学生编号 42

2021-2022 学年秋季学期

《计算思维实训（2）》(0830A030)

课程报告

成绩 (百分制)	

学　　号	20121034	学　　院	计算机学院
姓　　名	胡才郁	手工签名	胡才郁
报告题目	基于 Spring Boot 与 Vue.js 框架的个人博客		
实训报告成绩 50%	实训过程描述：清晰、全面。（5%）		
	报告主要部分：1、计算思维实例叙述清楚、有意义；2、分析到位；3、思路独特或有创意；4、关键实现技术描述清楚详细；5、内容丰富；6、不直接粘贴所有代码；7、代码有注释或说明。（25%）		
	实训收获体会：感受真实、深刻，建议有价值。（10%）		
	书写格式：书写规范、用词正确、无明显的错别字，图表、代码清晰规范，格式协调、效果好，参考文献书写、引用规范合理。（10%）		
工作实绩 50%	1、平时表现、进步情况（25%），2、工作实绩（25%）		

教师对该生工作实绩简要评述：

教师签名：

日期：　　年　　月　　日

Part 1. 本学期实训过程概述

1. 实训总体概况、实训过程

通过本学期的实训学习，我与张俊雄组成学习小组，共同开发了一个前后端分离的个人博客。在整个开发流程之中，我负责后端的开发，张俊雄负责前端开发。此个人博客提供了一个博客基本的功能，例如发表文章、文章展示、最热标签等功能。

在开发过程中，我与负责前端的同学使用 git 等版本控制工具同步开发进度，分工协作，从无到有开发出一个成型的项目。

实训结果总体而言锻炼了我代码能力与使用开发工具的工程能力，实现结果良好。

2. 对计算思维的认识

计算思维是一种选择合适的方式去陈述一个问题，或对一个问题的相关方面建模使其易于处理的思维方法。通过不断的实践，反复修改现有的程序，往既定的目标实现的一个过程。同时也是属于一种在自身不断实践中通过良好规划，并且充分使用推理的过程实现的一个方法。是按照预防、保护及通过冗余、容错、纠错的方式，并从最坏情况进行系统恢复的一种思维方法;是利用启发式推理寻求解答，也即在不确定情况下的规划、学习和调度的思维方法。

3. 实训体会及建议

整个学习过程中大多数时间都是看书、看网课、博客等方法自学，从查找资料到搭建环境、再到进一步阅读官方文档，运用框架实现各种操作，这个过程中待解决的问题有很多。一步一步建立开发的系统架构，并且根据自己的想法进行开发，并且成功实现，是一个非常令人满足的事，这也进一步提高了自己的自学能力与动手实践能力。

Part 2. 综合实训报告

基于 Spring Boot 与 Vue.js 框架的个人博客

20121034 胡才郁

(计算机工程与科学学院)

摘　要　前后端分离是目前主流的开发模式，本项目使用 Spring Boot 与 Vue.js 框架进行前后端分离开发，严格遵守 Restful 开发规范，后台采用 MVC 架构，完成了个人博客的开发

关键词　前后端分离, Spring Boot, Vue.js, MVC 架构, Restful 开发规范

1 引言

本次实训主要通过采用前后端分离开发模式,使用 Spring Boot+Vue.js 的技术栈，实现了一个小型博客的基础功能。前端采取 Vue.js 框架，Vue.js 是一个用于创建用户界面的开源 Model–view–view model 前端 JavaScript 框架，其核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。UI 采取 Element UI(饿了么),其美观与高复用性的组件库简化了开发流程。后端采取 Spring Boot 框架，是当前主流的后端开发框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程，不仅继承了 Spring 框架原有的优秀特性，而且还通过简化配置来进一步简化了 Spring 应用的整个搭建和开发过程。

2 开发工具与框架

2.1 Vue.js

随着前端技术的不断发展，前端开发能够处理的业务越来越多，网页也变得越来越强大和动态化，这些进步都离不开 JavaScript。在目前的开发中，已经把很多服务端的代码放到了浏览器中来执行，这就产生了成千上万行的 JavaScript 代码，他们连接着各式各样的 HTML 和 CSS 文件，但是缺乏正规的组织形式。这也是为什么越来越多的前端开发者使用 JavaScript 框架的原因，目前比较流行的前端框架有 Angular、React、Vue 等。

