

J2EE

课程设计报告

姓名： 张彬

班级： 22软工1班

成绩：

电子与计算机工程学院

School of Electronic & Computer Engineering

2025年4月

目录

[一、引言 4](#_Toc196868192)

[1.1 选题背景 4](#_Toc196868193)

[1.1.1技术演进驱动开发范式升级，全栈架构构建高效技术底座 4](#_Toc196868194)

[1.1.2用户需求迭代催生“交互型新闻生态”，技术赋能参与式体验升级 4](#_Toc196868195)

[1.1.3行业痛点与政策机遇双重驱动，构建技术领先的媒体融合平台 4](#_Toc196868196)

[1.2 选题依据 4](#_Toc196868197)

[1.2.1选题渊源 4](#_Toc196868198)

[1.2.2技术渊源 5](#_Toc196868199)

[1.3 主要研究内容 5](#_Toc196868200)

[1.3.1Vue架构设计与组件化开发 5](#_Toc196868201)

[1.3.2Vue的状态管理与数据交互 5](#_Toc196868202)

[1.3.3性能优化与用户体验提升 6](#_Toc196868203)

[1.3.4SpringBoot架构设计与接口开发 6](#_Toc196868204)

[二、开发环境与工具 6](#_Toc196868205)

[2.1硬件与操作系统 6](#_Toc196868206)

[2.2集成开发环境 IDE 6](#_Toc196868207)

[2.3项目依赖 7](#_Toc196868208)

[2.3.1前端 7](#_Toc196868209)

[2.3.2后端 7](#_Toc196868210)

[三、需求分析 8](#_Toc196868211)

[3.1系统定位与目标 8](#_Toc196868212)

[3.2核心功能需求 8](#_Toc196868213)

[3.2.1内容生产与管理 8](#_Toc196868214)

[3.2.2用户体验优化 8](#_Toc196868215)

[3.2.3技术架构与性能优化 9](#_Toc196868216)

[3.2.4商业化与支付功能 9](#_Toc196868217)

[3.2.5安全与合规 9](#_Toc196868218)

[3.3非功能需求 9](#_Toc196868219)

[3.4技术选型与实现方案 10](#_Toc196868220)

[3.4.1后端概括 10](#_Toc196868221)

[3.4.2前端概括 10](#_Toc196868222)

[3.5预期效果 10](#_Toc196868223)

[3.5.1用户体验 10](#_Toc196868224)

[3.5.2运营指标 10](#_Toc196868225)

[3.5.3技术价值 10](#_Toc196868226)

[3.6风险与应对策略 10](#_Toc196868227)

[3.6.1技术风险 10](#_Toc196868228)

[3.6.2运营风险 10](#_Toc196868229)

[四、系统设计 11](#_Toc196868230)

[4.1概要设计（B / S结构） 11](#_Toc196868231)

[4.1.1系统模块划分 11](#_Toc196868232)

[4.1.2类图 11](#_Toc196868233)

[4.1.3时序图 12](#_Toc196868234)

[4.1.3活动图 13](#_Toc196868235)

[4.1.1时序图 14](#_Toc196868236)

[4.2详细设计 15](#_Toc196868237)

[4.2.1 用户相关类 15](#_Toc196868238)

[4.2.2 文章相关类 15](#_Toc196868239)

[4.2.3 交互相关类 16](#_Toc196868240)

[4.2.4 其他类 16](#_Toc196868241)

[4.2.5 E-R图 17](#_Toc196868242)

[五、系统实现及测试 17](#_Toc196868243)

[5.1系统功能实现 17](#_Toc196868244)

[5.1.1 前端核心功能实现 17](#_Toc196868245)

# 一、引言

## 1.1 选题背景

### 1.1.1技术演进驱动开发范式升级，全栈架构构建高效技术底座

随着Web应用复杂度提升，前后端分离与工程化开发成为必然趋势。前端领域，Vue3凭借响应式系统与组合式API（Composition API）大幅提升组件复用性，搭配TypeScript的静态类型检查，显著增强大型项目的可维护性，尤其适合新闻平台复杂交互逻辑的模块化开发；后端采用Spring Boot框架，依托“约定优于配置”理念快速构建REST ful API，结合My Batis、Spring Security等生态组件，实现数据持久化、安全认证与接口管理的高效集成。三者结合形成“Vue3+TS+Spring Boot”全栈技术栈，既满足移动端优先的响应式设计需求（适配手机、平板、PC多端界面），又通过Vite打包优化与Redis缓存机制，保障高并发场景下的性能稳定性，为新闻平台的快速迭代与规模化扩展奠定技术基础。

### 1.1.2用户需求迭代催生“交互型新闻生态”，技术赋能参与式体验升级

现代用户对新闻消费的需求已从“单向信息获取”转向“深度交互参与”：一方面，个性化推荐、社交化互动（评论/点赞/关注）、实时弹幕与投票等功能，要求平台具备高效的数据实时交互与状态管理能力——前端通过Vue3的Pinia实现全局状态共享，结合WebSocket支持评论区实时更新；后端利用Spring Boot整合Redis缓存用户行为数据，基于协同过滤算法构建个性化推荐引擎，形成“内容生产-传播-互动”的闭环生态。另一方面，面对信息过载，前端通过TS严格定义数据接口（新闻标签、分类），结合虚拟滚动技术优化长列表渲染性能；后端集成Elasticsearch实现多维度快速检索（关键词、时间、来源筛选），帮助用户在海量内容中精准定位兴趣点，从“被动接收”转变为“主动筛选”，大幅提升信息获取效率。

### 1.1.3行业痛点与政策机遇双重驱动，构建技术领先的媒体融合平台

传统新闻平台普遍面临技术架构老旧（前端性能不足、后端扩展性差）、交互功能单一、数据安全机制薄弱等问题，难以适应自媒体崛起与UGC内容爆发的行业趋势。本选题基于主流技术栈构建可扩展架构：前端支持富文本编辑、多媒体上传等UGC功能，后端通过Spring Boot实现内容整合，使用MySQL结构化存储数据，平衡用户创作自由度与内容合规性；同时响应“互联网+媒体”融合政策，借助Spring Security实现分级权限管理和数据加密传输，满足《个人信息保护法》等法规要求。项目兼具技术前瞻性与实际价值——Vue3的高效渲染、TS的类型安全、Spring Boot的微服务化扩展，不仅解决传统平台痛点，更通过“PGC专业内容+UGC用户原创”的混合模式，助力新闻平台从信息载体升级为社交化、个性化的互动社区，为媒体数字化转型提供可复用的技术范本。

## 1.2 选题依据

### 1.2.1选题渊源

大二时，谢秀娟老师的ASP.NET课程设计课的项目统一要求“新闻网站”，这是我第一次接触新闻网站设计，这样简单的应用程序可以同时包含推荐算法、交互功能、状态管理，权限管理，跨域访问控制。我便对新闻网站开发产生了兴趣。

### 1.2.2技术渊源

B站自学Vue3 TS和Spring Boot技术

数据库原理——操凤萍老师

## 1.3 主要研究内容

### 1.3.1Vue架构设计与组件化开发

响应式布局设计：利用Vue3的响应式系统，结合Elements PLUS等组件工具，实现新闻网站在不同设备（手机、平板、PC）上的自适应布局。通过监听设备屏幕尺寸变化，动态调整页面元素的大小、位置和显示方式，确保用户在各种设备上都能获得良好的视觉体验。例如，在手机端采用单列布局，方便用户单手操作；在PC端采用多列布局，提高信息展示密度。

组件化构建：将新闻网站的各个功能模块拆分为独立的 Vue 组件，如新闻列表组件、新闻详情组件、评论组件等。通过组件化开发，提高代码的复用性和可维护性。每个组件都有明确的职责和接口，便于团队成员分工协作开发。例如，新闻列表组件负责展示新闻的标题、摘要和发布时间等信息，当有新的新闻数据时，组件能够自动更新显示内容。

