飞猪在线笔记系统 - 项目说明文档

---

1. 项目基本信息

1.1 项目概要

- 项目名称: 飞猪在线笔记系统
- 选题方向: Web应用开发 - 笔记系统
- 作者姓名: 朱嘉翔
- 学号: 102201604
- 开发时间: 2025年

1.2 项目访问/运行方式

本地运行步骤：

后端部署：

1. 确保已安装Java 8+、Maven 3.6+、MySQL 5.7+
2. 创建数据库并执行`schema.sql`脚本
3. 修改`application.yml`中的数据库连接配置
4. 运行命令：`mvn spring-boot:run`
5. 后端服务启动在：`http://localhost:9090`

前端访问：

1. 将HTML文件部署到Web服务器或直接用浏览器打开
2. 访问`index.html`开始使用系统
3. 测试账号：`admin/123456` 或 `test/123456`

2. 设计与实现概述

2.1 项目简介

本项目是一个基于Web的在线笔记管理系统，旨在帮助用户构建个人知识库。系统采用前后端分离架构，支持用户创建多个知识库，在每个知识库中管理相关笔记，并提供Markdown编辑器实现富文本编辑功能。

主要功能包括：

- 用户认证与权限管理
- 知识库的增删改查管理
- 笔记的创建、编辑、删除和搜索
- 实时Markdown预览
- 自动保存机制

2.2 技术选型

前端技术栈

- **HTML5 + CSS3**: 页面结构与样式设计
- 原生**JavaScript (ES6+)**: 业务逻辑实现，选择原生JS是为了深入理解Web开发基础
- **Marked.js**: Markdown解析与渲染
- **Font Awesome**: 图标库

后端技术栈

- **SpringBoot 2.7.14**: 快速构建Web应用框架
- **MyBatis Plus 3.5.3**: 简化数据库操作，提供强大的CRUD功能
- **JWT (JJWT 0.11.5)**: 无状态认证方案，适合前后端分离架构
- **MySQL**: 关系型数据库，数据持久化存储

选择关键技术的理由

1. **SpringBoot + MyBatis Plus**: 快速开发，减少样板代码，专注业务逻辑
2. **JWT**认证: 无状态设计，支持分布式部署，前后端完全分离
3. 原生**JavaScript**: 深入理解Web开发原理，不依赖框架的轻量级方案

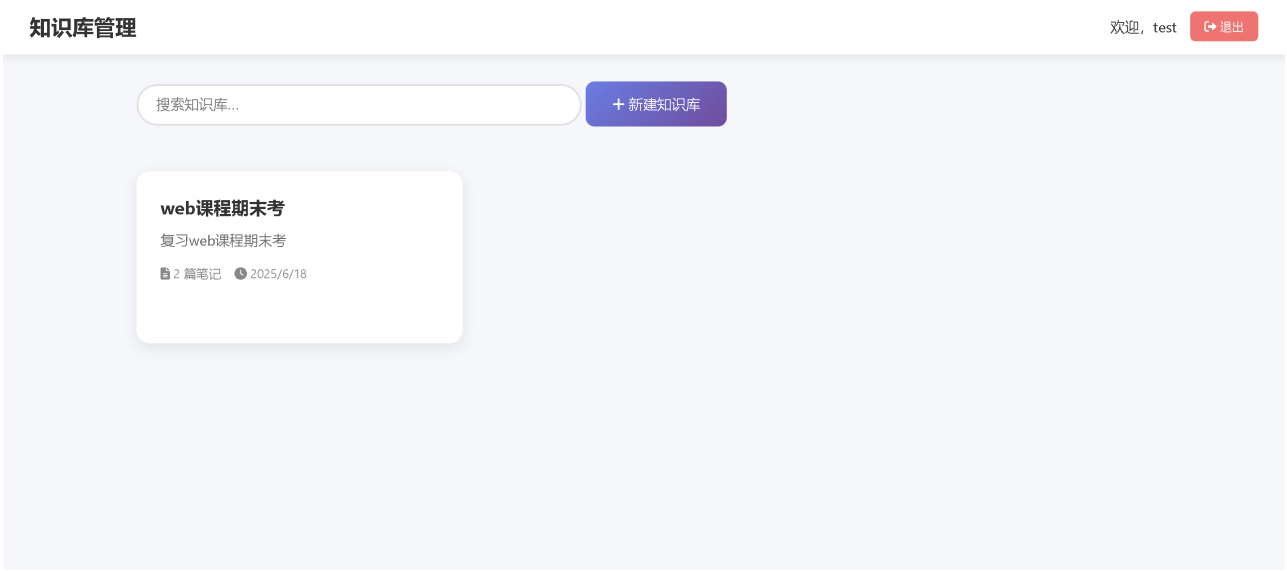
2.3 核心功能实现

2.3.1 用户登录

实现思路：

- 用户登录后生成JWT token，包含用户ID和用户名
- 前端将token存储在localStorage中
- 后端通过拦截器验证每个请求的token有效性
- token过期自动跳转到登录页

