# 项目结构说明文档

## 1. 项目概述

本项目是一个基于Spring Boot + Vue.js的Web应用系统，采用前后端分离架构。主要功能包括微信公众号文章爬取、文章管理、全文检索、智能摘要生成等。

## 2. 目录结构

### 2.1 后端结构

src/main/java/org/example/web2\_7/ ├── controller/ # 控制器层 │ ├── ArticleController.java # 文章管理 │ ├── ArticleSearchController.java # 文章搜索 │ ├── ChatController.java # 聊天功能 │ ├── CrawlerController.java # 爬虫控制 │ ├── HomeController.java # 首页 │ ├── SummaryController.java # 摘要生成 │ └── TagController.java # 标签管理 ├── service/ # 服务层 ├── model/ # 数据模型 ├── config/ # 配置类 └── utils/ # 工具类

## 3. 核心代码实现

### 3.1 控制器层代码

#### ArticleController.java

package org.example.web2\_7.controller; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.service.ArticleService; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import java.util.ArrayList; import java.util.HashMap; import java.util.List; import java.util.Map; @RestController@RequestMapping("/api/articles")@CrossOriginpublicclassArticleController { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(ArticleController.class); @Autowiredprivate ArticleService articleService; @Autowiredprivate ArticleMapper articleMapper; @GetMappingpublic ResponseEntity<?> getAllArticles() { try { List<Article> articles = articleMapper.findAll(); return ResponseEntity.ok(articles); } catch (Exception e) { logger.error("获取文章列表失败", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } @GetMapping("/{id}")public ResponseEntity<?> getArticleById(@PathVariable Integer id) { try { Articlearticle= articleService.getArticleById(id); if (article == null) { return ResponseEntity.notFound().build(); } return ResponseEntity.ok(article); } catch (Exception e) { logger.error("获取文章详情失败", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } @GetMapping("/{id}/html")public ResponseEntity<?> getArticleHtml(@PathVariable Integer id) { try { Stringhtml= articleService.getArticleHtml(id); if (html == null) { return ResponseEntity.notFound().build(); } return ResponseEntity.ok(Map.of("html", html)); } catch (Exception e) { logger.error("获取文章HTML内容失败", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } @DeleteMapping("/{id}")public ResponseEntity<?> deleteArticle(@PathVariable Integer id) { try { articleMapper.deleteById(id); return ResponseEntity.ok().build(); } catch (Exception e) { logger.error("删除文章失败", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } }

#### ArticleSearchController.java

package org.example.web2\_7.controller; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.service.ArticleSearchService; import org.example.web2\_7.service.LuceneIndexService; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import java.util.List; import java.util.Map; @RestController@RequestMapping("/api/search")@CrossOriginpublicclassArticleSearchController { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(ArticleSearchController.class); privatefinal ArticleSearchService searchService; @Autowiredprivate ArticleMapper articleMapper; @Autowiredprivate LuceneIndexService luceneIndexService; @AutowiredpublicArticleSearchController(ArticleSearchService searchService) { this.searchService = searchService; } @GetMappingpublic ResponseEntity<?> searchArticles(@RequestParam String keyword) { try { logger.info("搜索文章，关键词: {}", keyword); List<Article> results = searchService.searchArticles(keyword); logger.info("搜索结果数量: {}", results.size()); return ResponseEntity.ok(results); } catch (Exception e) { logger.error("搜索失败: ", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } }

#### ChatController.java

package org.example.web2\_7.controller; import org.example.web2\_7.service.DeepSeekService; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*; import java.util.Map; @RestController@RequestMapping("/api/chat")@CrossOrigin("\*")publicclassChatController { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(ChatController.class); @Autowiredprivate DeepSeekService deepSeekService; @PostMappingpublic Map<String, Object> chat(@RequestBody ChatRequest request) { logger.info("收到聊天请求, 用户ID: {}", request.getUserId()); try { Stringresponse= deepSeekService.generateChatResponse(request.getMessage()); return Map.of( "success", true, "message", response ); } catch (Exception e) { logger.error("生成回复时出错: {}", e.getMessage(), e); return Map.of( "success", false, "error", e.getMessage() ); } } publicstaticclassChatRequest { private String userId; private String message; public String getUserId() { return userId; } publicvoidsetUserId(String userId) { this.userId = userId; } public String getMessage() { return message; } publicvoidsetMessage(String message) { this.message = message; } } }

#### CrawlerController.java

package org.example.web2\_7.controller; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.service.CrawlerService; import org.example.web2\_7.service.LuceneIndexService; import org.example.web2\_7.crawler.WeChatArticleSpider; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.core.io.Resource; import org.springframework.http.MediaType; import org.springframework.http.ResponseEntity; import org.springframework.transaction.annotation.Transactional; import org.springframework.web.bind.annotation.\*; import org.springframework.http.HttpHeaders; import org.springframework.http.HttpStatus; import java.io.File; import java.util.List; import java.util.Map; import java.util.HashMap; @RestController@RequestMapping("/api/crawler")@CrossOriginpublicclassCrawlerController { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(CrawlerController.class); @Autowiredprivate CrawlerService crawlerService; @Autowiredprivate ArticleMapper articleMapper; @Autowiredprivate LuceneIndexService luceneIndexService; @PostMapping("/crawl")@Transactionalpublic ResponseEntity<?> crawlArticle(@RequestBody Map<String, String> request) { Stringurl= request.get("url"); if (url == null || url.trim().isEmpty()) { return ResponseEntity.badRequest().body(Map.of("error", "URL不能为空")); } try { logger.info("开始爬取文章: {}", url); crawlerService.crawlArticle(url); return ResponseEntity.ok(Map.of("message", "爬取成功")); } catch (Exception e) { logger.error("爬取文章失败: ", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } @GetMapping("/articles")public ResponseEntity<?> getAllArticles() { try { List<Article> articles = articleMapper.findAll(); return ResponseEntity.ok(articles); } catch (Exception e) { logger.error("获取文章列表失败", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } @GetMapping("/articles/{id}")public ResponseEntity<?> getArticleById(@PathVariable Integer id) { try { Articlearticle= articleMapper.findById(id); if (article == null) { return ResponseEntity.notFound().build(); } return ResponseEntity.ok(article); } catch (Exception e) { logger.error("获取文章详情失败", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } @DeleteMapping("/articles/{id}")public ResponseEntity<?> deleteArticle(@PathVariable Integer id) { try { articleMapper.deleteById(id); return ResponseEntity.ok().build(); } catch (Exception e) { logger.error("删除文章失败", e); return ResponseEntity.badRequest().body(Map.of("error", e.getMessage())); } } }