路由管理：使用Vue Router实现新闻网站的路由管理，根据用户的访问路径展示不同的页面内容。支持路由导航守卫，对用户的访问权限进行验证和控制。例如，对于需要登录才能访问的页面，当未登录用户尝试访问时，将自动跳转到登录页面。同时，支持路由懒加载，提高网站的加载速度，只有当用户访问特定页面时，才加载该页面的组件代码。

### 1.3.2Vue的状态管理与数据交互

引入Pinia作为状态管理库，对新闻网站的全局状态进行集中管理。例如，用户的登录状态、收藏列表、阅读历史等信息都可以存储在Pinia中。通过状态管理，确保不同组件之间的数据共享和同步，避免数据不一致的问题。同时，Pinia提供了响应式的状态更新机制，当状态发生变化时，相关的组件会自动更新显示内容。

使用Axios库与后端的Spring Boot服务进行数据交互。在前端组件中，通过发送HTTP请求获取新闻数据、提交评论、点赞等操作。处理请求的响应结果，根据不同的状态码进行相应的处理，如显示成功提示、错误提示等。例如，当用户提交评论时，前端组件会发送一个POST请求到后端的评论接口，后端处理请求并返回处理结果，前端根据结果显示评论成功或失败的提示信息。异步请求。

### 1.3.3性能优化与用户体验提升

采用虚拟列表、代码分割、懒加载等技术优化新闻网站的性能。虚拟列表技术可以只渲染用户可见区域的新闻列表项，减少 DOM 操作，提高页面的滚动流畅性。代码分割和懒加载可以将大的组件或模块拆分成多个小的文件，根据用户的访问需求动态加载，减少首屏加载时间。例如，对于新闻详情页面中的图片和视频等资源，可以采用懒加载的方式，只有当用户滚动到该位置时才加载资源。

为新闻网站添加动画效果和过渡效果，增强用户的交互体验。例如，在新闻列表中，当用户点击新闻标题时，使用淡入淡出的动画效果显示新闻详情页面；在评论组件中，当用户提交评论时，使用动画效果显示新评论的添加。同时，优化表单验证和错误提示，提供友好的用户反馈，让用户能够清楚地知道自己的操作结果。

### 1.3.4SpringBoot架构设计与接口开发

基于Spring Boot构建微服务架构，将新闻网站的不同业务功能拆分为多个独立的微服务，如新闻服务、用户服务、评论服务等。每个微服务都有自己独立的数据库和业务逻辑，通过RESTful API进行通信。采用微服务架构可以提高系统的可扩展性和维护性，便于团队进行并行开发和部署。例如，当需要对新闻服务进行性能优化时，可以单独对新闻服务进行升级和扩展，而不会影响其他服务的正常运行。

使用Spring MVC框架开发RESTful API，为前端提供数据接口。定义清晰的API接口规范，包括请求方法、请求参数、响应格式等。对API进行版本管理，确保不同版本的API之间的兼容性。例如，为新闻列表接口定义不同的请求参数，如分页参数、排序参数等，方便前端根据不同的需求获取新闻数据。

选择合适的数据库（MySQL）存储新闻网站的数据，包括新闻信息、用户信息、评论信息等。使用MyBatis或JPA等框架实现数据库的CRUD操作，提高开发效率。设计合理的数据库表结构和索引，优化数据库查询性能。例如，为新闻表的发布时间字段添加索引，提高按时间排序查询新闻的效率。

# 二、开发环境与工具

## 2.1硬件与操作系统

Legion Y7000P IAH7

Windows 11 专业版 22631.3447

## 2.2集成开发环境 IDE

|  |  |
| --- | --- |
| **名称** | **版本号** |
| **Visual Studio Code ×64** | 1.99.3 |
| **Node** | 20.18.3 |
| **IntelliJ IDEA Ultimate** | 2024.2 |
| **MySQL** | 8.0.37 |
| **Navicat Premium** | 15.0.12 |
| **Web Tunnel** | Undefine |
| **Redis** | 7.4.2 |

## 2.3项目依赖

### 2.3.1前端

|  |  |
| --- | --- |
| **名称** | **版本号** |
| **ckeditor/ckeditor5-vue** | ^7.3.0, |
| **element-plus/icons-vue** | ^2.3.1, |
| **fortawesome/fontawesome-free** | ^6.7.2, |
| **tailwindcss/postcss** | ^4.1.4, |
| **tailwindcss/vite** | ^4.0.13, |
| **tiptap/core** | ^2.11.5, |
| **tiptap/extension-font-family** | ^2.11.5, |
| **tiptap/extension-image** | ^2.11.5, |
| **tiptap/extension-table** | ^2.11.5, |
| **tiptap/extension-text-style** | ^2.11.5, |
| **tiptap/extension-youtube** | ^2.11.5, |
| **tiptap/starter-kit** | ^2.11.5, |
| **tiptap/vue-3** | ^2.11.5, |
| **vee-validate/i18n** | ^4.15.0, |
| **vee-validate/rules** | ^4.15.0, |
| **vue/compiler-sfc** | ^3.5.13, |
| **vueuse/core** | ^13.1.0, |
| **wangeditor/editor** | ^5.1.23, |
| **wangeditor/editor-for-vue** | ^5.1.12, |
| **axios** | ^1.8.3, |
| **chinese-lunar** | ^0.1.4, |
| **chinese-lunar-calendar** | ^1.0.1, |
| **dayjs** | ^1.11.13, |
| **element-plus** | ^2.9.6, |
| **lodash** | ^4.17.21, |
| **lunar-calendar** | ^0.1.4, |
| **lunar-javascript** | ^1.7.1, |
| **markdown-it** | ^14.1.0, |
| **pinia** | ^3.0.1, |
| **pinia-plugin-persistedstate** | ^4.2.0, |
| **qrcode-generator** | ^1.4.4, |
| **quill-image-super-solution-module** | ^2.0.1, |
| **vee-validate** | ^4.9.5, |
| **vue** | ^3.5.13, |
| **vue-cropper** | ^1.1.4, |
| **vue-loading-overlay** | ^6.0.6, |
| **vue-router** | ^4.5.0, |
| **vue3-puzzle-vcode** | ^1.1.7, |
| **vue3-toastify** | ^0.2.8 |

### 2.3.2后端

|  |  |
| --- | --- |
| **名称** | **版本号** |
| **spring-boot-starter-parent** | 3.3.4 |
| **SEUNewsWebsite** | 0.0.1-SNAPSHOT |
| **java** | 17 |
| **lombok** | 1.18.26 |
| **spring-ai** | 1.0.0-M3 |
| **spring-boot-starter-web** | 继承 |
| **spring-boot-starter-data-jpa** | 继承 |
| **spring-boot-starter-security** | 继承 |
| **spring-boot-starter-validation** | 继承 |
| **mysql-connector-j** | 8.3.0 |
| **commons-fileupload** | 1.5 |
| **commons-io** | 2.18.0 |
| **spring-boot-starter-test** | 继承 |
| **modelmapper** | 3.1.1 |
| **spring-boot-starter-actuator** | 继承 |
| **spring-boot-starter-data-redis** | 继承 |
| **spring-boot-starter-mail** | 继承 |
| **spring-boot-starter-freemarker** | 继承 |
| **spring-ai-ollama-spring-boot-starter** | 1.0.0-M3 |

