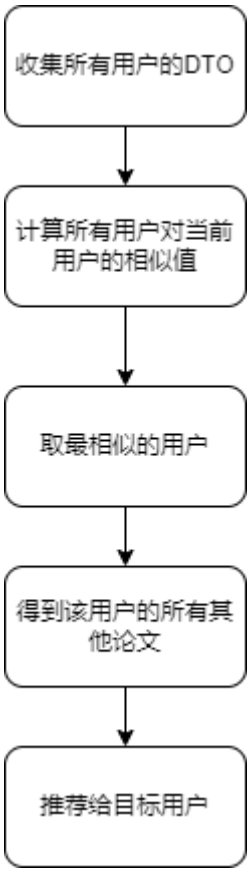
Detail 论文详细信息路由用于展示路由的详细信息。

Chat 聊天界面用于学生与导师进行沟通和交流。

Select 路由用于学生管理和查看自己已经选择的论文。

### 5.1.5 协同过滤推荐算法整体设计

协同过滤算法需要得到用于对论文的喜爱程度。所以我们需要设计一个实体类 relateDTO 表示用户对论文的喜爱程度。




我们需要的到所有用户的 relateDTO，通过 DTO 计算每个用户和目标推荐用户的相似值。

计算出所有相似值后，取最相似的用户。拿到该用户的所有论文。

对论文进行筛选。得到推荐论文列表。

对列表进行去重处理，推荐给目标用户。

## 5.2 详细设计

### 5.2.1 环境搭建

本项目前端使用 vue.js 框架，后端使用 springboot 框架。前端编辑器为 vscode，后端为 idea。

后端环境搭建：

使用 idea 创建 springboot 项目，设置项目名等基本信息，编程语言选择 java，通过 maven 进行项目管理且配置好 maven 的路径。Jdk 选择版本 21。在.yml 文件中配置端口号为 9090。搭建 MVC 三层模型。创建 controller、service、mapper 包。

定义前后端的信息传播模板，定义统一的返回类型。创建 result 类。

图表 7 返回类型字段

```
// 状态码
private String code;
// 提示信息
private String msg;
// 返回数据
private Object data;
```

后端返回给前端的数据都将使用 Result 包装，这样就统一了前后端的格式问题。

定义全局的异常处理，通过继承 RuntimeException 类实现自定义异常来向前端抛出错误，通过@ ExceptionHandler 注解实现异常处理器拦截异常，对异常进行打印输出以便在调试过程中更快的定位问题，维护一个统一的异常枚举类可以在编写代码时候很方便的抛出异常。

前端环境搭建：

下载 vue 后，通过 npm create vue@latest 指令创建 vue 项目。

项目中选择 vue-router 管理路由、pinia 状态管理以及 ESlint 管理代码格式规范。

运行 pnpm install 安装依赖后，pnpm dev 即可运行前端项目。

### 5.2.2 登录和注册

登录和注册功能是学生、导师和管理员三个角色都需要的功能。在实现登录注册功能时，最先考虑的是如何实现不同身份进行不同登录逻辑的考虑。

（1）前端设计

前端整体页面设计，相比于一个为每一个用户提供一个单独的用户登录页面，本系统选择将学生登录页面和导师登录页面合并，管理员页面进行页面复用。原因是管理员所需功能和教师、学生的功能有一些差异，一个页面进行三种逻辑的判断反而增大了代码的复杂度。而学生和导师的登录逻辑相似，相比于两个不同界面可以减少代码的复用率和项目的冗余度。因此本页面选择通过两个前端页面实现前端登录设计。

前端登录校验设计，本系统的前端登录表单通过 elementUI 中的 el-form 实现，使用 validate 进行表单内容的校验，使得一些不合法的内容根本无法提交给后端，校验规则为

- 用户名、密码不得为空
- 用户名不得出现特殊字符

- 用户需要选择自己的登录身份（学生、导师）
- 密码不得过短或者过长
- 用户需要勾选协议

通过了校验，用户才能向后端发送请求。

后端验证账号密码正确后，后端会返回用户的数据以及 JWT。前端通过 pinia 的 persist 持久化将信息存储在本地，登录后的请求中必须加入 token 到请求头，否则无法通过后端的请求。

（2）后端设计

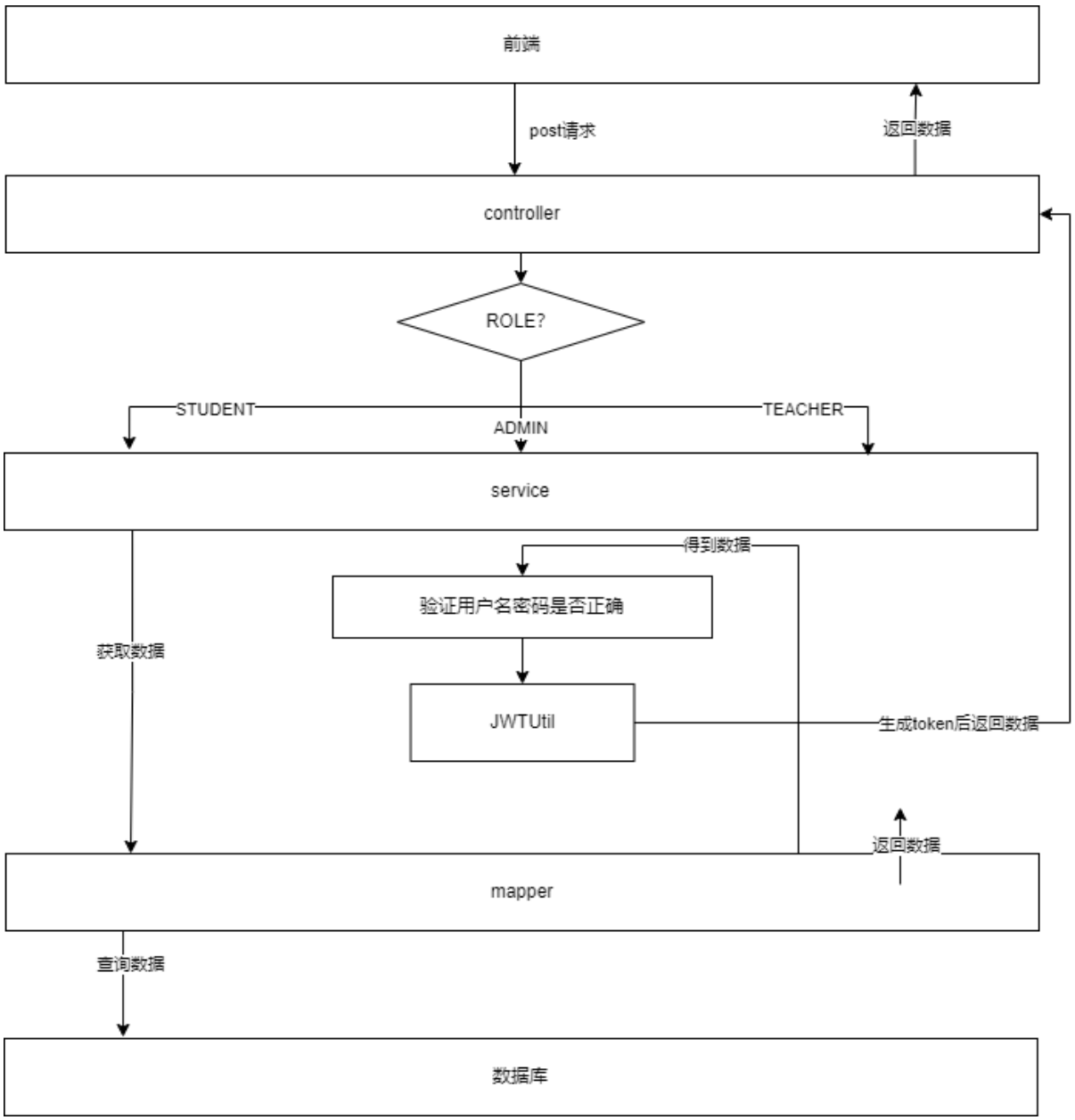
整体的后端设计采用 MVC 模式，通过 controller、service、mapper 三层框架实现后端代码的管理。