#### HomeController.java

package org.example.web2\_7.controller; import org.springframework.web.bind.annotation.RestController; import org.springframework.web.bind.annotation.GetMapping; @RestControllerpublicclassHomeController { @GetMapping("/")public String home() { return"欢迎访问微信公众号文章爬虫系统"; } }

#### ImageController.java

package org.example.web2\_7.controller; import org.example.web2\_7.utils.HtmlImageProcessor; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.core.io.Resource; import org.springframework.http.MediaType; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import org.springframework.http.HttpHeaders; import java.io.File; import java.util.Map; @RestController@RequestMapping("/api/images")@CrossOriginpublicclassImageController { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(ImageController.class); @Autowiredprivate HtmlImageProcessor imageProcessor; @GetMapping("/{filename}")public ResponseEntity<Resource> getImage(@PathVariable String filename) { try { FileimageFile= imageProcessor.getImageFile(filename); if (!imageFile.exists()) { return ResponseEntity.notFound().build(); } Resourceresource= imageProcessor.getImageResource(imageFile); StringcontentType= imageProcessor.getImageContentType(filename); return ResponseEntity.ok() .contentType(MediaType.parseMediaType(contentType)) .header(HttpHeaders.CONTENT\_DISPOSITION, "inline; filename=\"" + filename + "\"") .body(resource); } catch (Exception e) { logger.error("获取图片失败: ", e); return ResponseEntity.badRequest().build(); } } }

#### SummaryController.java

package org.example.web2\_7.controller; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.pojo.RelatedArticle; import org.example.web2\_7.service.DeepSeekService; import org.example.web2\_7.service.RelatedArticleService; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import java.util.HashMap; import java.util.List; import java.util.Map; @RestController@RequestMapping("/api/articles")@CrossOrigin(originPatterns = "\*", allowCredentials = "true")publicclassSummaryController { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(SummaryController.class); @Autowiredprivate DeepSeekService deepSeekService; @Autowiredprivate ArticleMapper articleMapper; @Autowiredprivate RelatedArticleService relatedArticleService; @PostMapping("/{id}/summarize")public ResponseEntity<Map<String, Object>> generateSummary( @PathVariable("id") Integer id, @RequestParam(value = "force", required = false, defaultValue = "false")boolean force) { logger.info("开始为文章ID:{}生成摘要, 强制模式: {}", id, force); Map<String, Object> response = newHashMap<>(); try { Articlearticle= articleMapper.findById(id); if (article == null) { logger.warn("未找到ID为{}的文章", id); response.put("success", false); response.put("message", "文章不存在"); return ResponseEntity.badRequest().body(response); } if (!force && article.getSummary() != null && !article.getSummary().trim().isEmpty()) { logger.info("文章ID:{}已有摘要，直接返回", id); response.put("success", true); response.put("summary", article.getSummary()); response.put("message", "文章已有摘要"); response.put("isExisting", true); return ResponseEntity.ok(response); } Stringcontent= article.getContent(); if (content == null || content.trim().isEmpty()) { logger.warn("文章ID:{}内容为空，无法生成摘要", id); response.put("success", false); response.put("message", "文章内容为空，无法生成摘要"); return ResponseEntity.badRequest().body(response); } String[] result = deepSeekService.generateSummaryAndKeywords(content); Stringsummary= result[0]; Stringkeywords= result[1]; intupdateResult= articleMapper.updateArticleSummaryAndKeywords(id, summary, keywords); if (updateResult > 0) { response.put("success", true); response.put("summary", summary); response.put("keywords", keywords); response.put("message", "摘要生成成功"); return ResponseEntity.ok(response); } else { response.put("success", false); response.put("message", "更新文章摘要失败"); return ResponseEntity.badRequest().body(response); } } catch (Exception e) { logger.error("生成摘要失败: ", e); response.put("success", false); response.put("message", "生成摘要失败: " + e.getMessage()); return ResponseEntity.badRequest().body(response); } } }

#### TagController.java

package org.example.web2\_7.controller; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.service.TagService; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import java.util.List; import java.util.Map; import java.util.HashMap; import java.util.concurrent.atomic.AtomicBoolean; @RestController@RequestMapping("/api/tags")@CrossOriginpublicclassTagController { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(TagController.class); privatefinalAtomicBooleancacheWarming=newAtomicBoolean(false); @Autowiredprivate TagService tagService; @Autowiredprivate ArticleMapper articleMapper; @GetMappingpublic ResponseEntity<List<String>> getAllTags() { logger.info("获取所有标签"); List<String> tags = tagService.getAllTags(); return ResponseEntity.ok(tags); } @GetMapping("/article/{id}")public ResponseEntity<?> getArticleTags(@PathVariable Integer id) { try { logger.info("获取文章ID={}的标签", id); Articlearticle= articleMapper.findById(id); if (article == null) { logger.warn("未找到ID为{}的文章", id); return ResponseEntity.notFound().build(); } List<String> tags = tagService.extractTags(article); Map<String, Object> response = newHashMap<>(); response.put("articleId", id); response.put("title", article.getTitle()); response.put("tags", tags); return ResponseEntity.ok(response); } catch (Exception e) { logger.error("获取文章标签失败", e); return ResponseEntity.status(500) .body("获取文章标签失败: " + e.getMessage()); } } }

### 3.2 服务层代码

#### ArticleService.java

package org.example.web2\_7.service; import org.example.web2\_7.pojo.Article; publicinterfaceArticleService { Article getArticleById(Integer id); /\*\* \* 获取文章原始HTML内容 \* @param articleId 文章ID \* @return 原始HTML内容 \*/ String getArticleHtml(Integer articleId); /\*\* \* 获取处理后的文章HTML内容（图片URL已替换为本地路径） \* @param articleId 文章ID \* @return 处理后的HTML内容 \*/ String getProcessedArticleHtml(Integer articleId); }