# 三、需求分析

## 3.1系统定位与目标

《东南新闻网》定位为区域性综合新闻服务平台，聚焦福建及周边地区的时政、民生、文化、经济等领域，依托 AI 技术实现内容生产与分发的智能化，目标包括：

提升用户粘性：通过个性化推荐、实时互动等功能，增强用户留存率。

强化内容竞争力：结合 AI 生成能力（如自动摘要、热点预测），提升内容生产效率与质量。

拓展商业化路径：通过会员订阅、付费内容、广告精准投放（待定）等模式，实现可持续运营。

## 3.2核心功能需求

### 3.2.1内容生产与管理

多模态内容发布：支持图文、视频、音频等格式，提供富文本编辑器与多媒体素材库。

AI辅助创作（待定）：调用Ollama部署的DeepSeek 1.5B模型自动生成新闻摘要、标题优化建议，提升编辑效率。

审核流程：引入多级人工审核+AI智能过滤（敏感词、低质内容识别，待定），确保内容合规。

内容运营工具：标签与分类管理：支持动态标签体系（如 “最新新闻”“独家报道”），结合SEO优化提升搜索引擎排名（待定）。

热点追踪：通过实时舆情监测（如社交媒体、RSS源）自动抓取热点事件，生成候选选题。（待定）

### 3.2.2用户体验优化

个性化推荐系统：混合推荐算法：结合协同过滤（用户行为相似性，待定）、基于内容（关键词匹配）、深度学习（多模态特征提取，待定），实现精准推送。

动态更新：实时响应用户行为（如点击、停留时长），调整推荐策略。

冷启动策略：对新用户基于地域、年龄等基础信息推荐热门内容，对新新闻通过相似内容关联提升曝光。

互动与社交功能：评论与点赞：支持多级回复、评论举报与审核，引入 Redis 缓存提升加载速度。

社交分享：集成微信、微博等第三方分享接口，支持一键转发至社交平台。

用户反馈：提供 “新闻纠错”“内容建议” 入口，结合NLP分析用户反馈优化内容（待定）。

### 3.2.3技术架构与性能优化

缓存与分布式系统

Redis缓存：对热门新闻、用户会话、推荐结果等进行缓存，设置合理过期时间（如 10 分钟）。

分布式锁：在高并发场景下保证数据一致性。

高可用部署

数据库优化：使用MySQL主从复制，分库分表存储海量新闻数据。

### 3.2.4商业化与支付功能

会员体系

分级权益：提供免费会员（基础内容）、付费会员（独家报道、无广告），支持按年 / 月订阅。

积分系统：用户通过评论、分享、付费等行为获取积分，可兑换优惠券或增值服务。

支付集成

支付宝支付：支持扫码支付、H5 支付，实现订单生成、支付回调、退款等全流程。

虚拟商品交易：用户可购买付费内容（如深度报告），支持按篇或打包购买。

### 3.2.5安全与合规

权限控制

RBAC 模型：管理员（内容终审）、编辑（内容发布）、普通用户（浏览评论）等角色权限隔离。

数据加密：用户密码采用BCrypt算法加密存储，敏感数据（如支付信息）通过 HTTPS 传输。

风险防控

防爬虫机制：限制 IP 访问频率，对异常请求进行拦截。

内容安全：使用阿里云内容安全（OSS）接口，实时检测图片、视频中的违规内容。

## 3.3非功能需求

|  |  |
| --- | --- |
| *维度* | *具体要求* |
| *性能* | 首页响应时间 ≤2 秒（95% 请求） |
| *支持 5000 + 并发访问* | Redis 缓存命中率≥90% |
| *可用性* | 系统可用性 ≥99.9% |
| *故障恢复时间* | ≤30 分钟（自动切换至备用节点） |
| *扩展性* | 支持水平扩展（新增服务器节点） |
| *支持功能模块解耦* | 独立推荐服务 |
| *兼容性* | 主流浏览器及移动设备（iOS/Android）适配 |
| *可维护性* | 代码模块化（MVC 架构） |
| *完善的日志与监控系统* | Spring Boot Actuator |

