大连东软信息学院

**毕业设计（论文）**

**论文题目**：**IT技术视频分享社区的设计与实现**

|  |  |
| --- | --- |
| **学 院：** | 大连东软信息学院 |
| **专 业：** | 软件工程 |
| **学生姓名：** | 范博 |
| **学生学号：** | 15180600501 |
| **指导教师：** | 彭志豪 |
| **导师职称：** | 副教授 |
| **完成日期：** |  |

大连东软信息学院

**Dalian Neusoft University of Information**

IT技术视频分享社区的设计与实现

# 摘 要

当今时代，社区已经成为互联网上的一个重要平台，它主要的功能是为用户提供一个互相交流、互相学习、互相帮助的平台。它使人与人之间的沟通、交流变得更加容易，尤其是在IT领域，我们只有能够很好的交流，技术才会一步一步的提升。因此，掌握如何设计并开发一个技术社区有着重要而且深远的意义。然而，大多数社区以文字和图片为主，较为单调，容易造成读者的视觉疲劳。

针对目前的问题，本课题在继承了传统社区优点下，增加了视频元素，让内容更加充实，完成了一套基于微服务架构的IT技术视频分享交流社区。本系统后端部分使用的集成开发环境是IDEA，使用MySQL作为数据库，采用Spring Cloud微服务架构搭配Spring Boot技术进行实现，使用Nacos作为注册中心，前端部分集成开发环境是WebStorm，使用Vue.js + Element-UI实现，使用Axios调用后端接口。整个系统包括社区系统和管理员系统两部分组成，社区系统包括：首页推荐模块、新闻模块、博客模块、论坛模块、用户中心模块、个人中心模块。管理员系统包括：新闻维护模块、帖子管理模块、博客管理模块、用户管理模块。除了完成了IT社区应该有的功能，还可以让当用户在编辑博客、编辑帖子的时候可以上传视频，并且用户可以自己决定视频插入的位置。

本系统实现的就是一个相互交流技术、获取和分享知识和技术、了解最新技术新闻的网站，提供的一个IT技术交流平台，为技术人员解决技术难题、相互交流、分享学习成果、获取IT界最新消息提供便利。并且为了更好的提高用户观感体验，在传统内容的基础上加入视频元素。

**关键词**：技术社区，Vue.js，SpringCloud，Mysql，Nacos

**Design and Implementation of IT Technology Video Sharing Community**

# Abstract

In today's times community has become an important platform on the Internet. Its main function is to provide users with a platform to communicate with each other, learn from each other and help each other. In the field of IT, we can only communicate well, technology will be step by step to improve. Therefore, mastering how to design and develop a technology community has important and far-reaching implications. However, most communities are mainly text and pictures. Too monotonous and easy to cause visual fatigue of readers.

This topic has inherited the advantages of the traditional community, increased the video elements. So that the content is more substantial. Completed the IT technology video sharing community based on microservice architecture. In the back-end, the system development environment section is IntelliJ IDEA. Using MySQL as the database. Using Spring Cloud with Spring Boot, using Nacos as the registration center. The front-end part of the integrated development environment is WebStorm, using Vue.js + Element-UI, using Axios to invoke the back-end interface. The system consists of two parts: community system and administrator system. Community system includes, home page recommendation module, news module, blog module, forum module, user center module and personal center module. Administrator system includes, news maintenance module, post management module, blog management module and user Management module. Users can upload videos when they edit a blog, edit a post, and users can decide where to insert the video themselves.

The system realizes a website that communicates technology, obtains and shares knowledge and technology, understands the latest technology news, provides IT technology exchange platform for technicians to solve technical problems, communicate with each other, share learning results, and get the latest news from the IT industry. Add video elements based on traditional content. Better improve the user's experience.

**Key words:** Technology community, Vue.js, Spring Cloud, MySQL, Nacos

目 录

[摘 要 I](#_Toc7625193)

[Abstract II](#_Toc7625194)

[第1章　项目概述 1](#_Toc7625195)

[1.1 问题描述 1](#_Toc7625196)

[1.2 项目目标 1](#_Toc7625197)

[1.3 项目适用范围 1](#_Toc7625198)

[1.4 涉众 1](#_Toc7625199)

[第2章　需求分析 2](#_Toc7625200)

[2.1 业务对象 2](#_Toc7625201)

[2.2 功能性需求 2](#_Toc7625202)

[2.2.1 用例概述 2](#_Toc7625203)

[2.2.2 用例描述 4](#_Toc7625204)

[2.3 非功能性需求 8](#_Toc7625205)

[2.4 运行环境 8](#_Toc7625206)

[第3章　系统设计 10](#_Toc7625207)

[3.1 技术路线 10](#_Toc7625208)

[3.2 软件体系结构设计 11](#_Toc7625209)

[3.2.1 功能结构 11](#_Toc7625210)

[3.2.2 系统架构 12](#_Toc7625211)

[3.3 数据库设计 13](#_Toc7625212)

[3.3.1 概念结构设计 13](#_Toc7625213)

[3.3.2 表的详细设计 14](#_Toc7625214)

[3.4 用例实现 18](#_Toc7625215)

[3.4.1 撰写博客用例实现 18](#_Toc7625216)

[3.4.2 修改个人信息用例实现 18](#_Toc7625217)

[3.4.3 修改用户博客用例实现 18](#_Toc7625218)

[3.4.4 删除用户博客用例实现 19](#_Toc7625219)

[3.4.5 发表评论用例实现（以博客为例） 19](#_Toc7625220)

[3.4.6 收藏用例实现（以博客为例） 20](#_Toc7625221)

[3.4.7 添加新闻用例实现 20](#_Toc7625222)

[3.4.8 修改新闻用例实现 21](#_Toc7625223)

[3.5 对象设计 21](#_Toc7625224)

[3.5.1 模块设计 21](#_Toc7625225)

[3.5.2 业务实体类的设计 22](#_Toc7625226)

[3.5.3 业务处理类的设计 22](#_Toc7625227)

[3.5.4 类的描述 24](#_Toc7625228)

[第4章　系统实现 28](#_Toc7625229)

[4.1 核心功能实现 28](#_Toc7625230)

[4.1.1 新闻/博客/帖子编辑 28](#_Toc7625231)

[4.1.2 新闻模块 29](#_Toc7625232)

[4.1.3 博客模块 31](#_Toc7625233)

[4.1.4 个人中心模块 32](#_Toc7625234)

[4.2 关键技术难点与解决方案 36](#_Toc7625235)

[第5章　系统测试 37](#_Toc7625236)

[5.1 功能测试 37](#_Toc7625237)

[5.2 非功能测试 38](#_Toc7625238)

[5.3 测试总结 38](#_Toc7625239)

[第6章　结论与展望 39](#_Toc7625240)

[参考文献 40](#_Toc7625241)

[致 谢 41](#_Toc7625242)

# 第1章　项目概述

## 1.1 问题描述

随着IT行业的飞速发展，技术不断更新，需要一个交流与分享的平台来让IT行业的人获取和分享技术我要做的就是一个IT行业的技术社区。而普通的社区的博文和帖子都以文字和图片等静态画面为主，容易造成用户的视觉疲劳，在博文和帖子中加入视频元素，更好的理解内容，还可以提高读者的阅读兴趣和感受。

## 1.2 项目目标

IT视频分享交流社区的开发目标是使用当前较为流行的技术，开发一套拥有较高容错性、扩展性、易维护的系统。功能上，实现新闻、博客、论坛于一体的IT技术社区，并且在传统社区功能的基础上，加入视频元素，提高博文和帖子的可读性。

本系统主要面向IT界的人群，对比传统社区的功能，完成一个相互交流技术、获取和分享知识和技术、了解最新技术新闻的网站，提供的一个IT技术交流平台，为技术人员解决技术难题、相互交流、分享学习成果、获取IT界最新消息提供便利。

## 1.3 项目适用范围

本项目适用于喜欢交流IT技术的人群，包括学生、IT工程师等，喜欢分享自己的知识，尤其是喜欢使用视频来展示自己所表达的内容的人们。

## 1.4 涉众

系统设计人员：根据用户的习惯设计系统的交互，在保证基本用户。

系统开发人员：严格按照需求及设计文档进行功能模块开发，避免功能缺陷。

后台维护人员：编辑、上传优质新闻，维护博文、帖子、评论、用户等信息，清理劣质博文、帖子、评论，清理行为恶劣的用户。

普通用户：浏览所有新闻、博文、帖子、评论和其他用户的信息，编写优质的博客、帖子、评论，收藏喜欢的新闻、博文、帖子。

# 第2章　需求分析

本章主要对项目需求进行分析，主要从业务对象、功能性需求、非功能性需求、运行环境四个方向进行具体分析。

## 2.1 业务对象

未注册用户仅能浏览，注册用户可以发表帖子和撰写博客，一位用户可以撰写多个博客，可以发表多个帖子，一位用户可以收藏多个新闻、博客、帖子，一位用户可以在一篇新闻、博客、帖子下评论多条。一个分类下有多条新闻和博文，一条博文或新闻只属于一个分类。一张轮播图关联一条新闻，领域模型如题2.1所示。

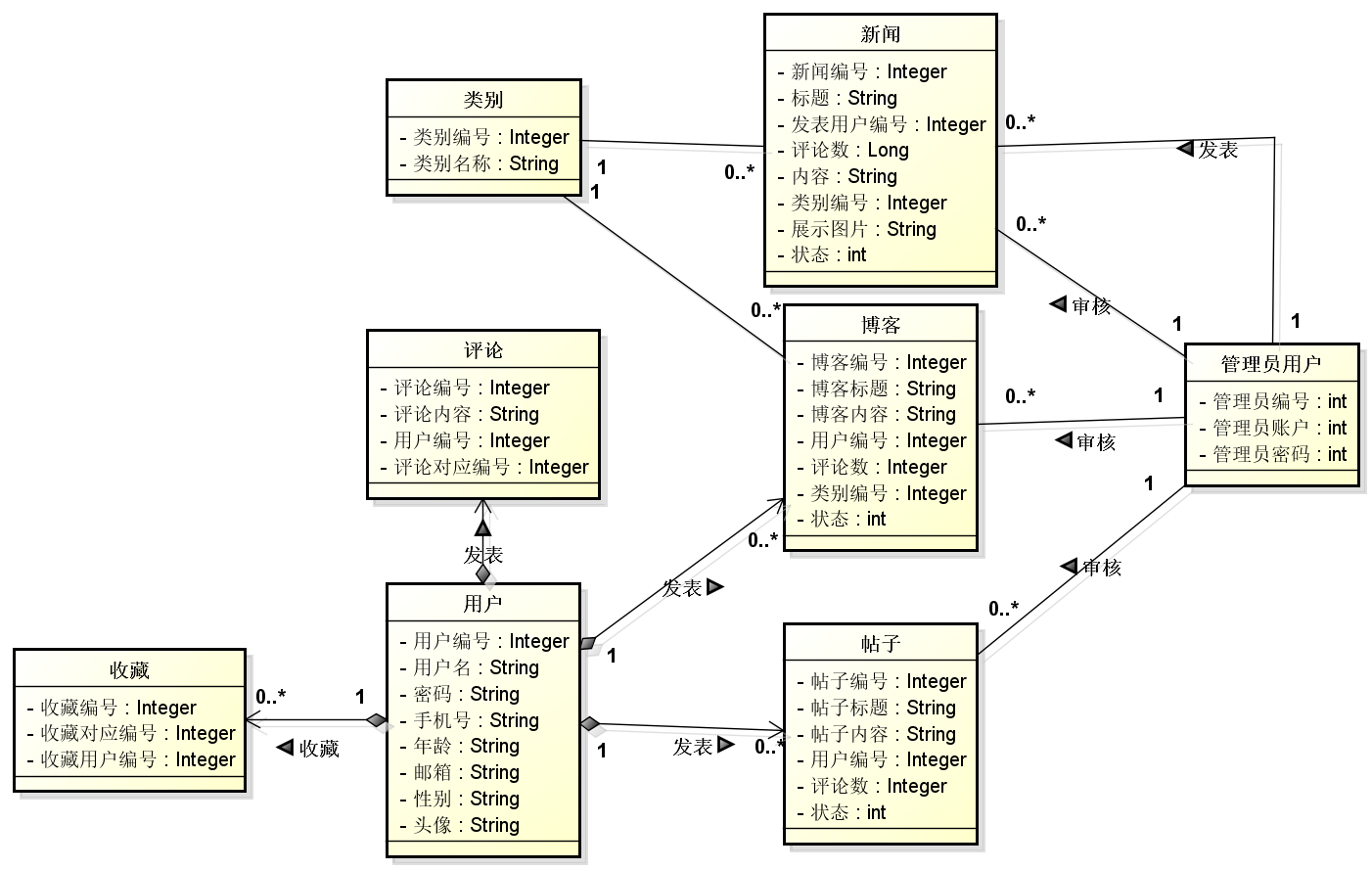


图2.1领域模型

## 2.2 功能性需求

### 2.2.1 用例概述

在图2.2高层用例图中抽取了满足用户基本业务需求的基本用例。这些用例可以从社区中心、后台管理两个方面完成整个IT技术视频分享社区的运行。

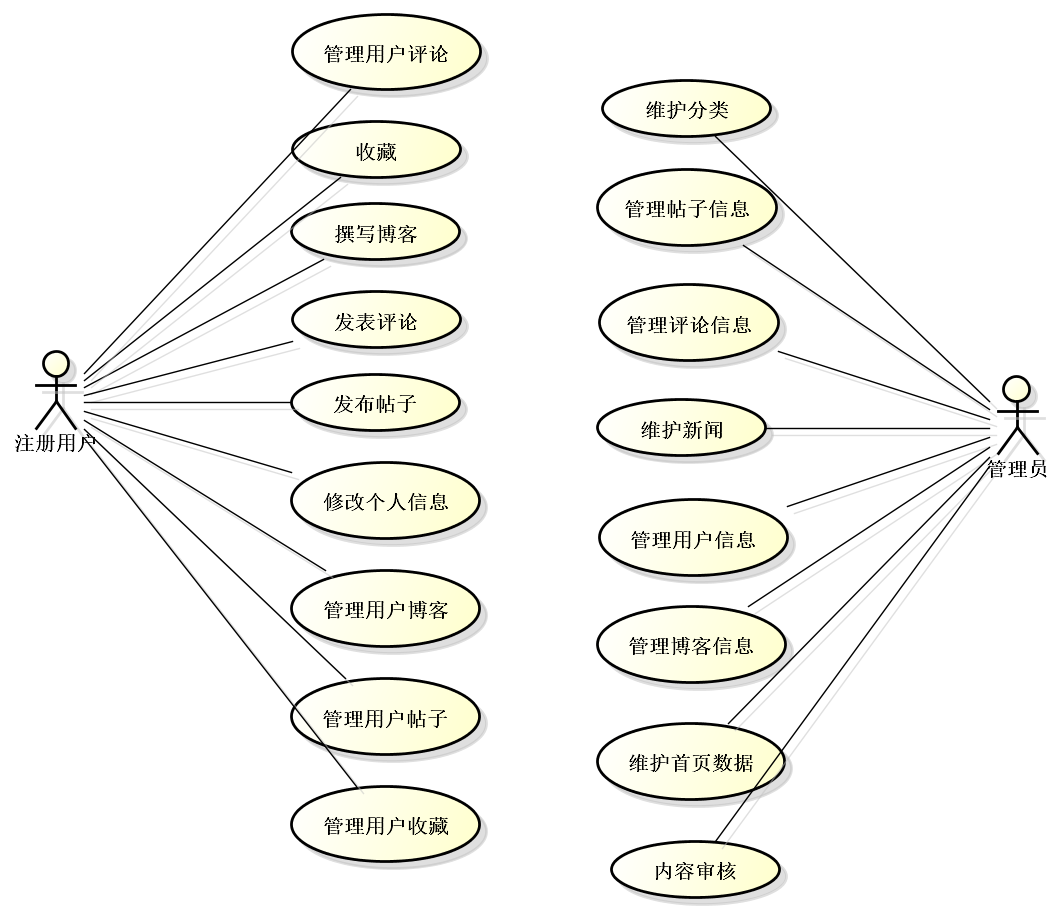


图2.2 高层用例图

用例摘要描述如表2.1所示。

表2.1 用例摘要描述

|  |  |  |
| --- | --- | --- |
| **用例标识（UC）** | **用例名称** | **摘要描述** |
| 1 | 撰写博客 | 用户可以在博客模块编写博文并发表 |
| 2 | 发表帖子 | 用户可以在论坛模块发表帖子 |
| 3 | 修改个人信息 | 用户可以对自己的个人信息进行修改。 |
| 4 | 管理用户博客 | 用户可以对自己的博客进行修改、删除。 |
| 5 | 管理用户收藏 | 用户可以对自己的收藏进行修改、删除。 |
| 6 | 管理用户评论 | 用户可以对自己的评论进行修改、删除。 |
| 7 | 管理用户帖子 | 用户可以对自己的帖子进行修改、删除。 |
| 8 | 发表评论 | 用户可以在新闻、帖子、博客下进行评论 |
| 9 | 收藏 | 用户可以收藏喜欢的新闻、帖子、博客。 |
| 10 | 查看其他用户 | 用户可以点击查看其他用户的信息和发表、评论、收藏记录。 |
| 11 | 维护新闻 | 管理员可以对新闻进行增删改查操作。 |
| 12 | 管理用户信息 | 管理员可以对用户信息进行查询、修改、删除操作。 |
| 13 | 管理帖子信息 | 管理员可以对帖子进行查询、删除操作。 |
| 14 | 管理博客信息 | 管理员可以对博客进行查询、删除操作。 |
| 15 | 管理评论信息 | 管理员可以对评论进行查询、删除操作。 |
| 16 | 维护首页数据 | 管理员可以对主页的轮播图进行查询、修改操作。 |
| 17 | 维护分类 | 管理员可以对分类进行增删改查操作 |
| 18 | 内容审核 | 管理员可以对帖子、博客、新闻内容进行审核。 |

### 2.2.2 用例描述

（1）撰写博客用例

注册用户可以通过登录用例进行系统登录，登录用例描述如表2.2所示。

表2.2 撰写博客用例

|  |  |
| --- | --- |
| **用例标识** | 4 |
| **用例名称** | 撰写博客 |
| **参与者** | 注册用户 |
| **前置条件** | 用户已经登录系统并进入博客模块。 |
| **后置条件** | 用户撰写一篇博客。 |
| **用例概述** | 用户登录后可以撰写博客。 |
| **基本事件流** | 1. 用户点击撰写博客按钮。  2. 系统显示博客编辑页面。  3. 用户输入博客标题、博客类别、博客内容。  4. 系统检测录入信息的正确性。  5. 系统显示成功。 |
| **备选事件流** | 4a 系统检测用户录入信息有误，显示错误信息，转入步骤2 |
| **备注** | 无 |