#### ArticleSearchService.java

package org.example.web2\_7.service; import org.apache.lucene.analysis.Analyzer; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.utils.LuceneIndexManager; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service; import jakarta.annotation.PostConstruct; import org.wltea.analyzer.lucene.IKAnalyzer; import java.time.format.DateTimeFormatter; import java.util.List; @ServicepublicclassArticleSearchService { privatefinal LuceneIndexManager indexManager; privatefinal ArticleMapper articleMapper; privatefinal Analyzer analyzer; privatestaticfinalDateTimeFormatterDATE\_FORMATTER= DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"); @AutowiredpublicArticleSearchService(LuceneIndexManager indexManager, ArticleMapper articleMapper) { this.indexManager = indexManager; this.articleMapper = articleMapper; this.analyzer = newIKAnalyzer(false); } @PostConstructpublicvoidinitIndex() { try { List<Article> articles = articleMapper.findAllOrderByPublishTime(); if (articles != null && !articles.isEmpty()) { indexManager.createOrUpdateIndex(articles); System.out.println("成功初始化 " + articles.size() + " 篇文章的搜索索引"); } else { System.out.println("没有找到文章，跳过索引初始化"); } } catch (Exception e) { System.err.println("初始化搜索索引失败: " + e.getMessage()); e.printStackTrace(); } } public List<Article> searchArticles(String keyword)throws Exception { return indexManager.searchArticles(keyword, analyzer); } }

#### CrawlerService.java

package org.example.web2\_7.service; publicinterfaceCrawlerService { voidcrawlArticle(String url)throws Exception; /\*\* \* 检查链接状态 \* @param url 要检查的URL \* @return 链接是否有效 \*/booleancheckLinkStatus(String url)throws Exception; }

#### DeepSeekService.java

package org.example.web2\_7.service; /\*\* \* DeepSeek服务接口 \* 用于调用DeepSeek API生成文章摘要 \*/publicinterfaceDeepSeekService { /\*\* \* 根据输入文本生成摘要 \* \* @param textToSummarize 需要摘要的文本内容 \* @return 生成的摘要文本 \*/ String summarizeText(String textToSummarize); /\*\* \* 根据文本内容生成关键词 \* \* @param textContent 文本内容 \* @return 生成的关键词，以逗号分隔 \*/ String generateKeywords(String textContent); /\*\* \* 同时生成摘要和关键词 \* \* @param textContent 文本内容 \* @return 包含摘要和关键词的数组，索引0为摘要，索引1为关键词 \*/ String[] generateSummaryAndKeywords(String textContent); /\*\* \* 生成聊天回复 \* \* @param message 用户消息 \* @return 生成的回复文本 \*/ String generateChatResponse(String message); }

#### LuceneIndexService.java

package org.example.web2\_7.service; import org.apache.lucene.analysis.standard.StandardAnalyzer; import org.apache.lucene.document.\*; import org.apache.lucene.index.\*; import org.apache.lucene.queryparser.classic.MultiFieldQueryParser; import org.apache.lucene.queryparser.classic.QueryParser; import org.apache.lucene.search.\*; import org.apache.lucene.store.Directory; import org.apache.lucene.store.FSDirectory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.beans.factory.annotation.Value; import org.springframework.stereotype.Service; import jakarta.annotation.PostConstruct; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import java.io.IOException; import java.nio.file.Path; import java.nio.file.Paths; import java.util.ArrayList; import java.util.List; @ServicepublicclassLuceneIndexService { @Autowiredprivate ArticleMapper articleMapper; @Value("${lucene.index.directory:lucene/indexes}")private String indexDir; private Directory directory; private StandardAnalyzer analyzer; private IndexWriter writer; @PostConstructpublicvoidinit()throws IOException { PathindexPath= Paths.get(indexDir); directory = FSDirectory.open(indexPath); analyzer = newStandardAnalyzer(); IndexWriterConfigconfig=newIndexWriterConfig(analyzer); writer = newIndexWriter(directory, config); } publicvoidaddToIndex(String title, String content, String url)throws Exception { Documentdoc=newDocument(); doc.add(newTextField("title", title != null ? title : "", Field.Store.YES)); doc.add(newTextField("content", content != null ? content : "", Field.Store.YES)); doc.add(newTextField("url", url != null ? url : "", Field.Store.YES)); doc.add(newTextField("author", getAuthorFromUrl(url), Field.Store.YES)); doc.add(newTextField("accountName", getAccountNameFromUrl(url), Field.Store.YES)); doc.add(newLongPoint("timestamp", System.currentTimeMillis())); writer.addDocument(doc); writer.commit(); } private String getAuthorFromUrl(String url) { return""; // 需要实现从数据库获取作者信息的逻辑 } private String getAccountNameFromUrl(String url) { return""; // 需要实现从数据库获取公众号名称的逻辑 } }

#### RelatedArticleService.java

package org.example.web2\_7.service; import org.example.web2\_7.pojo.RelatedArticle; import java.util.List; /\*\* \* 相关文章服务接口 \* 用于处理相关文章的爬取和管理 \*/publicinterfaceRelatedArticleService { /\*\* \* 爬取并保存相关文章 \* @param articleId 文章ID \* @param keywords 关键词，用于搜索相关文章 \* @return 爬取的相关文章数量 \*/intcrawlAndSaveRelatedArticles(Integer articleId, String keywords); /\*\* \* 获取文章的相关文章列表 \* @param articleId 文章ID \* @return 相关文章列表 \*/ List<RelatedArticle> getRelatedArticles(Integer articleId); /\*\* \* 添加相关文章 \* @param articleId 文章ID \* @param relatedUrl 相关文章URL \* @param title 相关文章标题 \* @return 是否添加成功 \*/booleanaddRelatedArticle(Integer articleId, String relatedUrl, String title); /\*\* \* 删除文章的所有相关文章 \* @param articleId 文章ID \* @return 是否删除成功 \*/booleandeleteRelatedArticles(Integer articleId); }

#### TagService.java

package org.example.web2\_7.service; import org.example.web2\_7.pojo.Article; import java.util.List; /\*\* \* 标签服务接口 \* 用于处理文章标签的提取和管理 \*/publicinterfaceTagService { /\*\* \* 获取所有可用标签 \* @return 标签列表 \*/ List<String> getAllTags(); /\*\* \* 从文章中提取标签 \* @param article 文章对象 \* @return 提取的标签列表 \*/ List<String> extractTags(Article article); /\*\* \* 根据标签筛选文章 \* @param articles 文章列表 \* @param tag 标签 \* @return 符合标签的文章列表 \*/ List<Article> filterArticlesByTag(List<Article> articles, String tag); /\*\* \* 清除标签缓存 \*/voidclearCache(); }