本项目基于 Vue.js 进行开发。Vue.js 是一款友好的、多用途且高性能的 JavaScript 框架，它能够帮助开发者创建可维护性和可测试性更强的代码库。Vue 是渐进式的 JavaScript 框架，也就是说，如果开发者已经有了现成的服务端应用，也可以将 Vue 作为该应用的一部分嵌入其中，带来更加丰富的交互体验。和其他框架一样，Vue.js 允许将一个网页分割成可复用的组件，每个组件都包含属于自己的 HTML、CSS、JavaScript，以用来渲染网页中相应的地方。如果我们构建了一个大型的应用，可能需要将东西分割成为各自的组件和文件，使用 Vue 的命令行工具，使快速初始化一个真实的工程变得非常简单。

2.2 Element-UI

Element-UI 是一套为开发者、设计师和产品经理准备的基于 Vue 2.0 的桌面端组件库使用组件 Demo 快速体验交互细节；使用前端框架封装的代码帮助工程师快速开发。在线主题编辑器，可视化定制和管理站点主题、组件样式

2.3 Spring Boot

在使用传统的 Spring 去做 Java EE（Java Enterprise Edition）开发中，大量的 XML 文件存在于项目之中，导致 JavaEE 项目变得慢慢笨重起来，繁琐的配置和整合第三方框架的配置，导致了开发和部署效率的降低。

Spring Boot 并不是用来替代 Spring 的解决方案，而是和 Spring 框架紧密结合用于提升 Spring 开发者体验的工具。同时它集成了大量常用的第三方库配置，Spring Boot 应用中这些第三方库几乎可以是零配置的开箱即用（out-of-the-box），大部分的 Spring Boot 应用都只需要非常少量的配置代码（基于 Java 的配置），开发者能够更加专注于业务逻辑。另外 SpringBoot 通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。

2.4 Swagger

无论对于前端还是后端开发，都需要详尽接口文档。前端经常遇到后端给的接口文档与实际情况不一致。而对于后端编写及维护接口文档会耗费不少精力，文档也难以做到实时更行。随着时间推移，项目版本迭代，接口文档往往跟不上代码了。

Swagger 是一个接口文档生成工具，提供了生成多种语言的客户端和服务端的代码，以及在线接口调试页面等等。按照新的开发模式，在进行代码版本迭代时候，只需要更新 Swagger 描述文件，就可以自动生成接口文档和客户端服务端代码，做到调用端代码、服务端代码以及接口文档的一致性。