（2）修改个人信息用例

注册用户可以通过修改个人信息用例修改个人信息，用例描述如表2.3所示。

表2.3 修改个人信息用例

|  |  |
| --- | --- |
| **用例标识** | 6 |
| **用例名称** | 修改个人信息 |
| **参与者** | 注册用户 |
| **前置条件** | 用户已登录并且进入个人中心。 |
| **后置条件** | 用户修改自己的信息。 |
| **用例概述** | 用户可以修改自己的个人信息。 |
| **基本事件流** | 1. 用户点击修改信息按钮。  2. 系统显示编辑界面。  3. 用户修改信息，并提交。  4. 系统检测录入信息的正确性。  5. 系统提示修改成功并回到个人中心 |
| **备选事件流** | 4a 系统检测用户录入信息有误，显示错误信息，转入步骤2 |
| **备注** | 无 |

（3）管理用户博客用例

注册用户可以通过管理用户博客用例对博客进行修改和删除（管理用户收藏、帖子、评论用例与其类似），用户博客用例图如图2.3所示



图2.3 管理用户博客用例图

用例描述如表2.4、2.5所示。

表2.4 修改用户博客用例

|  |  |
| --- | --- |
| **用例标识** | 4.1 |
| **用例名称** | 修改用户博客 |
| **参与者** | 注册用户 |
| **前置条件** | 用户已登录并且进入个人中心。 |
| **后置条件** | 用户修改自己的博客。 |
| **用例概述** | 用户可以修改自己的博客。 |
| **基本事件流** | 1. 用户点击我的博客。  2. 系统显示我的博客页面。  3. 用户点击要修改的博客。  4. 系统显示博客详情。  5. 用户点击修改按钮  6. 系统显示博客编辑页面。  7. 用户对博客进行修改，并提交。  8. 系统检测录入信息的正确性。  9. 系统提示修改成功 |
| **备选事件流** | 8a 系统检测用户录入信息有误，显示错误信息，转入步骤6 |
| **备注** | 无 |

表2.5 删除用户博客用例

|  |  |
| --- | --- |
| **用例标识** | 4.2 |
| **用例名称** | 删除用户博客 |
| **参与者** | 注册用户 |
| **前置条件** | 用户已登录并且进入个人中心。 |
| **后置条件** | 用户删除自己的博客。 |
| **用例概述** | 用户可以删除自己的博客。 |
| **基本事件流** | 1. 用户点击我的博客。  2. 系统显示我的博客页面。  3. 用户点击要删除博客后的删除按钮。  4. 系统显示删除提示框。  5. 用户点击确认。  6. 系统提示删除成功。 |
| **备选事件流** | 5a 用户点击取消，转入2 |
| **备注** | 无 |

（4）发表评论用例

注册用户可以在博客、帖子、新闻下发表评论，用例描述如表2.6所示。

表2.6 发表评论用例

|  |  |
| --- | --- |
| **用例标识** | 8 |
| **用例名称** | 发表评论 |
| **参与者** | 注册用户 |
| **前置条件** | 用户已登录并进入博客（新闻、帖子）详情 |
| **后置条件** | 用户可以对博客、新闻、帖子进行评伦。 |
| **用例概述** | 用户评论博客、新闻、帖子 |
| **基本事件流** | 1. 用户在评论输入框中输入评论内容。  2. 用户点击评论。  3. 系统检测录入信息的正确性。  4. 系统提示评论成功。 |
| **备选事件流** | 3a 系统检测用户录入信息有误，显示错误信息，转入步骤1 |
| **备注** | 无 |

（5）收藏用例

注册用户可以通过管理用户博客用例对博客进行修改和删除，用例描述如表2.7所示。

表2.7 收藏用例

|  |  |
| --- | --- |
| **用例标识** | 12 |
| **用例名称** | 收藏 |
| **参与者** | 注册用户 |
| **前置条件** | 用户已登录并且进入博客（帖子、新闻）详情。 |
| **后置条件** | 用户收藏博客（帖子、新闻）。 |
| **用例概述** | 用户可以收藏博客、帖子、新闻 |
| **基本事件流** | 1. 用户点击收藏按钮。  2. 系统检测用户状态的正确性。  3. 收藏成功，收藏按钮变为取消按钮。 |
| **备选事件流** | 无 |
| **备注** | 无 |

（6）维护新闻用例

管理员可以通过维护新闻用例对新闻进行增删改查操作，维护新闻用例图如图2.4所示。



图2.4 管理新闻用例图

维护新闻用例描述如表2.8-2.11所示。

表2.8 添加新闻用例

|  |  |
| --- | --- |
| **用例标识** | 17.1 |
| **用例名称** | 添加新闻 |
| **参与者** | 管理员 |
| **前置条件** | 已经以管理员身份登录并进入新闻模块。 |
| **后置条件** | 在新闻表中添加一条新闻。 |
| **用例概述** | 管理员可以添加新闻。 |
| **基本事件流** | 1. 管理员在新闻模块页点击新闻添加。  2. 系统显示新闻添加页。  3. 管理员输入新闻标题、新闻类别、新闻内容后，提交添加请求。  4. 系统检测录入信息的正确性。  5. 系统显示添加新闻成功。 |
| **备选事件流** | 4a 系统检测用户录入信息有误，显示错误信息，转入步骤2 |
| **备注** | 无 |

表2.9 查询新闻用例

|  |  |
| --- | --- |
| **用例标识** | 17.2 |
| **用例名称** | 查询新闻 |
| **参与者** | 管理员 |
| **前置条件** | 已经以管理员身份登录并进入新闻模块。 |
| **后置条件** | 在新闻表中查询一条新闻。 |
| **用例概述** | 管理员可以查询新闻。 |
| **基本事件流** | 1. 管理员在新闻模块页的搜索框中填入想要查询的新闻。  2. 系统显示查询结果。  3. 管理员点击详情进入新闻详情页。  4. 系统显示新闻详情。 |
| **备选事件流** | 2a管理员没有找到想要查询的新闻，可以转入步骤1重新查询或结束查询  2b管理员可以使用用例“修改新闻信息”对新闻进行修改转入修改信息页, 执行完毕结束  2c管理员可以使用用例“删除新闻信息”对新闻进行删除转入删除确认页，执行完毕结束 |
| **备注** | 无 |

表2.10 修改新闻信息用例

|  |  |
| --- | --- |
| **用例标识** | 17.2.1 |
| **用例名称** | 修改新闻信息 |
| **参与者** | 管理员 |
| **前置条件** | 管理员已经在新闻模块查询出结果。 |
| **后置条件** | 在新闻表中修改一条新闻。 |
| **用例概述** | 管理员可以修改新闻。 |
| **基本事件流** | 1. 管理员在查询结果页点击编辑。  2. 系统显示编辑界面。  3. 用户点击详情。  4. 系统检测录入信息的正确性。  5. 系统显示编辑新闻成功。 |
| **备选事件流** | 无 |
| **备注** | 无 |

表2.11 删除新闻信息用例

|  |  |
| --- | --- |
| **用例标识** | 17.2.2 |
| **用例名称** | 删除新闻信息 |
| **参与者** | 管理员 |
| **前置条件** | 管理员已经在新闻模块查询出结果。 |
| **后置条件** | 在新闻表中删除一条新闻。 |
| **用例概述** | 管理员可以删除新闻。 |
| **基本事件流** | 1. 管理员在查询结果页点击删除。  2. 系统显示删除确认。  3. 用户点击确认。  4. 系统显示删除新闻成功。 |
| **备选事件流** | 3a 用户点击取消，结束流程 |
| **备注** | 无 |

## 2.3 非功能性需求

正确性：页面不允许出现显示非正确信息的情况

高可用性：在系统的某个模块出现故障时，不能影响其他模块正常使用，故障模块应该有保护措施，并有短时间修复的能力。

性能：正常情况下系统响应时间不得超过2秒。

安全性：用户敏感信息需要保密，管理员账户只能由管理员登录

兼容性：可运行在IE9以上的浏览器中。

## 2.4 运行环境

系统需要的硬件环境如下：

（1）应用服务器

处理器 4 vCPU以上

内存 16G以上

存储器容量 40G以上

带宽 1.5 Gbps以上

（2）数据库服务器

处理器 2 vCPU以上

内存 8G以上

存储器容量 100G以上

带宽 1.5 Gbps以上

系统运行所需的软件环境如下：

操作系统：centos7

JDK版本：jdk1.8

数据库：mysql5.7

Node.js版本:v10.15.3

Naocs版本:0.8.0

# 第3章　系统设计

## 3.1 技术路线

本系统采用前后端分离的开发方式，后端架构使用Spring Cloud微服务架构，服务发现使用Nacos，配合Spring Boot快速开发，持久层框架使用基于Mybatis的Mybatis-Plus。前端使用Vue.js + Element-UI实现，使用Axios异步请求后端接口。

微服务是近两年流行起来的最优秀的web系统架构[3]。Spring Cloud对微服务架构提供了全面的技术支持,是实践微服务架构案例最优技术框架[1]。Spring Boot是一个用于简化、加速Spring开发的框架[4]，通过少量的代码和配置就能常见一个独立的、产品级别的Spring应用。Spring Boot Starter将经常用的分组进行了整合，将其合并到一个依赖中[7]。Spring Boot为Spring平台及第三方库提供开箱即用的设置，这样你就可以有条不紊地开始。多数Spring Boot应用只需要很少的Spring配置。

Spring Cloud是一个微服务框架，它提供了全套的分布式系统解决方案。Spring Cloud对为服务基础框架Netflix的多个开源组件进行封装，同时实现了云端平台以及和Spring Boot开发框架的集成。它利用Spring Boot的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用Spring Boot的开发风格做到一键启动和部署。Spring MVC是Spring WEB模块的一个应用框架,它分离了控制器、模型对象、分派器以及处理程序对象的角色,从而实现程序MVC模式的分层定制[5]。

Nacos是Alibaba团队开发的一个更易于构建云原生应用的动态服务发现、配置管理和服务管理平台。其功能包括：动态配置服务、服务发现及管理、动态DNS服务。动态配置服务能够以中心化、外部化和动态化的方式管理所有环境的配置。动态配置消除了配置变更时重新部署应用和服务的需要。配置中心化管理让实现无状态服务更简单，也让按需弹性扩展服务更容易。动态服务发现对以服务为中心的（例如微服务和云原生）应用架构方式非常关键。Nacos支持DNS-Based和RPC-Based（Dubbo、gRPC）模式的服务发现。Nacos也提供实时健康检查，以防止将请求发往不健康的主机或服务实例。借助Nacos，可以更容易地服务实现断路器。通过支持权重路由，动态DNS服务能轻松实现中间层负载均衡、更灵活的路由策略、流量控制以及简单数据中心内网的简单DNS解析服务。动态DNS服务还能让您更容易地实现以DNS协议为基础的服务发现，以消除耦合到厂商私有服务发现API上的风险。尤其Nacos能够无缝支持Spring Cloud，易于使用。因此选择Nacos而放弃Eureka和Zookeeper。

MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射[8]。Mybatis-Plus是baomihua开发的基于Mybatis框架，只做增强不做改变，引入它不会对现有工程产生影响，通过简单的配置，即可快速进行增删改查操作，节省了大量时间，Mybatis-Plus有丰富的附加功能，包括热加载、代码生成、分页、性能分析等功能一应俱全。

Vue.js是一个构建用户界面的框架(库),它的目标是通过尽可能简单的API实现响应的数据绑定和组合的视图集合 Vue自身不是一个全能框架的核心是只关注视图层，因此它非常容易学习，非常容易与其它库或已有项目整合 Vue在与相关工具和支持库一起使用时, 也能完美地驱动复杂的单页应用。

## 3.2 软件体系结构设计

### 3.2.1 功能结构

根据需求调研结果确定本系统主要包括以下功能模块，如图3.1所示。

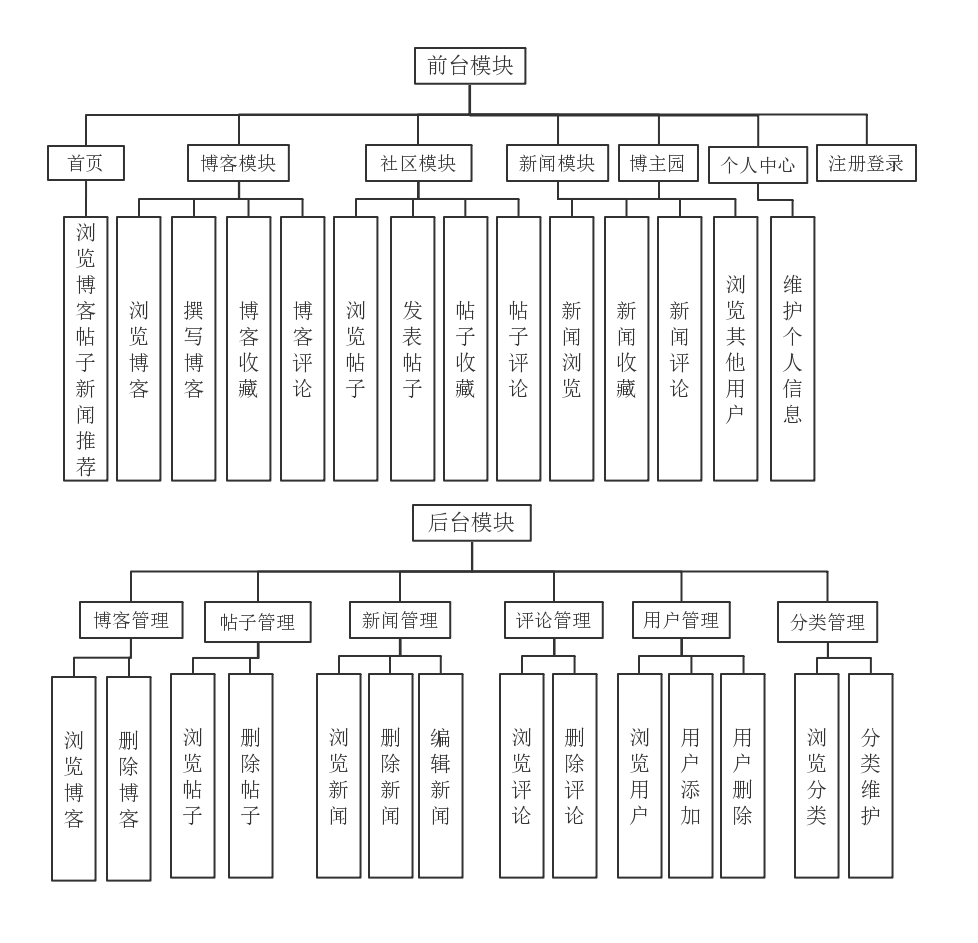


图3.1系统功能结构图

### 3.2.2 系统架构

前端页面使用Vue.js + Element-UI作为页面显示，Vue基于MVVM模式，通过数据驱动进行双向绑定。UI是饿了么团队提供的一套基于Vue2.0的组件库，可以快速搭建网站，提高开发效率。Vue将页面拆分成多个模块，主模块分为头部的菜单栏和主体部分，主体部分通过路由切换不同的模块，模块与模块之间通过Bus传递数据，与服务器之间的通信使用Axios。

前端系统架构图如图3.2所示。

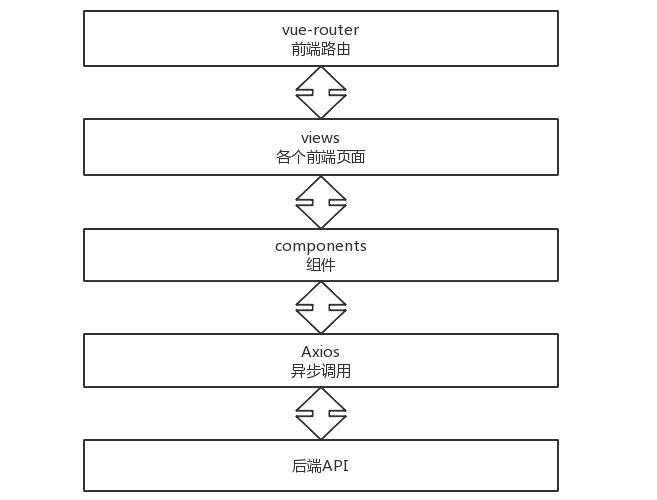


图3.2 前端系统架构图

后端服务器使用Spring Cloud微服务架构，配合Spring Boot进行快速搭建单个服务。服务与服务之间通过Nacos作为服务发现，它作为一个中间人，管理所有服务，服务与服务之间通过Feign调用HTTP API，Feign是Netflix开发的声明式、模块化的HTTP客户端，Spring Cloud支持Feign的Spring MVC的注解，通过注解调用HTTP API。

服务分为消费服务和生产服务，消费服务负责接收前端请求，身份验证、权限控制，通过调用生产服务，对数据进行整理返回给前端，生产服务负责访问数据库，为消费服务提供接口。数据库访问使用Mybatis框架，对数据库进行增删改查的操作。系统中还包含通用模块，模块中包含服务间通用的实例，例如VO、DTO等，还包括通用的工具模块，例如Json转换模块，时间模块等。