### 3.3 服务实现层代码

#### ArticleServiceImpl.java

package org.example.web2\_7.service.impl; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.service.ArticleService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service; @ServicepublicclassArticleServiceImplimplementsArticleService { @Autowiredprivate ArticleMapper articleMapper; @Overridepublic Article getArticleById(Integer id) { return articleMapper.findById(id); } @Overridepublic String getArticleHtml(Integer articleId) { Articlearticle= articleMapper.findById(articleId); return article != null ? article.getContent() : null; } @Overridepublic String getProcessedArticleHtml(Integer articleId) { Articlearticle= articleMapper.findById(articleId); if (article == null) { returnnull; } // 处理HTML内容，替换图片URL为本地路径return processHtmlContent(article.getContent()); } private String processHtmlContent(String html) { // 实现HTML内容处理逻辑return html; } }

#### CrawlerServiceImpl.java

package org.example.web2\_7.service.impl; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.service.CrawlerService; import org.example.web2\_7.service.LuceneIndexService; import org.example.web2\_7.utils.HtmlImageProcessor; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service; import org.springframework.transaction.annotation.Transactional; import java.time.LocalDateTime; @ServicepublicclassCrawlerServiceImplimplementsCrawlerService { @Autowiredprivate ArticleMapper articleMapper; @Autowiredprivate LuceneIndexService luceneIndexService; @Autowiredprivate HtmlImageProcessor imageProcessor; @Override@TransactionalpublicvoidcrawlArticle(String url)throws Exception { // 实现文章爬取逻辑Articlearticle=newArticle(); article.setUrl(url); article.setTitle("文章标题"); article.setContent("文章内容"); article.setPublishTime(LocalDateTime.now()); // 保存文章 articleMapper.insert(article); // 处理文章中的图片StringprocessedContent= imageProcessor.processHtmlContent(article.getContent()); article.setContent(processedContent); // 更新文章 articleMapper.update(article); // 添加到搜索索引 luceneIndexService.addToIndex(article.getTitle(), article.getContent(), article.getUrl()); } @OverridepublicbooleancheckLinkStatus(String url)throws Exception { // 实现链接状态检查逻辑returntrue; } }

#### DeepSeekServiceImpl.java

package org.example.web2\_7.service.impl; import org.example.web2\_7.service.DeepSeekService; import org.springframework.beans.factory.annotation.Value; import org.springframework.stereotype.Service; import org.springframework.web.client.RestTemplate; @ServicepublicclassDeepSeekServiceImplimplementsDeepSeekService { @Value("${deepseek.api.url}")private String apiUrl; @Value("${deepseek.api.key}")private String apiKey; privatefinal RestTemplate restTemplate; publicDeepSeekServiceImpl(RestTemplate restTemplate) { this.restTemplate = restTemplate; } @Overridepublic String summarizeText(String textToSummarize) { // 调用DeepSeek API生成摘要return callDeepSeekApi("summarize", textToSummarize); } @Overridepublic String generateKeywords(String textContent) { // 调用DeepSeek API生成关键词return callDeepSeekApi("keywords", textContent); } @Overridepublic String[] generateSummaryAndKeywords(String textContent) { // 同时生成摘要和关键词Stringsummary= summarizeText(textContent); Stringkeywords= generateKeywords(textContent); returnnewString[]{summary, keywords}; } @Overridepublic String generateChatResponse(String message) { // 调用DeepSeek API生成聊天回复return callDeepSeekApi("chat", message); } private String callDeepSeekApi(String endpoint, String text) { // 实现API调用逻辑return"API响应结果"; } }

#### RelatedArticleServiceImpl.java

package org.example.web2\_7.service.impl; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.Dao.RelatedArticleMapper; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.pojo.RelatedArticle; import org.example.web2\_7.service.RelatedArticleService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service; import org.springframework.transaction.annotation.Transactional; import java.util.List; @ServicepublicclassRelatedArticleServiceImplimplementsRelatedArticleService { @Autowiredprivate RelatedArticleMapper relatedArticleMapper; @Autowiredprivate ArticleMapper articleMapper; @Override@TransactionalpublicintcrawlAndSaveRelatedArticles(Integer articleId, String keywords) { // 实现相关文章爬取和保存逻辑return0; } @Overridepublic List<RelatedArticle> getRelatedArticles(Integer articleId) { return relatedArticleMapper.findByArticleId(articleId); } @Override@TransactionalpublicbooleanaddRelatedArticle(Integer articleId, String relatedUrl, String title) { RelatedArticlerelatedArticle=newRelatedArticle(); relatedArticle.setArticleId(articleId); relatedArticle.setRelatedUrl(relatedUrl); relatedArticle.setTitle(title); return relatedArticleMapper.insert(relatedArticle) > 0; } @Override@TransactionalpublicbooleandeleteRelatedArticles(Integer articleId) { return relatedArticleMapper.deleteByArticleId(articleId) > 0; } }

### 3.4 数据访问层代码

#### ArticleMapper.java

package org.example.web2\_7.Dao; import org.apache.ibatis.annotations.\*; import org.example.web2\_7.pojo.Article; import java.util.List; @MapperpublicinterfaceArticleMapper { @Select("SELECT \* FROM article WHERE id = #{id}") Article findById(Integer id); @Select("SELECT \* FROM article ORDER BY publish\_time DESC") List<Article> findAll(); @Select("SELECT \* FROM article ORDER BY publish\_time DESC") List<Article> findAllOrderByPublishTime(); @Insert("INSERT INTO article (title, content, url, publish\_time) VALUES (#{title}, #{content}, #{url}, #{publishTime})")@Options(useGeneratedKeys = true, keyProperty = "id")intinsert(Article article); @Update("UPDATE article SET title = #{title}, content = #{content}, url = #{url}, publish\_time = #{publishTime} WHERE id = #{id}")intupdate(Article article); @Update("UPDATE article SET summary = #{summary}, keywords = #{keywords} WHERE id = #{id}")intupdateArticleSummaryAndKeywords(@Param("id") Integer id, @Param("summary") String summary, @Param("keywords") String keywords); @Delete("DELETE FROM article WHERE id = #{id}")intdeleteById(Integer id); }