## 3.4技术选型与实现方案

### 3.4.1后端概括

框架：Spring Boot 3.3.4（支持 Java 17）+ Spring Security（权限控制）。

数据层：MySQL 8.0（主从复制）+ Redis 7.0（缓存与实时数据）。

AI 集成：Spring AI 1.0.0-M3 + Ollama 模型（本地部署或调用 API），实现内容生成与推荐。

### 3.4.2前端概括

技术栈：Vue.ts 3 + Element Plus（响应式设计）+ vite（打包优化）。

性能优化：图片懒加载、CSS Sprite、Gzip 压缩。

部署

容器化：Docker + Kubernetes（自动扩缩容，待定）。

监控：Prometheus + Grafana（实时性能监控，待定）。

## 3.5预期效果

### 3.5.1用户体验

个性化推荐：用户点击率提升 30%+，日均使用时长增加 20%。

内容质量：AI 生成内容占比达 20%，人工审核效率提升 50%。

### 3.5.2运营指标

流量增长：上线首年 PV 突破 1000 万，注册用户超 50 万。

商业化：付费会员转化率达 5%，年营收增长 50%。

### 3.5.3技术价值

高可用性：系统故障率降低至 0.1% 以下，平均恢复时间（MTTR）缩短至 15 分钟。

可扩展性：支持 10 万 + 并发访问，单节点日均处理请求量达百万级。

## 3.6风险与应对策略

### 3.6.1技术风险

AI 模型效果不佳：定期进行 A/B 测试，优化推荐算法参数。

高并发性能瓶颈：采用 Redis 缓存热点数据，引入消息队列（RabbitMQ，待定）异步处理请求。

### 3.6.2运营风险

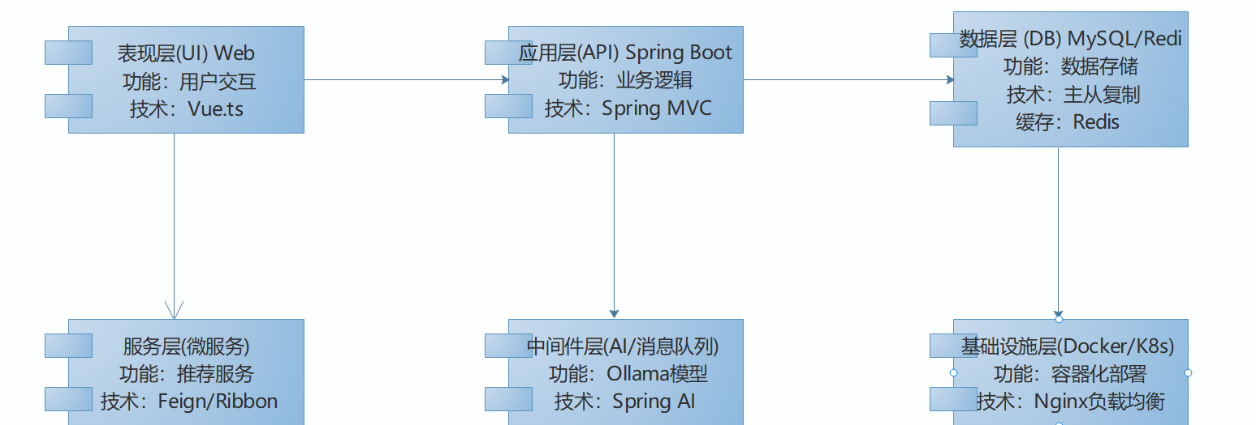
内容合规问题：强化人工审核与 AI 过滤，定期进行内容安全审计。

用户流失：通过用户画像分析流失原因，针对性优化功能与内容。

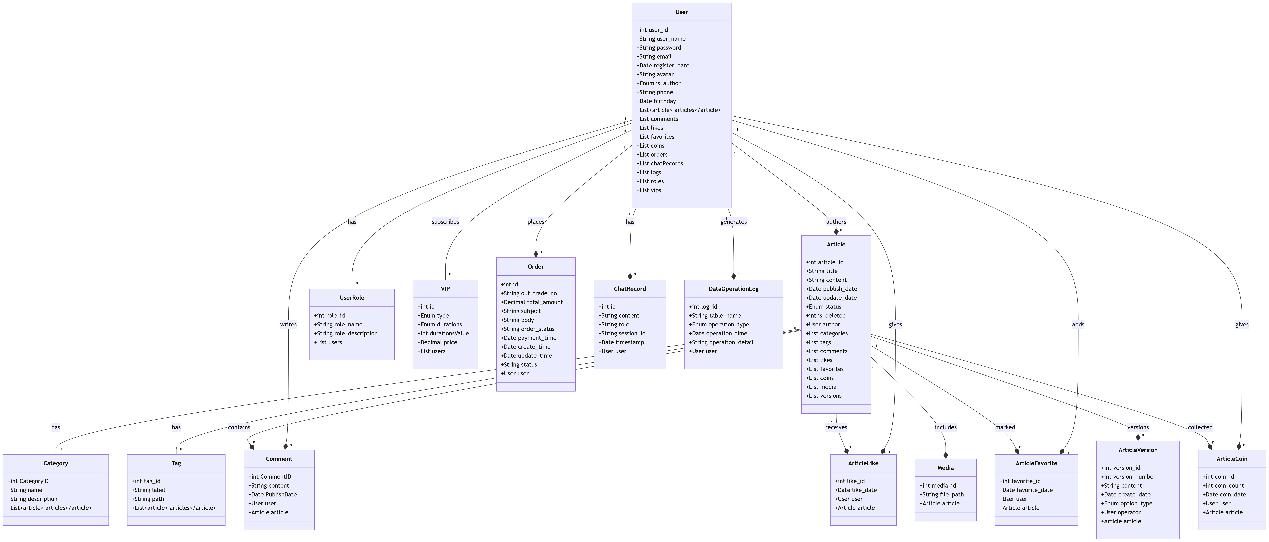
# 四、系统设计

## 4.1概要设计（B / S结构）

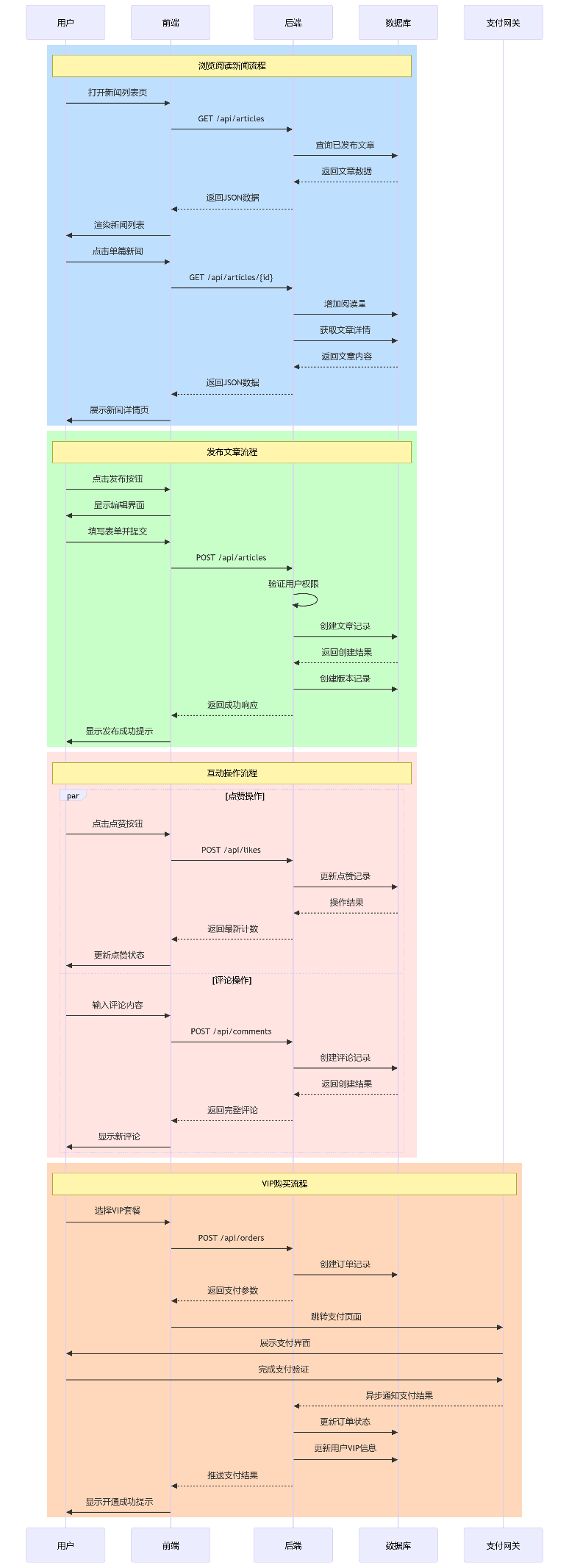
### 4.1.1系统模块划分



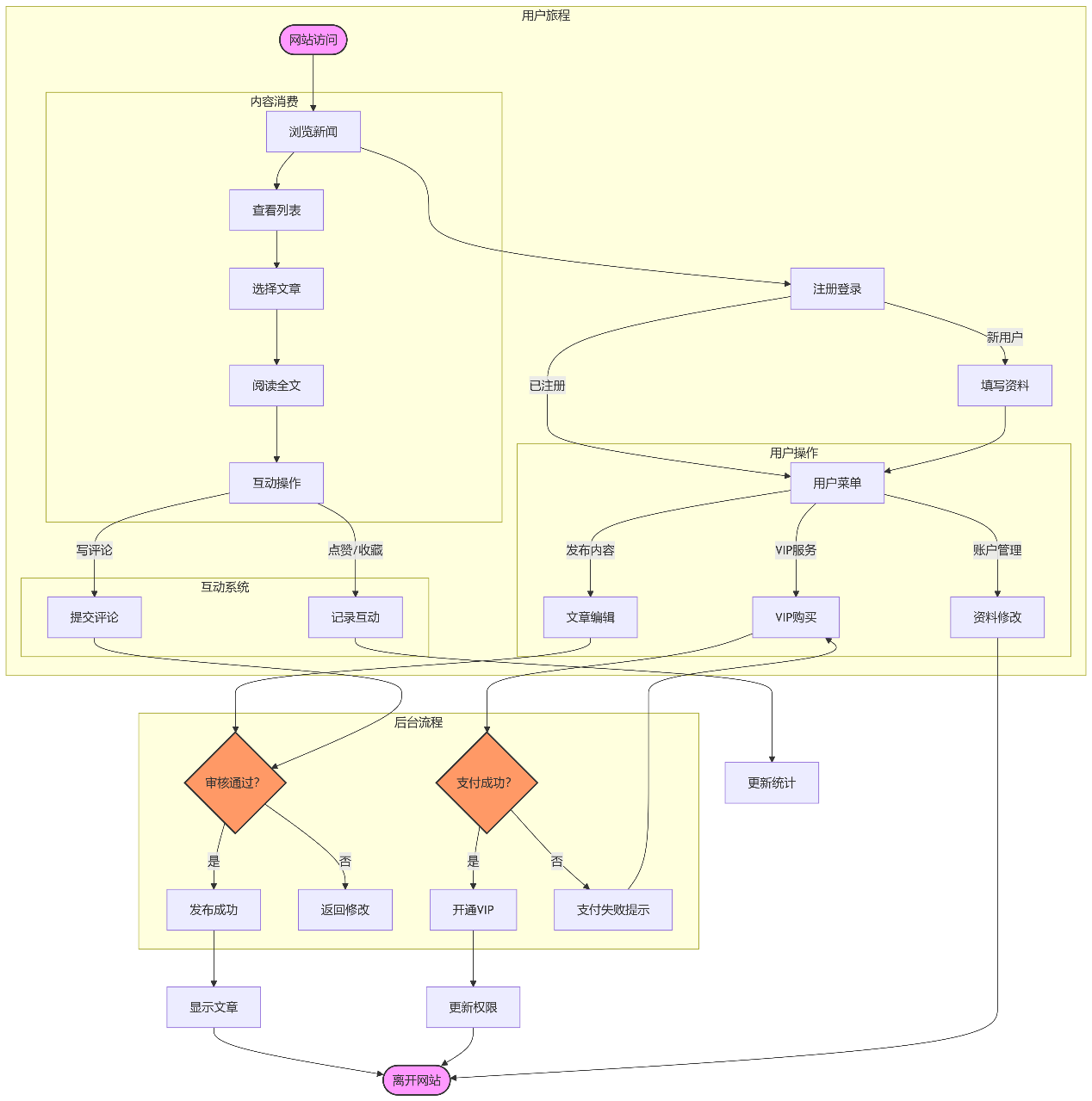
### 4.1.2类图

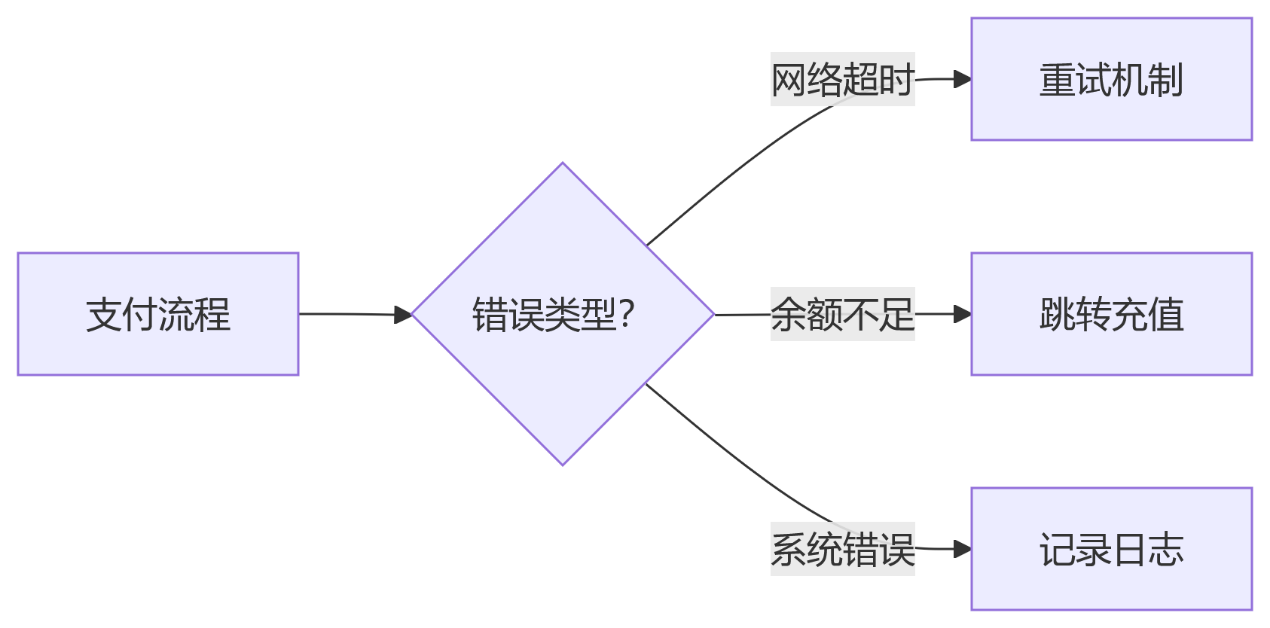


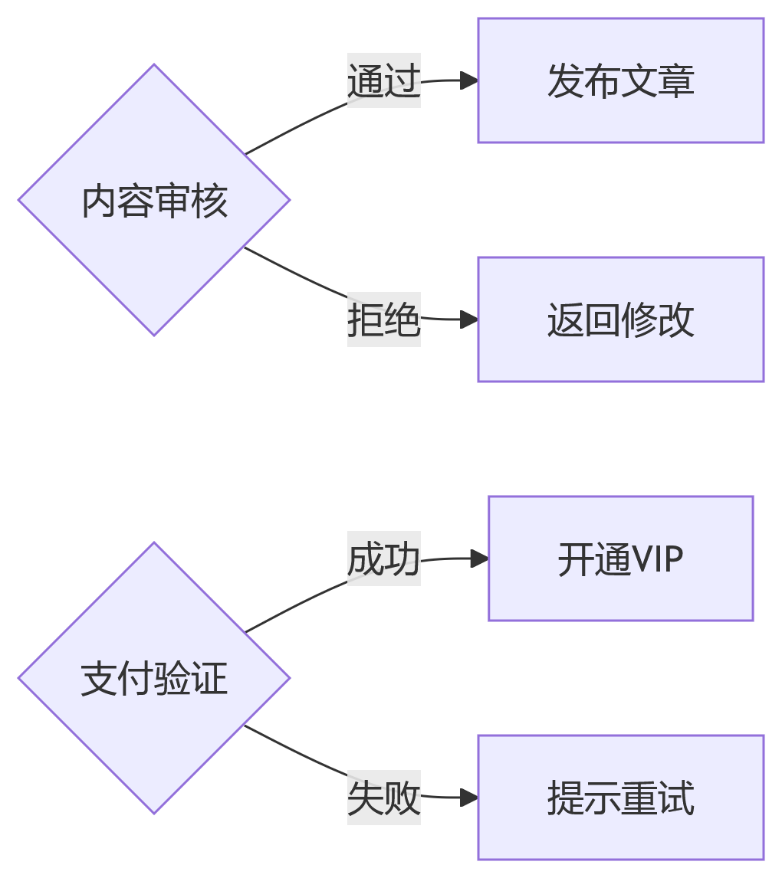
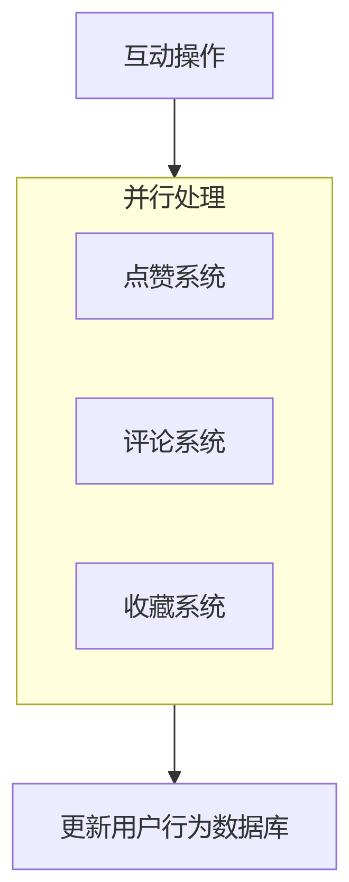
### 4.1.3时序图



