核心代码清单

1. 后端核心功能

1.1 控制器层 (Controller)

ArticleController.java

```
@RestController
@RequestMapping("/api/articles")
@CrossOrigin
```

```
public class ArticleController {
    @Autowired
    private ArticleService articleService;

    @GetMapping("/{id}")
    public ResponseEntity<Article> getArticle(@PathVariable Integer id) {
        Article article = articleService.getArticleById(id);
        return ResponseEntity.ok(article);
    }

    @GetMapping("/{id}/html")
    public ResponseEntity<String> getArticleHtml(@PathVariable Integer id) {
        String html = articleService.getArticleHtml(id);
        return ResponseEntity.ok(html);
    }
}
```

ArticleSearchController.java

```
@RestController
@RequestMapping("/api/search")
@CrossOrigin
public class ArticleSearchController {
    @Autowired
    private ArticleSearchService searchService;

    @GetMapping
    public ResponseEntity<?> searchArticles(@RequestParam String keyword) {
        try {
            List<Article> results = searchService.searchArticles(keyword);
            return ResponseEntity.ok(results);
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of("error",
e.getMessage()));
        }
    }
}
```

CrawlerController.java

```
@RestController
@RequestMapping("/api/crawler")
@CrossOrigin
public class CrawlerController {
    @Autowired
    private CrawlerService crawlerService;

    @PostMapping("/crawl")
    public ResponseEntity<?> crawlArticle(@RequestBody Map<String, String> request)
{
```

```
        try {
            crawlerService.crawlArticle(request.get("url"));
            return ResponseEntity.ok("文章爬取成功");
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
    }
}
```

1.2 服务层 (Service)

ArticleService.java

```
public interface ArticleService {
    Article getArticleById(Integer id);
    String getArticleHtml(Integer articleId);
    String getProcessedArticleHtml(Integer articleId);
}

@Service
public class ArticleServiceImpl implements ArticleService {
    @Autowired
    private ArticleMapper articleMapper;

    @Autowired
    private LuceneIndexService luceneIndexService;

    @Override
    public Article getArticleById(Integer id) {
        return articleMapper.findById(id);
    }

    @Override
    public String getArticleHtml(Integer articleId) {
        Article article = articleMapper.findById(articleId);
        return article != null ? article.getFullHtml() : null;
    }

    @Override
    public String getProcessedArticleHtml(Integer articleId) {
        Article article = articleMapper.findById(articleId);
        return article != null ? article.getProcessedHtml() : null;
    }
}
```

ArticleSearchService.java

```
@Service
public class ArticleSearchService {
    @Autowired
    private LuceneIndexManager indexManager;

    @Autowired
    private ArticleMapper articleMapper;

    public List<Article> searchArticles(String keyword) throws Exception {
        if (keyword == null || keyword.trim().isEmpty()) {
            return new ArrayList<>();
        }

        String[] fields = {"title^2.0", "content", "author", "accountName"};
        MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, analyzer);
        Query query = parser.parse(keyword);

        IndexReader reader = indexManager.getIndexReader();
        IndexSearcher searcher = new IndexSearcher(reader);
        TopDocs hits = searcher.search(query, 100);

        List<Article> results = new ArrayList<>();
        for (ScoreDoc hit : hits.scoreDocs) {
            Document doc = searcher.doc(hit.doc);
            Article article =
articleMapper.findById(Integer.parseInt(doc.get("id")));
            if (article != null && !article.getIsDeleted()) {
                results.add(article);
            }
        }

        return results;
    }
}
```

CrawlerService.java

```
public interface CrawlerService {
    void crawlArticle(String url);
}

@Service
public class CrawlerServiceImpl implements CrawlerService {
    @Autowired
    private DatabasePipeline databasePipeline;

    @Autowired
    private InvalidLinkPipeline invalidLinkPipeline;

    @Override
    public void crawlArticle(String url) {
        Spider spider = Spider.create(new WeChatArticleSpider())
```

```
                .addUrl(url)
                .thread(1)
                .addPipeline(invalidLinkPipeline)
                .addPipeline(databasePipeline);
        spider.run();
    }
}
```

1.3 数据访问层 (DAO)

ArticleMapper.java

```
@Mapper
public interface ArticleMapper {
    Article findById(Integer id);
    List<Article> findAllOrderByPublishTime();
    void insert(Article article);
    void update(Article article);
    void deleteById(Integer id);
}

@Mapper
public interface TagMapper {
    List<String> findAllTags();
    List<Article> findByTag(String tag);
}
```