#### TagMapper.java

package org.example.web2\_7.Dao; import org.apache.ibatis.annotations.\*; import org.example.web2\_7.pojo.Tag; import java.util.List; @MapperpublicinterfaceTagMapper { @Select("SELECT \* FROM tag") List<Tag> findAll(); @Select("SELECT \* FROM tag WHERE article\_id = #{articleId}") List<Tag> findByArticleId(Integer articleId); @Insert("INSERT INTO tag (name, article\_id) VALUES (#{name}, #{articleId})")intinsert(Tag tag); @Delete("DELETE FROM tag WHERE article\_id = #{articleId}")intdeleteByArticleId(Integer articleId); }

### 3.5 模型层代码

#### Article.java

package org.example.web2\_7.pojo; import java.time.LocalDateTime; publicclassArticle { private Integer id; private String title; private String content; private String url; private LocalDateTime publishTime; private String summary; private String keywords; // Getters and Setterspublic Integer getId() { return id; } publicvoidsetId(Integer id) { this.id = id; } public String getTitle() { return title; } publicvoidsetTitle(String title) { this.title = title; } public String getContent() { return content; } publicvoidsetContent(String content) { this.content = content; } public String getUrl() { return url; } publicvoidsetUrl(String url) { this.url = url; } public LocalDateTime getPublishTime() { return publishTime; } publicvoidsetPublishTime(LocalDateTime publishTime) { this.publishTime = publishTime; } public String getSummary() { return summary; } publicvoidsetSummary(String summary) { this.summary = summary; } public String getKeywords() { return keywords; } publicvoidsetKeywords(String keywords) { this.keywords = keywords; } }

#### RelatedArticle.java

package org.example.web2\_7.pojo; publicclassRelatedArticle { private Integer id; private Integer articleId; private String relatedUrl; private String title; // Getters and Setterspublic Integer getId() { return id; } publicvoidsetId(Integer id) { this.id = id; } public Integer getArticleId() { return articleId; } publicvoidsetArticleId(Integer articleId) { this.articleId = articleId; } public String getRelatedUrl() { return relatedUrl; } publicvoidsetRelatedUrl(String relatedUrl) { this.relatedUrl = relatedUrl; } public String getTitle() { return title; } publicvoidsetTitle(String title) { this.title = title; } }

#### Tag.java

package org.example.web2\_7.pojo; publicclassTag { private Integer id; private String name; private Integer articleId; // Getters and Setterspublic Integer getId() { return id; } publicvoidsetId(Integer id) { this.id = id; } public String getName() { return name; } publicvoidsetName(String name) { this.name = name; } public Integer getArticleId() { return articleId; } publicvoidsetArticleId(Integer articleId) { this.articleId = articleId; } }

### 3.6 工具类代码

#### HtmlImageProcessor.java

package org.example.web2\_7.utils; import org.jsoup.Jsoup; import org.jsoup.nodes.Document; import org.jsoup.nodes.Element; import org.jsoup.select.Elements; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Value; import org.springframework.core.io.Resource; import org.springframework.core.io.UrlResource; import org.springframework.http.MediaType; import org.springframework.stereotype.Component; import java.io.File; import java.io.IOException; import java.net.URL; import java.nio.file.Files; import java.nio.file.Path; import java.nio.file.Paths; import java.util.UUID; @ComponentpublicclassHtmlImageProcessor { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(HtmlImageProcessor.class); @Value("${image.storage.path:images}")private String imageStoragePath; @Value("${image.base.url:http://localhost:8080/api/images}")private String imageBaseUrl; public String processHtmlContent(String html) { if (html == null || html.trim().isEmpty()) { return html; } try { Documentdoc= Jsoup.parse(html); Elementsimages= doc.select("img"); for (Element img : images) { Stringsrc= img.attr("src"); if (src != null && !src.isEmpty()) { try { StringlocalPath= downloadAndSaveImage(src); if (localPath != null) { img.attr("src", localPath); } } catch (Exception e) { logger.error("处理图片失败: " + src, e); } } } return doc.html(); } catch (Exception e) { logger.error("处理HTML内容失败", e); return html; } } private String downloadAndSaveImage(String imageUrl)throws IOException { try { URLurl=newURL(imageUrl); StringfileName= UUID.randomUUID().toString() + getFileExtension(imageUrl); PathtargetPath= Paths.get(imageStoragePath, fileName); // 确保目录存在 Files.createDirectories(targetPath.getParent()); // 下载并保存图片try (varinputStream= url.openStream()) { Files.copy(inputStream, targetPath); } // 返回可访问的URLreturn imageBaseUrl + "/" + fileName; } catch (Exception e) { logger.error("下载图片失败: " + imageUrl, e); returnnull; } } private String getFileExtension(String url) { intlastDotIndex= url.lastIndexOf('.'); if (lastDotIndex > 0) { return url.substring(lastDotIndex); } return".jpg"; // 默认扩展名 } public File getImageFile(String filename) { return Paths.get(imageStoragePath, filename).toFile(); } public Resource getImageResource(File imageFile)throws IOException { returnnewUrlResource(imageFile.toURI()); } public String getImageContentType(String filename) { Stringextension= filename.substring(filename.lastIndexOf('.') + 1).toLowerCase(); switch (extension) { case"jpg": case"jpeg": return MediaType.IMAGE\_JPEG\_VALUE; case"png": return MediaType.IMAGE\_PNG\_VALUE; case"gif": return MediaType.IMAGE\_GIF\_VALUE; default: return MediaType.APPLICATION\_OCTET\_STREAM\_VALUE; } } }