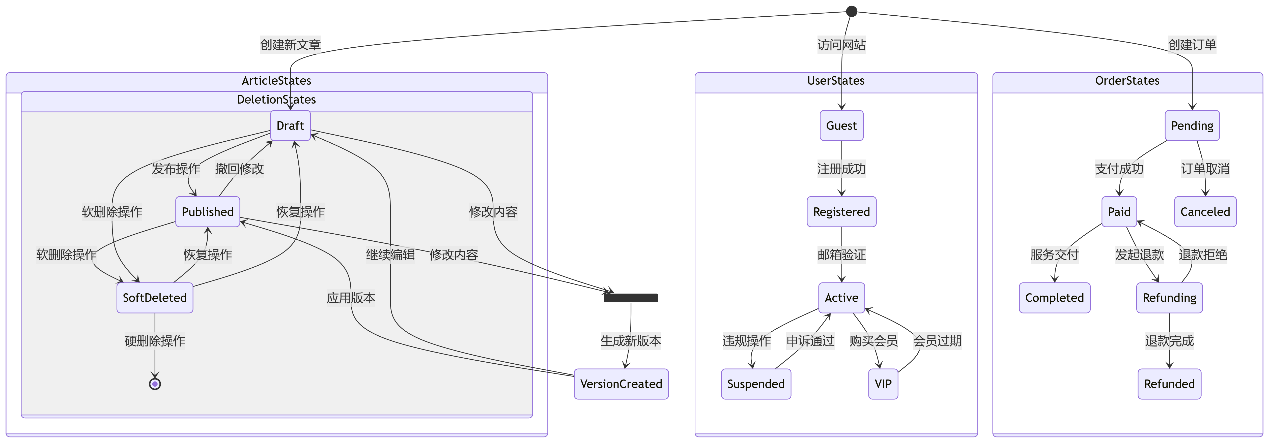
### 4.1.3活动图

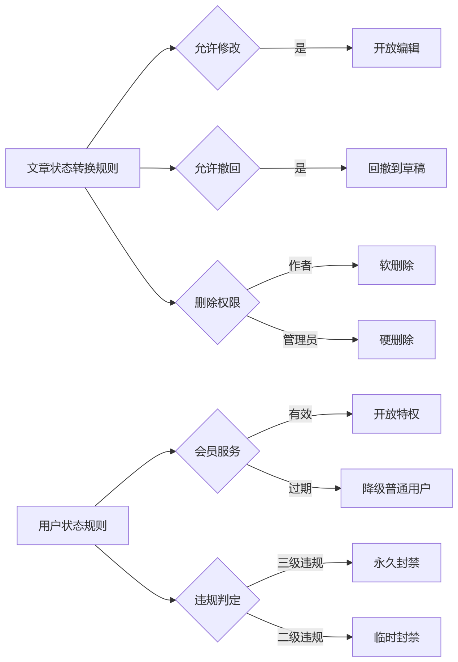






### 4.1.1时序图





## 4.2详细设计

### 4.2.1 用户相关类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User | UserRole | UserVipRelation | | Vip |
| userId:Integer | roleId:Integer | id:Integer | id:Integer | |
| userName:String | roleName:String | uid:User | type:String | |
| password:String | roleDescription:String | vipId:Vip | durations:String | |
| email:String |  |  | durationsValue:Integer | |
| registerDate:Date |  |  | price:BigDecimal | |
| avatar:String |  |  |  | |
| isAuthor:Boolean |  |  |  | |
| phone:String |  |  |  | |
| birthday:Date |  |  |  | |
| 关联： | 关联： | 关联： | 关联： | |
| roles(UserRole,多对多) | users(User,多对多) | user(User,1:1) | relations(UserVipRelation,1:多) | |
| articles(Article,1:多) |  | vip(Vip,1:1) |  | |
| coins(ArticleCoin,1:多) |  |  |  | |
| favorites(ArticleFavorite,1:多) |  |  |  | |
| likes(ArticleLike,1:多) |  |  |  | |
| comments(Comment,1:多) |  |  |  | |
| orders(Order,1:多) |  |  |  | |
| chatRecords(ChatRecord,1:多) |  |  |  | |
| operationLogs(DataOperationLog,1:多) |  |  |  | |

### 4.2.2 文章相关类

|  |  |  |  |
| --- | --- | --- | --- |
| Article | Category | Tag |  |
| articleId:Integer | categoryId:Integer | tagId:Integer |  |
| title:String | name:String | label:String |  |
| content:String | description:String | path:String |  |
| publishDate:Date | **关联**： | **关联**： |  |
| updateDate:Date | articles(Article,多对多) | articles(Article,多对多) |  |
| author:User |  |  |  |
| status:String("DRAFT","PUBLISHED") |  |  |  |
| isDeleted:Boolean |  |  |  |
| lastOperator:User |  |  |  |
| 关联： |  |  |  |
| categories(Category,多对多) |  |  |  |
| tags(Tag,多对多) |  |  |  |
| coins(ArticleCoin,1:多) |  |  |  |
| favorites(ArticleFavorite,1:多) |  |  |  |
| likes(ArticleLike,1:多) |  |  |  |
| versions(ArticleVersion,1:多) |  |  |  |
| comments(Comment,1:多) |  |  |  |
| media(Media,1:多) |  |  |  |

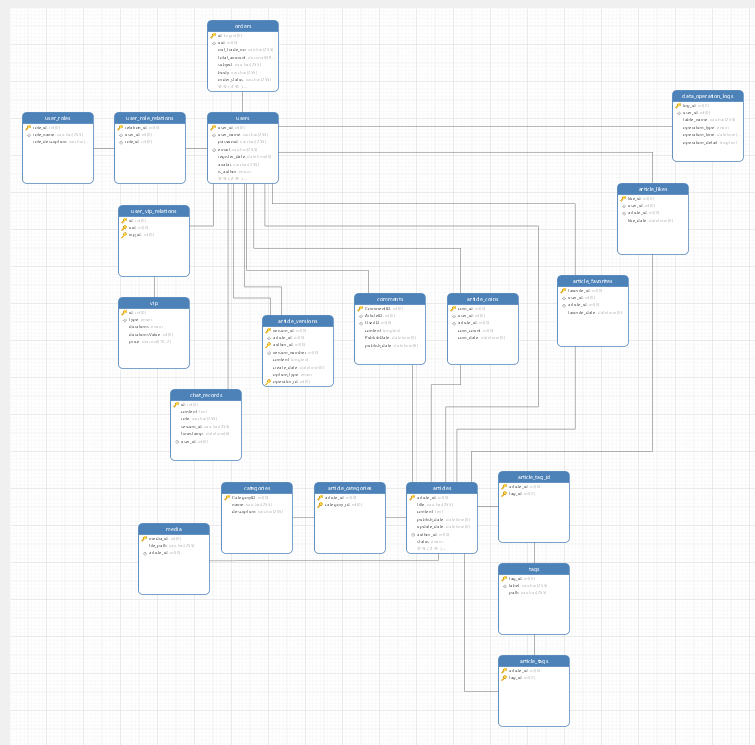
### 4.2.3 交互相关类

|  |  |  |  |
| --- | --- | --- | --- |
| ArticleCoin | ArticleFavorite | ArticleLike | ArticleVersion |
| coinId:Integer | favoriteId:Integer | likeId:Integer | versionId:Integer |
| user:User | user:User | user:User | article:Article |
| article:Article | article:Article | article:Article | author:User |
| coinCount:Integer | favoriteDate:Date | likeDate:Date | versionNumber:Integer |
| coinDate:Date | **关联**： | **关联**： | content:String |
| 关联： | user(User,1:1) | user(User,1:1) | createDate:Date |
| user(User,1:1) | article(Article,1:1) | article(Article,1:1) | optionType:String |
| article(Article,1:1) |  |  | operator:User |

### 4.2.4 其他类

|  |  |  |  |
| --- | --- | --- | --- |
| Comment | Media | Order | ChatRecord |
| commentId:Integer | mediaId:Integer | id:Long | id:Integer |
| article:Article | filePath:String | uid:User | content:String |
| user:User | article:Article | outTradeNo:String | role:String |
| content:String | **关联**： | totalAmount:BigDecimal | sessionId:String |
| publishDate:Date | article(Article,1:1) | subject:String | timestamp:Date |
| 关联： |  | body:String | user:User |
| article(Article,1:1) |  | orderStatus:String |  |
| user(User,1:1) |  | paymentTime:Date |  |

### 4.2.5 E-R图



# 五、系统实现及测试

## 5.1系统功能实现

### 5.1.1 前端核心功能实现

用户登录与权限控制

关键代码（Vue3 TS + Pinia）：

--typescript--------------------------------------------------------------------------------------------------------

// src/store/userStore.ts

import { ElMessage } from 'element-plus';

import { defineStore } from 'pinia';

import { ref, computed } from 'vue';

import userCache, { type UserInfo } from '@/cache/userCache';

export const useUserStore = defineStore('user', () => {

  const userInfo = ref<UserInfo | null>(userCache.getUserCache());

  const isLoggedIn = computed(() => !!userInfo.value);

  const login = (user: UserInfo) => {

    userInfo.value = user;

    userCache.saveUserCache(user);

    ElMessage.success('登录成功！');

  };

  const logout = (router: any) => { // 接收 router 参数

    userInfo.value = null;

    userCache.clearUserCache();

    ElMessage.success('退出登录成功！');

    router.push('/home');