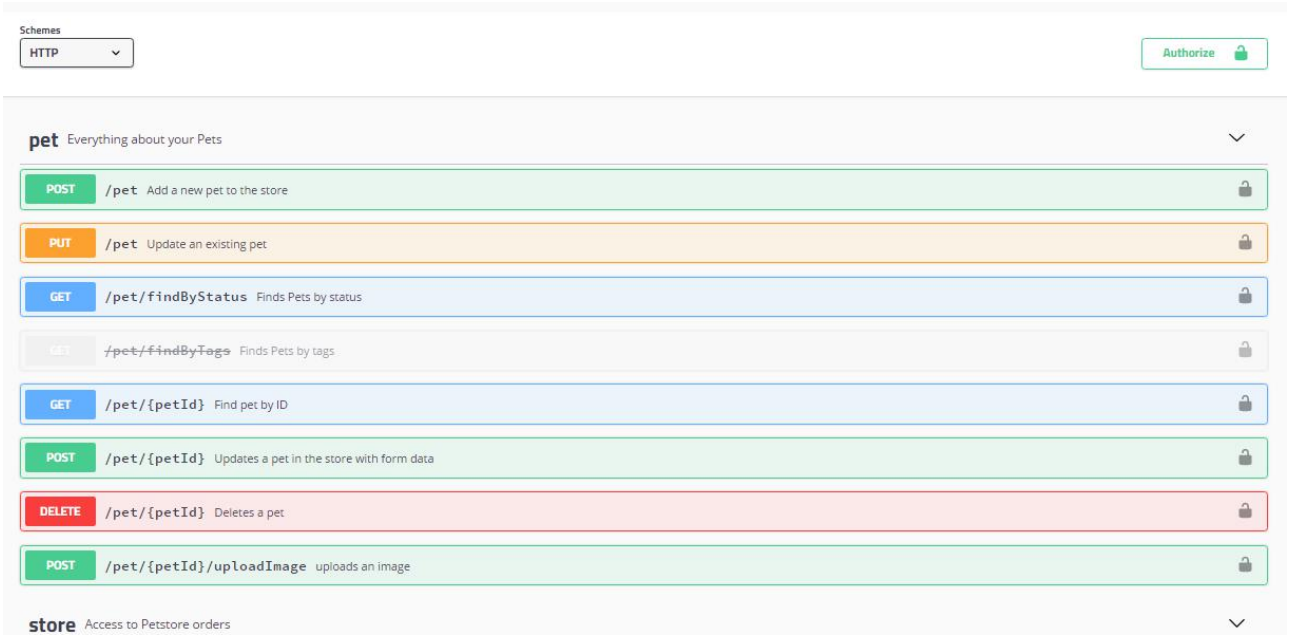

图 1. Swagger 生成的接口文档图示

2.5 Git & GitHub

Git 是用于 Linux 内核开发的版本控制工具。与 CVS、Subversion 一类的集中式版本控制工具不同，它采用了分布式版本库的作法，不需要服务器端软件，就可以运作版本控制，使得源代码的发布和交流极其方便。

而 Github 是一款基于 Git 的远程代码仓库管理工具，本项目在前端开发与后端开发进行联调时，通过访问 Github 上的远程仓库，做到了版本控制，减少了前后端之间沟通的时间。

2.6 前后端分离架构

在较为过时的前后端未分离架构模式之中，在前端人员完成 js、css、html 的实现之后，还需要后端开发人员直接兼顾前端的工作，将前端人员提供的页面改写为 JSP 页面，一边实现 API 接口，一边开发页面，根据不同的 URL 动态拼接页面，这也导致后台的开发压力大大增加。前后端工作分配不均。存在着前端无法单独调试、JSP 等程序第一次运行响应时间较慢、开发效率低、项目维护成本较高等缺点。

而前后端分离架构是目前开发中的主要开发模式，他将前后端的工作进行了明确的划分。前端关注界面展现，后端关注业务逻辑，分工明确，职责清晰。

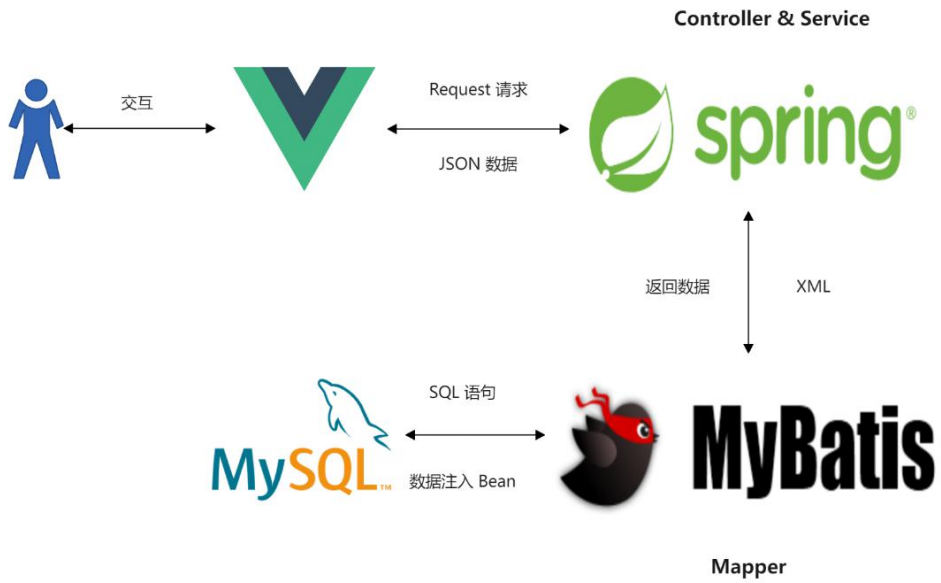

图 2. 前后端分离架构图示

3 项目构建

3.1 项目简介

3.1.1 架构简介

本次实训主要通过采用前后端分离开发模式,使用 Spring Boot+Vue.js 的技术栈，实现了一个小型博客的基础功能。严格遵守 MVC 开发模式，将整个系统划分为 View 层，Controller 层，Service 层，DAO 层