代码在后端需要验证用户的账号和密码是否正确，逻辑处理在 service 层，因此后端设计在 controller 层通过依赖注入注入 studentService、teacherService、adminService。然后在各自的 service 层实现登录的逻辑。

以 studentService 为例，首先通过 username 字段查询该用户信息，如果未查到信息则抛出异常——用户不存在。若是存在用户信息，则核对密码是否一致，不一致则抛出异常——密码错误。

若账号密码正确，则通过 JWTUtil 生成 token，连同用户信息一同传递给前端。

图表 8 登录逻辑图




JWTUtil 的实现：

JWT 的组成分为三个部分：头部、负载和签名。

头部使用用户 id（唯一标识符）+用户身份作为头部如（1-ADMIN），代表管理员 1。定义 token 的有效期为 1 天。使用 HMAC256 作为加密算法，使用用户的密码作为加密密钥生成 JWT。

验证 JWT 同理也是对三个部分进行解密，验证用户 id、用户身份、密码以及时间的正确性。

（3）多状态管理和持久化设计

用户信息中含有 token，在登录后的所有请求中都应该携带 token 进行身份验证，且用户信息在其他页面中也需要频繁用到，所以考虑将用户信息实现本地持久化。这样既保证了开发效率，也提升了用户体验。

pinia 多状态管理，pinia 是 vue 最新的状态管理工具。可以帮我们管理通用数据用户信息。

在 store 中创建 accountInfo，定义用户信息 accountInfo 以及对 accountInfo 的操作方法。并配置 persist 持久化。保存在浏览器的 localStorage 中。

### 5.2.3 路由控制详细设计

（1）概念和架构设计

在开发单页面系统时，vue-router 可以在不重新请求页面的情况下，更新页面视图。

路由的设计主要包括四个部分——学生板块、导师板块、管理员板块、通用板块（登录、404）具体的模块设计如下图。

（2）具体配置过程

通过 pnpm 依赖管理工具配置 router 依赖。

在 router 目录下创建 index.js 按照 vue-router 官网的规定函数配置 router。

在 main.js 中引入 index 文件使用 app.use()完成对 router 的配置。

（3）配置路由守卫

路由守卫室 Vue-router 提供的一种机制，用于在导航过程中对路由进行控制和管理，通过使用路由可以在路由切换前、切换中以及错误处理时处理相

应的逻辑。

Vue Router 提供三种类型的路由守卫。

1. 全局前置守卫

2. 路由独享守卫

3. 组件内的守卫

本文中主要运用全局前置守卫。已实现对用于路由访问的控制。

本文中的路由守卫主要负责防止没有 token 的用户访问用户信息。限制学生、导师、管理员三种身份的访问权限，控制三者只能访问该身份的页面。

路由守卫的第一个作用是确认浏览器是否存有 token，没有 token 说明用户没有通过身份认证，则直接拒绝访问，并重定向到登录界面。

### 5.2.4 论文管理和选择详细设计

#### 5.2.4.1 论文管理功能

本文的研究目标是为毕业论文选择提供可靠的解决方案。为高校同学和高校导师在论文选题阶段提供便捷性，让同学能够快捷清晰地选择到心仪的论文，导师可以方便地选择到合适的人选。

学生选择论文时，最为在意的是论文的属性。为此，在设计表示论文的数据结构时，设置了论文所需前置课程 course 字段、论文所涉及的相关编程语言 Language 字段、论文所涉及的先关技术 technology 字段。

学生可以通过筛选这些字段来在众多论文中快速预览到自己心仪的论文。以此更加方便快捷准确的选择论文。

需要的考虑的是，一个论文往往对应着多个 course、language、technology。

由于 course、language、technology 的实现逻辑是一致的。下面将说明 course 的实现过程，language 和 technology 的实现方法与 course 完全相同 。

在数据库表存储时，属性无法设置为数组形式。所以无法使用单个属性存储多个 courseId，除非以字符串的形式存储，以指定的分隔符分割，取出数据后做 split 操作。但是这种解决方案的复杂度很高，且会导致代码和项目的可读性降低，提高理解成本。

使用外键只能实现单对单的表结构，无法实现一个表的字段对应另一个表多个字段的需求。

本文采用中间表存储实现一个 paper 与多个 course 的对应。创建数据表 paper_course，记录 paperId 和 courseId 的对应关系，查询时，通过关联 paperId 字段查询到相应的 courseId 字。

论文实体类中通过 courseIds 以存储 paper 关联的 course，查询数据时，将查询到的 course 以 course 实体类的数据结构存储到 courses 中，则拿到了所有的 course 信息。

这种做法伴随着一个问题——使用 mybatis 进行查询时，需要先查询中间表，通过中间表的 paperId 对应的 courseId 查询 course，同理 language 和 technology 也是如此，这已经关联了三个表结构，涉及到导师和学生信息时甚至要关联五个表结构。这使得 SQL 查询语句复杂度极高。大大提高代码的维护难度。而且这种查询方式无法实现将整个 course 添加到 paper 类中的 List<course>中。

本文通过定义 resultMap 解决这个问题。

首先在 courseMapper 中定义查询函数 selectCourseByPaperId。通过关联 paper_course 中间表查询到该 courseId 对应的 paperId。

图表 9 resultMap 数据结构

```
<resultMap id="PaperResultMap" type="com.paper.entity.Paper">
    <id property="id" column="id" />
    <result property="teacherId" column="teacher_id" />
    <result property="studentId" column="student_id" />
    <result property="name" column="name" />
    <result property="resource" column="resource" />
    <result property="content" column="content" />
    <result property="studentGroup" column="student_group" />
    <result property="type" column="type" />
    <result property="gpa" column="gpa" />
    <result property="requirement" column="requirement" />
    <collection property="courses" ofType="com.paper.entity.Course"
                select="com.paper.mapper.CourseMapper.selectCourseByPaperId"
                column="id">
    </collection>
    <collection property="languages" ofType="com.paper.entity.Language"
                select="com.paper.mapper.LanguageMapper.selectLanguageByPaperId"
                column="id">
    </collection>
    <collection property="technologies" ofType="com.paper.entity.Technology"
                select="com.paper.mapper.TechnologyMapper.selectTechnologyByPaperId"
                column="id">
    </collection>
</resultMap>
```