后端服务端系统架构图如图3.3所示。

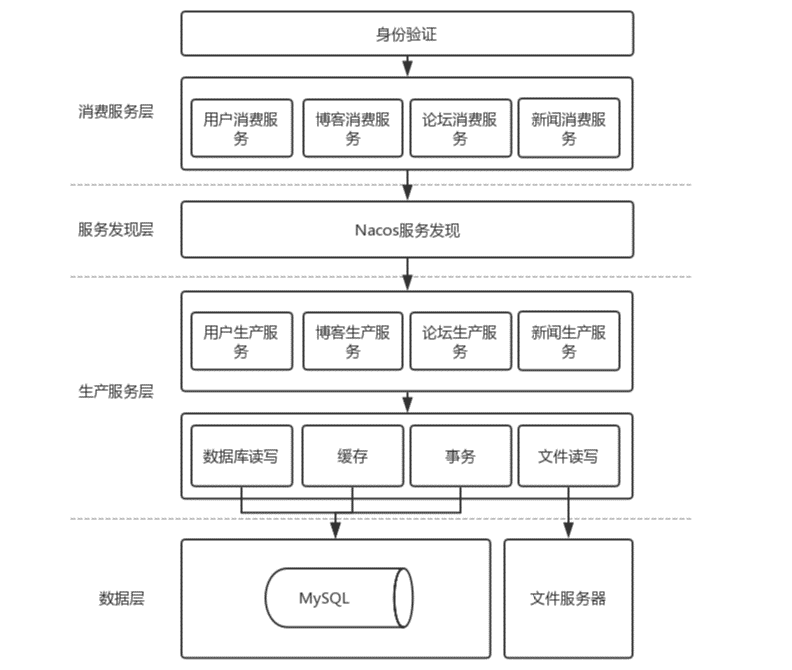


图3.3 服务端系统架构图

## 3.3 数据库设计

### 3.3.1 概念结构设计

新闻、帖子、博客与评论、收藏关系大体相同，E-R图省略了帖子和博客部分。E-R图如图3.4所示。

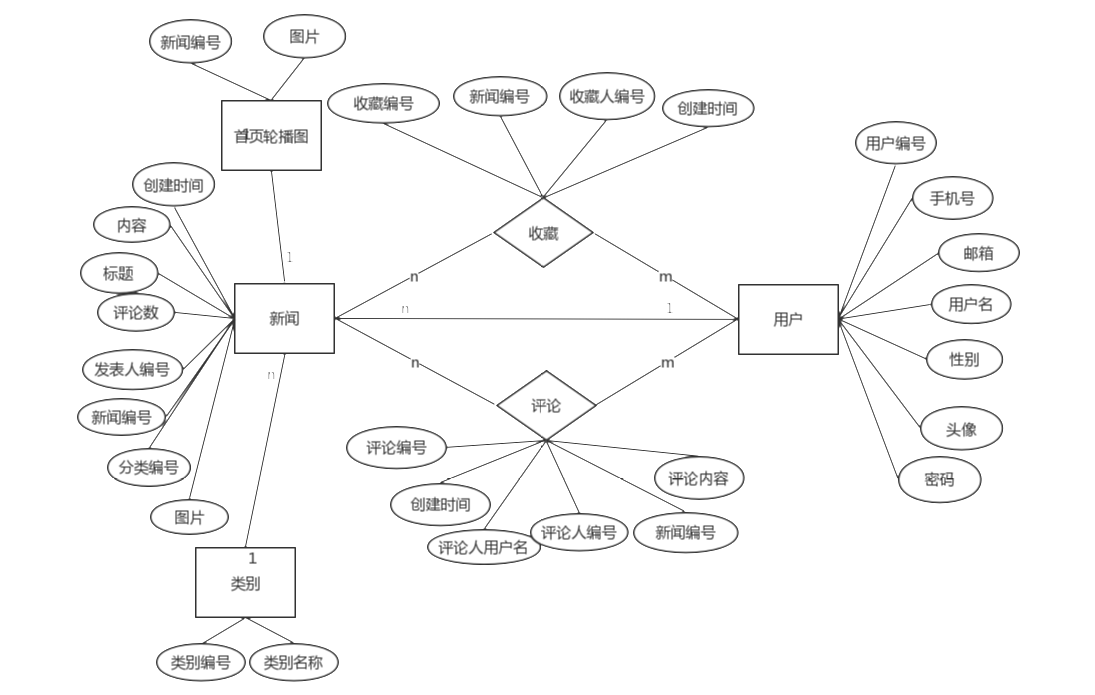


图3.4 E-R图

### 3.3.2 表的详细设计

通过对系统的业务逻辑及领域模型的分析，设计了数据库表结构，如表3.1-表3.12所示。系统使用的数据库为MySQL。

（1）用户信息表(user)

用户信息表用于存储用户的登录信息和个人信息，主要包括：用户名、密码、手机号、邮箱、年龄、性别、头像。表结构如表3.1所示。

表3.1 用户信息表（user）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| user\_id | int | 11 | 否 | 是 | 用户编号 |
| user\_name | varchar | 50 | 否 | 否 | 用户名 |
| passwd | varchar | 50 | 否 | 否 | 密码 |
| mobile | varchar | 11 | 是 | 否 | 手机号 |
| age | varchar | 2 | 是 | 否 | 年龄 |
| email | varchar | 50 | 是 | 否 | 邮箱 |
| sex | varchar | 4 | 是 | 否 | 性别 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |
| img\_url | varchar | 50 | 是 | 否 | 头像地址 |

（2）博客信息表(blog)

博客信息表用于存储博客信息，主要包括：博客标题、博客内容、发布人编号、发布人用户名、所属分类编号、所属分类名、创建时间。表结构如表3.2所示。

表3.2 博客信息表（blog）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| blog\_id | int | 11 | 否 | 是 | 博客编号 |
| title | varchar | 50 | 否 | 否 | 博客标题 |
| content | longtext | 0 | 否 | 否 | 博客内容 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |
| user\_id | int | 11 | 否 | 否 | 发布人编号 |
| read\_count | int | 11 | 否 | 否 | 评论数 |
| category\_id | int | 11 | 否 | 否 | 分类编号 |
| category\_name | varchar | 50 | 否 | 否 | 分类名 |

（3）博客评论表(blog\_comment)

博客评论表用于存储博客评论信息，主要包括：评论内容、评论人编号、评论人用户名、关联博客编号、关联博客标题、创建时间。表结构如表3.3所示。

表3.3 博客评论表（blog\_comment）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| comment\_id | int | 11 | 否 | 是 | 评论编号 |
| content | varchar | 50 | 否 | 否 | 评论内容 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |
| user\_id | int | 11 | 否 | 否 | 评论人编号 |
| blog\_id | int | 11 | 否 | 否 | 关联博客编号 |
| blog\_title | varchar | 50 | 否 | 否 | 关联博客标题 |

（4）博客收藏表(blog\_collection)

博客收藏表用于存储用户收藏博客信息，主要包括：博客编号、博客标题、收藏人编号、收藏人用户名、创建时间。表结构如表3.4所示。

表3.4 博客收藏表（blog\_collection）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| collection\_id | int | 11 | 否 | 是 | 收藏编号 |
| user\_id | int | 11 | 否 | 否 | 收藏人编号 |
| user\_name | varchar | 50 | 否 | 否 | 收藏人用户名 |
| blog\_id | int | 11 | 否 | 否 | 收藏博客编号 |
| blog\_title | varchar | 11 | 否 | 否 | 收藏博客标题 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |

（5）新闻信息表(news)

新闻信息表用于存储新闻信息，主要包括：新闻标题、新闻内容、发布人编号、发布人用户名、所属分类编号、所属分类名、创建时间。表结构如表3.5所示。

表3.5 新闻信息表（news）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| news\_id | int | 11 | 否 | 是 | 新闻编号 |
| title | varchar | 50 | 否 | 否 | 新闻标题 |
| content | longtext | 0 | 否 | 否 | 新闻内容 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |

续表3.5 新闻信息表（news）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| user\_id | int | 11 | 否 | 否 | 发布人编号 |
| read\_count | int | 11 | 否 | 否 | 评论数 |
| user\_name | varchar | 50 | 否 | 否 | 发布人用户名 |
| category\_id | int | 11 | 否 | 否 | 分类编号 |
| category\_name | varchar | 50 | 否 | 否 | 分类名 |

（6）新闻评论表(news\_comment)

新闻评论表用于存储新闻评论信息，主要包括：评论内容、评论人编号、评论人用户名、关联新闻编号、关联新闻标题、创建时间。表结构如表3.6所示。

表3.6 新闻评论表（news\_comment）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| comment\_id | int | 11 | 否 | 是 | 评论编号 |
| content | varchar | 50 | 否 | 否 | 评论内容 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |
| user\_id | int | 11 | 否 | 否 | 评论人编号 |
| user\_name | varchar | 50 | 否 | 否 | 发布人用户名 |
| news\_id | int | 11 | 否 | 否 | 关联新闻编号 |
| news\_title | varchar | 50 | 否 | 否 | 关联新闻标题 |

（7）新闻收藏表(news\_collection)

新闻收藏表用于存储用户收藏新闻信息，主要包括：新闻编号、新闻标题、收藏人编号、收藏人用户名、创建时间。表结构如表3.7所示。

表3.7 新闻收藏表（news\_collection）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| collection\_id | int | 11 | 否 | 是 | 收藏编号 |
| user\_id | int | 11 | 否 | 否 | 收藏人编号 |
| news\_id | int | 11 | 否 | 否 | 收藏新闻编号 |
| news\_title | varchar | 11 | 否 | 否 | 收藏新闻标题 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |

（8）帖子信息表(post)

帖子信息表用于存储帖子信息，主要包括：帖子标题、帖子内容、发布人编号、发布人用户名、创建时间。表结构如表3.8所示。

表3.8 帖子信息表（post）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| post\_id | int | 11 | 否 | 是 | 帖子编号 |
| title | varchar | 50 | 否 | 否 | 帖子标题 |
| content | longtext | 0 | 否 | 否 | 帖子内容 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |
| user\_id | int | 11 | 否 | 否 | 发布人编号 |
| read\_count | int | 11 | 否 | 否 | 评论数 |

（9）帖子评论表(post\_comment)

帖子评论表用于存储帖子评论信息，主要包括：评论内容、评论人编号、评论人用户名、关联帖子编号、关联帖子标题、创建时间。表结构如表3.9所示。

表3.9 帖子评论表（post\_comment）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| comment\_id | int | 11 | 否 | 是 | 评论编号 |
| content | varchar | 50 | 否 | 否 | 评论内容 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |
| user\_id | int | 11 | 否 | 否 | 评论人编号 |
| user\_name | varchar | 50 | 否 | 否 | 发布人用户名 |
| post\_id | int | 11 | 否 | 否 | 关联帖子编号 |
| post\_title | varchar | 50 | 否 | 否 | 关联帖子标题 |

（10）帖子收藏表(post\_collection)

帖子收藏表用于存储用户收藏帖子信息，主要包括：帖子编号、帖子标题、收藏人编号、收藏人用户名、创建时间。表结构如表3.10所示。

表3.10 帖子收藏表（post\_collection）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| collection\_id | int | 11 | 否 | 是 | 收藏编号 |
| user\_id | int | 11 | 否 | 否 | 收藏人编号 |
| user\_name | varchar | 50 | 否 | 否 | 收藏人用户名 |
| post\_id | int | 11 | 否 | 否 | 收藏帖子编号 |
| post\_title | varchar | 11 | 否 | 否 | 收藏帖子标题 |
| create\_time | datetime | 0 | 否 | 否 | 创建时间 |

（11）分类表(category)

分类表用于存储分类信息，主要包括：分类编号、分类名、创建时间。表结构如表3.11所示。

表3.11 分类表（category）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| category\_id | int | 11 | 否 | 是 | 收藏编号 |
| category\_name | int | 11 | 否 | 否 | 收藏人编号 |
| create\_time | varchar | 50 | 否 | 否 | 收藏人用户名 |

（12）首页轮播图图片表(post\_collection)

首页轮播图图片表用于存储首页轮播图图片地址，主要包括：图片编号、图片地址、关联新闻编号。表结构如表3.12所示。

表3.12 帖子收藏表（post\_collection）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否主键** | **说明** |
| head\_id | int | 11 | 否 | 是 | 轮播图编号 |
| img\_url | varchar | 50 | 否 | 否 | 图片地址 |
| news\_id | news\_id | 11 | 否 | 否 | 关联新闻编号 |

## 3.4 用例实现

### 3.4.1 撰写博客用例实现

用户撰写博客时，首先点击编辑博客进入博客编辑页面，用户可以点击上传图片或视频，通过文件控制器将文件上传到文件服务器，用户编辑完后，点击提交，服务器对数据进行处理后在页面上提示编辑成功。撰写博客用例的实现顺序图如图3.5所示。

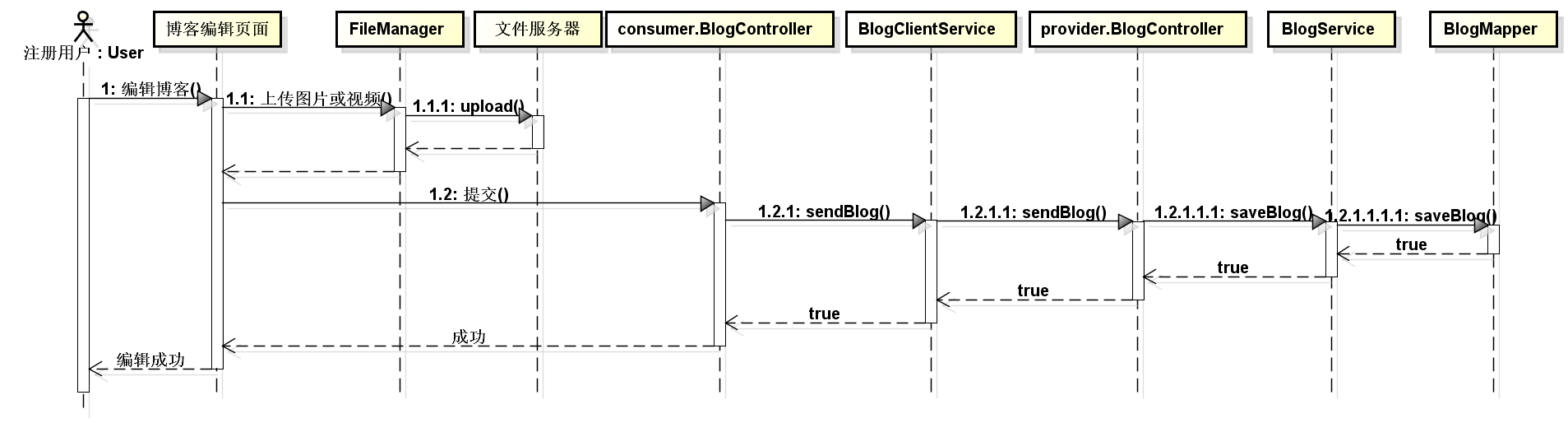


图3.5撰写博客用例顺序图

### 3.4.2 修改个人信息用例实现

用户修改个人信息时，首先点击修改信息，系统显示编辑界面，用户填写修改信息后点击提交，服务器对数据进行处理，处理完成后，界面提示修改成功。修改个人信息用例的实现顺序图如图3.6所示。

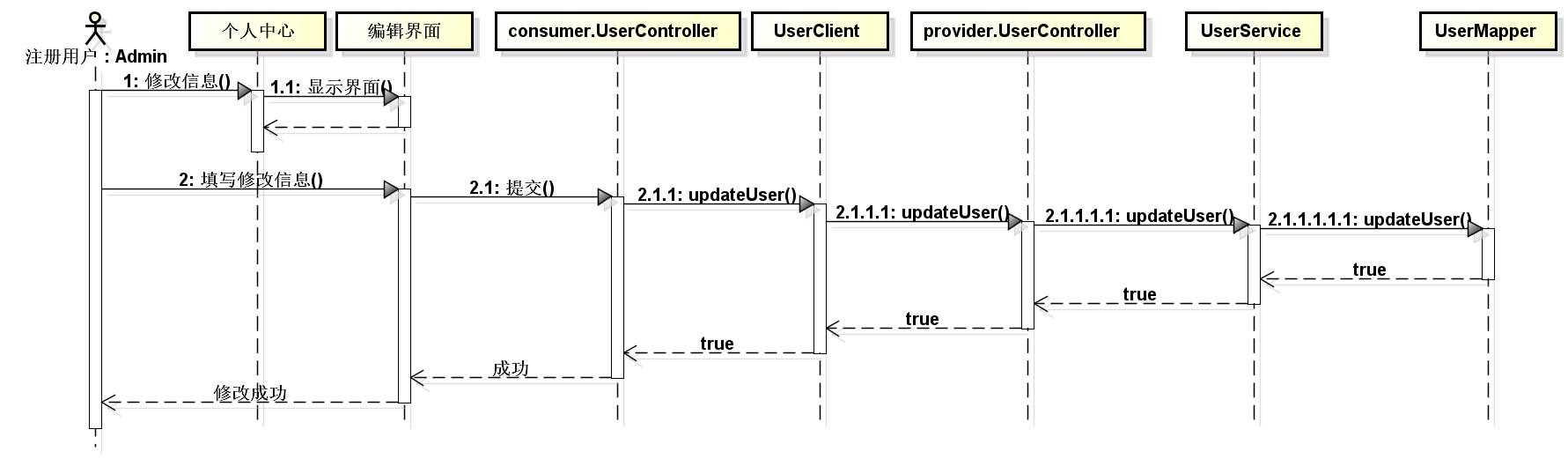
****

图3.6撰写博客用例顺序图

### 3.4.3 修改用户博客用例实现

用户要修改自己的博客时，用户通过点击我的博客，查看博客，前端通过调用服务器接口，从数据库读取博客详情数据返回给前端，前端通过数据显示博客详情。用户对博客修改后提交，后台服务将修改后的数据存入数据库，成功后返回给前端，前端页面显示修改成功。修改用户博客用例的实现顺序图如图3.7所示。