四层，使用 Spring MVC 负责请求的转发和视图管理，Spring 实现业务对象管理，Mybatis 作为数据对象的持久化引擎，并通过配置文件实现管理。此次的 web 项目由分工合作完成，对于协同开发而言，控制代码版本以及前后端沟通接口参数十分重要。因此，本项目使用 git 进行版本控制，并使用 github 完成代码远程同步。对于协同开发而言，前后端联调更是重要的一环，前端部分使用 mock 这款 API 接口开发调试工具，使得前端部分无需等待后端的接口，后端部分使用 postman，使得后端部分无需等待前端展示即可测试接口可用性。

1. 前端采取 Vue.js 框架，其核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。

2. UI 采取 Element UI(饿了么),其美观与高复用性的组件库简化了开发流程。

3. 后端采取 Spring Boot 框架，是当前主流的后端开发框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程，不仅继承了 Spring 框架原有的优秀特性，而且还通过简化配置来进一步简化了 Spring 应用的整个搭建和开发过程。另外 SpringBoot 通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。

3.1.2 功能简介

本个人博客实现的基础功能有：发表文章、文章列表、热门文章、最新文章、标签分类、评论区、要想完成这些功能，需要先创建对应数据表（tab_article，tab_favorite，tab_tags）等并存入信息，本项目采用了 mysql5.7 版本。其次，前端页面使用 Vue.js 搭建，本项目使用了部分 Element UI(饿了么)的 UI 组件。再者，在某些时刻，还需要通过 ajax 请求返回 json 数据来达到目的。

3.2 项目简介

3.2.1 Vue 项目搭建

Vue 的搭建依赖于 node.js 框架，所以在安装好 node.js 后，在需要创建工程文件的地方使用 cmd 运行命令：vue create blog 即可生成 Vue 的 blog 工程文件。

在工程文件创建完毕之后，使用 cmd 命令行工具运行 npm run serve 即可启动 Vue 工程，启动后的效果如下：

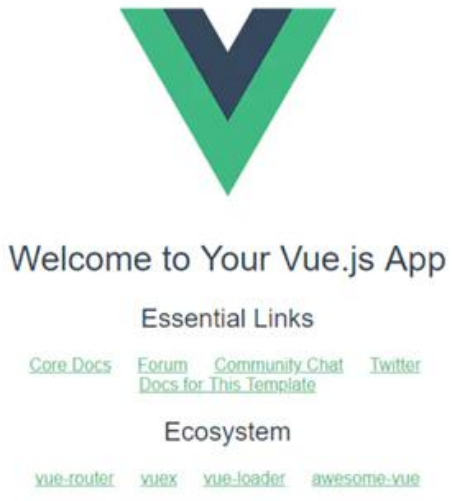

图 3. Vue 启动界面图示

3.2.2 SpringBoot 项目搭建

在 IDEA 中新建 maven 工程，导入 SpringBoot， Mybatis-Plus 相关依赖

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-starter</artifactId>
    <version>3.4.3</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
```

图 4. Maven 依赖坐标

```yaml
server:
  port: 8888
  servlet:
    context-path: /myblog
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/blog
    username: root
    password: 12345
    driver-class-name: com.mysql.cj.jdbc.Driver
  redis:
    host: localhost
    port: 6379