2.3.2 知识库管理



提供知识库的创建，删除，搜素，笔记数量和修改实际的展示

搜索功能支持针对知识库名称和说明进行模糊搜素，后端搜素为主，前端降级兜底。

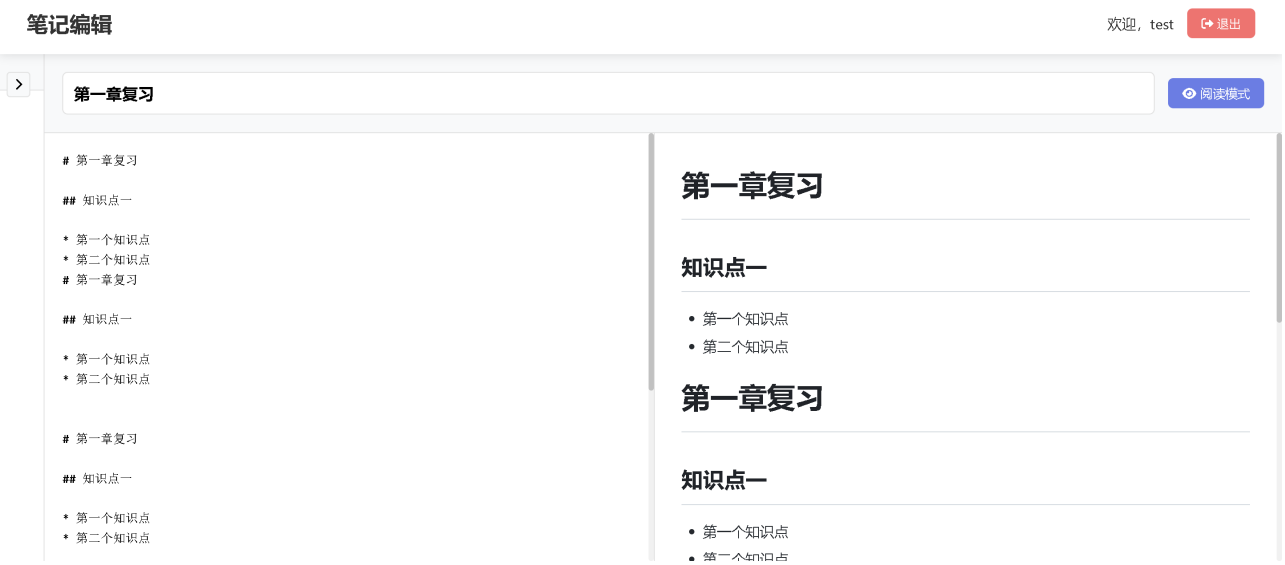
数据库设计：

```
CREATE TABLE `knowledge_base` (
    `id` bigint(20) NOT NULL AUTO_INCREMENT,
    `title` varchar(100) NOT NULL,
    `description` text,
    `user_id` bigint(20) NOT NULL,
    `note_count` int(11) DEFAULT 0,
    PRIMARY KEY (`id`)
);
```

关键逻辑：

- 每个知识库属于特定用户，通过`user_id`关联
- 维护`note_count`字段，实时统计笔记数量
- 支持按标题和描述进行模糊搜索

2.3.3 Markdown实时渲染系统



技术选型： 选择Marked.js作为Markdown解析引擎，原因包括：轻量级(50KB)、标准兼容、扩展性强、内置安全防护。

核心实现：

javascript

```
// 实时渲染函数
function updateMarkdownPreview() {
    const markdownContent = elements.markdownEditor.value;
    elements.markdownPreview.innerHTML = marked.parse(markdownContent);
    syncScrollPosition(); // 滚动同步
}

// 防抖优化，避免频繁渲染
elements.markdownEditor.addEventListener('input', (e) => {
    clearTimeout(renderTimeout);
    renderTimeout = setTimeout(() => {
        updateMarkdownPreview();
        autoSave(); // 触发自动保存
    }, 100);
});

// 利用parser函数实现渲染
function updateMarkdownPreview() {
    const markdownContent = elements.markdownEditor.value;
    elements.markdownPreview.innerHTML = marked.parse(markdownContent);
}
```