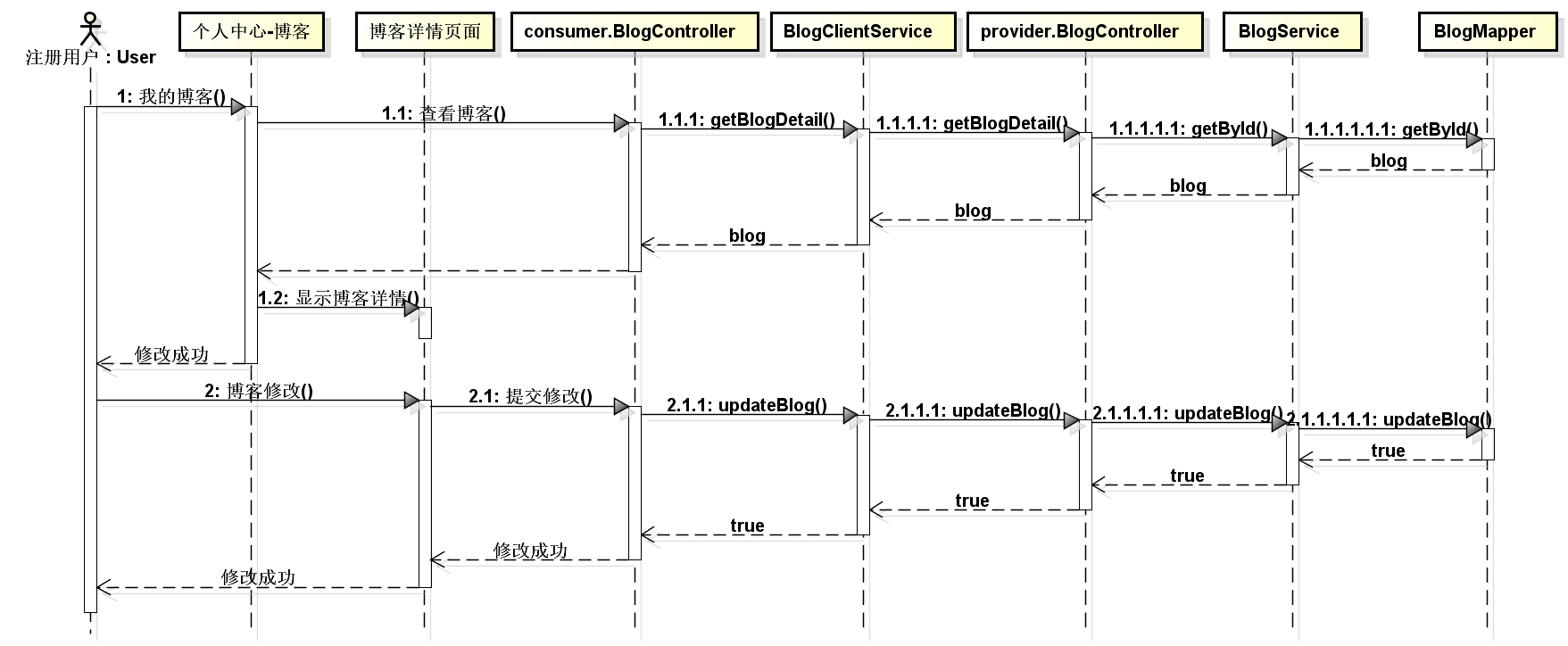


图3.7修改用户博客用例顺序图

### 3.4.4 删除用户博客用例实现

用户删除自己的博客时，点击我的博客进入个人中心-博客界面，选择要删除的博客，点击删除，服务器操作数据库，对博客进行删除，删除成功后显示成功提示。删除用户博客用例的实现顺序图如图3.8所示。

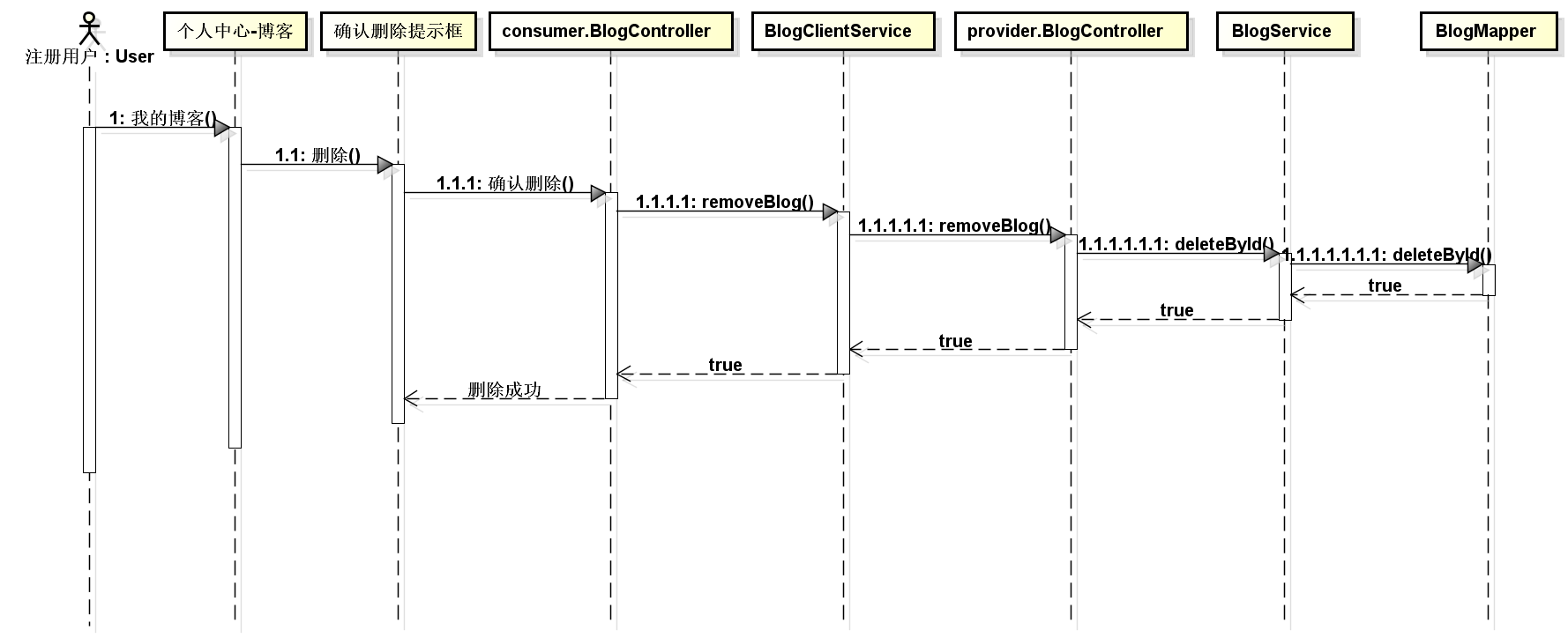


图3.8删除用户博客用例顺序图

### 3.4.5 发表评论用例实现（以博客为例）

用户在博客下发表评论，首先在评论框输入评论，点击发表后，前端将数据发送给后台服务器，后台服务器将评论信息存入数据库，存入成功后，操作博客表，将博客评论数加一，成功后返回评论成功。发表评论用例的实现顺序图如图3.9所示。

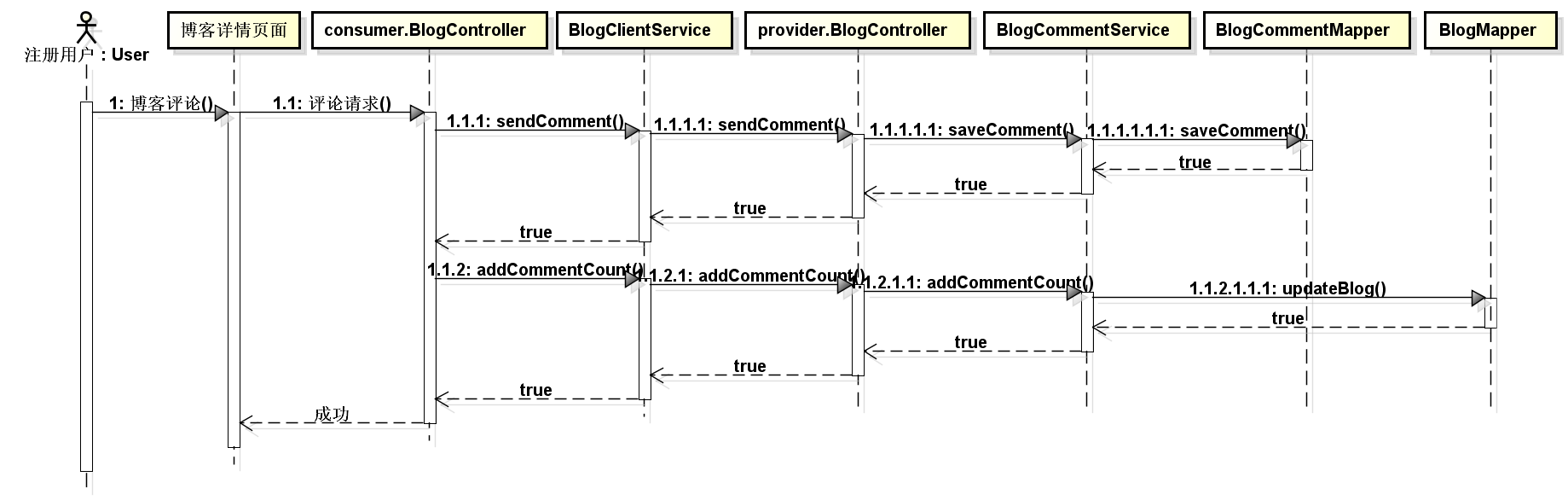


图3.9发表评论用例顺序图

### 3.4.6 收藏用例实现（以博客为例）

用户要收藏博客时，首先点击博客详情页面的收藏按钮，前端页面将数据传递给后台，后台操作数据库存储收藏信息，存储成功后返回给前端页面，前端页面显示收藏成功。收藏用例的实现顺序图如图3.10所示。

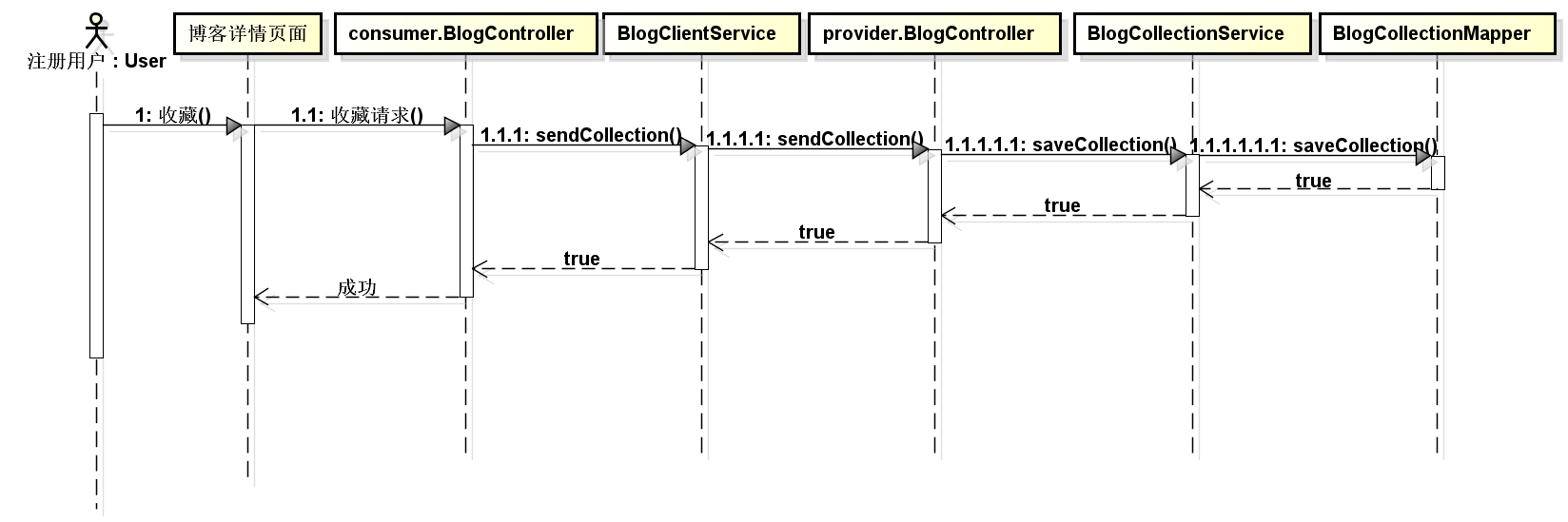


图3.10删除用户博客用例顺序图

### 3.4.7 添加新闻用例实现

管理员在后台管理端添加新闻，首先管理员在新闻管理页面点击新闻添加，进入新闻添加页，管理员在编辑新闻后，点击提交，前端页面将数据发送给后端服务器，通过操作数据库，将新闻数据存储到数据库中，存储成功后返回添加成功。添加新闻用例的实现顺序图如图3.11所示。

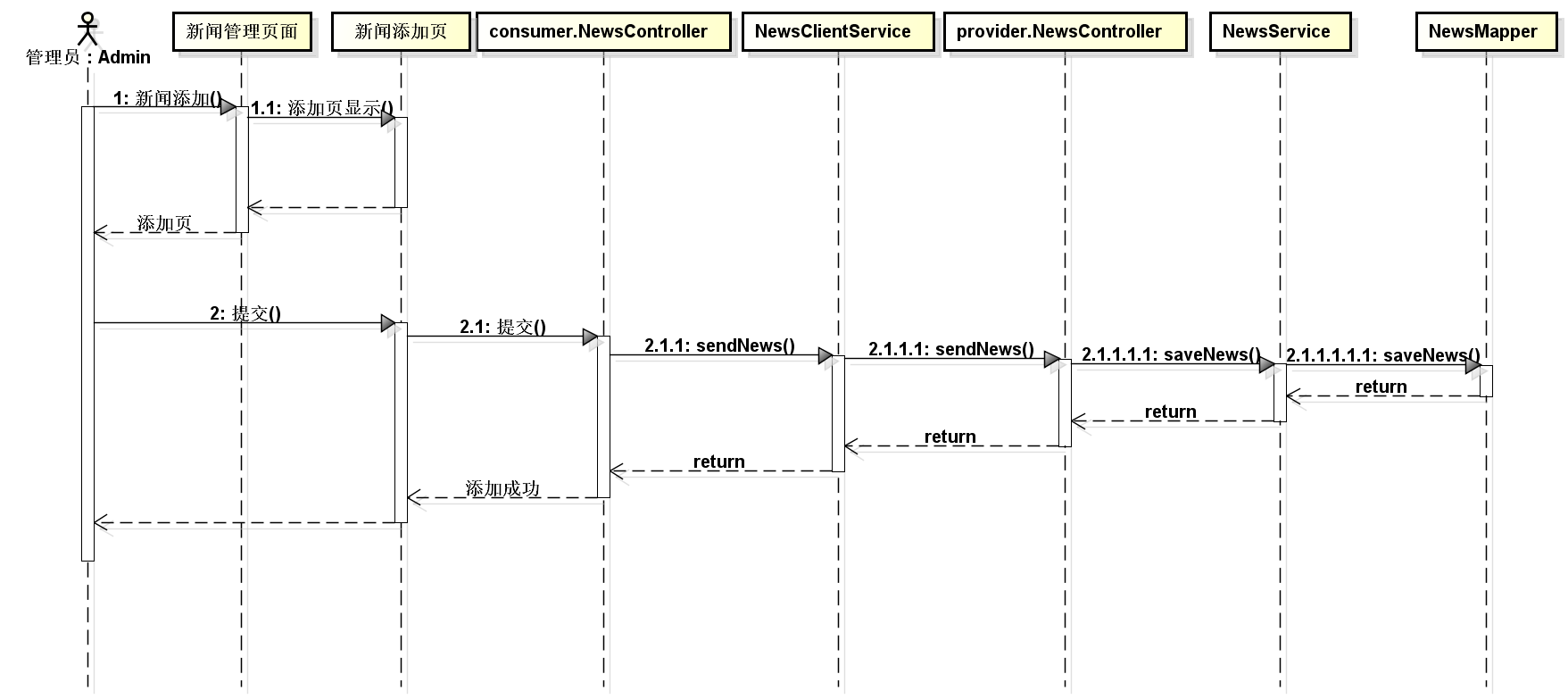


图3.11添加新闻用例顺序图

### 3.4.8 修改新闻用例实现

管理员修改新闻时，首先点击需要编辑的新闻，进入新闻编辑界面，管理员编辑后点击提交，前端将数据发送黑后台服务器，后台服务器操作数据库，对该条新闻数据进行更新，更新成功后，前端显示修改成功。修改新闻用例的实现顺序图如图3.12所示。

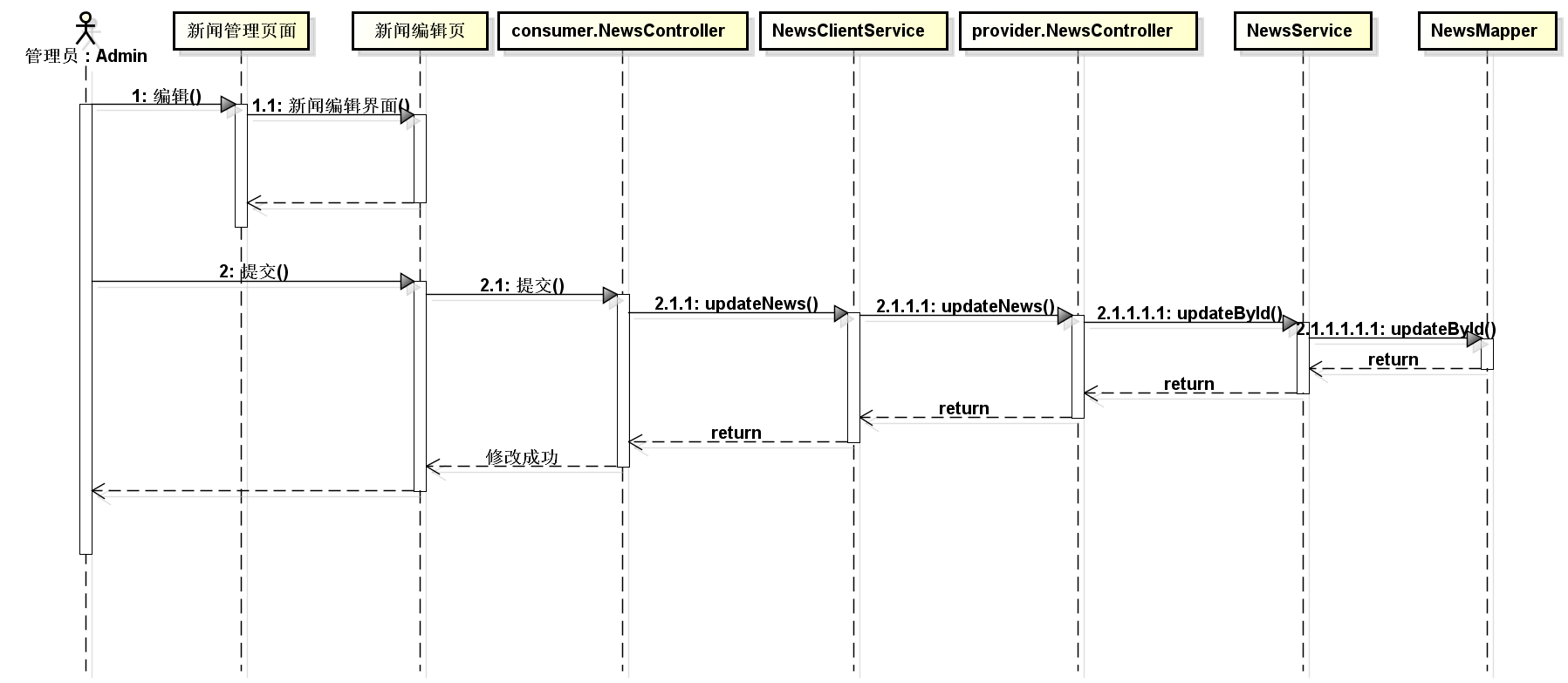


图3.12修改新闻用例顺序图

## 3.5 对象设计

### 3.5.1 模块设计

本系统分为消费者控制模块、Feign客户端模块、生产者控制模块、生产者业务模块、生产者持久模块、业务实体模块。模块间调用及依赖关系为：消费者控制模块依赖Feign客户端模块，Feign客户端通过Http调用生产者控制模块，生产者控制模块依赖生产者业务模块，生产者业务模块依赖生产者持久模块，业务实体模块包含数据库实体和业务模块处理后的业务实体，其他模块均依赖业务实体模块。依赖关系如图3.13所示。