mybatis-plus:
  configuration:
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
    map-underscore-to-camel-case: true
  global-config:
    db-config:
      table-prefix: ms_
  mapper-locations: classpath*:mapper/*.xml

```

图 5. application.yml 配置文件

3.3 功能简介

本项目实现了个人博客应有的基本功能，限于篇幅，本处选取几个功能进行介绍，详细功能实现在项目演示时完成。

图 6. 博客首页效果图

3.2.1 首页文章展示

在登录进入网站首页之后，会显示出已经发表过的文章，效果图如下：

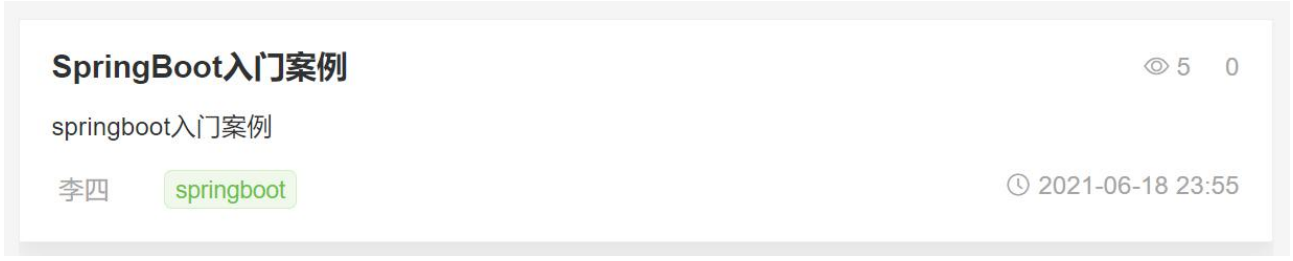

图 7. 首页文章效果图

前端实现如下：

1. Api 创建

```
export function getArticles(query, page) {
  return request({
    url: '/articles', // 后端接口
    method: 'post', // 请求方法
    data: { // 数据参数
      page: page.pageNumber,
      pageSize: page.pageSize,
      name: page.name,
      sort: page.sort,
      year: query.year,
      month: query.month,
      tagId: query.tagId,
      categoryId: query.categoryId
    }
  })
}

```

图 8. Api 创建

2. 新建 ArticleItem.vue 文件，使用饿了么 el-card 组件标签

```
<el-card class="me-area" :body-style="{ padding: '16px' }">
    <div class="me-article-header">
        <a @click="view(id)" class="me-article-title">{{title}}</a>
        <el-button v-if="weight > 0" class="me-article-icon" type="text">置顶</el-button>
        <span class="me-pull-right me-article-count">
            <i class="me-icon-comment"></i> {{commentCounts}}
        </span>
        <span class="me-pull-right me-article-count">
            <i class="el-icon-view"></i> {{viewCounts}}
        </span>
    </div>
    <div class="me-artile-description">
        {{summary}}
    </div>
    <div class="me-article-footer">
        <span class="me-article-author">
            <i class="me-icon-author"></i> {{author}}
        </span>
        <el-tag v-for="t in tags" :key="t.tagName" size="mini" type="success">{{t.tagName}}</el-tag>
        <span class="me-pull-right me-article-count">
            <i class="el-icon-time"></i> {{createDate | format}}
        </span>
    </div>
</el-card>

```

图 9. 饿了么 el-card 组件标签

3. 给 a 绑定方法：当文章标题被点击时自动进行路由跳转，文章详情页面

```
view(id) {
    this.$router.push({path: `/view/${id}`})
}
```

图 10. 路由跳转的绑定

4. 后端按照 controller、service、dao 进行分层，实现如下：

```java
// ArticleController
@PostMapping
@LogAnnotation(module = "文章", operation = "获取文章列表")
public Result listArticle(@RequestBody PageParams page){
    List<ArticleVo> articleVoList = articleService.getArticleList(page);
    return Result.success(articleVoList);
}

// ArticlService
@Override
public List<ArticleVo> listArticlesPage(PageParams pageParams) {
    QueryWrapper<Article> queryWrapper = new QueryWrapper<>();
    Page<Article> page = new Page<>(pageParams.getPage(),pageParams.getPageSize());
    Page<Article> articlePage = articleMapper.selectPage(page, queryWrapper);
    List<ArticleVo> articleVoList = copyList(articlePage.getRecords(),true,false,true);
    return articleVoList;
}

// ArticleMapper
public interface ArticleMapper extends BaseMapper<Article> {}
```

图 11. 首页文章列表后端实现

3.2.2 最热标签

对于每一篇博文而言，都有相应的标签，根据每篇文章标签数量的多少进行排序，并返回最多者，效果如下：

图 12. 最热文章效果图

前端实现如下：

```javascript
export function getHotTags() {
  return request({
    url: '/tags/hot/5',
    method: 'get',
  })
}
```

图 13. Api 创建