  };

  return {

    userInfo,

    isLoggedIn,

    login,

    logout

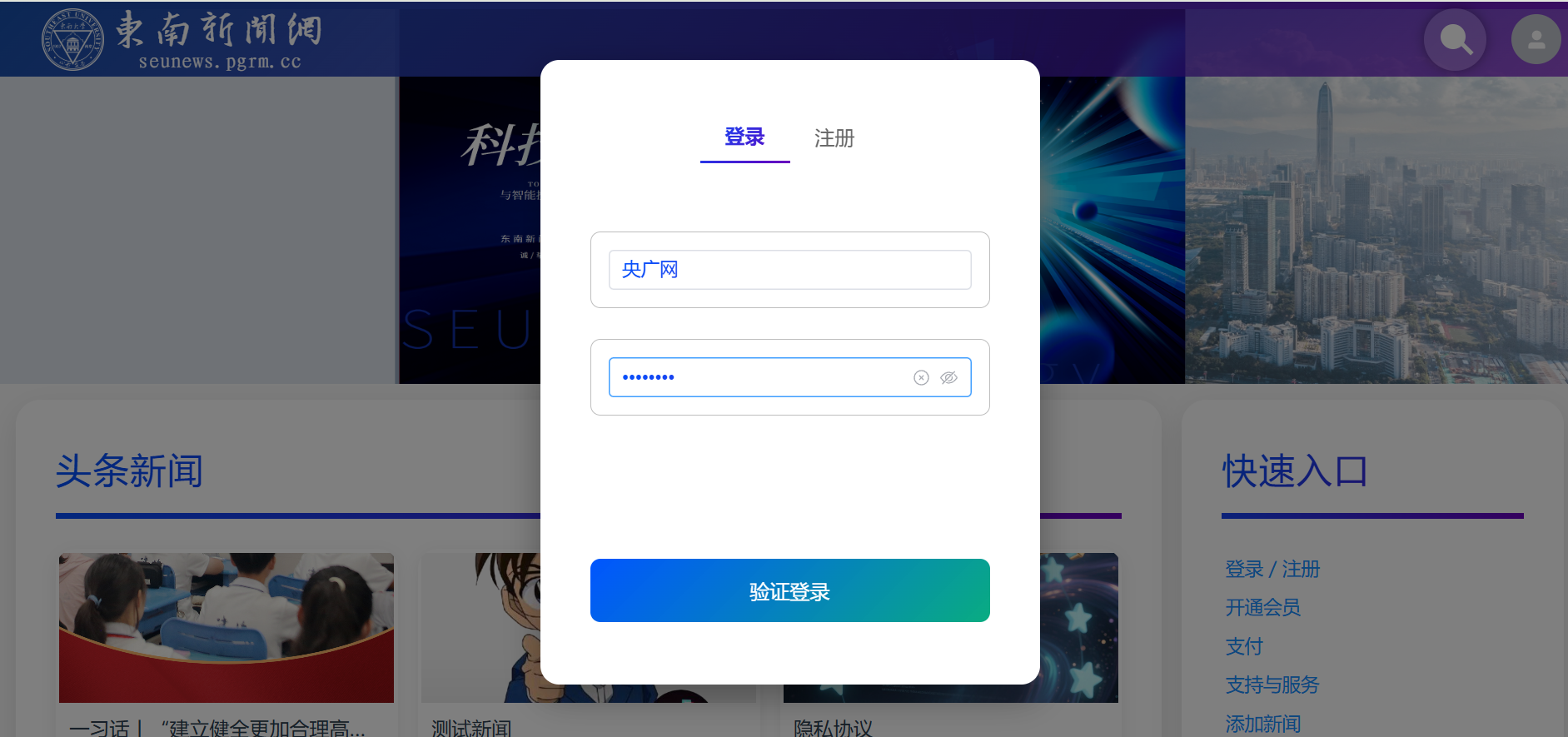
  };

});

----------------------------------------------------------------------------------------------------------------------

实际效果：

用户输入账号密码后，前端通过 Axios 调用登录接口，成功则跳转到首页；未登录用户访问受限页面时自动重定向至登录页。





新闻列表与虚拟滚动优化

<template>

<div class="news-list" @scroll="handleScroll">

<div :style="{ height: `${totalHeight}px` }">

<div v-for="item in visibleItems" :key="item.id" :style="getItemStyle(item)">

<NewsCard :data="item" />

</div>

</div>

</div>

</template>

<script setup lang=”ts”>

import { useVirtualList } from "@vueuse/core";

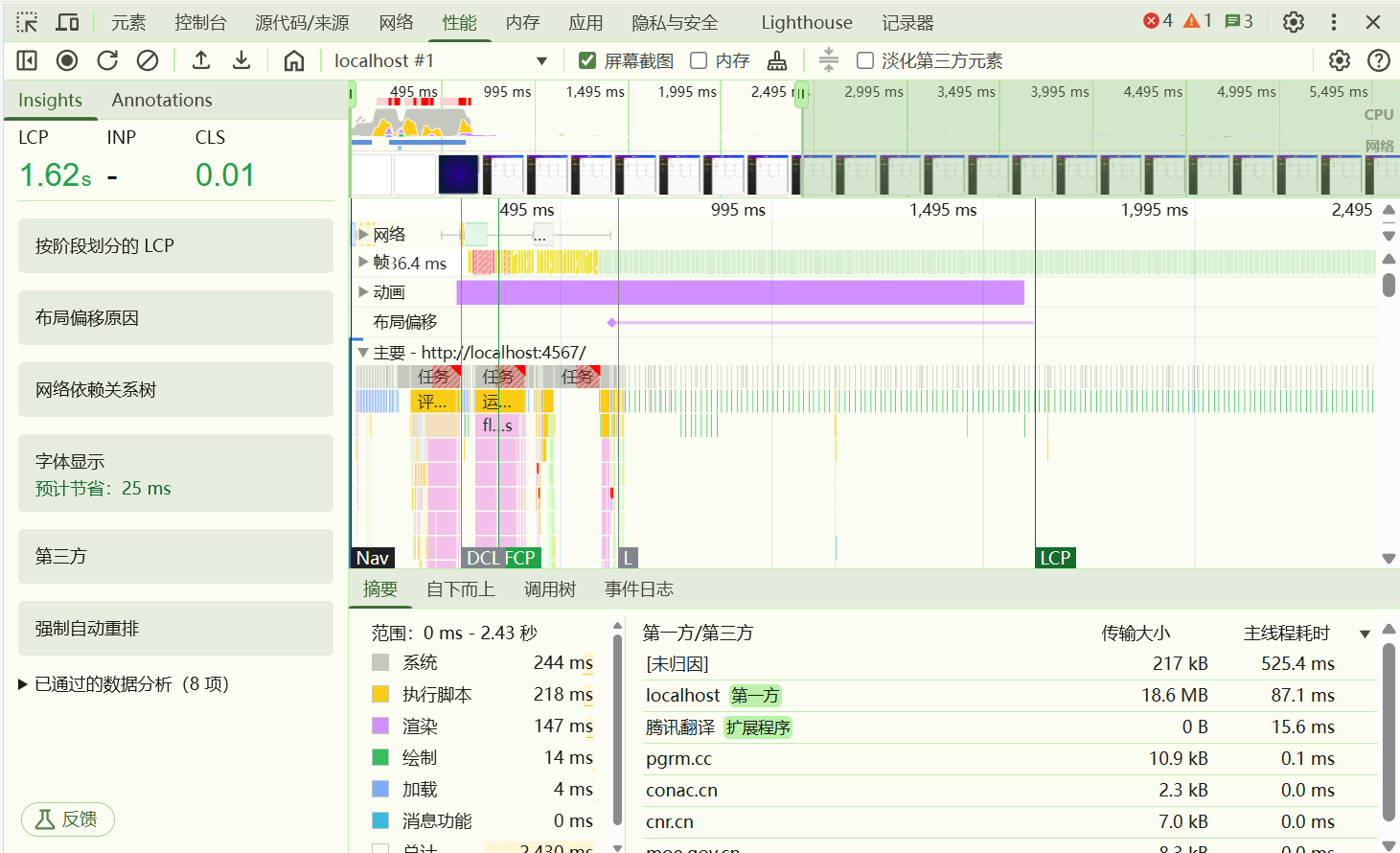
const { list: visibleItems, containerProps, wrapperProps } = useVirtualList(