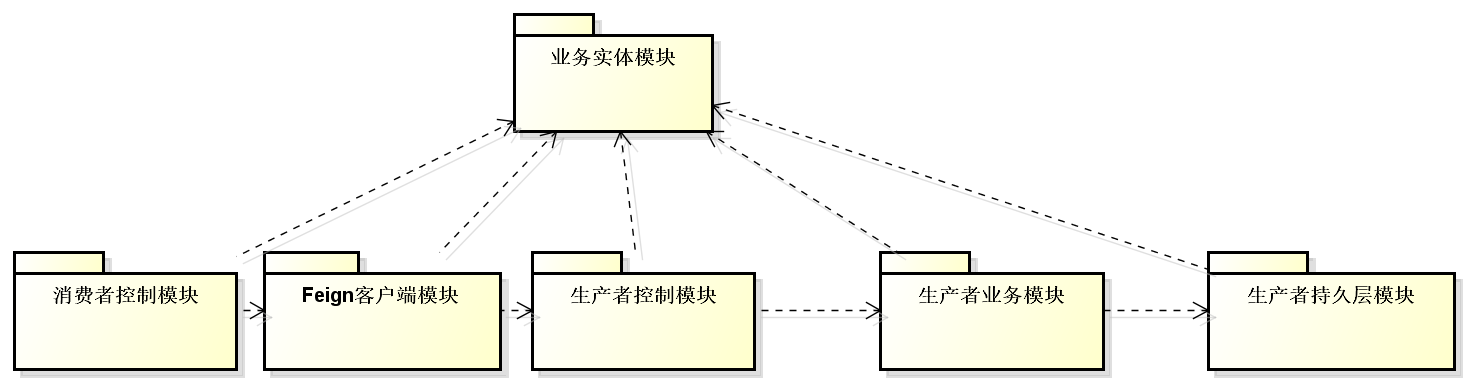


图3.13模块依赖关系图

### 3.5.2 业务实体类的设计

业务实体类的设计与数据库设计相同，即数据库实体与业务实体为同一实体。业务实体类的设计关系如图3.14。

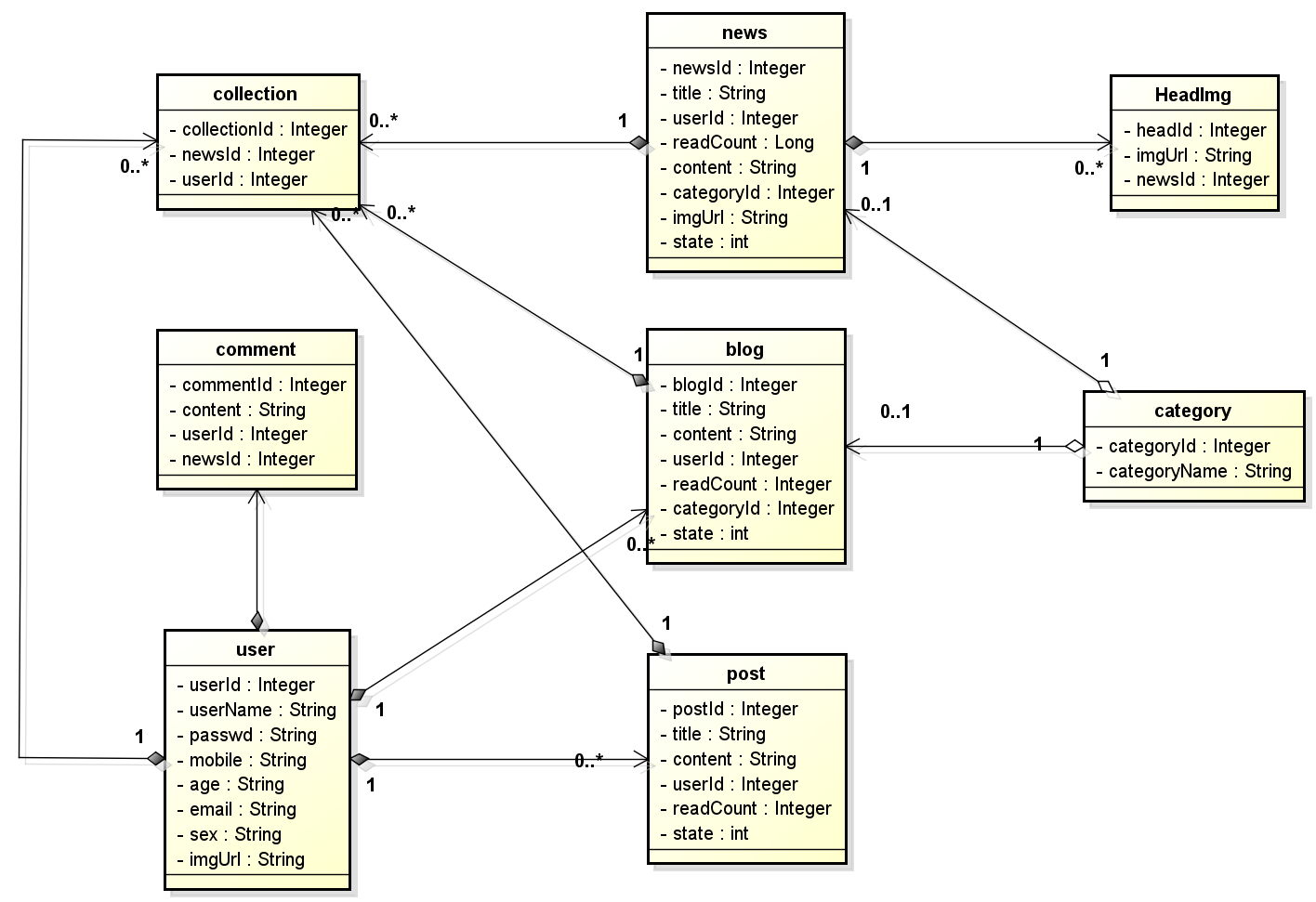


图3.14业务实体类的设计关系图

### 3.5.3 业务处理类的设计

（1）、新闻、博客、帖子业务处理类设计

系统中新闻、博客、帖子的业务类似，均由消费者服务通过Feign客户端调用对应的生产者服务获取或处理数据。所有数据均有生产者服务从MySQL数据库获取。业务处理类的设计如图3.15所示（以新闻为例）。

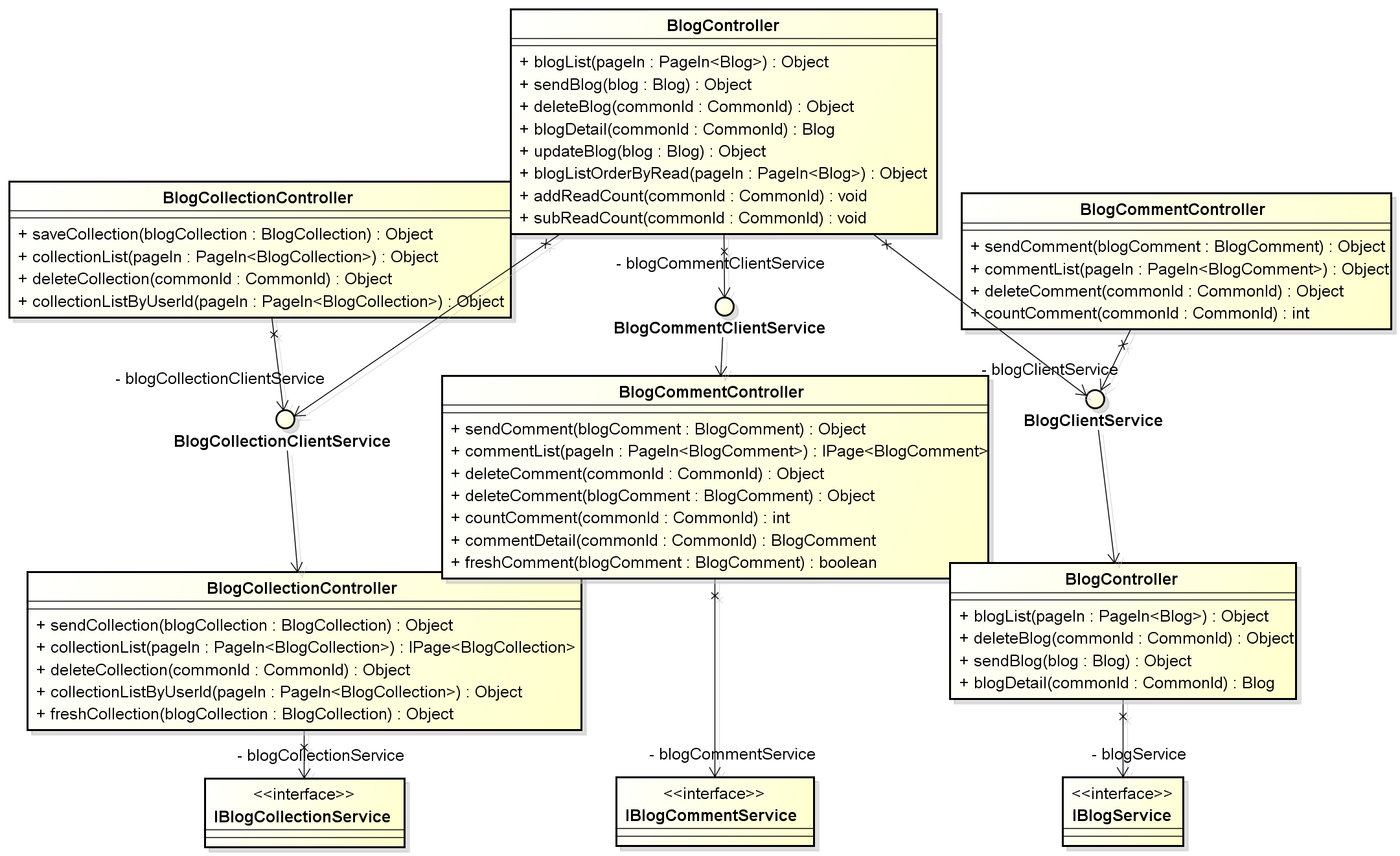


图3.15新闻业务处理类设计图

（2）用户业务处理类设计

用户业务包括用户对博客、新闻、帖子、其他用户、用户本身数据的获取、添加及更新。所有数据均有生产者服务从MySQL数据库获取。用户业务处理类的设计如图3.16所示。

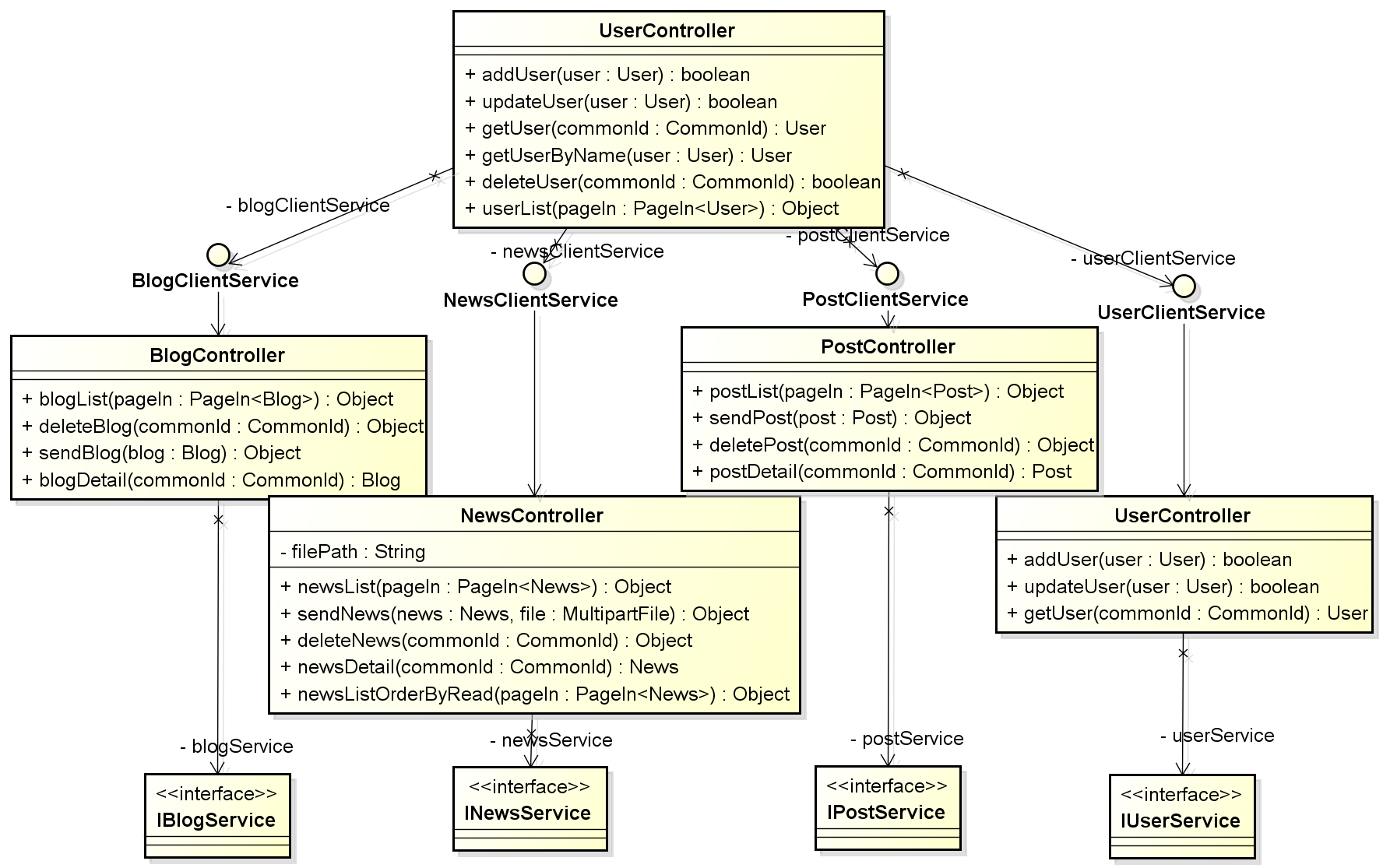


图3.16用户业务处理类设计图

### 3.5.4 类的描述

（1）业务实体类描述

业务实体包中类的描述如表3.13所示。

表3.13 业务实体包中类的描述

|  |  |  |
| --- | --- | --- |
| **类名** | **属性/方法** | **描述** |
| Blog | Integer blogId | 博客编号 |
| String title | 博客标题 |
| String content | 博客内容 |
| LocalDateTime createTime | 创建时间 |
| Integer userId | 用户编号 |
| Integer readCount | 评论数 |
| Integer state | 状态 |
| String userName | 用户名 |
| Integer categoryId | 类别编号 |
| String categoryName | 类别名 |
| BlogCollection | Integer collectionId | 收藏编号 |
| Integer userId | 用户编号 |
| String userName | 用户名） |
| Integer blogId | 博客编号 |
| LocalDateTime createTime | 创建时间 |
| String blogTitle | 博客标题 |
| BlogComment | Integer commentId | 评论编号 |
| String content | 评论内容 |
| Integer userId | 用户编号 |
| LocalDateTime createTime | 创建时间 |
| Integer blogId | 博客编号 |
| String userName | 用户名 |
| String blogTitle | 博客标题 |
| News | Integer newsId | 新闻编号 |
| String title | 新闻标题 |
| String content | 新闻内容 |
| LocalDateTime createTime | 创建时间 |
| Integer userId | 用户编号 |
| Integer readCount | 评论数 |
| Integer state | 状态 |
| String userName | 用户名 |
| Integer categoryId | 类别编号 |
| String categoryName | 类别名 |
| String imgUrl | 展示图 |
| NewsCollection | Integer collectionId | 收藏编号 |
| Integer userId | 用户编号 |
| String userName | 用户名） |
| Integer newsId | 新闻编号 |
| LocalDateTime createTime | 创建时间 |
| String newsTitle | 新闻标题 |
| NewsComment | Integer commentId | 评论编号 |
| String content | 评论内容 |

续表3.13 业务实体包中类的描述

|  |  |  |
| --- | --- | --- |
| **类名** | **属性/方法** | **描述** |
| NewsComment | Integer userId | 用户编号 |
| LocalDateTime createTime | 创建时间 |
| Integer newsId | 新闻编号 |
| String userName | 用户名 |
| String newsTitle | 新闻标题 |
| Post | Integer postId | 帖子编号 |
| String content | 帖子内容 |
| LocalDateTime createTime | 创建时间 |
| Integer userId | 用户编号 |
| Integer state | 状态 |
| Integer readCount | 评论数 |
| String userName | 用户名 |
| PostCollection | Integer collectionId | 收藏编号 |
| Integer userId | 用户编号 |
| String userName | 用户名） |
| Integer postId | 帖子编号 |
| LocalDateTime createTime | 创建时间 |
| String postTitle | 帖子标题 |
| PostComment | Integer commentId | 评论编号 |
| String content | 评论内容 |
| Integer userId | 用户编号 |
| LocalDateTime createTime | 创建时间 |
| Integer postId | 帖子编号 |
| String userName | 用户名 |
| String postTitle | 帖子标题 |
| Category | Integer categoryId | 分类编号 |
| String categoryName | 分类名 |
| LocalDateTime createTime | 创建时间 |
| HeadImg | Integer headId | 轮播图编号 |
| String imgUrl | 图片地址 |
| Integer newsId | 新闻编号 |
| User | Integer userId | 用户编号 |
| String userName | 用户名 |
| String passwd | 密码 |
| String mobile | 手机号 |
| String age | 年龄 |
| String email | 邮箱 |
| String sex | 姓名 |
| LocalDateTime createTime | 创建时间 |
| String imgUrl | 头像 |