在 paperMapper 中定义一个 ResultMap，相当于自己定义一个数据类型。我们可以自定义这个数据类型的属性。

我们使用 collection 标签，通过 select 属性关联到 courseMapper 中的 selectCourseByPaperId 方法。这样我们就实现了将查询到的 course 以 course 的形式插入到 paper 中的 List 泛型中了。

我们只需要在查询 paper 的返回类型中定义 resultMap 为 PaperResultMap 即可。

Paper 定义好后，即可以实现导师对论文的管理。

（1）添加论文

前端采用 form 表单的形式收集导师输入的论文信息。

图表 10 前端表单字段


ElementUI 提供了表单组件以及输入框、选择框、计数框。

对于前置课程 course 的选择。教师可以自定义添加。此功能只需要调用 course 的增加接口即可。

前端收集到对应 paper 数据结构的数据。

图表 11 前端 paper 数据结构


通过点击按钮发送事件，传递到后端。

后端接受数据后再 service 层处理。需要遍历 List<course>，以此调用中间表的插入接口插入每一个 paperId 和 courseId。将数据写入数据库则添加论文功能实现完毕。

修改论文同样，前端依靠 paperData 双向绑定表单数据。无论是增加还是修

改，修改的都是 paperData 数据。在前端收集到 paperData 数据后提交到后端。

对于修改 paperId 和 courseId 的功能，需要修改 paper_course 中间表的数据。相比于修改，直接删除之前的所有数据再重新插入效率更高。

查询逻辑在构建 paper 数据结构时已经说明。后端查询数据后返回前端渲染即可。

图表 12 前端组件化开发


前端的论文渲染中，所有的论文展示模块都是相同的。

所以可以将论文渲染部分设计为一个组件。提高组件的隔离性以及代码的复用性，组件中需要通过 defineprop 传入 paper 数据进行渲染。

删除功能依赖着 defineEmits 来向父组件传递事件。子组件会向父组件传递 onDel 事件，并且传入参数 paperId。

在父组件中调用删除论文的 api，传递到后端删除即可。

这里没有在子组件调用 api，遵循接口隔离原则。删除论文的功能应该在父组件集成实现。

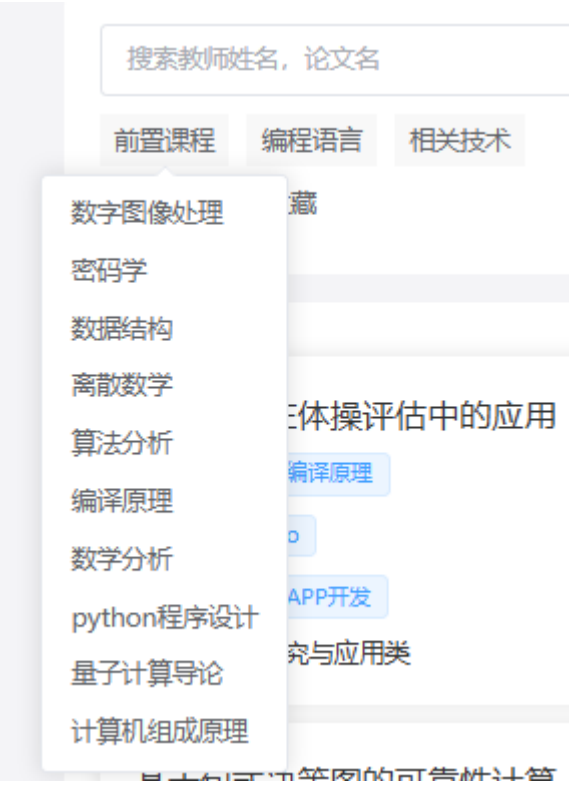
#### 5.2.4.2 学生选择和筛选论文

实现了导师对论文的管理功能后。后端已经拥有正确的数据，学生端此时应该可以正确选择导师的论文。

学生端通过后端查询接口来获取数据进行渲染。学生端论文渲染和导师不同，需要设计新的组件进行渲染。

学生应该可以快捷地检索到自己想选择的论文。所以系统中提供了根据字段进行筛选和根据字段搜索的功能。

图表 13 学生端字段检索功能图




这部分论文需要渲染导师的信息，所以 paper 需要通过外键 teacherId 关联 teacher 表。以及通过中间表检索到对应的 course。

前面已经设计好了 resultMap，在 mapper 中可以直接复用 resultMap,只需要填充好查询条件。完整的查询条件如下。

通过设置 keyword 关键词进行模糊查询实现检索功能。

通过 foreach 标签检索对应的 courseId，language 和 technology 同理。

该接口同样支持 paperId 字段和 teacherId 字段筛选。

图表 14 查询条件

```
<where>
    <if test="studentId != null">
        and p.id in (
        select paper_id from `collect` where student_id = #{studentId}
        )
    </if>
    <if test="courseIds != null and courseIds.size() > 0">
        and pc.course_id IN
        <foreach collection="courseIds" item="item" open="(" close=")" separator=",">#{item}</foreach>
    </if>
    <if test="languageIds != null and languageIds.size() > 0">
        and pl.language_id IN
        <foreach collection="languageIds" item="item" open="(" close=")" separator=",">#{item}</foreach>
    </if>
    <if test="technologyIds != null and technologyIds.size() > 0">
        and pl.language_id IN
        <foreach collection="technologyIds" item="item" open="(" close=")" separator=",">#{item}</foreach>
    </if>
    <if test="keyword != null">
        and (t.name like concat('%', #{keyword},'%') or p.name like concat('%', #{keyword} , '%'))
    </if>
    <if test="id != null">
        and p.id = #{id}
    </if>
    <if test="teacherId != null">
        and t.id = #{teacherId}
    </if>
</where>
```