功能特色：

1. 双栏编辑界面：左侧编辑器，右侧实时预览
2. 编辑/阅读模式切换：满足不同使用场景
3. 滚动同步：编辑区与预览区位置同步
4. 语法支持：完整Markdown语法(标题、段落、列表、代码块、删除线、分割线等) + 代码高亮
5. 性能优化：防抖渲染 + DOM批量更新

移动端适配： 通过CSS媒体查询实现响应式设计，小屏设备自动切换为单栏模式，提供流畅的移动端编辑体验。

2.3.4 笔记编辑与管理

提供笔记的增加，删除，搜素，预览和编辑。

这里还提供了自动保存机制，在编辑后2秒自动保存：

```
const autoSave = () => {
    if (currentNote && hasUnsavedChanges()) {
        clearTimeout(saveTimeout);
        saveTimeout = setTimeout(async () => {
            await saveCurrentNote();
        }, 2000); // 2秒后自动保存
    }
};
```

2.3.5 数据持久化

- 使用MyBatis Plus的BaseMapper提供基础CRUD操作
- 自定义方法实现复杂查询（如搜索、统计）
- 事务管理确保数据一致性

2.4 遇到的挑战与解决

挑战1：JWT token管理与前后端状态同步

问题描述： 初期使用Session认证，后来改成更为安全JWT认证。

解决方案：

1. 改为JWT无状态认证
2. 创建统一的API客户端类管理token
3. 实现自动token验证和过期处理机制
4. 使用拦截器统一处理认证逻辑

```
// 统一的API请求处理
async request(url, options = {}) {
    const response = await fetch(fullUrl, config);
    if (response.status === 401) {
        this.redirectToLogin(); // 自动处理token过期
    }
    return response;
}
```

挑战2：前端数据状态管理与用户体验优化

问题描述： 原生JavaScript管理复杂的前端状态，需要处理加载状态、错误处理、数据同步等问题。

解决方案：

1. 设计清晰的状态管理模式，使用全局变量管理应用状态
2. 实现乐观更新：先更新前端状态，再调用API
3. 添加加载状态和错误提示，提升用户体验
4. 实现自动保存和本地缓存机制

3. 项目结构

3.1 整体项目结构

```
note-system/
├─ src/main/
|   ├─ java/fun/flyingpig/note/        # 后端Java代码
|   └─ resources/
|       ├─ static/                     # 前端静态资源
|       └─ application.yml             # 配置文件
├─ target/                             # 编译输出目录
├─ pom.xml                            # Maven依赖配置
└─ schema.sql                         # 数据库初始化脚本
```

3.2 后端结构 (src/main/java/fun/flyingpig/note/)

```
fun/flyingpig/note/
├─ config/                              # 配置类
|   ├─ MybatisPlusConfig.java          # MyBatis Plus配置
|   └─ GlobalExceptionHandler.java     # 全局异常处理
├─ controller/                          # 控制器层
|   ├─ AuthController.java             # 认证相关接口
|   ├─ KnowledgeBaseController.java    # 知识库管理接口
|   └─ NoteController.java             # 笔记管理接口
├─ dto/                                # 数据传输对象
|   ├─ LoginDTO.java                   # 登录请求DTO
|   ├─ LoginResponseDTO.java           # 登录响应DTO
|   ├─ KnowledgeBaseDTO.java           # 知识库DTO
|   └─ NoteDTO.java                    # 笔记DTO
|   ├─ Result.java                     # 统一响应结果
|   └─ PageResult.java                 # 分页结果
├─ entity/                             # 实体类
|   ├─ BaseEntity.java                 # 基础实体类
|   ├─ User.java                       # 用户实体
|   ├─ KnowledgeBase.java              # 知识库实体
|   └─ Note.java                       # 笔记实体
├─ mapper/                             # 数据访问层
|   ├─ UserMapper.java                 # 用户数据访问
|   ├─ KnowledgeBaseMapper.java        # 知识库数据访问
|   └─ NoteMapper.java                 # 笔记数据访问
├─ service/                            # 业务逻辑层
|   ├─ UserService.java                # 用户服务接口
|   ├─ KnowledgeBaseService.java       # 知识库服务接口
|   ├─ NoteService.java                # 笔记服务接口
|   └─ impl/                           # 服务实现类
|      ├─ UserServiceImpl.java
|      ├─ KnowledgeBaseServiceImpl.java
|      └─ NoteServiceImpl.java
├─ util/                               # 工具类
|   ├─ JwtUtil.java                    # JWT工具类
|   └─ UserContext.java                # 用户上下文工具
|   ├─ JwtProperties.java              # JWT配置属性
|   └─ JwtInterceptor.java             # JWT认证拦截器
└─ NoteSystemApplication.java          # SpringBoot启动类
```