newsList,

{ itemHeight: 120 }

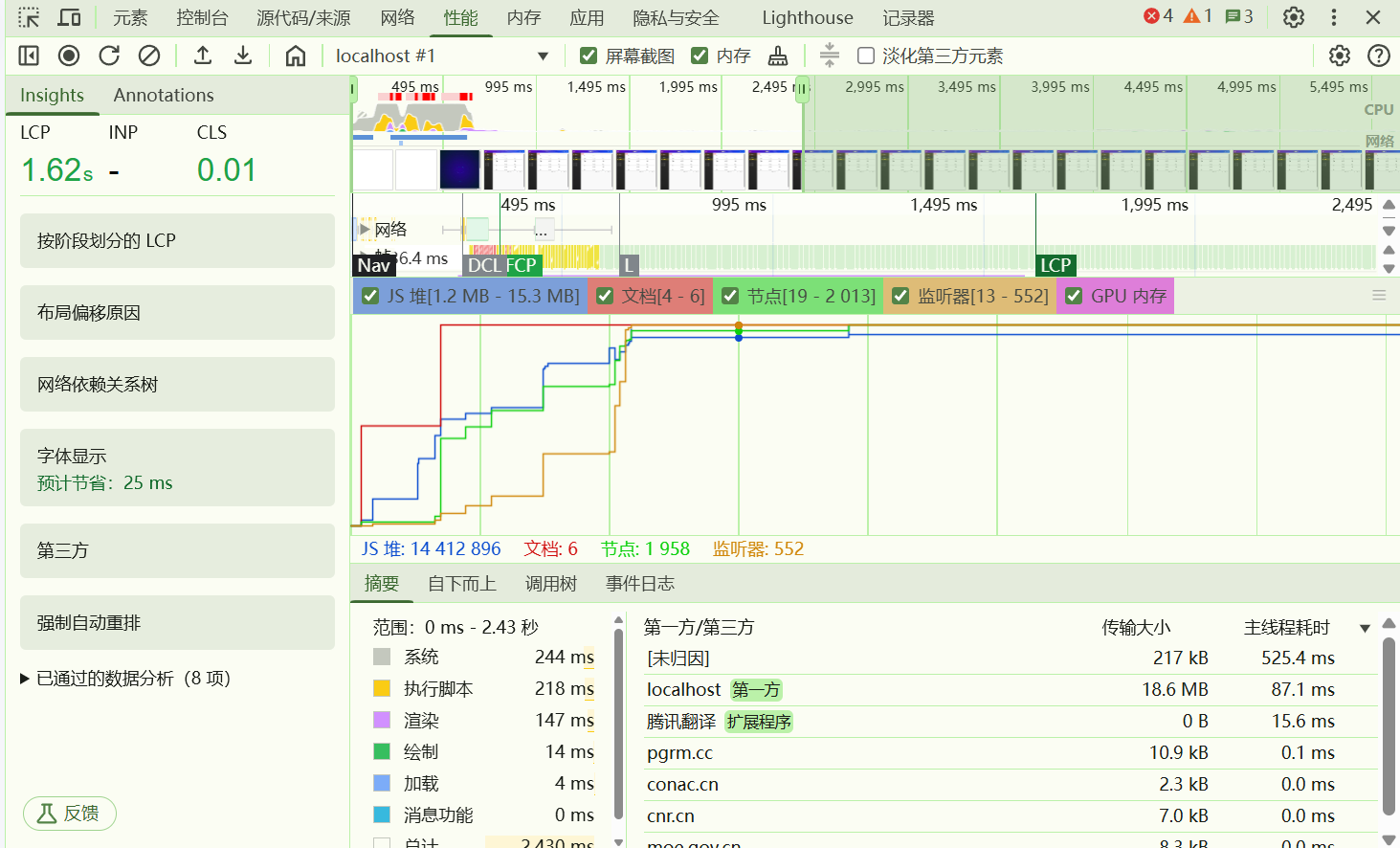
);

</script>



实现思路：

使用 @vueuse/core 的虚拟滚动库，仅渲染可视区域内的新闻卡片，减少 DOM 节点数量。



（需插入新闻列表界面截图及 Chrome Performance 性能对比图）

5.1.2 后端核心功能实现

个性化推荐接口（Spring Boot + Redis）

关键代码（Java）：

@RestController

@RequestMapping("/api/recommend")

public class RecommendController {

@Autowired private RedisTemplate<String, String> redisTemplate;

@GetMapping

public ResponseEntity<List<News>> getRecommendations(@RequestParam String userId) {

String key = "user:recommend:" + userId;

List<News> cached = redisTemplate.opsForList().range(key, 0, -1);

if (cached != null && !cached.isEmpty()) return ResponseEntity.ok(cached);

// 协同过滤算法计算推荐结果

List<News> result = recommendService.calculate(userId);

redisTemplate.opsForList().rightPushAll(key, result);

redisTemplate.expire(key, 10, TimeUnit.MINUTES);

return ResponseEntity.ok(result);

}

}

实现思路：

用户首次请求推荐数据时，通过协同过滤算法生成结果并缓存至 Redis，后续请求直接读取缓存，降低数据库压力。

支付功能集成（支付宝 SDK）

关键代码（Java）：

@PostMapping("/pay")

public ResponseEntity<String> createOrder(@RequestBody OrderRequest request) {

AlipayTradePagePayResponse response = alipayClient.pageExecute(

new AlipayTradePagePayRequest()

.setBizContent(JSON.toJSONString(request))

);

return ResponseEntity.ok(response.getBody());

}

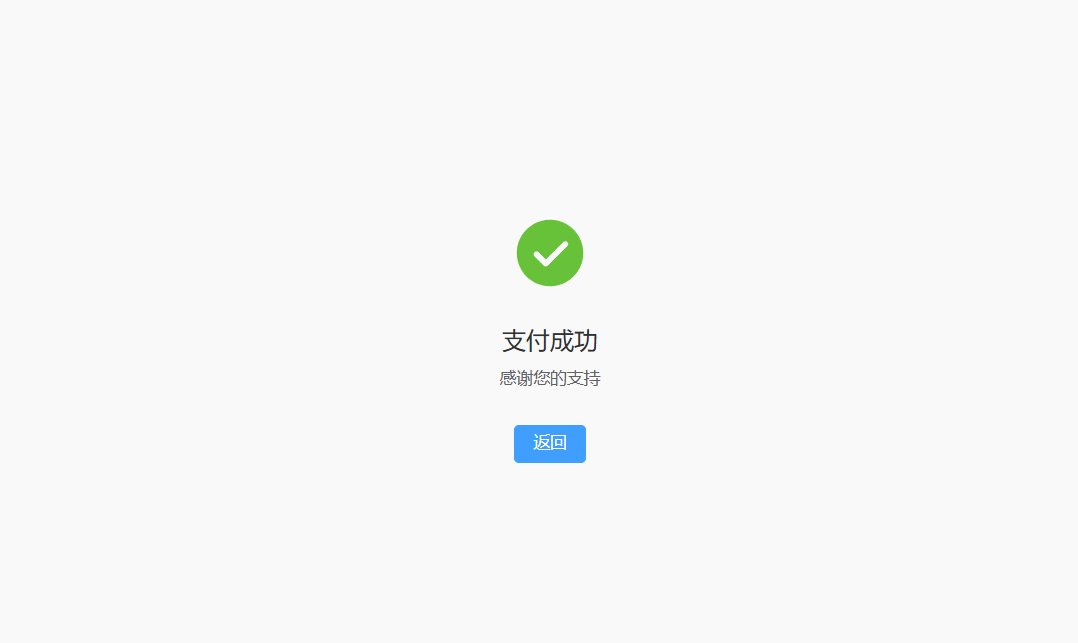
实际效果：

用户提交订单后生成支付宝支付链接，跳转至支付宝完成支付，后端通过异步通知验证支付结果。









5.2 系统测试

5.2.1 单元测试（JUnit + Jest）

后端测试（Spring Boot）：

@Test

void testRecommendService() {

List<News> result = recommendService.calculate("user123");

assertNotNull(result);

assertEquals(10, result.size()); // 默认返回10条推荐

}

前端测试示例（Jest）：

test("Login should store token", async () => {

await authStore.login("test", "123456");

expect(localStorage.getItem("token")).not.toBeNull();

});