#### 5.2.4.3 论文详情页

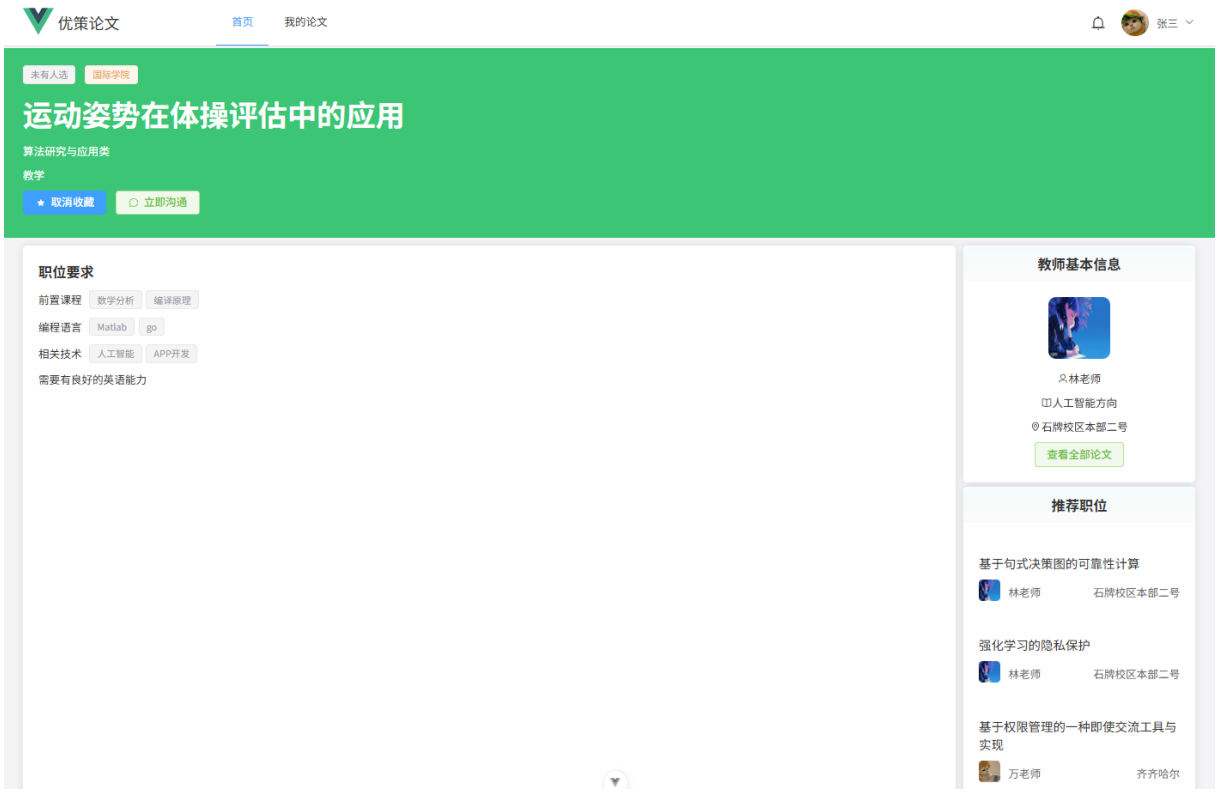
学生可以在前端查询和筛选论文。预览页学生只能浏览到论文的简要信息。当学生对论文很感兴趣时，学生可以点击论文题目进入详情页了解论文的详细信息。

前端需要设计详情页面展示论文详细信息。该页面需要展示各种不同的论文，其数据结构完全一致、展示内容不一致取决于展示的论文。

通过在路由跳转前在路由后面携带 paperId，路由跳转到新页面后，通过 router 获取 param 字段来获得跳转前的 paperId，通过 paperId 获取详细的论文信息完成前端页面渲染。

该页面不仅需要完整的论文信息还需要导师的相关信息。因此在后端查询的过程中需要通过外键关联 teacher 表，以及通过 paper_course、paper_language、paper_technology 中间表关联前置课程、编程语言以及相关技术字段。

图表 15 论文详情页




当学生认为该论文是心仪论文时，可以选择收藏该论文。当学生想要选择该论文时，则可以点击立即沟通选择与导师沟通。沟通合适后，导师可以为学生开放选择权限，开放权限后学生可以选择该论文。

收藏功能的实现遵循项目开发框架。后端开发模板 controller、service、mapper 操作 collect 表。前端点击收藏论文向后端发送插入请求，后端在 mapper 层插入数据。点击取消收藏向后端发送删除请求，后端在 mapper 层删除数据。

#### 5.2.4.4 查看导师全部论文

学生认为导师是心仪的导师，学生可以选择查看该老师的全部论文，以方便的选择到最合适的论文。

该功能的实现依靠在路由跳转前携带参数 teacherId，在跳转页面后通过 route 中的 param 获取参数 teacherId，在页面初始化时通过携带参数 teacherId 请求查询接口，获得该导师所有论文的数据。

前端已经设计好了论文组件，获得所有的 papers 后，通过 v-for 语法可以直接完成渲染。大大提高开发效率，体现了组件化开发的思想。

#### 5.2.4.5 论文选择权限同步功能

由于论文需要设置强健的唯一选择功能。所以在数据库表设计时给论文添加 studentId 字段。无人选择的论文为 null，通过这个字段来校验论文是否已经被学生选择。

导师希望沟通到的对象都是合适的学生。所以我们在设计时为学生的选择添加限制、一个学生只能选择五个论文。学生需要先选择论文才能和老师进行沟通。

学生选择论文后，则可以选择和老师沟通。但是最终的论文决定权在导师手中。导师端可以在对话界面对学生进行选择。

当导师选择了一个论文后。这个论文将不会再出现在学生端的首页供大家选择。在渲染论文的时候通过 studentId 是否为空来判断，如果不为空则说明论文已经被选择了，那么则对这个论文的访问、选择、沟通功能进行禁用操作。因而实现了论文选择逻辑的同步操作。

### 5.2.5 沟通模块的详细设计

沟通模块的核心问题在于表结构的设计。

要确定一个会话，需要五个字段——发起者 id、发起者身份、接受者 id、接受者身份、论文 id。一个会话对应着学生、导师、论文三个要素。

所以在对会话进行管理时，需要用这五个字段进行联合查询。

会话数据和会话信息数据的更新时机在每一次发消息时。

所以对会话信息和会话数据 mapper 层的方法调用时机实在 websocket 协议的 handleTextMessage 方法中，在每一次发送信息的时候都要更新。

在前端查询会话信息的时候，需要关联论文信息。而论文信息又要同时关联导师、学生信息。如果使用 sql 实现是非常繁琐的。

系统中通过在 service 层添加方法，通过依赖注入的方式获取 StudentService、teacherService、perService。对查询到的数据进行数据装饰而解决这个问题。

在设计时出现了这样一个问题。学生 A 和老师 Z 创建了一个会话。

那么导师 Z 和学生 A 创建对话的时候，是再创建一个导师 Z 对学生 A 的对话，还是继续使用之前的对话？

如果创建两个对话，会导致信息存储混乱的问题。且会话的联合主键是由发起者 id、发起者身份、接受者 id、接受者身份、论文 id。发起者和接受者是同一个人，所以其实是创建了两次。

为了防止会话混乱，设计一个防止混乱的函数。在得到数据的时候，将发送者和接受者的身份顺序重新赋值。通过 JWTutil 获取当前身份，如果和 contact 的身份不一样，则进行交换。确保得到的数据 userId 永远是当前用