1.4 爬虫模块

WeChatArticleSpider.java

```
public class WeChatArticleSpider implements PageProcessor {
    private final Site site = Site.me()
            .setRetryTimes(3)
            .setSleepTime(2000)
            .setUserAgent("Mozilla/5.0...");

    @Override
    public void process(Page page) {
        try {
            Document doc = Jsoup.parse(page.getHtml().get());

            // 提取文章信息
            String title = doc.select("meta[property=og:title]").attr("content");
            String author = extractAuthor(doc);
            String accountName = extractAccountName(doc);
```

```
            String publishTime = extractPublishTime(doc);
            String content = doc.select("div.rich_media_content").text();

            // 处理图片
            Elements imgs = doc.select("img[data-src], img[src]");
            List<String> imageUrls = new ArrayList<>();
            for (Element img : imgs) {
                String imgUrl = img.attr("data-src");
                if (imgUrl.isEmpty()) {
                    imgUrl = img.attr("src");
                }
                if (!imgUrl.isEmpty()) {
                    imageUrls.add(cleanWeChatUrl(imgUrl));
                }
            }

            // 存储结果
            page.putField("title", title);
            page.putField("author", author);
            page.putField("accountName", accountName);
            page.putField("publishTime", publishTime);
            page.putField("content", content);
            page.putField("imageUrls", imageUrls);
        } catch (Exception e) {
            page.setSkip(true);
        }
    }
}
```

1.5 搜索模块

LuceneIndexService.java

```
@Service
public class LuceneIndexService {
    @Autowired
    private ArticleMapper articleMapper;

    private Directory directory;
    private StandardAnalyzer analyzer;
    private IndexWriter writer;

    public void addToIndex(String title, String content, String url) throws
Exception {
        Document doc = new Document();
        doc.add(new TextField("title", title, Field.Store.YES));
        doc.add(new TextField("content", content, Field.Store.YES));
        doc.add(new TextField("url", url, Field.Store.YES));
        writer.addDocument(doc);
        writer.commit();
    }
```

```
    public List<SearchResult> search(String queryStr) throws Exception {
        DirectoryReader reader = DirectoryReader.open(directory);
        IndexSearcher searcher = new IndexSearcher(reader);

        String[] fields = {"title", "content", "author", "accountName"};
        MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, analyzer);
        Query query = parser.parse(queryStr);

        TopDocs hits = searcher.search(query, 50);
        List<SearchResult> results = new ArrayList<>();

        for (ScoreDoc hit : hits.scoreDocs) {
            Document doc = searcher.doc(hit.doc);
            results.add(new SearchResult(
                doc.get("title"),
                doc.get("content"),
                doc.get("url"),
                doc.get("author"),
                doc.get("accountName"),
                hit.score
            ));
        }

        return results;
    }
}
```

2. 前端核心功能

2.1 首页组件

Home.vue

```
<template>
  <div class="home">
    <!-- 爬虫表单 -->
    <el-card class="crawler-form">
      <el-form :model="crawlerForm">
        <el-form-item label="文章URL">
          <el-input v-model="crawlerForm.url" placeholder="请输入微信公众号文章URL">
</el-input>
        </el-form-item>
        <el-button type="primary" @click="crawlArticle" :loading="crawling">
          开始爬取
        </el-button>
      </el-form>
    </el-card>

    <!-- 搜索框 -->
```

```
    <el-card class="search-form">
      <el-input
        v-model="searchKeyword"
        placeholder="请输入搜索关键词"
        @keyup.enter="searchArticles"
      >
        <template #append>
          <el-button @click="searchArticles">搜索</el-button>
        </template>
      </el-input>
    </el-card>

    <!-- 文章列表 -->
    <el-card class="article-list">
      <div class="waterfall-container">
        <div v-for="article in filteredArticles" :key="article.id" class="article-
card">
          <el-card shadow="hover">
            <div class="article-title">
              <router-link :to="{ name: 'ArticleDetail', params: { id: article.id
}}">
                {{ article.title }}
              </router-link>
            </div>
            <div class="article-meta">
              <span>作者：{{ article.author }}</span>
              <span>发布时间：{{ formatDate(article.publishTime) }}</span>
            </div>
          </el-card>
        </div>
      </div>
    </el-card>
  </div>
</template>
```

2.2 文章详情组件

ArticleDetail.vue

```
<template>
  <div class="article-detail" v-if="article">
    <el-card>
      <template #header>
        <div class="article-header">
          <h1>{{ article.title }}</h1>
          <div class="article-meta">
            <span>作者：{{ article.author }}</span>
            <span>发布时间：{{ formatDate(article.publishTime) }}</span>
          </div>
        </div>
      </template>
```

```
      <div class="article-content" v-html="processedContent"></div>

      <div class="article-actions">
        <el-button type="primary" @click="openArticle">阅读原文</el-button>
      </div>
    </el-card>
  </div>
</template>
```

2.3 搜索组件

Search.vue

```
<template>
  <div class="search-container">
    <div class="search-header">
      <el-input
        v-model="keyword"
        placeholder="请输入关键词搜索"
        @keyup.enter="handleSearch"
      >
        <template #append>
          <el-button @click="handleSearch">搜索</el-button>
        </template>
      </el-input>
    </div>

    <div v-if="articles.length" class="search-results">
      <el-table :data="articles">
        <el-table-column prop="title" label="标题">
          <template #default="scope">
            <router-link :to="{ name: 'ArticleDetail', params: { id: scope.row.id
}}">
              {{ scope.row.title }}
            </router-link>
          </template>
        </el-table-column>
        <el-table-column prop="author" label="作者"></el-table-column>
        <el-table-column prop="publishTime" label="发布时间">
          <template #default="scope">
            {{ formatDate(scope.row.publishTime) }}
          </template>
        </el-table-column>
      </el-table>
    </div>
  </div>
</template>
```