（2）消费者控制层类的描述

消费者控制层类的描述如表3.14所示。

表3.14 消费者控制层类的描述

|  |  |  |
| --- | --- | --- |
| **类名** | **属性/方法** | **描述** |
| BlogCollectionController | saveCollection(BlogCollection blogCollection): Object | 收藏博客请求处理 |
| collectionList(PageIn<BlogCollection> pageIn): Object | 获取博客收藏列表请求处理 |
| deleteCollection(CommonId commonId): Object | 删除博客收藏请求处理 |
| collectionListByUserId(PageIn<BlogCollection> pageIn): Object | 根据用户编号获取博客收藏列表请求处理 |
| BlogCommentController | sendComment(BlogComment blogComment) : Object | 发表博客评论请求处理 |
| commentList(PageIn<BlogComment> pageIn) : Object | 获取评论列表请求处理 |
| deleteComment(CommonId commonId) : Object | 删除评论请求处理 |
| countComment(CommonId commonId) : Object | 计算评论数请求处理 |
| BlogController | blogList(PageIn<Blog> pageIn) : Object | 博客列表请求处理 |
| sendBlog(Blog blog) : Object | 发表博客请求处理 |
| deleteBlog(CommonId commonId) : Object | 删除博客请求处理 |
| blogDetail(CommonId commonId) : Object | 获取博客详情请求处理 |
| updateBlog(Blog blog) : Object | 更新博客请求处理 |
| blogListOrderByRead(PageIn<Blog> pageIn) : Object | 获取按评论数排序的博客列表请求处理 |
| addReadCount(CommonId commonId) : Object | 评论数+1请求处理 |
| subReadCount(CommonId commonId) : Object | 评论数-1请求处理 |
| NewsCollectionController | collectionList(PageIn<NewsCollection> pageIn) : Object | 获取新闻收藏请求处理 |
| deleteCollection(CommonId commonId) : Object | 删除新闻收藏请求处理 |
| collectionListByUserId(PageIn<NewsCollection> pageIn) : Object | 根据用户编号获取新请求处理闻收藏 |
| sendCollection(NewsCollection newsCollection) : Object | 收藏新闻请求处理 |
| NewsCommentController | sendComment(NewsComment newsComment) : Object | 发表新闻评论请求处理 |
| commentList(PageIn<NewsComment> pageIn) : Object | 新闻评论列表请求处理 |
| deleteComment(CommonId commonId) : Object | 删除新闻评论请求处理 |
| countComment(CommonId commonId) : Object | 计算评论数请求处理 |
| NewsController | newsList(PageIn<News> pageIn) : Object | 获取新闻列表请求处理 |
| sendNews(News news, MultipartFile file) : Object | 发表新闻请求处理 |
| deleteNews(CommonId commonId) : Object | 删除新闻请求处理 |
| newsDetail(CommonId commonId) : Object | 获取新闻详情请求处理 |
| updateNews(News news, MultipartFile file) : Object | 更新新闻请求处理 |
| newsListOrderByRead(PageIn<News> pageIn) : Object | 获取按评论数排序的请求处理新闻列表 |
| addReadCount(CommonId commonId) : Object | 评论数+1请求处理 |

续表3.14 消费者控制层类的描述

|  |  |  |
| --- | --- | --- |
| **类名** | **属性/方法** | **描述** |
| CategoryController | addCategory(Category category) : Object | 添加分类请求处理 |
| updateCategory(Category category) : Object | 更新分类请求处理 |
| getCategory(CommonId commonId) : Object | 根据分类编号获取分类请求处理 |
| getCategoryByName(Category category) : Object | 根据分类名获取分类请求处理 |
| deleteCategory(CommonId commonId) : Object | 删除分类请求处理 |
| categoryList(PageIn<Category> pageIn) : Object | 获取分类列表请求处理 |
| categoryList(): Object | 获取所有分类请求处理 |
| PostCollectionController | sendCollection(PostCollection postCollection) : Object | 获取帖子收藏请求处理 |
| collectionList(PageIn<PostCollection> pageIn) : Object | 删除帖子收藏请求处理 |
| deleteCollection(CommonId commonId) : Object | 根据用户编号获取帖子收藏请求处理 |
| collectionListByUserId(PageIn<PostCollection> pageIn) : Object | 收藏帖子请求处理 |
| PostCommentController | sendComment(PostComment postComment) : Object | 发表帖子评论请求处理 |
| commentList(PageIn<PostComment> pageIn) : Object | 获取评论列表请求处理 |
| deleteComment(CommonId commonId) : Object | 删除评论请求处理 |
| countComment(CommonId commonId) : Object | 计算评论数请求处理 |
| PostController | postList(PageIn<Post> pageIn) : Object | 获取帖子列表请求处理 |
| sendPost(Post post) : Object | 发表帖子请求处理 |
| deletePost(CommonId commonId) : Object | 删除帖子请求处理 |
| postDetail(CommonId commonId) : Object | 获取帖子详情请求处理 |
| updatePost(Post post) : Object | 更新帖子请求处理 |
| postListOrderByRead(PageIn<Post> pageIn) : Object | 获取按评论数排序的帖子列表请求处理 |
| addReadCount(CommonId commonId) : Object | 评论数+1请求处理 |
| subReadCount(CommonId commonId) : Object | 评论数-1请求处理 |
| UserController | addUser(User user) : Object | 添加用户请求处理 |
| updateUser(User user) : Object | 更新用户请求处理 |
| getUser(CommonId commonId) : Object | 根据用户编号获取用户请求处理 |
| getUserByName(User user) : Object | 根据用户名密码获取用户请求处理 |
| deleteUser(CommonId commonId) : Object | 删除用户请求处理 |
| userList(PageIn<User> pageIn) : Object | 获取用户列表请求处理 |

（3）生产者业务层与生产者持久层的类由Mybatis-Plus实现

# 第4章　系统实现

## 4.1 核心功能实现

### 4.1.1 新闻/博客/帖子编辑

本系统中最核心的功能为在新闻、博文和帖子中插入视频。

内容编辑区可以插入各类格式的文字、代码块，可以对文字进行段落编辑，如前后缩进等，可以插入链接、图片和视频。如图4.1所示。

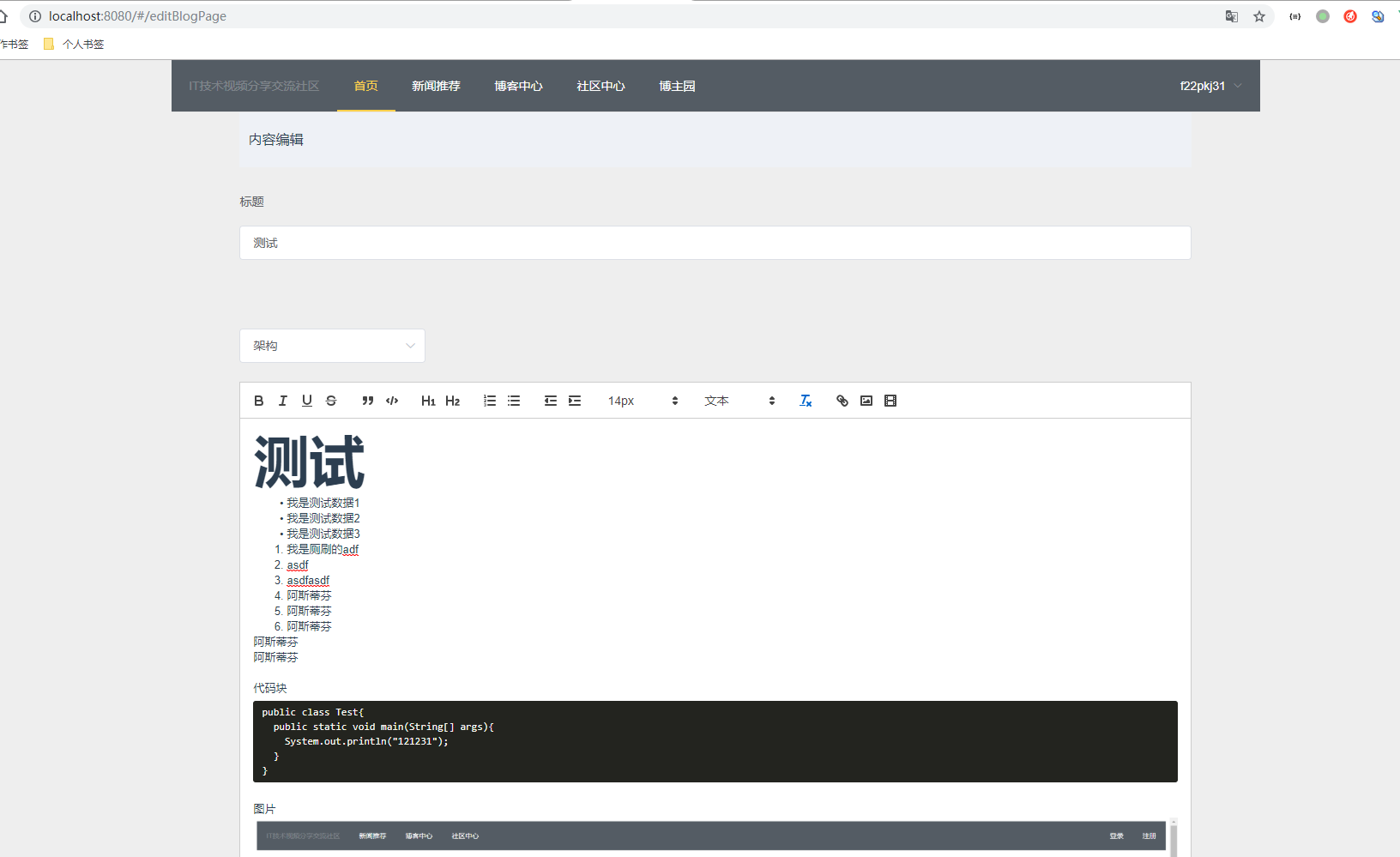


图4.1 编辑展示图

相关前端代码如下：

<!-- 视频上传 -->

<div class="floatBox" v-show='upvideoShow'>

<div class="floatsmBox">

<b-form-file accept="video/\*" placeholder="选择视频文件" ref="videofilereset"

v-model="videofile"></b-form-file>

<span @click='upVideo' class="btn btn-outline-success" style="margin-top: 10px;">确认</span>

<span @click='cancelupVideo' class="btn btn-outline-success" style="margin-top: 10px;">取消</span>

</div>

</div>

<!-- 图片上传 -->

<div class="floatBox" v-show='upimgShow'>

<div class="floatsmBox">

<b-form-file accept="image/\*" placeholder="选择图片文件" ref="imgfilereset" v-model="imgfile"></b-form-file>

<span @click='upImg' class="btn btn-outline-success" style="margin-top: 10px;">确认</span>

<span @click='cancelupImg' class="btn btn-outline-success" style="margin-top: 10px;">取消</span>

</div>

</div>

<div @click='upvideoShow=true' id="upvideoshow" style="display: none;"></div>

<div @click='upimgShow=true' id="upimgshow" style="display: none;"></div>

### 4.1.2 新闻模块

用户可以在新闻推荐页查看新闻，在列表页可以看到新闻的标题、发布人、创建时间和评论数，功能演示如图4.2

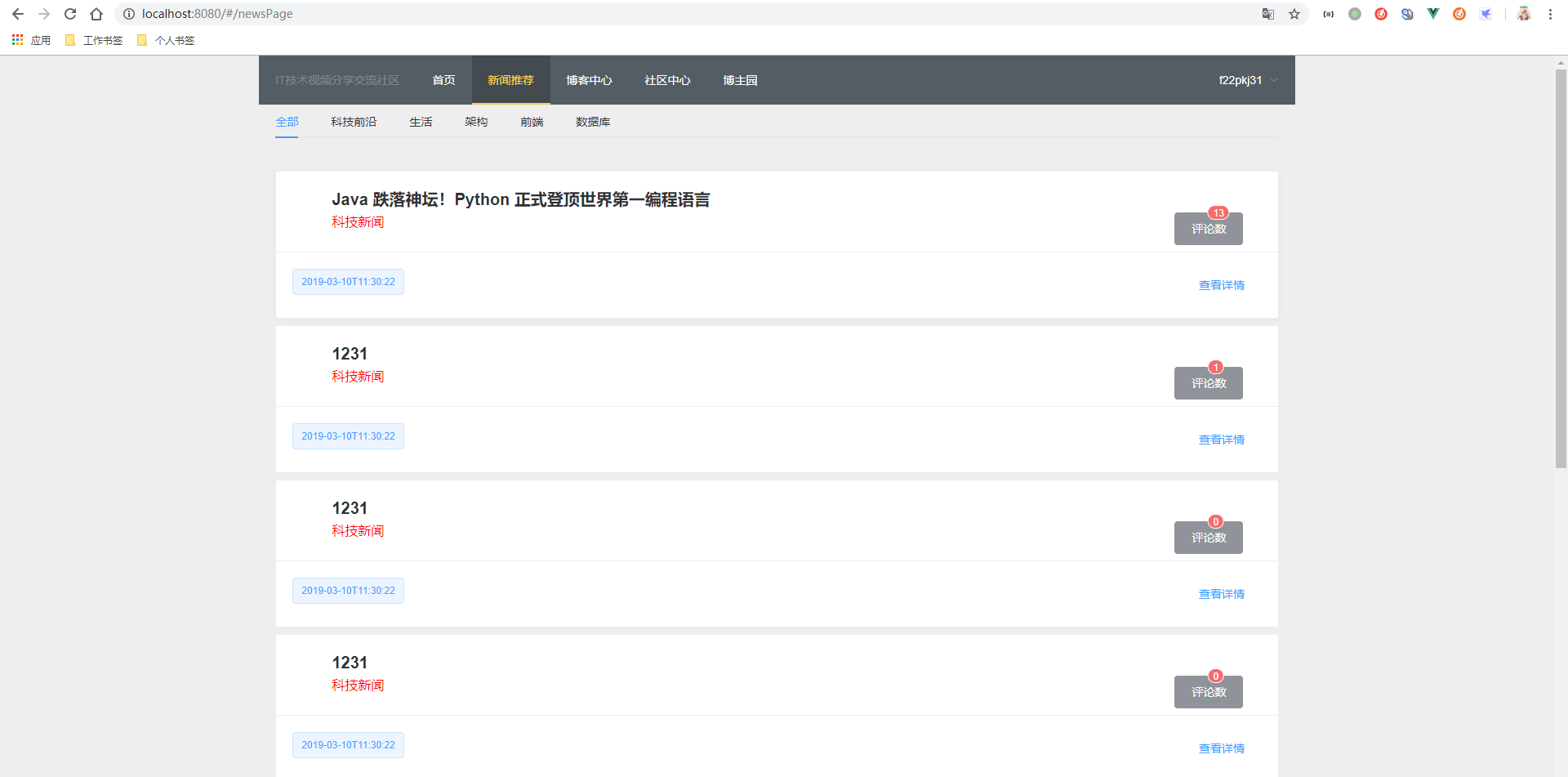


图4.2 新闻列表展示图

用户可以点击任意新闻进入新闻详情页，功能演示如图4.3：



图4.3 新闻详情展示图

已登录用户可以在新闻详情页下对新闻进行评论，如图4.4

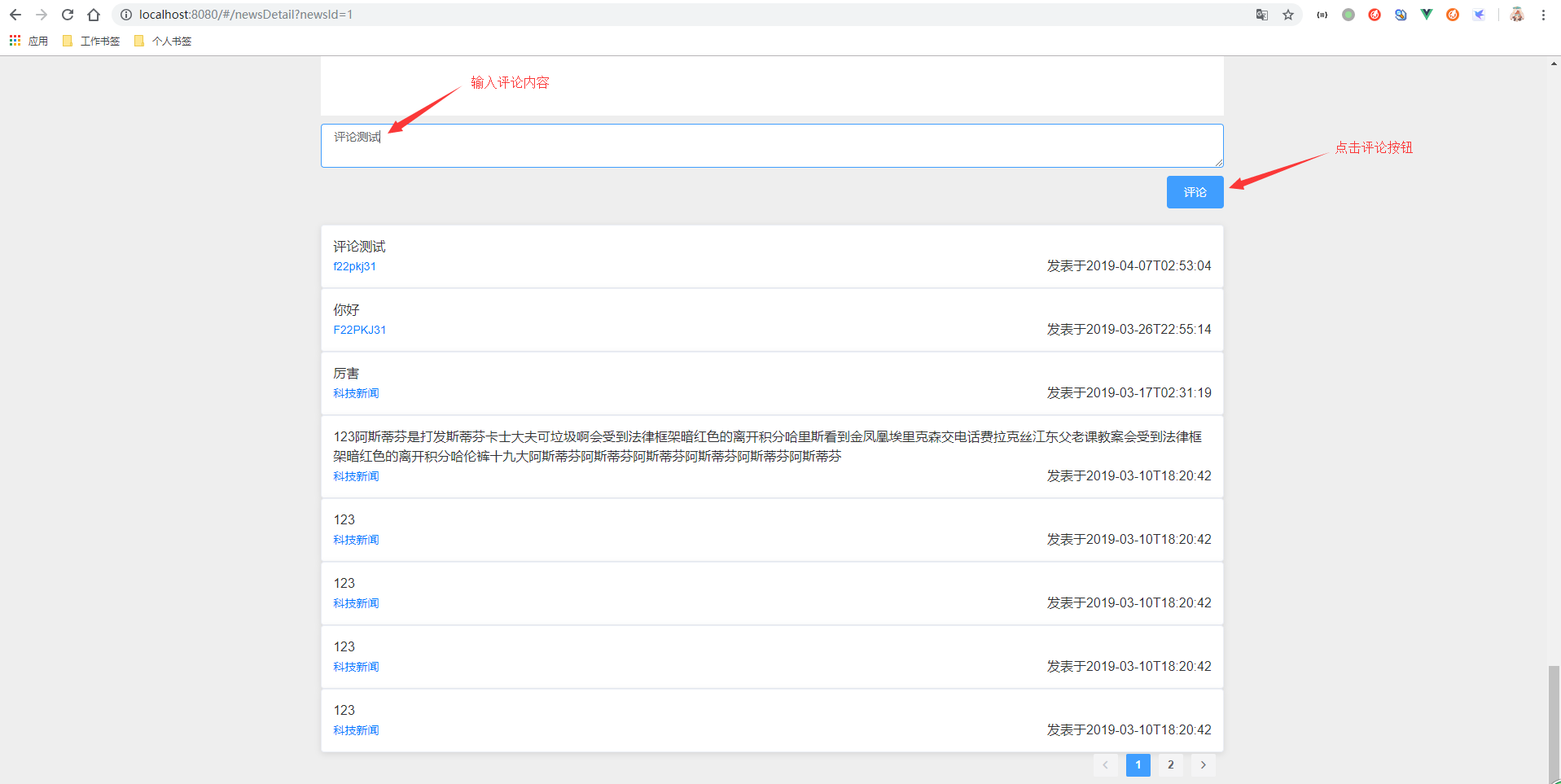


图4.4 新闻评论展示图

收藏和评论的关键后端代码如下：

@RequestMapping("saveCollection")

public Object sendCollection(@RequestBody NewsCollection newsCollection) {

return newsClientService.sendCollection(newsCollection.setCreateTime(LocalDateTime.now()));