户。

### 5.2.6 协同过滤推荐算法实现

本文第二章相关技术部分已经介绍了协同过滤推荐算法的原理。

通过计算所有用户的相似度，根据用户相似度进行排序。取与目标用户最相似的用户，查询该用户收藏、选择的全部论文，为目标用户推荐该最相似用户的论文。

本系统通过用户的收藏行为和选择行为来作为用户的行为数据。

定义用户 A 对论文 a 的喜爱程度指数 index，用户 A 收藏论文 a，喜爱程度指数 index+1，用户 A 选择该论文 a，喜爱程度指数 index+2。计算用户 A 对论文 a 的喜爱度 index。

依此方法计算所有用户对所有论文的喜爱度。得到一个喜爱度矩阵。通过计算矩阵向量的余弦值作为用户之间的相似度。

创建新的实体类 RelateDTO。该实体类有三个字段。studentId、paperId、index。代表用户、论文和喜爱程度指数。

算法实现：

用于计算相关系数的数学类 CoreMath：

类中有三个方法 computeNeighbor 计算相关系数并排序、relateDist 计算两个序列间的相关系数、cosineSimilarity 余弦相似度计算方法。

本文将从底层到顶层讲解该类。

（1）cosineSimilarity 余弦相似度计算

余弦相似度是通过计算两个向量的夹角余弦值来度量他们之间的相似性。

该函数接受两个向量(List<Integer> xs, List<Integer> ys)

```
public static double cosineSimilarity(List<Integer> xs, List<Integer> ys) {
    // 初始化点积为 0.0，点积用于衡量两个向量在方向上的一致性
    double dotProduct = 0.0;
    // 初始化向量 xs 的范数（模长）为 0.0，范数表示向量的长度
    double normA = 0.0;
    // 初始化向量 ys 的范数（模长）为 0.0，范数表示向量的长度
    double normB = 0.0;
    // 遍历向量中的每个元素
    for (int i = 0; i < xs.size(); i++) {
        // 计算点积，将两个向量对应位置的元素相乘并累加
        dotProduct += xs.get(i) * ys.get(i);
        // 计算向量 xs 每个元素的平方和，用于后续计算范数
        normA += Math.pow(xs.get(i), 2);
        // 计算向量 ys 每个元素的平方和，用于后续计算范数
        normB += Math.pow(ys.get(i), 2);
    }
    // 计算向量 xs 的范数，即平方和的平方根
    double normASqrt = Math.sqrt(normA);
    // 计算向量 ys 的范数，即平方和的平方根
    double normBSqrt = Math.sqrt(normB);
    // 计算并返回余弦相似度，公式为点积除以两个向量范数的乘积
    return dotProduct / (normASqrt * normBSqrt);
}
```

$$sim(i,j) = \frac{\vec{r}_i \cdot \vec{r}_j}{|\vec{r}_i||\vec{r}_j|} = \frac{\sum_{u=1}^{m} r_{ui} r_{uj}}{\sqrt{\sum_{u=1}^{m} r_{ui}^2} \sqrt{\sum_{u=1}^{m} r_{uj}^2}}$$

根据余弦相似度的公式对两个向量进行计算。

（2）relateDist 计算两个序列的相关系数：

```java
/**
 * 计算两个序列间的相关系数。
 * 此方法的核心是基于给定的两个包含 RelateDTO 对象的列表，根据指定的类型（type）筛选出匹配的元素，
 * 提取其中的索引信息形成两个整数列表，最后调用 cosineSimilarity 方法计算这两个整数列表所代表向量的余弦相似度，
 * 以此作为两个序列间的相关系数。
 *
 * @param xList 第一个包含 RelateDTO 对象的列表，代表一个序列。
 * @param yList 第二个包含 RelateDTO 对象的列表，代表另一个序列。
 * @param type  筛选匹配元素的类型，取值为 0 或其他。
 *              当 type 为 0 时，会根据 RelateDTO 对象的 paperId 属性进行匹配；
 *              当 type 不为 0 时，会根据 RelateDTO 对象的 studentId 属性进行匹配。
 * @return 两个序列间的相关系数，通过调用 cosineSimilarity 方法计算得到，是一个双精度浮点数。
 */
private static double relateDist(List<RelateDTO> xList, List<RelateDTO> yList, int type) {
    // 初始化一个新的 ArrayList 用于存储从 xList 中筛选出的元素的 index 值
    List<Integer> xs = new ArrayList<>();
    // 初始化一个新的 ArrayList 用于存储从 yList 中筛选出的元素的 index 值
    List<Integer> ys = new ArrayList<>();
    // 遍历 xList 中的每个元素
    xList.forEach(x -> {
        // 对于 xList 中的每个元素，再遍历 yList 中的每个元素
        yList.forEach(y -> {
            // 根据 type 的值进行不同的匹配操作
            if (type == 0) {
                // 当 type 为 0 时，检查 x 和 y 的 paperId 是否相等
                if (x.getPaperId().equals(y.getPaperId())) {
                    // 如果相等，将 x 的 index 值添加到 xs 列表中
                    xs.add(x.getIndex());
                    // 如果相等，将 y 的 index 值添加到 ys 列表中
                    ys.add(y.getIndex());
                }
            } else {
                // 当 type 不为 0 时，检查 x 和 y 的 studentId 是否相等
                if (x.getStudentId().equals(y.getStudentId())) {
                    // 如果相等，将 x 的 index 值添加到 xs 列表中
                    xs.add(x.getIndex());
                    // 如果相等，将 y 的 index 值添加到 ys 列表中
                    ys.add(y.getIndex());
                }
            }
        });
    });
    // 调用 cosineSimilarity 方法计算 xs 和 ys 列表所代表向量的余弦相似度，并将其作为相关系数返回
    return cosineSimilarity(xs, ys);
}
```

该函数接受两个 RelateDTO 列表和 type。

type 决定着是根据用户还是论文进行匹配。

将所有的 index 也就是权值处理为 cosineSimilarity 函数可以处理的数据然后计算余弦相似度。

（3）computeNeighbor 计算相关系数并排序：

```
/**
 * 计算相关系数并排序。
 * 此方法的主要功能是针对给定的目标键（key），在包含多个序列的映射（map）中，计算除目标键对应序列外的其他序列与目标序列的相关系数，
 * 并将结果存储在一个新的映射中，同时该映射会根据键的自然顺序进行排序。
 *
 * @param key  目标键，用于从映射中获取目标序列，该序列将与其他序列进行相关系数的计算。
 * @param map  一个映射，键为整数类型，值为包含 RelateDTO 对象的列表，代表多个序列。
 * @param type 用于传递给 relateDist 方法，指定在计算相关系数时筛选匹配元素的类型。
 *             取值为 0 或其他，当 type 为 0 时，会根据 RelateDTO 对象的 paperId 属性进行匹配；
 *             当 type 不为 0 时，会根据 RelateDTO 对象的 studentId 属性进行匹配。
 * @return 一个 TreeMap，键为整数类型，代表映射中的各个键；值为双精度浮点数，代表对应序列与目标序列的相关系数的绝对值（关系距离）。
 *         该 TreeMap 会根据键的自然顺序进行排序。
 */
public static Map<Integer, Double> computeNeighbor(Integer key, Map<Integer, List<RelateDTO>> map, int type) {
    // 初始化一个 TreeMap，用于存储每个键对应的相关系数的绝对值（关系距离），TreeMap 会根据键的自然顺序进行排序
    Map<Integer, Double> distMap = new TreeMap<>();
    // 从映射中获取目标键对应的序列
    List<RelateDTO> userItems = map.get(key);
    // 检查目标序列是否不为空
    if (CollectionUtil.isNotEmpty(userItems)) {
        // 遍历映射中的每个键值对
        map.forEach((k, v) -> {
            // 排除目标键对应的序列，避免与自身计算相关系数
            if (!k.equals(key)) {
                // 调用 relateDist 方法计算当前序列（v）与目标序列（userItems）的相关系数
                double coefficient = relateDist(v, userItems, type);
                // 计算相关系数的绝对值，作为关系距离
                double distance = Math.abs(coefficient);
                // 将当前键和对应的关系距离存入 distMap 中
                distMap.put(k, distance);
            }
        });
    }
    // 返回存储关系距离的映射
    return distMap;
}
```