#### LuceneIndexManager.java

package org.example.web2\_7.utils; import org.apache.lucene.analysis.Analyzer; import org.apache.lucene.document.\*; import org.apache.lucene.index.\*; import org.apache.lucene.queryparser.classic.QueryParser; import org.apache.lucene.search.\*; import org.apache.lucene.store.Directory; import org.apache.lucene.store.FSDirectory; import org.example.web2\_7.pojo.Article; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Value; import org.springframework.stereotype.Component; import java.io.IOException; import java.nio.file.Path; import java.nio.file.Paths; import java.util.ArrayList; import java.util.List; @ComponentpublicclassLuceneIndexManager { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(LuceneIndexManager.class); @Value("${lucene.index.directory:lucene/indexes}")private String indexDir; private Directory directory; private IndexWriter writer; publicvoidinit()throws IOException { PathindexPath= Paths.get(indexDir); directory = FSDirectory.open(indexPath); IndexWriterConfigconfig=newIndexWriterConfig(newStandardAnalyzer()); writer = newIndexWriter(directory, config); } publicvoidcreateOrUpdateIndex(List<Article> articles)throws IOException { if (writer == null) { init(); } try { // 清除现有索引 writer.deleteAll(); // 添加新文档for (Article article : articles) { Documentdoc=newDocument(); doc.add(newTextField("title", article.getTitle(), Field.Store.YES)); doc.add(newTextField("content", article.getContent(), Field.Store.YES)); doc.add(newStringField("url", article.getUrl(), Field.Store.YES)); doc.add(newLongPoint("publishTime", article.getPublishTime().toEpochSecond(java.time.ZoneOffset.UTC))); doc.add(newStoredField("publishTime", article.getPublishTime().toEpochSecond(java.time.ZoneOffset.UTC))); writer.addDocument(doc); } writer.commit(); logger.info("成功更新索引，共{}篇文章", articles.size()); } catch (Exception e) { logger.error("更新索引失败", e); throw e; } } public List<Article> searchArticles(String keyword, Analyzer analyzer)throws Exception { if (directory == null) { init(); } try (IndexReaderreader= DirectoryReader.open(directory)) { IndexSearchersearcher=newIndexSearcher(reader); // 构建查询QueryParsertitleParser=newQueryParser("title", analyzer); QueryParsercontentParser=newQueryParser("content", analyzer); QuerytitleQuery= titleParser.parse(keyword); QuerycontentQuery= contentParser.parse(keyword); BooleanQuery.BuilderqueryBuilder=newBooleanQuery.Builder(); queryBuilder.add(titleQuery, BooleanClause.Occur.SHOULD); queryBuilder.add(contentQuery, BooleanClause.Occur.SHOULD); // 执行搜索TopDocsresults= searcher.search(queryBuilder.build(), 100); // 处理结果 List<Article> articles = newArrayList<>(); for (ScoreDoc hit : results.scoreDocs) { Documentdoc= searcher.doc(hit.doc); Articlearticle=newArticle(); article.setTitle(doc.get("title")); article.setContent(doc.get("content")); article.setUrl(doc.get("url")); article.setPublishTime(java.time.LocalDateTime.ofEpochSecond( doc.getField("publishTime").numericValue().longValue(), 0, java.time.ZoneOffset.UTC)); articles.add(article); } return articles; } } publicvoidclose()throws IOException { if (writer != null) { writer.close(); } if (directory != null) { directory.close(); } } }

### 3.7 配置类代码

#### RestTemplateConfig.java

package org.example.web2\_7.config; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.http.client.ClientHttpRequestFactory; import org.springframework.http.client.SimpleClientHttpRequestFactory; import org.springframework.web.client.RestTemplate; @ConfigurationpublicclassRestTemplateConfig { @Beanpublic RestTemplate restTemplate(ClientHttpRequestFactory factory) { returnnewRestTemplate(factory); } @Beanpublic ClientHttpRequestFactory simpleClientHttpRequestFactory() { SimpleClientHttpRequestFactoryfactory=newSimpleClientHttpRequestFactory(); factory.setReadTimeout(5000); // 读取超时时间：5秒 factory.setConnectTimeout(5000); // 连接超时时间：5秒return factory; } }

#### WebConfig.java

package org.example.web2\_7.config; import org.springframework.context.annotation.Configuration; import org.springframework.web.servlet.config.annotation.CorsRegistry; import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry; import org.springframework.web.servlet.config.annotation.WebMvcConfigurer; @ConfigurationpublicclassWebConfigimplementsWebMvcConfigurer { @OverridepublicvoidaddCorsMappings(CorsRegistry registry) { registry.addMapping("/\*\*") .allowedOrigins("\*") .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS") .allowedHeaders("\*") .maxAge(3600); } @OverridepublicvoidaddResourceHandlers(ResourceHandlerRegistry registry) { // 配置静态资源处理 registry.addResourceHandler("/images/\*\*") .addResourceLocations("file:images/"); // 配置Swagger UI资源 registry.addResourceHandler("swagger-ui.html") .addResourceLocations("classpath:/META-INF/resources/"); registry.addResourceHandler("/webjars/\*\*") .addResourceLocations("classpath:/META-INF/resources/webjars/"); } }

### 3.8 爬虫模块代码

#### WeChatArticleSpider.java

package org.example.web2\_7.crawler; import org.example.web2\_7.pojo.Article; import org.example.web2\_7.service.LuceneIndexService; import org.example.web2\_7.utils.HtmlImageProcessor; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Component; import us.codecraft.webmagic.Page; import us.codecraft.webmagic.Site; import us.codecraft.webmagic.Spider; import us.codecraft.webmagic.processor.PageProcessor; import us.codecraft.webmagic.selector.Html; import java.time.LocalDateTime; @ComponentpublicclassWeChatArticleSpiderimplementsPageProcessor { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(WeChatArticleSpider.class); @Autowiredprivate LuceneIndexService luceneIndexService; @Autowiredprivate HtmlImageProcessor imageProcessor; privateSitesite= Site.me() .setRetryTimes(3) .setSleepTime(1000) .setTimeOut(10000) .setUserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"); @Overridepublicvoidprocess(Page page) { try { Htmlhtml= page.getHtml(); // 提取文章标题Stringtitle= html.xpath("//h1[@class='rich\_media\_title']/text()").toString().trim(); // 提取文章内容Stringcontent= html.xpath("//div[@class='rich\_media\_content']").toString(); // 提取发布时间StringpublishTimeStr= html.xpath("//em[@id='publish\_time']/text()").toString(); LocalDateTimepublishTime= LocalDateTime.parse(publishTimeStr); // 处理文章内容中的图片 content = imageProcessor.processHtmlContent(content); // 创建文章对象Articlearticle=newArticle(); article.setTitle(title); article.setContent(content); article.setUrl(page.getUrl().toString()); article.setPublishTime(publishTime); // 保存到数据库 page.putField("article", article); // 添加到搜索索引 luceneIndexService.addToIndex(title, content, page.getUrl().toString()); logger.info("成功爬取文章: {}", title); } catch (Exception e) { logger.error("爬取文章失败: " + page.getUrl(), e); page.setSkip(true); } } @Overridepublic Site getSite() { return site; } publicvoidcrawl(String url) { Spider.create(this) .addUrl(url) .addPipeline(newDatabasePipeline()) .addPipeline(newInvalidLinkPipeline()) .thread(1) .run(); } }

