计算机科学与技术学院课程设计成绩单

课程名称：数据库系统课程设计

姓名		性别		学号		班级	
电话		综合成绩		成绩等级			
程序运行情况 （占总成绩 20%）							
程序功能完善程度 （占总成绩 10%）							
数据库逻辑结构的合理性 （占总成绩 10%）							
对问题的答辩情况 （占总成绩 40%）							
学生的工作态度与独立工作能力 （占总成绩 10%）							
设计报告的规范性 （占总成绩 10%）							

A：90~100 分　A-：85~89 分　B+：82~84 分　B：78~81 分　B-：75~77 分

C+：72~74 分　C：68~71 分　C-：64~67 分　D：60~63 分　F：<60 分

武汉科技大学计算机科学与技术学院制表

基于 Web 的博客网站

1 需求分析

1.1 设计意义

写博客可以记录自己的成长史，我们可以随时清晰的看到自己的努力，是很有意义的一件事情。写博客也可以分享自己的知识，利用博客的论坛交流，你可以和博客上的人互相交流经验，获取一些目前的行业知识，拓展自己的知识面 。
虽然有很多关于博客平台，但我更希望拥有一个属于自己的博客网站来记录与分享自己的学习历程。

1.2 功能实现

整体功能

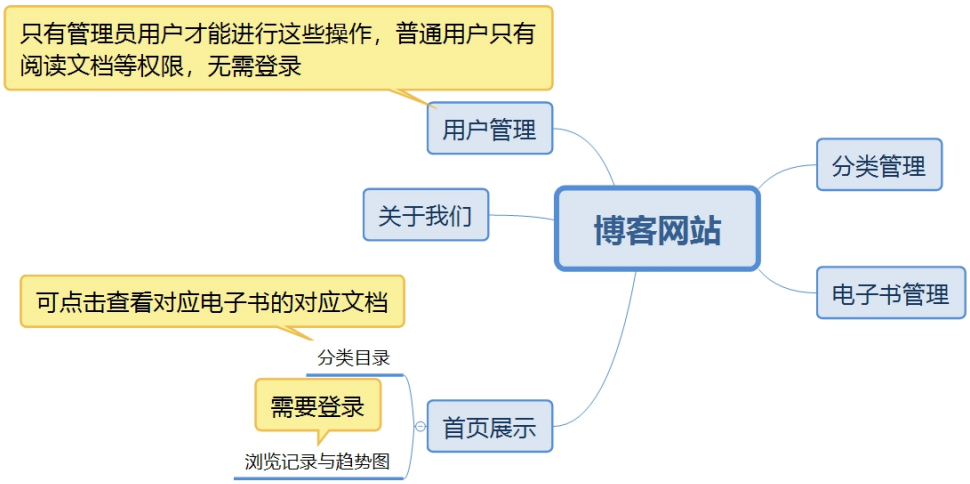

图 1.1 总体功能图

1.2.1 用户管理

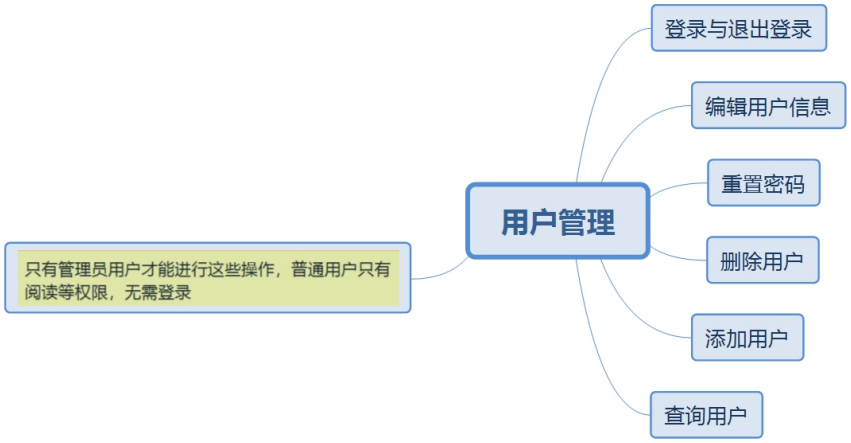

图 1.1 用户管理功能图

1.2.2 分类管理

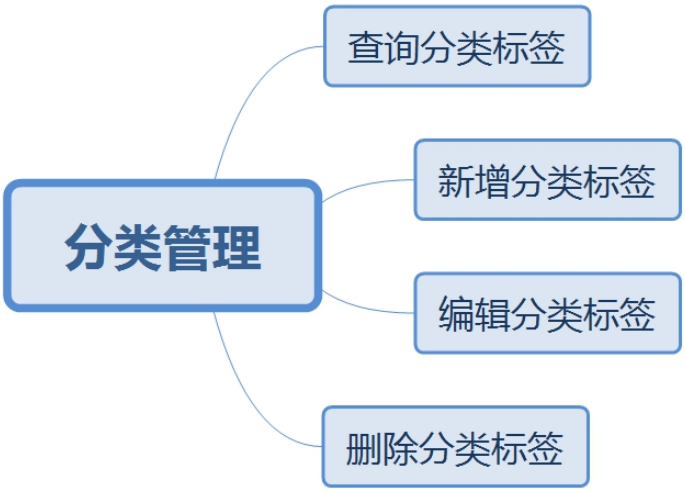

图 1.2 分类管理功能图

1.2.3 电子书管理

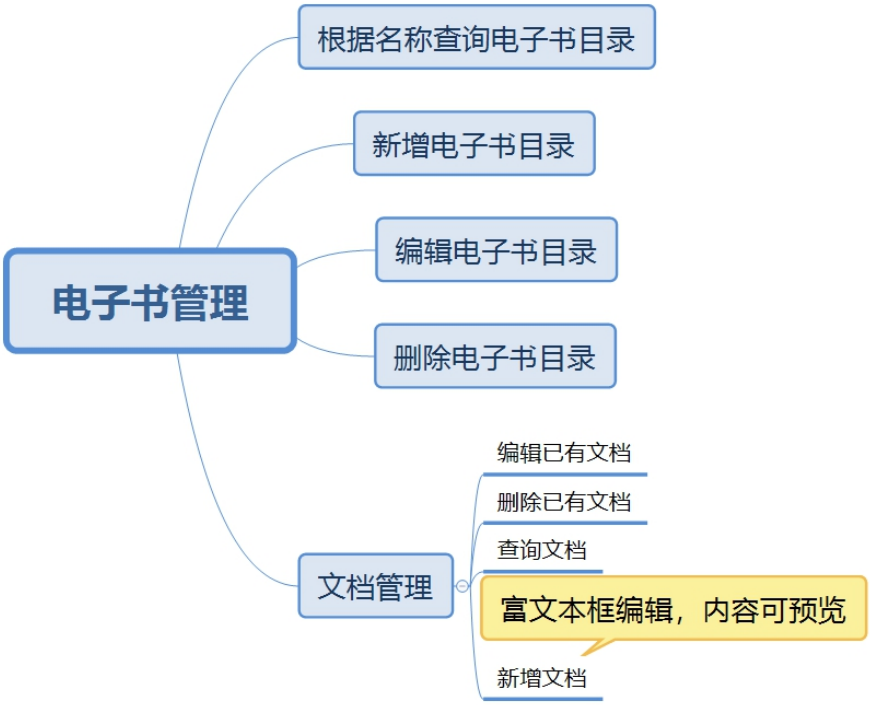

图 1.3 电子书管理功能图

2 概念结构设计

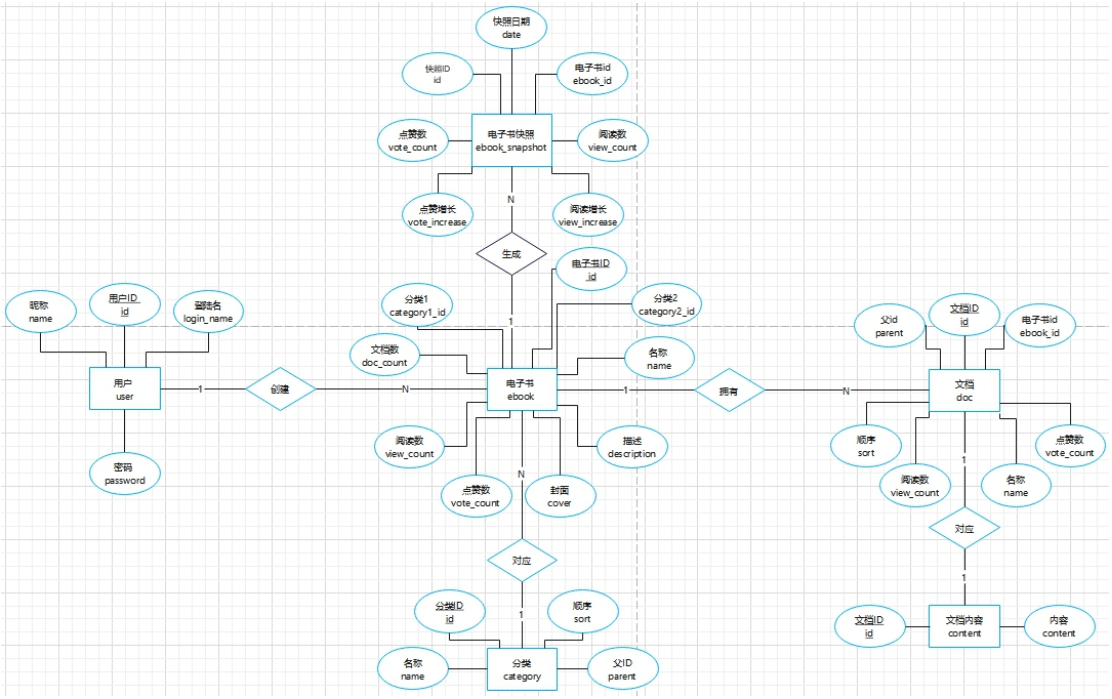

图 1.4 ER 图

3 逻辑结构设计

表 3.1 user (用户表)