3.3 前端结构 (src/main/resources/static/)

```
static/
├─ login.html                    # 登录页面
├─ dashboard.html                # 知识库管理页面
├─ notes.html                    # 笔记编辑页面
├─ css/                          # 样式文件目录
|   ├─ login.css                 # 登录页样式
|   ├─ dashboard.css             # 知识库管理页样式
|   └─ notes.css                 # 笔记编辑页样式
└─ js/                           # JavaScript文件目录
|   ├─ login.js                  # 登录页脚本
|   ├─ dashboard.js              # 知识库管理页脚本
|   └─ notes.js                  # 笔记编辑页脚本
```

3.4 数据库结构

```
-- 三张核心表的关系
User（用户表）
 ├─ id（主键）
 ├─ username
 ├─ password
 └─ email

KnowledgeBase（知识库表）
 ├─ id（主键）
 ├─ title
 ├─ description
 ├─ user_id（外键 → User.id）
 └─ note_count

Note（笔记表）
 ├─ id（主键）
 ├─ title
 ├─ content
 └─ knowledge_base_id（外键 → KnowledgeBase.id）
```

4. 学习总结与自我评估

4.1 主要学习收获

技术能力提升

1. 前后端分离架构理解：深入理解了现代Web应用的架构模式
2. **RESTful API**设计：掌握了标准的API设计规范和最佳实践
3. **JWT**认证机制：理解无状态认证的原理和实现方式
4. 数据库设计：学会了合理的表结构设计和关联关系处理
5. 前端状态管理：在没有框架的情况下实现复杂的前端状态管理

工程能力发展

1. 项目架构设计：学会了分层架构和职责分离
2. 代码组织：掌握了清晰的代码结构和命名规范
3. 错误处理：实现了完善的异常处理和用户反馈机制
4. 用户体验设计：关注加载状态、操作反馈等细节

4.2 项目完成情况自我评估

完成度评估 **(95%)**

已完成功能：

- ✅ 用户认证与权限控制
- ✅ 知识库完整CRUD操作
- ✅ 笔记编辑与管理
- ✅ Markdown实时预览
- ✅ 搜索功能
- ✅ 自动保存机制
- ✅ 响应式界面设计
- ✅ 完善的错误处理

技术亮点：

1. **JWT**无状态认证：实现了安全、可扩展的认证机制
2. 自动保存：2秒延迟自动保存，提升用户体验

3. 智能搜索: 后端搜索+前端降级的双重保障
4. 状态管理: 原生JS实现的复杂状态管理

代码质量评估

- 可维护性: 良好的分层架构，职责清晰
- 可扩展性: 支持添加新功能，如标签、分享等
- 安全性: JWT认证，输入验证，SQL注入防护
- 用户体验: 加载状态，错误提示，自动保存

4.3 未来可改进或扩展的方向

功能扩展

1. 富文本编辑器: 集成更强大的编辑器如TinyMCE
2. 文件上传: 支持图片、附件上传
3. 笔记分享: 实现笔记的公开分享功能
4. 标签系统: 为笔记添加标签分类
5. 协作功能: 支持多用户协作编辑

技术优化

1. 前端框架: 可考虑迁移到Vue.js或React
2. 缓存机制: 添加Redis缓存提升性能
3. 搜索优化: 集成ElasticSearch实现全文搜索
4. 微服务架构: 拆分为多个微服务
5. 容器化部署: 使用Docker进行部署

5. 原创性声明与引用

5.1 原创性声明

本项目为个人独立完成，所有核心业务逻辑、数据库设计、前端界面均为原创开发。项目架构设计、技术选型、功能实现等均体现了个人的学习成果和技术理解。

5.2 第三方资源引用

1. **Marked.js**: Markdown解析库 - https://marked.js.org/
2. **Font Awesome**: 图标库 - https://fontawesome.com/
3. **JJWT**: JWT处理库 - https://github.com/jwtk/jjwt
4. **MyBatis Plus**: 数据库操作框架 - https://baomidou.com/

5.3 参考资料

1. Spring Boot官方文档
2. MDN Web文档 (JavaScript API参考)
3. JWT标准规范 (RFC 7519)

所有代码实现均为原创，未直接复制任何第三方代码片段。第三方库仅用于基础功能支持，核心业务逻辑完全自主实现。

6.项目总结

本项目成功实现了一个功能完整的在线笔记和知识库系统，通过前后端分离架构，为了方便部署集成在一个JAVA项目中，启动JAR即可一键式部署，展示了现代Web应用开发的完整流程。