#### DatabasePipeline.java

package org.example.web2\_7.crawler; import org.example.web2\_7.Dao.ArticleMapper; import org.example.web2\_7.pojo.Article; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Component; import us.codecraft.webmagic.ResultItems; import us.codecraft.webmagic.Task; import us.codecraft.webmagic.pipeline.Pipeline; @ComponentpublicclassDatabasePipelineimplementsPipeline { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(DatabasePipeline.class); @Autowiredprivate ArticleMapper articleMapper; @Overridepublicvoidprocess(ResultItems resultItems, Task task) { try { Articlearticle= resultItems.get("article"); if (article != null) { // 保存文章到数据库 articleMapper.insert(article); logger.info("文章已保存到数据库: {}", article.getTitle()); } } catch (Exception e) { logger.error("保存文章到数据库失败", e); } } }

#### InvalidLinkPipeline.java

package org.example.web2\_7.crawler; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import us.codecraft.webmagic.Page; import us.codecraft.webmagic.ResultItems; import us.codecraft.webmagic.Task; import us.codecraft.webmagic.pipeline.Pipeline; @ComponentpublicclassInvalidLinkPipelineimplementsPipeline { privatestaticfinalLoggerlogger= LoggerFactory.getLogger(InvalidLinkPipeline.class); @Overridepublicvoidprocess(ResultItems resultItems, Task task) { Pagepage= resultItems.get("page"); if (page != null && page.isSkip()) { Stringurl= page.getUrl().toString(); logger.warn("检测到无效链接: {}", url); // 可以在这里添加无效链接的处理逻辑// 例如：保存到数据库、发送通知等 } } }

## 4. 前端代码实现

### 4.1 核心文件

#### main.js

import { createApp } from'vue'importAppfrom'./App.vue'import router from'./router'import axios from'axios'// 配置axios axios.defaults.baseURL = 'http://localhost:8080/api' axios.defaults.timeout = 5000// 创建Vue应用实例const app = createApp(App) // 全局配置 app.config.globalProperties.$axios = axios // 使用路由 app.use(router) // 挂载应用 app.mount('#app')

#### App.vue

<template> <div id="app"> <nav class="navbar"> <div class="nav-brand">微信公众号文章爬虫系统</div> <div class="nav-links"> <router-link to="/">首页</router-link> <router-link to="/articles">文章列表</router-link> <router-link to="/search">搜索</router-link> </div> </nav> <main class="main-content"> <router-view></router-view> </main> </div> </template> <script> export default { name: 'App' } </script> <style> .navbar { display: flex; justify-content: space-between; align-items: center; padding: 1rem 2rem; background-color: #2c3e50; color: white; } .nav-brand { font-size: 1.5rem; font-weight: bold; } .nav-links { display: flex; gap: 1rem; } .nav-links a { color: white; text-decoration: none; padding: 0.5rem 1rem; border-radius: 4px; transition: background-color 0.3s; } .nav-links a:hover { background-color: #34495e; } .main-content { max-width: 1200px; margin: 0 auto; padding: 2rem; } </style>

### 4.2 视图组件

#### Home.vue

<template> <div class="home"> <h1>欢迎使用微信公众号文章爬虫系统</h1> <div class="features"> <div class="feature-card"> <h3>文章爬取</h3> <p>支持微信公众号文章的自动爬取和保存</p> </div> <div class="feature-card"> <h3>全文检索</h3> <p>提供强大的文章搜索功能</p> </div> <div class="feature-card"> <h3>智能摘要</h3> <p>自动生成文章摘要和关键词</p> </div> </div> <div class="quick-actions"> <button @click="$router.push('/articles')">浏览文章</button> <button @click="$router.push('/search')">搜索文章</button> </div> </div> </template> <script> export default { name: 'Home' } </script> <style scoped> .home { text-align: center; } .features { display: grid; grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); gap: 2rem; margin: 2rem 0; } .feature-card { padding: 1.5rem; border-radius: 8px; background-color: #f8f9fa; box-shadow: 0 2px 4px rgba(0,0,0,0.1); } .quick-actions { display: flex; gap: 1rem; justify-content: center; margin-top: 2rem; } button { padding: 0.75rem 1.5rem; border: none; border-radius: 4px; background-color: #3498db; color: white; cursor: pointer; transition: background-color 0.3s; } button:hover { background-color: #2980b9; } </style>

#### ArticleList.vue

<template> <div class="article-list"> <h2>文章列表</h2> <div class="articles"> <div v-for="article in articles" :key="article.id" class="article-card"> <h3>{{ article.title }}</h3> <p class="publish-time">发布时间：{{ formatDate(article.publishTime) }}</p> <div class="article-actions"> <button @click="viewArticle(article.id)">查看详情</button> <button @click="generateSummary(article.id)">生成摘要</button> </div> </div> </div> </div> </template> <script> import axios from 'axios' export default { name: 'ArticleList', data() { return { articles: [] } }, methods: { async fetchArticles() { try { const response = await axios.get('/articles') this.articles = response.data } catch (error) { console.error('获取文章列表失败:', error) } }, formatDate(date) { return new Date(date).toLocaleString() }, viewArticle(id) { this.$router.push(`/articles/${id}`) }, async generateSummary(id) { try { await axios.post(`/articles/${id}/summarize`) this.$router.push(`/articles/${id}`) } catch (error) { console.error('生成摘要失败:', error) } } }, mounted() { this.fetchArticles() } } </script> <style scoped> .article-list { max-width: 800px; margin: 0 auto; } .articles { display: grid; gap: 1.5rem; } .article-card { padding: 1.5rem; border-radius: 8px; background-color: #f8f9fa; box-shadow: 0 2px 4px rgba(0,0,0,0.1); } .publish-time { color: #666; font-size: 0.9rem; } .article-actions { display: flex; gap: 1rem; margin-top: 1rem; } button { padding: 0.5rem 1rem; border: none; border-radius: 4px; background-color: #3498db; color: white; cursor: pointer; transition: background-color 0.3s; } button:hover { background-color: #2980b9; } </style>