使用treemap数据结构，自动排序

排除与自身计算的情况

调用relateDist计算

计算绝对值作为关系距离

在调用该函数前，会将数据处理为一个 key 对应着所有的 RelateDTO 的数据结构。该函数返回一个 Map，对应着该 key 值对所有 RelateDTO 的关系距离。在函数设计中接受 key 和 map，type 的作用是控制是按 studentId 计算还是按 paperId 计算。

因为最终我们需要找到关系距离最近的节点，我们返回的 key 使用到了 TreeMap 数据结构，会根据键的自然顺序进行排序。

在计算每两个节点之间的距离时，我们要排除与自身的计算，这个计算没有意义。计算两个节点之间的距离函数已经封装好了，直接调用即可。

CoreMath 是底层的数学计算类。在计算前，我们还需要对数据结构进行处理，处理为 CoreMath 可以直接处理的数据结构。

创建 UserCF 类，该类有一个方法 recommend，用于获取到为该用户推荐的岗位。

接收参数 userId，以及所有的 RelateDTO。

```
public class UserCF {
    /**
     * 获取推荐岗位。此函数基于用户协同过滤（User Collaborative Filtering）算法，根据用户与论文的关联数据，
     * 为指定用户推荐可能感兴趣的论文。具体步骤为：先将用户与论文的关联数据按用户分组，接着找出与指定用户关系最近的其他用户，
     * 再获取该最近邻用户关联的论文列表，最后排除指定用户已经选择过的论文，得到推荐的论文列表。
     *
     * @param userId 要获取推荐的用户的 ID，用于从关联数据中定位该用户的相关信息。
     * @param list 包含用户与论文关联信息的列表，每个元素是一个 RelateDTO 对象，存储了用户 ID、论文 ID 等信息。
     * @return 返回一个整数列表，代表为指定用户推荐的论文 ID 列表。若无法找到合适的推荐，则返回一个空列表。
     */
    public static List<Integer> recommend(Integer userId, List<RelateDTO> list) {
        // 按用户分组，将关联数据列表根据用户 ID 进行分组，存储在一个 Map 中，键为用户 ID，值为该用户关联的 RelateDTO 列表
        Map<Integer, List<RelateDTO>> userMap = list.stream().collect(Collectors.groupingBy(RelateDTO::getStudentId));
        // 获取其他用户与当前用户的关系值，调用 CoreMath 类的 computeNeighbor 方法，计算其他用户与指定用户的相关系数（关系值）
        Map<Integer, Double> userDisMap = CoreMath.computeNeighbor(userId, userMap, 0);
        // 检查关系值映射是否为空，若为空则无法进行推荐，直接返回空列表
        if (CollectionUtil.isEmpty(userDisMap)) {
            return Collections.emptyList();
        }
        // 获取关系最近的用户，找出关系值映射中的最大值，即与指定用户关系最近的相关系数
        double maxValue = Collections.max(userDisMap.values());
        // 获取所有关系值等于最大值的用户 ID，筛选出关系值等于最大值的键值对，提取其中的键（用户 ID），并存储在一个集合中
        Set<Integer> userIds = userDisMap.entrySet().stream().filter(e -> e.getValue() == maxValue)
                .map(Map.Entry::getKey).collect(Collectors.toSet());
        // 取关系最近的用户中的一个，从关系最近的用户集合中任选一个用户 ID 作为最近邻用户 ID
        Integer nearestUserId = userIds.stream().findAny().orElse(null);
        // 检查最近邻用户 ID 是否为空，若为空则无法进行推荐，直接返回空列表
        if (nearestUserId == null) {
            return Collections.emptyList();
        }
        // 最近邻用户有关联的论文列表，获取最近邻用户关联的所有论文 ID，并存储在一个列表中
        List<Integer> neighborItems = userMap.get(nearestUserId).stream().map(RelateDTO::getPaperId)
                .collect(Collectors.toCollection(ArrayList::new));
        // 指定用户关联的论文列表，获取指定用户关联的所有论文 ID，并存储在一个列表中
        List<Integer> userItems = userMap.get(userId).stream().map(RelateDTO::getPaperId)
                .collect(Collectors.toCollection(ArrayList::new));
        // 排除用户已经选择的论文，从最近邻用户关联的论文列表中移除指定用户已经选择过的论文
        neighborItems.removeAll(userItems);
        // 返回推荐的论文列表
        return neighborItems;
    }
}
```