```
<el-card :body-style="{ padding: '8px 18px' }">
    <div slot="header" class="me-tag-header">
        <span>最热标签</span>
        <a @click="moreTags" class="me-pull-right me-tag-more">查看全部</a>
    </div>

    <ul class="me-tag-list">
        <li class="me-tag-item" v-for="t in tags" :key="t.id">
            <!--type="primary"-->
            <el-button @click="tag(t.id)" size="mini" type="primary" round plain>{{t.tagName}}</el-button>
        </li>
    </ul>
</el-card>
```

图 14. 饿了么 el-card 组件标签

```
moreTags() {
    this.$router.push('/tag/all')
}

tag(id) {
    this.$router.push({path: `/tag/${id}`})
}
```

图 15. 路由跳转的绑定

3.2.3 统一异常处理

不管是 controller 层还是 service，dao 层，都有可能报异常，如果是预料中的异常，可以直接捕获处理，如果是意料之外的异常，需要统一进行处理，进行记录，并给用户提示相对比较友好的信息。

/对加了@Controller 注解的方法进行拦截处理 AOP 的实现如下

```
@ControllerAdvice
public class AllExceptionHandler {
    //进行异常处理，处理Exception.class的异常
    @ExceptionHandler(Exception.class)
    @ResponseBody //返回json数据
    public Result doException(Exception ex){
        ex.printStackTrace();
        return Result.fail(-999,"系统异常");
    }
}
```

图 16. AOP 切片异常处理实现

3.2.3 文章发表

本博客实现了发表文章功能，并且支持 Markdown 语法，提供了舒适的写作环境，且可以在文章中插入本地图片，使用了七牛云的 cdn 图床服务进行图片的 URL 存储，效果图如下：

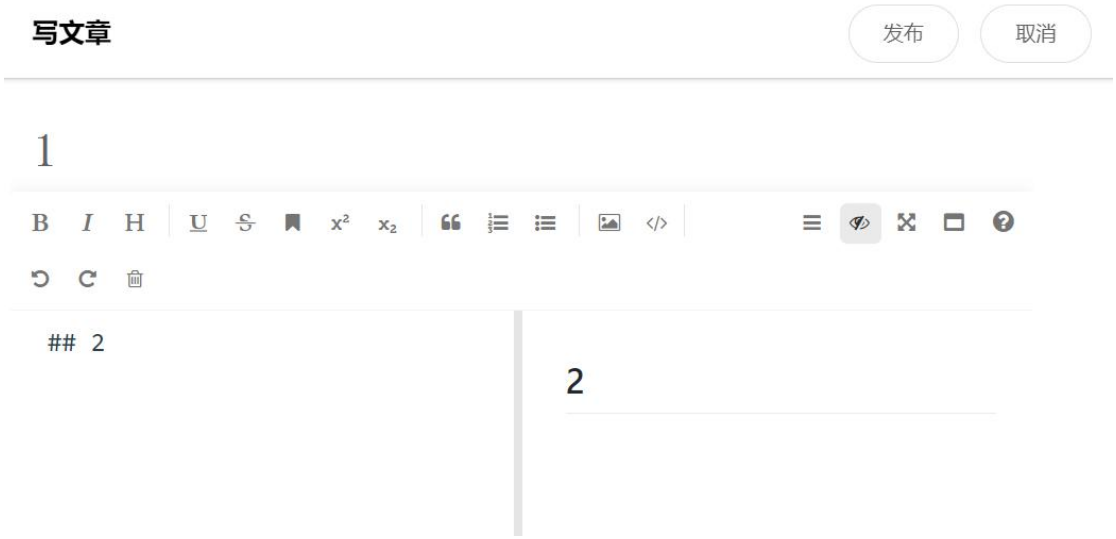

图 17. 发表文章效果图

```
export function publishArticle(article,token) {
  return request({
    headers: {'Authorization': token},
    url: '/articles/publish',
    method: 'post',
    data: article
  })
}
import MarkdownEditor from '@/components/markdown/MarkdownEditor'
```

图 18. Api 创建