}

@RequestMapping("sendComment")

public Object sendComment(@RequestBody NewsComment newsComment) {

CommonId commonId = new CommonId();

commonId.setId(newsComment.getNewsId());

newsClientService.addReadCount(commonId);

return newsClientService.sendComment(newsComment.setCreateTime(LocalDateTime.now()));

}

@RequestMapping("saveCollection")

public Object sendCollection(@RequestBody NewsCollection newsCollection) {

return newsCollectionService.save(newsCollection);

}

@RequestMapping("sendComment")

public Object sendComment(@RequestBody NewsComment newsComment) {

return newsCommentService.save(newsComment);

}

### 4.1.3 博客模块

用户可以在博客中心页查看博客，在列表页可以看到博客的标题、发布人、创建时间和评论数，功能演示如图4.5

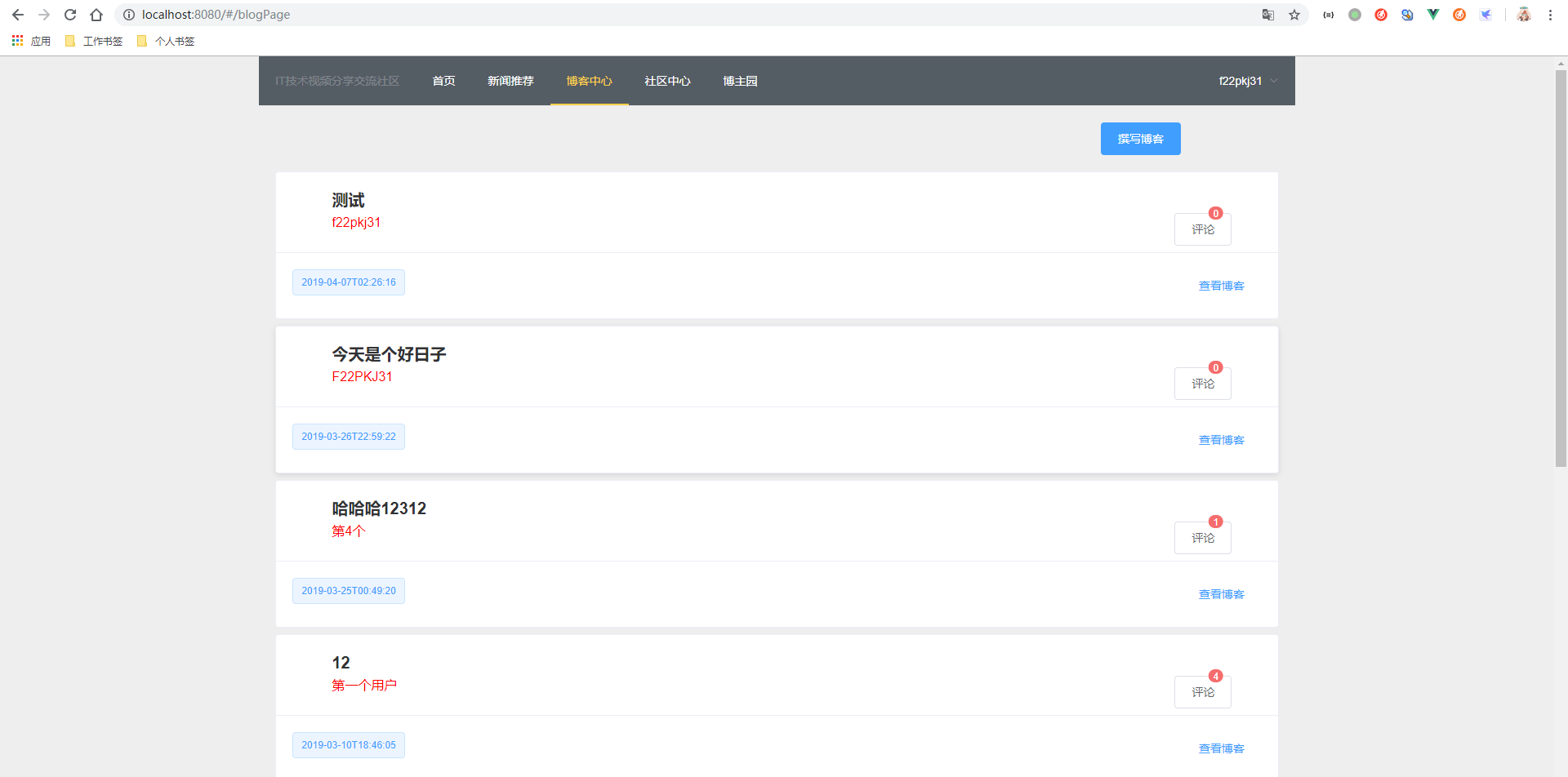


图4.5 博客列表展示图

博客模块的详情、收藏、评论功能与新闻相似。展示图如图4.6

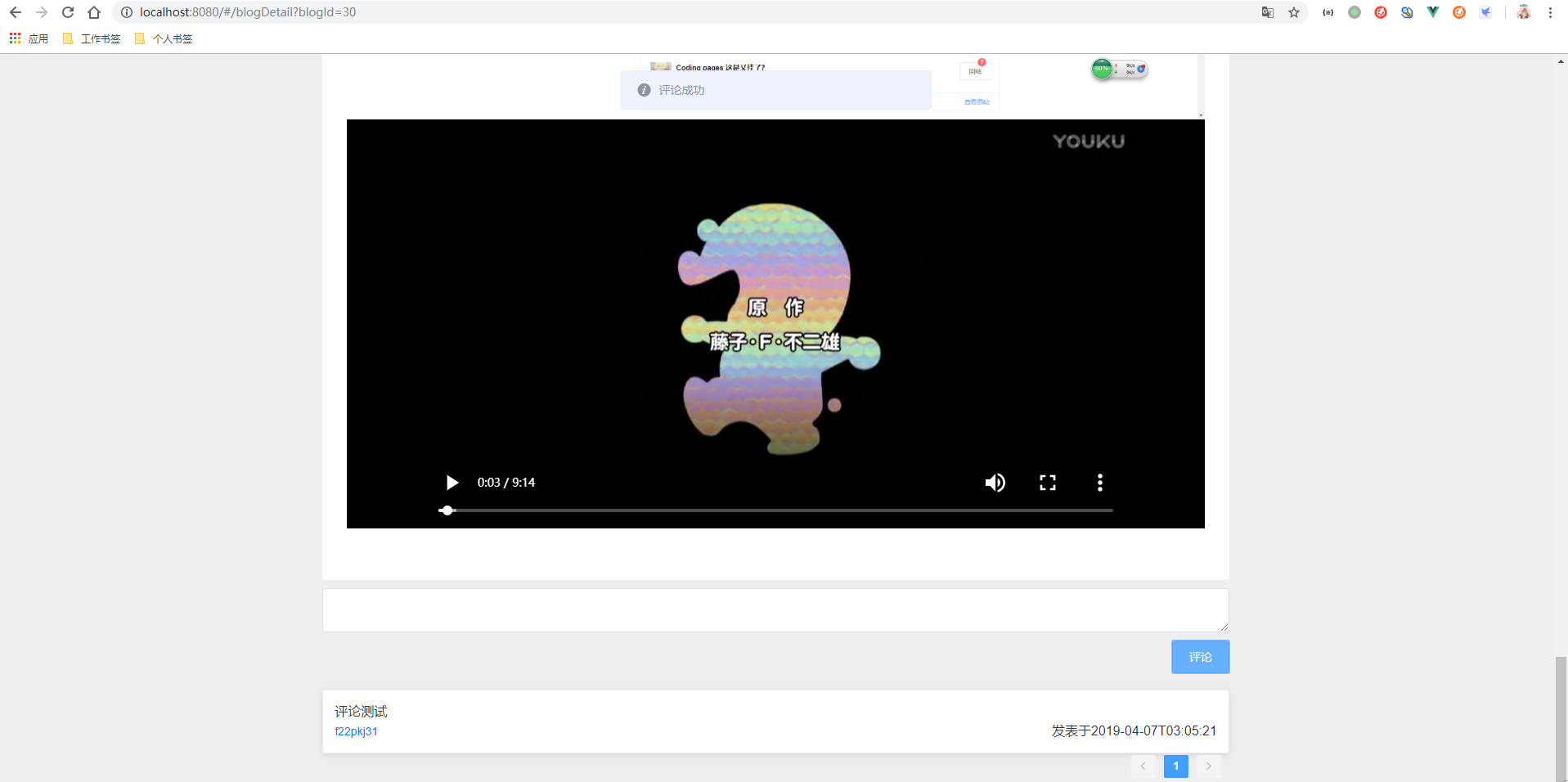


图4.6 博客评论展示图

关键后端代码如下：

@RequestMapping("blogListOrderByRead")

public Object blogListOrderByRead(@RequestBody PageIn<Blog> pageIn){

return blogClientService.blogListOrderByRead(pageIn);

}

@RequestMapping("saveCollection")

public Object saveCollection(@RequestBody BlogCollection blogCollection) {

return blogClientService.sendCollection(blogCollection.setCreateTime(LocalDateTime.now()));

}

@RequestMapping("sendComment")

public Object sendComment(@RequestBody BlogComment blogComment) {

CommonId commonId = new CommonId();

commonId.setId(blogComment.getBlogId());

blogClientService.addReadCount(commonId);

return blogClientService.sendComment(blogComment.setCreateTime(LocalDateTime.now()));

}

### 4.1.4 个人中心模块

用户可以在个人中心页查看自己的个人信息，包括头像、用户名、你年龄、性别、手机、邮箱，并可以对信息进行修改。功能演示如图4.7

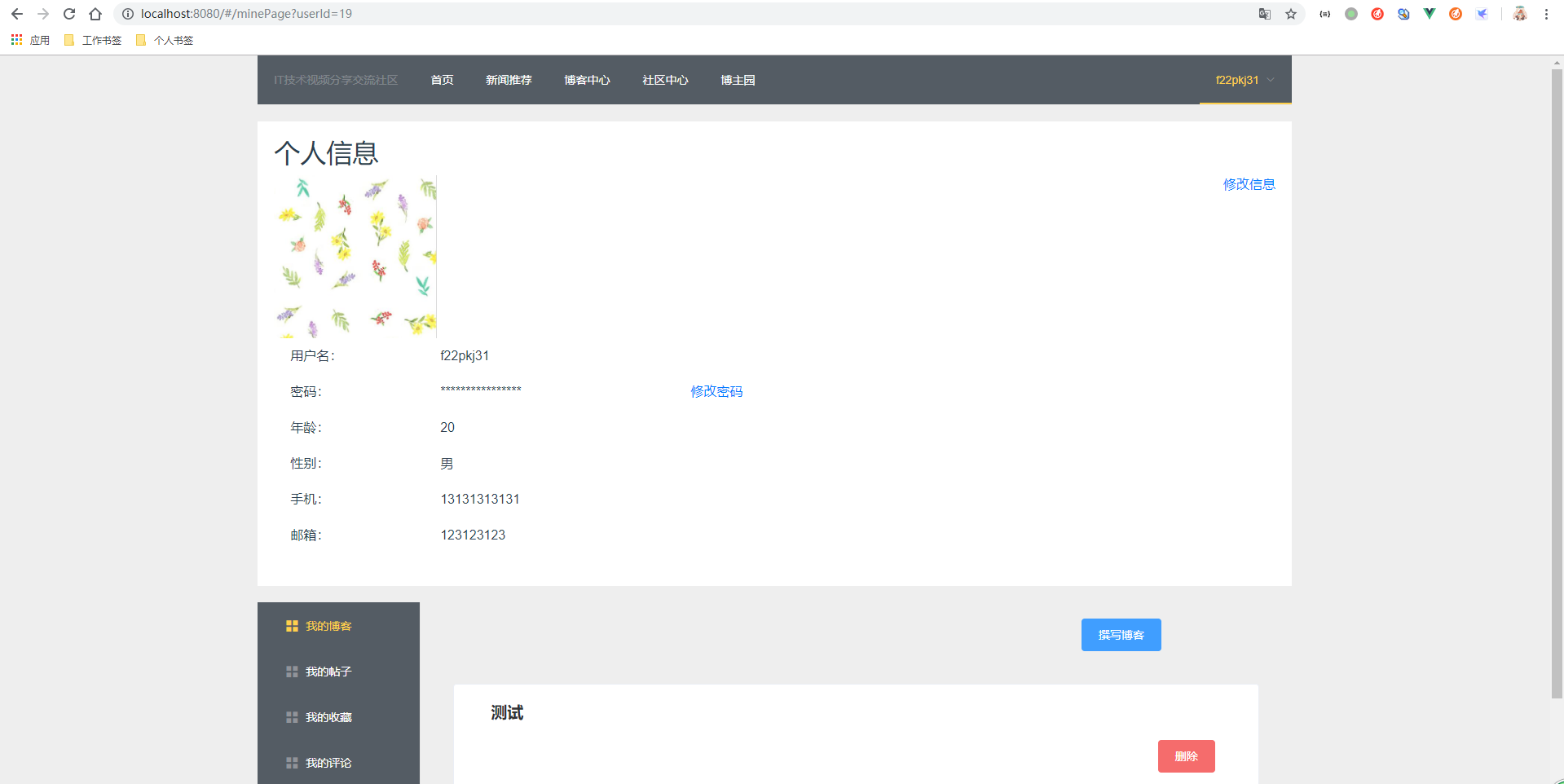


图4.7 个人中心展示图

关键代码如下：

<el-header style="height: auto; margin:20px 0; padding: 20px; background-color: white">

<h2>个人信息</h2>

<a @click="editInfo = true;userForm = JSON.parse(JSON.stringify(user))" href="javascript:void(0)"

style="float: right;"

v-if="isOwner">修改信息</a>

<div>

<img :src="user.imgUrl" class="head-img" style="float: left;"/>

<table style="border: none;margin: 20px;width: 100%;">

<tr>

<td width="15%">用户名：</td>

<td width="25%">{{user.userName}}</td>

</tr>

<tr v-if="isOwner">

<td>密码：</td>

<td>\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*</td>

<td><a @click="editPassword = true"