函数第一步将 List 数据处理为 userId 对应一个 map 的形式，以便后续的处理。我们要按照 studentId 处理，所以 type 赋值为 0。

调用 CoreMath.computeNeighbor 方法得到该用户与其他用户的相关系数。

得到后要对这个结果进行判空校验和处理。

获取所有相关系数中最大值，通过最大值定位用户。

查询该用户所收藏、选择的所有论文。将论文中与本用户相同的论文去重后返回所有论文。

在 paperService 中创建 recommend 方法，用于生成推荐的论文。

```
public List<Paper> recommend() {
    // 获取当前登录的学生账户信息
    Account curStudent = JWTUtil.getCurAccount();
    List<Student> students = studentMapper.selectByPage(new Student());
    List<Paper> papers = paperMapper.selectByFilter(new Paper());
    List<Collect> collects = collectMapper.selectByCollect(new Collect());
    List<Select> selects = selectMapper.selectBySelect(new Select());
    // 创建一个空的 ArrayList 用于存储 RelateDTO 对象，这些对象将记录学生与论文之间的关联信息
    List<RelateDTO> data = new ArrayList<>();
    for (Paper paper : papers) {
        // 获取当前论文的 ID
        Integer paperId = paper.getId();
        for (Student student : students) {
            Integer studentId = student.getId();
            // 初始化关联指数为 1
            int index = 1;
            // 如果该用户收藏过，权重+1
            // 从收藏信息列表中筛选出当前学生收藏当前论文的记录
            List<Collect> collectList = collects.stream()
                    .filter(x -> x.getPaperId().equals(paperId) && x.getStudentId().equals(studentId))
                    .toList();
            // 如果收藏记录不为空，说明该学生收藏过此论文，关联指数加 1
            if (CollectionUtil.isNotEmpty(collectList)) {
                index += 1;
            }
            // 从选择信息列表中筛选出当前学生选择当前论文的记录
            List<Select> selectList = selects.stream()
                    .filter(x -> x.getPaperId().equals(paperId) && x.getStudentId().equals(studentId))
                    .toList();
            // 如果选择记录不为空，说明该学生选择过此论文，关联指数加 2
            if (CollectionUtil.isNotEmpty(selectList)) {
                index += 2;
            }
            // 如果关联指数大于 1，说明该学生与论文有一定关联，创建 RelateDTO 对象并添加到数据列表中
            if (index > 1) {
                RelateDTO relateDTO = new RelateDTO(studentId, paperId, index);
                data.add(relateDTO);
            }
        }
    }
    // 调用 UserCF 类的 recommend 方法，传入当前学生的 ID 和关联数据列表，获取推荐的论文 ID 列表
    List<Integer> paperIds = UserCF.recommend(curStudent.getId(), data);
    // 从所有论文列表中筛选出 ID 在推荐 ID 列表中的论文，形成推荐论文列表
    List<Paper> result = papers.stream().filter(x -> paperIds.contains(x.getId())).toList();
```

首先要获取信息，通过 JWTUtil 获取当前用户，通过 mapper 获取用户数据、收藏数据、选择数据、论文数据。

将数据处理为 RelateDTO 的数据形式、调用 UserCF 中的 recommend 方法即可。

将返回的推荐的 paperId 转换为 paper。

前端页面中设定的推荐论文个数为三个。但是最终推荐的论文数量可能大

于三也可能小于三。

如果大于三直接截取数组即可。

当系统中选择数据较少的时候，协同过滤推荐算法有可能无法生成推荐论文。所以当不足三个的时候将随机添加论文添加至三个。

```
/**
 * 随机获取指定数量的论文
 * 该方法用于在推荐结果不足时，从所有论文列表中随机选择指定数量的论文进行补充。
 * 会排除已经在结果列表中的论文，以避免重复推荐。
 *
 * @param num   需要随机获取的论文数量
 * @param papers   所有论文的列表
 * @param result   已经推荐的论文列表
 * @return   随机选择的论文列表
 */
public List<Paper> getRandomPaper(int num, List<Paper> papers, List<Paper> result) {
    // 打乱所有论文列表的顺序
    Collections.shuffle(papers);
    // 如果已经有推荐结果，从所有论文列表中排除这些结果
    if (CollectionUtil.isNotEmpty(result)) {
        papers = papers.stream().filter(x -> !result.contains(x)).collect(Collectors.toList());
    }
    // 如果剩余论文数量大于需要的数量，截取前 num 个论文；否则返回剩余的所有论文
    return papers.size() > num ? papers.subList(0, num) : papers;
}
```

通过 getRandomPaper 方法对论文列表进行补充。

第一个参数 num 代表需要随机获取的论文数量。

函数实现首先打乱所有 paper，保证获取论文的随机性。

如果已经有推荐的论文，则先排除这些论文，以免推荐重复论文，而在论文列表中添加随机论文即可。

### 5.2.7 Websocket 实时通信

#### 5.2.7.1 选择 websocket 协议的必要性

由于 HTTP 协议是基于请求 - 响应模型的，客户端发送请求，服务器返回响应，连接在响应完成后即结束。对于实时通信，需要不断地建立新连接来发送和接收数据，这会带来额外的延迟和开销。WebSocket 建立的是持

久连接，在连接成功后，客户端和服务器可以随时相互发送数据，无需频繁地建立和断开连接，大大提高了实时性。

HTTP 协议本身是单向的，客户端发起请求，服务器响应请求。如果服务器有数据要主动推送给客户端，需要通过轮询等方式来实现，即客户端定时向服务器发送请求，询问是否有新数据，这会浪费大量的带宽和服务器资源，且实时性较差。WebSocket 支持双向通信，服务器可以在任何时候主动向客户端发送数据，客户端也可以随时向服务器发送数据，实现了真正意义上的实时双向通信。

#### 5.2.7.2 Websocket 配置

后端配置：

Websocket 后端需要配置拦截器和处理器。用于拦截 webSocket 请求以及处理 websocket 请求。

创建 wsHandler 处理器和 wsInterceptor 拦截器。

（1）wsInterceptor 拦截器

拦截器继承 HttpSessionHandshakeInterceptor 重写 beforeHandshake 方法和 afterHandshake 方法，处理在建立 webSocket 的连接前后的逻辑。beforeHandshake 方法，在连接前将 token 通过获取 request 确定用户信息，将其记录在 session 中，用于在 handler 中进行处理。

（2）wsHandler 处理器

处理器继承 AbstractWebSocketHandler 重写 afterConnectionEstablished、handleTextMessage 方法。处理当建立连接后、处理信息时的逻辑。

在 handler 中依赖注入 messageService 和 recentContactService。用于将信息、对话与数据库同步。

定义 sessionMap 记录 session。当用户 A 向用户 B 发送消息后，在 map 中找到用户 B，发送更新通知即可。

在 afterConnectionEstablished 方法中，取出 beforeHandshake 中记录的用户信息。将用户信息和会话信息存储在 sessionMap 哈希表中。

handleTextMessage 函数处理后端收到前端消息的逻辑。

当收到消息后，将消息处理为 Message 格式插入消息表。调用 recentContactService.updateOrAdd，如果当前会话已经存在则更新会话、否则新建会话。在 sessionMap 表中查看目标用户是否存在、如果存在则通知目标用户更新数据以实现实时通信。

前端配置：

创建 chat.js，初始化 WebSocket，确定后端端口号、携带 token。

在 chat.onopen 事件中处理连接逻辑，在 onmessage 事件中处理收到后端发送信息逻辑。

当收到后端的更新信号后、在前端重新更新数据即可同步信息，实现实时通信。

# 结论

本文针对高校学生与导师的需求。开发了高校本科生论文选题系统。基于 Springboot 和 Vue3 搭建，采用前后端分离的方式。实现了导师端进行简单快捷的论文创建与管理，学生端能够方便筛选、选择论文。并通过权限同步解决一个论文多个同学选择的选择混乱问题。

本系统的创新点和优势在于通过协同过滤推荐算法，基于用户的操作数据，为同学推荐合适的论文。相比于传统的论文选题系统，能够为学生用户提供更加合适的论文。基于 webSocket 协议实现的实时通信系统。支持学生和导师线上实时通过，大大提高沟通效率。

虽然系统实现了论文的创建、选择、推荐、管理功能。但是系统仍然存在一些有待改进的部分。在实时沟通功能中，用户可以发生信息和表情。但是系统不支持发送简历附件。在这方面仍有优化的空间。另一方面，前端的页面设计不够精美。可以通过颜色和结构的设计使得用户体验更加优秀。