#### ArticleDetail.vue

<template> <div class="article-detail"> <div v-if="article" class="article-content"> <h1>{{ article.title }}</h1> <div class="article-meta"> <span>发布时间：{{ formatDate(article.publishTime) }}</span> <span v-if="article.keywords">关键词：{{ article.keywords }}</span> </div> <div v-if="article.summary" class="article-summary"> <h3>摘要</h3> <p>{{ article.summary }}</p> </div> <div class="article-body" v-html="article.content"></div> <div class="article-actions"> <button @click="generateSummary" v-if="!article.summary">生成摘要</button> <button @click="$router.push('/articles')">返回列表</button> </div> </div> <div v-else class="loading"> 加载中... </div> </div> </template> <script> import axios from 'axios' export default { name: 'ArticleDetail', data() { return { article: null } }, methods: { async fetchArticle() { try { const response = await axios.get(`/articles/${this.$route.params.id}`) this.article = response.data } catch (error) { console.error('获取文章详情失败:', error) } }, formatDate(date) { return new Date(date).toLocaleString() }, async generateSummary() { try { await axios.post(`/articles/${this.$route.params.id}/summarize`) this.fetchArticle() } catch (error) { console.error('生成摘要失败:', error) } } }, mounted() { this.fetchArticle() } } </script> <style scoped> .article-detail { max-width: 800px; margin: 0 auto; padding: 2rem; } .article-meta { color: #666; margin: 1rem 0; } .article-summary { background-color: #f8f9fa; padding: 1rem; border-radius: 4px; margin: 1rem 0; } .article-body { line-height: 1.6; margin: 2rem 0; } .article-actions { display: flex; gap: 1rem; margin-top: 2rem; } button { padding: 0.5rem 1rem; border: none; border-radius: 4px; background-color: #3498db; color: white; cursor: pointer; transition: background-color 0.3s; } button:hover { background-color: #2980b9; } .loading { text-align: center; padding: 2rem; color: #666; } </style>

#### Search.vue

<template> <div class="search"> <h2>文章搜索</h2> <div class="search-box"> <input type="text" v-model="keyword" @keyup.enter="search" placeholder="输入关键词搜索文章..." > <button @click="search">搜索</button> </div> <div v-if="searching" class="loading"> 搜索中... </div> <div v-else-if="results.length > 0" class="search-results"> <div v-for="article in results" :key="article.id" class="result-item"> <h3>{{ article.title }}</h3> <p class="publish-time">发布时间：{{ formatDate(article.publishTime) }}</p> <button @click="viewArticle(article.id)">查看详情</button> </div> </div> <div v-else-if="searched" class="no-results"> 未找到相关文章 </div> </div> </template> <script> import axios from 'axios' export default { name: 'Search', data() { return { keyword: '', results: [], searching: false, searched: false } }, methods: { async search() { if (!this.keyword.trim()) return this.searching = true this.searched = true try { const response = await axios.get('/search', { params: { keyword: this.keyword } }) this.results = response.data } catch (error) { console.error('搜索失败:', error) } finally { this.searching = false } }, formatDate(date) { return new Date(date).toLocaleString() }, viewArticle(id) { this.$router.push(`/articles/${id}`) } } } </script> <style scoped> .search { max-width: 800px; margin: 0 auto; padding: 2rem; } .search-box { display: flex; gap: 1rem; margin: 2rem 0; } input { flex: 1; padding: 0.75rem; border: 1px solid #ddd; border-radius: 4px; font-size: 1rem; } button { padding: 0.75rem 1.5rem; border: none; border-radius: 4px; background-color: #3498db; color: white; cursor: pointer; transition: background-color 0.3s; } button:hover { background-color: #2980b9; } .search-results { display: grid; gap: 1.5rem; } .result-item { padding: 1.5rem; border-radius: 8px; background-color: #f8f9fa; box-shadow: 0 2px 4px rgba(0,0,0,0.1); } .publish-time { color: #666; font-size: 0.9rem; margin: 0.5rem 0; } .loading, .no-results { text-align: center; padding: 2rem; color: #666; } </style>

### 4.3 路由配置

#### index.js

import { createRouter, createWebHistory } from'vue-router'importHomefrom'../views/Home.vue'importArticleListfrom'../views/ArticleList.vue'importArticleDetailfrom'../views/ArticleDetail.vue'importSearchfrom'../views/Search.vue'const routes = [ { path: '/', name: 'Home', component: Home }, { path: '/articles', name: 'ArticleList', component: ArticleList }, { path: '/articles/:id', name: 'ArticleDetail', component: ArticleDetail, props: true }, { path: '/search', name: 'Search', component: Search } ] const router = createRouter({ history: createWebHistory(), routes }) // 全局前置守卫 router.beforeEach((to, from, next) => { // 可以在这里添加路由守卫逻辑// 例如：检查用户权限、记录访问日志等next() }) exportdefault router

### 4.4 工具函数

#### api.js

import axios from'axios'// 创建axios实例const api = axios.create({ baseURL: 'http://localhost:8080/api', timeout: 5000, headers: { 'Content-Type': 'application/json' } }) // 请求拦截器 api.interceptors.request.use( config => { // 在发送请求之前做些什么// 例如：添加token、显示加载动画等return config }, error => { // 对请求错误做些什么console.error('请求错误:', error) returnPromise.reject(error) } ) // 响应拦截器 api.interceptors.response.use( response => { // 对响应数据做点什么return response.data }, error => { // 对响应错误做点什么console.error('响应错误:', error) returnPromise.reject(error) } ) // API请求函数exportconst articleApi = { // 获取文章列表getArticles() { return api.get('/articles') }, // 获取文章详情getArticleById(id) { return api.get(`/articles/${id}`) }, // 获取文章HTML内容getArticleHtml(id) { return api.get(`/articles/${id}/html`) }, // 生成文章摘要generateSummary(id) { return api.post(`/articles/${id}/summarize`) }, // 删除文章deleteArticle(id) { return api.delete(`/articles/${id}`) } } exportconst searchApi = { // 搜索文章searchArticles(keyword) { return api.get('/search', { params: { keyword } }) } } exportconst crawlerApi = { // 爬取文章crawlArticle(url) { return api.post('/crawler/crawl', { url }) }, // 检查链接状态checkLinkStatus(url) { return api.get('/crawler/check', { params: { url } }) } } exportconst tagApi = { // 获取所有标签getAllTags() { return api.get('/tags') }, // 获取文章标签getArticleTags(articleId) { return api.get(`/tags/article/${articleId}`) } } exportdefault api

// ... 其他代码部分保持不变 ...