href="javascript:void(0)">修改密码</a></td>

</tr>

<tr> <td>年龄：</td>

<td>{{user.age}}</td> </tr>

<tr>

<td>性别：</td>

<td>{{user.sex}}</td>

</tr>

<tr>

<td>手机：</td>

<td>{{user.mobile}}</td>

</tr>

<tr>

<td>邮箱：</td>

<td>{{user.email}}</td>

</tr>

</table>

</div>

</el-header>

用户可以查看自己发的博客、帖子、收藏、评论，通过该模块，用户可以管理自己发表、收藏、评论的信息。效果展示图如图4.8：

用户撰写的所有博客展示效果如下：

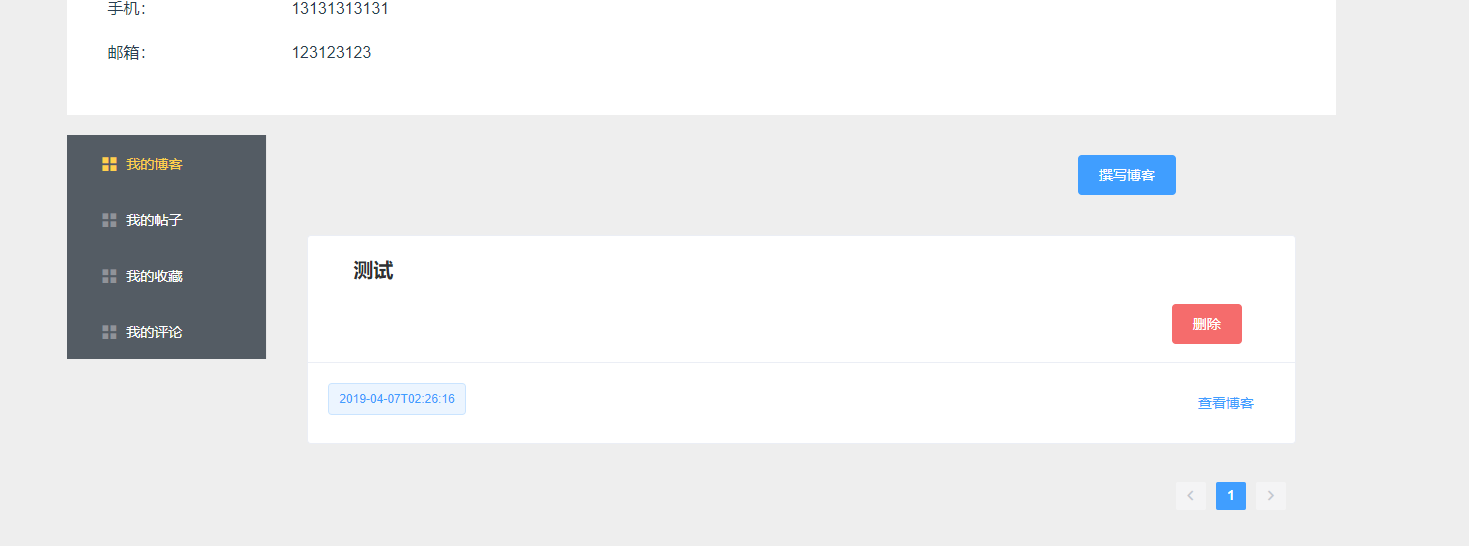


图4.8我的博客展示图

用户发布的所有帖子展示效果如图4.9：

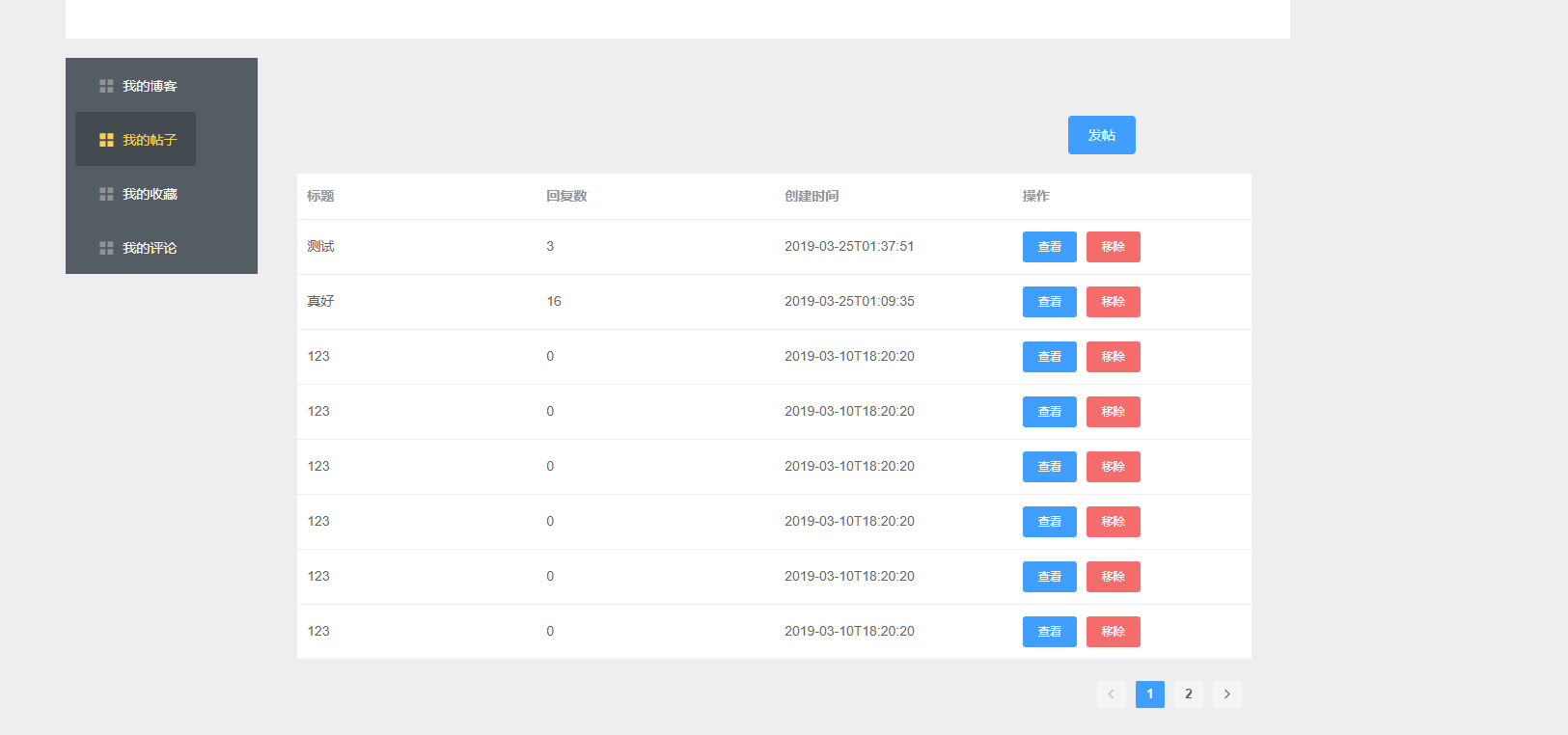


图4.9我的帖子展示图

用户的所有收藏展示效果如图4.10：



图4.10我的收藏展示图

用户的所有评论展示效果如图4.11：

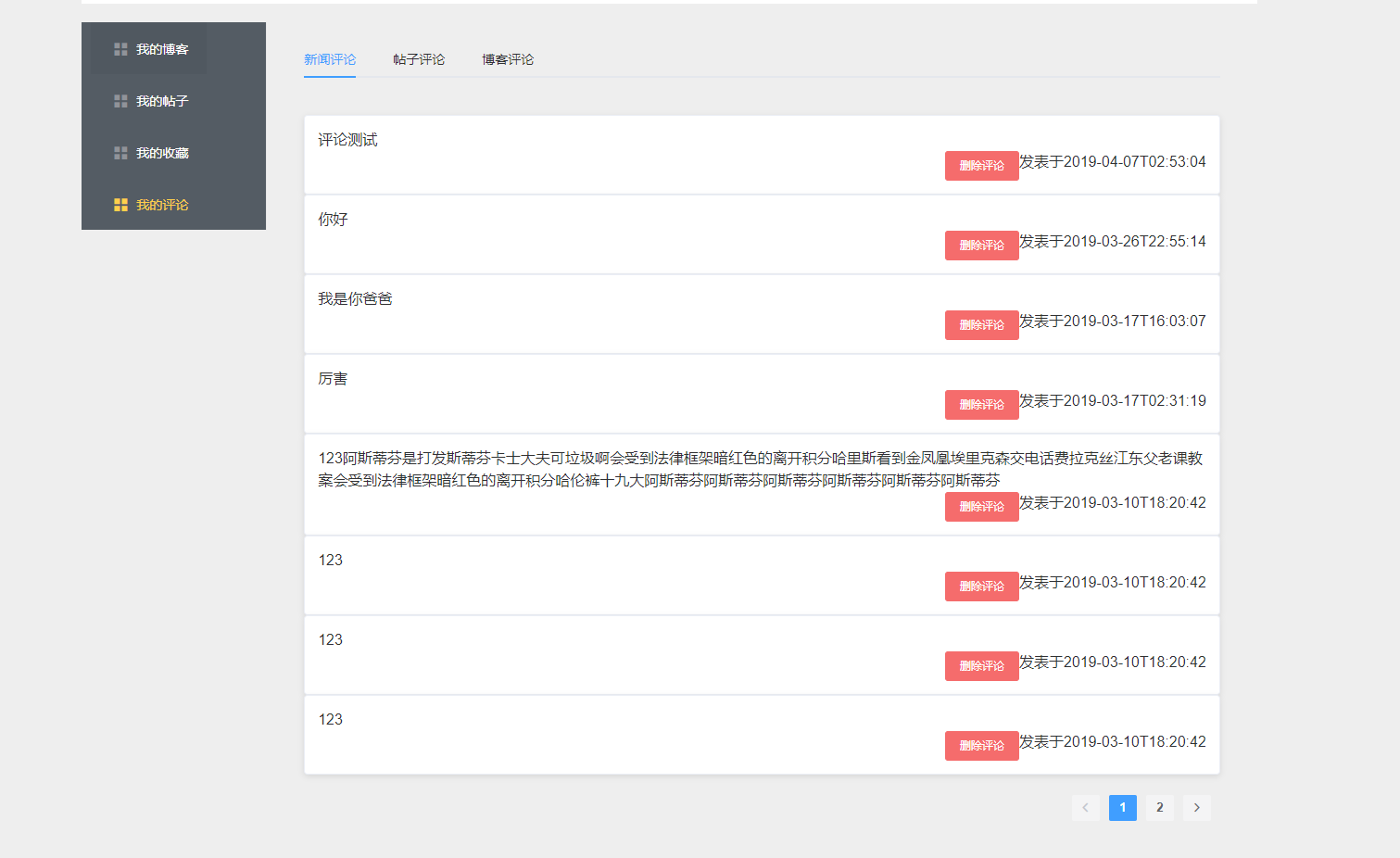


图4.11我的评论展示图

关键前端代码如下：

<el-main class="main">

用户收藏模块：

<v-my-collection :user="user" ref="myCollection" v-if="showCollection"></v-my-collection>

用户帖子模块：

<v-my-post :user="user" ref="myPost" v-if="showPost"></v-my-post>

用户博客模块：

<v-my-blog :user="user" ref="myBlog" v-if="showBlog"></v-my-blog>

用户评论模块：

<v-my-comment :user="user" ref="myComment" v-if="showComment"></v-my-comment>

</el-main>

## 4.2 关键技术难点与解决方案

本系统关键难点在于，在富文本中插入图片和视频。富文本使用了quill-editor，只支持上传图片和视频地址，系统中需要上传你本地图片和视频并插入到主体内容中。

我的解决方案为劫持插入图片和视频的方法，先弹出上传框，获取上传图片后，向后台发送请求上传图片，上传成功后通过获取quillEditor将图片插入到光标所在位置并将光标调到最后。

前端核心代码如下：

upImg() {

this.$axios.post('http://127.0.0.1:8010/file/uploadFile', param, config)

.then(response => {

that.$refs.imgfilereset.reset();//清除文件

console.log(response.data);

if (response.data.error === '0') {

// 获取光标所在位置

let quill = that.$refs.myTextEditor.quill;

let length = quill.getSelection().index;

var a = quill.insertEmbed(length, 'image', response.data.url);

console.log(a);

quill.setSelection(length + 1);

that.showFloat = false;

that.upimgShow = false

} else {

that.showFloat = false;

that.upimgShow = false;

this.$message('插入失败,请重试')

}

}).catch(function (error) {

this.$message(error);

that.showFloat = false;

that.upimgShow = false

})

}

# 第5章　系统测试

本系统的功能测试使用“W”测试模型，在开发的时候同步进行，功能开发完后对非功能测试，如兼容性测试等。以下是对核心功能测试、非功能测试进行简单说明。

## 5.1 功能测试

本系统主要对核心模块：内容编辑模块、个人中心模块进行测试，测试内容如下：

（1）内容编辑模块功能测试

内容编辑模块功能测试分为：图片插入、视频插入。测试用例如下：

表5.1图片插入用例

|  |  |  |  |
| --- | --- | --- | --- |
| **用例**  **编号** | **测试用例描述** | **操作过程及数据** | **预期结果** |
| addImg\_01 | 选择正确图片格式上传 | 点击上传图片，选择png格式的文件进行插入 | 系统提示上传成功并成功插入 |
| addImg \_02 | 选择其他格式文件上传 | 点击上传图片，选择txt格式的文件进行插入 | 系统提示：“请选择图片格式” |

表5.2 视频插入用例

|  |  |  |  |
| --- | --- | --- | --- |
| **用例**  **编号** | **测试用例描述** | **操作过程及数据** | **预期结果** |
| addVideo\_01 | 选择正确视频格式上传 | 点击上传视频，选择mp4格式的文件进行插入 | 系统提示添加成功 |
| addVideo\_02 | 选择其他格式文件上传 | 点击上传视频，选择json格式的文件进行插入 | 系统提示：“请选择视频格式” |

（2）个人中心模块功能测试

个人中心模块功能测试包括：修改密码、编辑信息。测试用例如下。

表5.3 修改密码用例

|  |  |  |  |
| --- | --- | --- | --- |
| **用例**  **编号** | **测试用例描述** | **操作过程及数据** | **预期结果** |
| updatePassword\_01 | 修改密码 | 点击修改密码，输入正确原密码，输入新密码并正确输入重复密码 | 系统提示修改成功 |
| updatePassword \_02 | 输入错误原密码 | 点击修改密码，输入错误原密码，输入新密码并正确输入重复密码 | 系统提示：“原密码错误” |
| updatePassword \_03 | 输入重复密码错误 | 点击修改密码，输入正确原密码，输入新密码并输入错误重复密码 | 系统提示：“两次输入密码不一致!” |

表5.4 编辑信息用例

|  |  |  |  |
| --- | --- | --- | --- |
| **用例**  **编号** | **测试用例描述** | **操作过程及数据** | **预期结果** |
| updateInfo\_01 | 正确填写信息 | 正确填写信息并提交 | 系统提示修改成功 |
| updateInfo \_02 | 年龄填写小于12 | 年龄填写10并提交 | 系统提示：“必须年满12岁” |
| updateInfo\_03 | 用户头像选择非图片格式文件 | 用户头像上传选择txt文件 | 系统提示：“上传头像图片只能是 JPG 或 PNG 格式!” |

## 5.2 非功能测试

表5.5兼容新测试用例

|  |  |
| --- | --- |
| **测试用例名称** | **浏览器版本** |
| 兼容性\_001 | Google Chrome 72.0.3626.121 |
| 兼容性\_002 | Firefox 9 |
| 兼容性\_003 | Microsoft Edge 44.17763.1.0 |
| 兼容性\_004 | IE 11 |

## 5.3 测试总结

软件测试是为了发现错误而执行程序的过程。目的是为了在投入生产运行之前，尽可能多地发现并排除软件中潜藏的错误，从而提高软件的质量。测试对于整个开发流程都是非常重要的。进行功能测试时采用“W”模型，边开发边测试， 这种测试模型将测试贯穿到整个软件的生命周期中，切除了代码要测试，需求、设计等都要测试，同时能够更早的接入到开发中，尽早发现问题所在并进行修复。

# 第6章　结论与展望

经过几个月时间的努力，同时在网上翻阅了大量资料，最终完成了IT视频分享交流社区的开发与设计。作为社区类的系统，相对于主流的CSDN、博客园、简书等，有一定的差距，但在功能上已经完成度比较高了，基本能够满足用户的要求。本系统有一下几个亮点：

（1）整个系统采用前后端分离的开发方式，后端采用Spring Cloud微服务架构，配合Nacos作为微服务的服务发现，我将后端前端采用Vue.js + Element UI，后台提供完整数据，前端负责展示。这样的架构设计，可扩展性很强，并且具有很高的可用性。

（2）本系统使用quill-editor作为富文本编辑器，并对该编辑器做了一定的重构，修改了上传图片和视频的功能，利用quill-editor插入上传的图片或视频。

除了这几个亮点，本系统还有很多的问题和不足：

（1）本系统在架构上虽然使用了Spring Cloud，但其潜力还没有被开发出来，只是使用了简单的服务发现功能。如果以后对系统进行改进，可以加入熔断器、动态路由和权限控制组件。

（2）系统中身份验证较为简陋，只在前端通过LocalStorage保存用户信息，在以后对系统的改进上，我希望能够加入Spring的Security模块来进行权限控制，还可以用来区分管理员和普通用户并分配权限。

（3）前端Vue代码编写逻辑较为混乱。由于Vue的编写经验不足，导致代码逻辑不清晰，代码耦合度很高，模块层次不清晰，模块之间数据传递不安全且单一。在这方面我会继续学习前端知识，争取早日重构前端代码。

本次课题是我在微服务和前后端分离开发的一次尝试。就技术风向而言，微服务和前后端分离一定是大势所趋，包括现在有很多的公司已经在使用各类微服务框架，前后端分离技术也已经很成熟了。因此，我希望本次毕业设计能够做出完成度比较高的项目。

总之，在这次毕业设计期间，我有了很多的收获，学到了很多技术，做了很多的代码练习。希望在以后的工作和学习中能够继续努力。

# 参考文献

[1]． 王方旭. 基于SpringCloud实现业务系统微服务化的设计与实现[J]. 电子技术与软件工程, 2018(8).

[2]． 邱生姬. 浅谈JAVA微服务SpringCloud开发[J]. 电脑迷, 2017(17).

[3]． 张峰. 微服务技术构建大规模web系统的研究[J]. 科技创新与应用, 2017(22):48-49.

[4]． 王永和, 张劲松, 邓安明,等. Spring Boot研究和应用[J]. 信息通信, 2016(10):91-94.

[5]． 周燕玲. Spring MVC框架开发WEB应用程序的探索与研究[J]. 科技广场, 2016(6):25-28.

[6]． BrianGoetz. JAVA并发编程实践[M]. 电子工业出版社, 2007.

[7]． 沃尔斯. Spring实战:第4版[M]. 人民邮电出版社, 2016.

[8]． 杨开振. 深入浅出MyBatis技术原理与实战[M]. 电子工业出版社, 2016.

[9]． 马志强, 张然, 李雷孝. Java核心技术[J]. 计算机教育, 2015(21).

[10]． 汪云飞. JavaEE开发的颠覆者:Spring Boot实战[M]. 电子工业出版社, 2016.

# 致 谢

从论文开题，到最终的提交论文，一共经历了大概半年的时间，在这半年里，我学会了很多知识，其中包括技术方面，学会了很多新的框架，接触到很多好用的、易用的工具。在整个流程开发中，我将大学中学到的软件工程知识运用到了项目中。这些让我觉得大学四年的学习没有白费。

整个毕业设计从选题到最终的开发、论文完成，获得了很多人的帮助。其中，我的导师彭志豪在我的选题和论文阶段给予我的帮助最大，在我确定了开发社区的方向时，导师向我提出意见，希望我能够对比各类社区网站，做一些自己的创新功能，因此我在社区中加入了视频元素，也得到了导师的赞同。最终撰写论文阶段，导师指出我论文中的各种问题，并建议我在论文中突出核心模块。其次，在开发的过程中，我实习期间的导师谢小菁给了我技术上的支持。在实习期间，我与导师探讨毕业设计使用的技术框架时，导师建议我挑战自己，使用当前阶段比较新的、热门的技术。导师还给了我许多的学习资源，让我去学习相关知识，能够快速的上手，完成项目的开发。

最后，还要感谢我的父母，一直在背后默默地支持我，感谢我的室友，在我困难的时候及时伸出援手，感谢辅导员张思柔老师，给我们毕业年级发布各种招聘简章，能够让我们顺利毕业。在我的大学四年里，有你们的帮助和支持，我在能顺利的走到现在。衷心的谢谢你们。

希望大连东软信息学院能够越办越好，从东软走出来的都是IT行业的精英。

大连东软信息学院

毕业设计（论文）原创承诺书

1、本人承诺：所提交的毕业设计（论文）是认真学习理解学校的《毕业设计（论文）工作规范》后，在教师的指导下，独立地完成了任务书中规定的内容，不弄虚作假，不抄袭别人的工作内容。

2、本人在毕业设计（论文）中引用他人的观点和研究成果，均在文中加以注释或以参考文献形式列出，对本文的研究工作做出重要贡献的个人和集体均已在文中注明。

3、在毕业设计（论文）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

4、本人完全了解学校关于保存、使用毕业设计（论文）的规定，即：按照学校要求提交论文和相关材料的印刷本和电子版本；同意学校保留毕业设计（论文）的复印件和电子版本，允许被查阅和借阅；学校可以采用影印、缩印或其他复制手段保存毕业设计（论文），可以公布其中的全部或部分内容。

5、本人完全了解《毕业（设计）论文工作规范》关于“学生毕业设计（论文）出现购买、他人代写、或者抄袭、剽窃等作假情形的，取消其学位申请资格；已经获得学位的，依法撤销其学位。取消学位申请资格或者撤销学位者，从处理决定之日起3年内，学校不再接受学生学位申请”的规定内容。

6、本人完全了解《学生手册》中关于在“毕业设计（论文）等环节中被认定抄袭他人成果者”不授予学士学位，并且“毕业学年因违纪受处分影响学位的学生不授予学士学位，并且无学士学位申请资格”的规定内容。

以上承诺的法律结果、不能正常毕业及其他不可预见的后果由学生本人承担！

学生本人签字：

年 月 日