```
<el-container class="me-area me-write-box">
    <el-main class="me-write-main">
        <div class="me-write-title">
            <el-input resize="none"
                      type="textarea"
                      autosize
                      v-model="articleForm.title"
                      placeholder="请输入标题"
                      class="me-write-input">
            </el-input>
        </div>
        <div id="placeholder" style="visibility: hidden;height: 89px;display: none;"></div>
        <markdown-editor :editor="articleForm.editor" class="me-write-editor"></markdown-editor>
    </el-main>
</el-container>
```

图 19. 饿了么 el-card 组件标签

```
publish(articleForm) {
    let that = this
    this.$refs[articleForm].validate((valid) => {
        if (valid) {
            let tags = this.articleForm.tags.map(function (item) {
                return {id: item};
            });
            let article = {
                id: this.articleForm.id,
                title: this.articleForm.title,
                summary: this.articleForm.summary,
                category: this.articleForm.category,
                tags: tags,
                body: {
                    content: this.articleForm.editor.value,
                    contentHtml: this.articleForm.editor.ref.d_render
                }
            }
            this.publishVisible = false;
            let loading = this.$loading({
                lock: true,
                text: '发布中，请稍后...'
            })
            } else {
                return false;
            }
    });
}
```

图 20. 路由跳转的绑定

```
@PostMapping("publish")
    public Result publish(@RequestBody ArticleParam articleParam){
        return articleService.publish(articleParam);
    }
@Override
    @Transactional
    public Result publish(ArticleParam articleParam) {
        SysUser sysUser = UserThreadLocal.get();
        Article article = new Article();
        article.setAuthorId(sysUser.getId());
        article.setCategoryId(articleParam.getCategory().getId());
        article.setCreateDate(System.currentTimeMillis());
        article.setCommentCounts(0);
        article.setSummary(articleParam.getSummary());
        article.setTitle(articleParam.getTitle());
        article.setViewCounts(0);
        article.setWeight(Article.Article_Common);
        article.setBodyId(-1L);
        this.articleMapper.insert(article);
        //tags
        List<TagVo> tags = articleParam.getTags();
        if (tags != null) {
            for (TagVo tag : tags) {
                ArticleTag articleTag = new ArticleTag();
                articleTag.setArticleId(article.getId());
                articleTag.setTagId(tag.getId());
                this.articleTagMapper.insert(articleTag);
            }
        }
        return Result.success(articleVo);
    }
public interface ArticleBodyMapper extends BaseMapper<ArticleBody> {}
```

图 21. 发表文章后端实现

4 结语

通过本次思维实训项目的开发，我学习到了后端开发技术，了解了后端开发流程，明白了前后端分离开发的架构。此次开发中，我增强了信息搜集能力，工具使用能力，工程能力等。这为我今后独立完成开发探索打下了良好基础。

致谢 在本次实训之中，大部分的课程实验内容资料查询代码实现之中完成的。在本项目之中，我负责后端界面的开发。同时，感谢我的前端搭档张俊雄，如果没有他，此项目的前端展示将无法完成。此外，正是互联网的开源共享精神，有丰富的学习资料可供参考。最后，也十分感谢在百忙之中抽出时间审阅本文的老师。由于本人的学识和写作的水平有限，在本论文的写作中难免有僻陋，恳请老师多指教。

参 考 文 献

[1]刘亚茹,张军.Vue.js 框架在网站前端开发中的研究[J].电脑编程技巧与维护,2022(01):18-19+39.DOI:10.16184/j.cnki.comprg.2022.01.009.

[2]罗斌,温丰蔚,曾晓钰,张亮,韦通明.基于 Vue.js 的培训可视化系统开发与设计[J].现代工业经济和信息化,2021,11(12):54-56.DOI:10.16525/j.cnki.14-1362/n.2021.12.020.

[3]郑玉娟,张亚东.基于 Vue.js 的微商城前端设计与实现[J].信息技术与信息化,2021(11):101-103.

[4]单树倩,任佳勋.基于 SpringBoot 和 Vue 框架的数据库原理网站设计与实现[J].电脑知识与技术,2021,17(30):40-41+50.DOI:10.14004/j.cnki.ckt.2021.2868.

[5]喻佳,吴丹新.基于 SpringBoot 的 Web 快速开发框架[J].电脑编程技巧与维护,2021(09):31-33.DOI:10.16184/j.cnki.comprg.2021.09.013.