# 致谢

感谢我的导师郝振明老师，谢谢他对我的悉心指导。在论文撰写过程中提出很多中肯、严谨的意见。帮助我成功完成我的论文。

另外还要感谢我的女朋友，她在我困难的时候给予帮助，开心的时候给予陪伴。她的理解、包容和爱是我不断前进的动力。感谢她的一路陪伴，希望未来的日子我也能和她携手走过更多美好时光。

我还要感谢我的父母，他们一直在背后支持我，爱着我，让我感受到温暖。她们总在我最难的时候提供帮助，从不索求回报。

最后要感谢我的辅导员老师和同学们。感谢你们在生活中的帮助。在此一并感谢。

# 附录A

/*是正文主体的补充项目，并不是必需的。下列内容可以作为附录：

（1）为了整篇材料的完整，插入正文又有损于编排条理性和逻辑性的材料；

（2）由于篇幅过大，或取材于复制件不便编入正文的材料；

（3）对一般读者并非必须阅读，但对本专业人员有参考价值的资料；（如外文文献复印件及中文译文、公式的推导、程序流程图、图纸、数据表格等）

附录按“附录 A，附录 B，附录 A1“等编号。

请单击样式“附录 1”为第 1 级的附录编号，样式“附录 2”为第二级的附录编号，样式“附录 3”控制第三级别的样式。*/

# 参考文献

/* 如需要撰写参考文献的帮助，请单击插入 → 自动图文集→ 参考文献，选择“参考文献著录格式说明”词条，将插入详细的各种参考文献著录格式说明与示例，也可选择插入常用的文献类型示例词条，如“期刊论文著录示例”词条。

引用文献标示应置于所引内容最末句的右上角。所引文献编号用阿拉伯数字置于方括号“[ ]”中，如“二次铣削[1]”。如同一处引用了多个文献，文献编号间用逗号分隔，如“二次铣削[1，3] ”。当提及的参考文献为文中直接说明时，其序号应该与正文排齐，如“由文献[8，10～14]可知”。

经济、管理类论文引用文献，若引用的是原话，要加引号，一般写在段中；若引的不是原文只是原意，文前只需用冒号或逗号，而不用引号。在参考文献之外，若有注释的话，建议采用夹注，即紧接文句，用圆括号标明。或者以脚注的形式排在页面底端，按①,②,③编号。

可使用如下两种方法之一插入参考文献，如参考文献较多且在写作过程中更改较大，建议采用第一种方法。

方法一：选择插入 → 引用→ 脚注与尾注，将显示“脚注与尾注”对话框，选择“尾注”，输入参考文献内容（系统会自动插入参考文献的编号，并跳转到参考文献内容输入处）请通过“字体”对话框取消参考文献内容前的编号的上标格式，并加上方括号。如果文中多处引用了同一篇文献，从第二处起请采用插入 → 引用→ 交叉引用的方法插入文献标示。这样当增删参考文献的时候，编号会自动调整。

方法二：在文中直接插入引用文献序号并将其设为上标，在文后输入参考文献的内容。这种方法的缺点是当增删改参考文献时，需要手工修改参考文献的编号。*/

**A.期刊论文**

［序号］作者．文献题名．刊名，出版年份，卷号(期号)：起止页码

[1] 袁庆龙，候文义．Ni-P 合金镀层组织形貌及显微硬度研究．太原理工大学学报，2001，32(1)：51-53

**B.专著**

［序号］作者．书名．版本（第 1 版不标注）．出版地：出版者，出版年．页码

[3] 蒋有绪，郭泉水，马娟，等．中国森林群落分类及其群落学特征 ．北京：科学出版社，1998．179-193

**C.学位论文**

［序号］作者．论文题名：学位论文级别．保存地点：保存单位，答辩年份

[7] 张和生．地质力学系统理论：博士学位论文．太原：太原理工大学，1998

**D.报纸文章**

［序号］作者．题名．报纸名，出版日期(版次)

[13] 谢希德．创造学习的思路．人民日报，1998-12-25(10)

**E 会议论文集**

［序号］作者．文章名．见（英文用 In）：主编．论文集名．(供选择项：会议名，会址，开会年)出版地：出版者，出版年．起止页码

[6] 孙品一．高校学报编辑工作现代化特征．见：中国高等学校自然科学学报研究会．科技编辑学论文集(2)．北京：北京师范大学出版社，1998．10-22

**F.报告**

［序号］ 主要责任者．文献题名．报告地：报告会主办单位，年份

［9］冯西桥．核反应堆压力容器的 LBB 分析．北京：清华大学核能技术设计研究院，1997

**G. 专利文献**

[序号] 专利申请者或所有者．专利题名．专利国别，专利号．发布日期

[11] 姜锡洲．一种温热外敷药制备方案．中国，881056078 ．1983-08-12

**H.国际、国家标准**

［序号］ 标准代号．标准名称．出版地：出版者，出版年

［1］GB/T 16159—1996．汉语拼音正词法基本规则．北京：中国标准出版社，1996

**I 翻译类文献**

[序号]﹎作者．书名．译者．版次（第一版应省略）．出版地：出版者，出版年．引用部分起止页

[2] 斯蒂芬·P·罗宾斯. 管理学．黄卫伟，等译．第七版．北京：中国人民大学出版社，2003

**J.专著中析出的文献**

［序号］ 析出责任者．析出题名．见：专著责任者．书名．出版地：出版者，出版年．起止页码

［12］罗云．安全科学理论体系的发展及趋势探讨．见：白春华，何学秋，吴宗之．21 世纪安全科学与技术的发展趋势．北京：科学出版社，2000．1-5

**K.电子文献**

1、电子文献转载其他非电子文献（如电子图书、电子报刊），应在源文献的著录格式后著录电子文献的引用日期和获取和访问路径，其文献类型标志使用复合标志，即[文献类型标志/文献载体标志]。

[1] 江向东．互联网环境下的信息处理与图书管理系统解决方案[J/OL]．情报学报，1999，18(2)：4[2000-01-18] ．http://www.chinainfo.gov.cn/periodical/gbxb/gbxb99/gbxb990203．

2、 非第 1 种情况者使用下面著录格式：（注：联机文献中无出版地、出版者、出版年的可省略。）

［序号］主要责任者．题名[文献类型/载体类型]．出版地：出版者，出版年(更新或修改日期)[引用日期] ．获取和访问路径．

[21] 萧钮．出版业信息化迈入快车道[EB/OL] ．(2001-12-19)[2002-04-15]

http://www. creader.com/news/20011219/200112190019.html.

附：参考文献著录中的文献类别代码

普通图书：M 会议录：C 汇编：G 报纸：N 期刊：J 学位论文：D 报告：R

标准：S 专利：P 数据库：DB 计算机程序：CP 电子公告：EB

电子文献载体类型标志如下：磁带 MT，磁盘 DK，光盘 CD，联机网络 OL。