属性名	含义	类型	说明
id	用户表主键 ID	bigint	由雪花算法生成，唯一，不能为空
login_name	用户登录名	varchar	不能为空
name	用户昵称	varchar	
password	用户密码	char	不能为空，MD5 加密存储
create_time	创建时间	datetime	用户创建时填入
update_time	更新时间	datetime	用户创建和信息更新时填入

表 3.2 ebook (电子书表)

属性名	含义	类型	说明
id	电子书表主键 ID	bigint	由雪花算法生成，唯一，不能为空
name	电子书名称	varchar	
category1_id	分类 1（父分类）	bigint	
category2_id	分类 2（子分类）	bigint	
description	电子书描述	varchar	
cover	电子书封面	varchar	图片 URL
doc_count	文档数	int	该电子书对应下的文档总数
view_count	阅读数	int	
vote_count	点赞数	int	
create_time	创建时间	datetime	创建时填入
update_time	更新时间	datetime	创建和信息更新时填入

表 3.3 doc (文档表)

属性名	含义	类型	说明
id	文档表主键 ID	bigint	由雪花算法生成，唯一，不能为空
ebook_id	电子书 ID	bigint	外键，引用 ebook 电子书表 ID
parent	父 ID	bigint	父文档 ID，引用 doc

			文档表的 id，不能为空，默认为 0
name	文档名称	varchar	
sort	顺序	int	值大的会优先展示
view_count	阅读数	int	
vote_count	点赞数	int	
create_time	创建时间	datetime	创建时填入
update_time	更新时间	datetime	创建和信息更新时填入

表 3.4 content (文档内容表)

属性名	含义	类型	说明
id	内容表主键 ID	bigint	外键，引用 doc 文档表主键 ID
content	文档内容	mediumtext	存储 HTML 代码

表 3.5 category(分类表)

属性名	含义	类型	说明
id	文档表主键 ID	bigint	由雪花算法生成，唯一，不能为空
parent	父 ID	bigint	父 ID，引用 category 文档表的 id，不能为空，默认为 0
name	分类名称	varchar	
sort	顺序	int	值大的会优先展示

表 3.6 ebook_snapshot(电子书快照表)

属性名	含义	类型	说明
id	文档表主键 ID	bigint	由雪花算法生成，唯一，不能为空
ebook_id	电子书 ID	bigint	外键，引用 ebook 电子书表 ID
date	快照日期	date	
view_count	阅读数	int	
vote_count	点赞数	int	
view_increase	阅读增长	int	
vote_increase	点赞增长	int	

4 开发工具

开发环境：Windows10，JDK1.8，MySql8.0.23，Redis6.0
开发语言：Java，Html，JavaScript，SQL

5 具体实现

5.1 登录与首页功能展示

5.1.1 效果展示

浏览器输入网址进入主页（未登录）

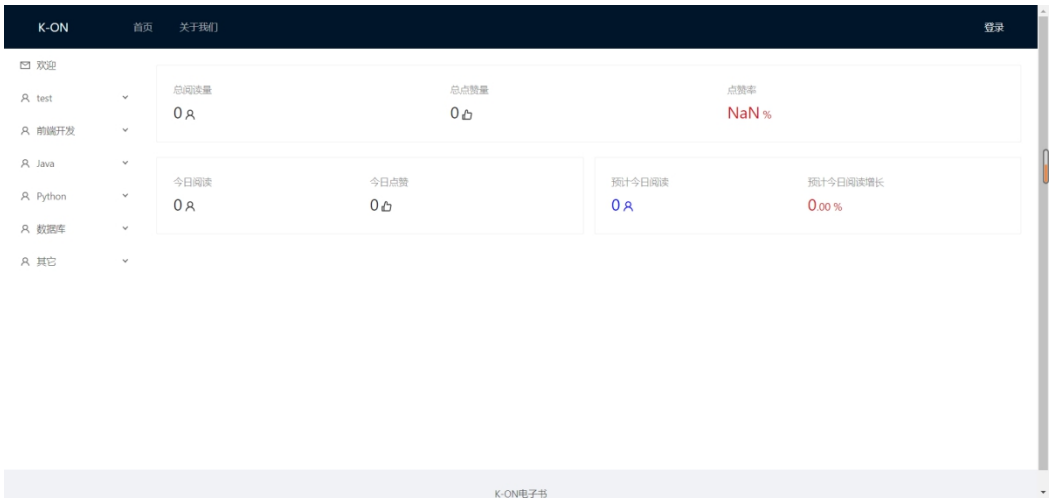

图 5.1 首页界面展示图（未登录）

点击右上角登录，输入账号和密码

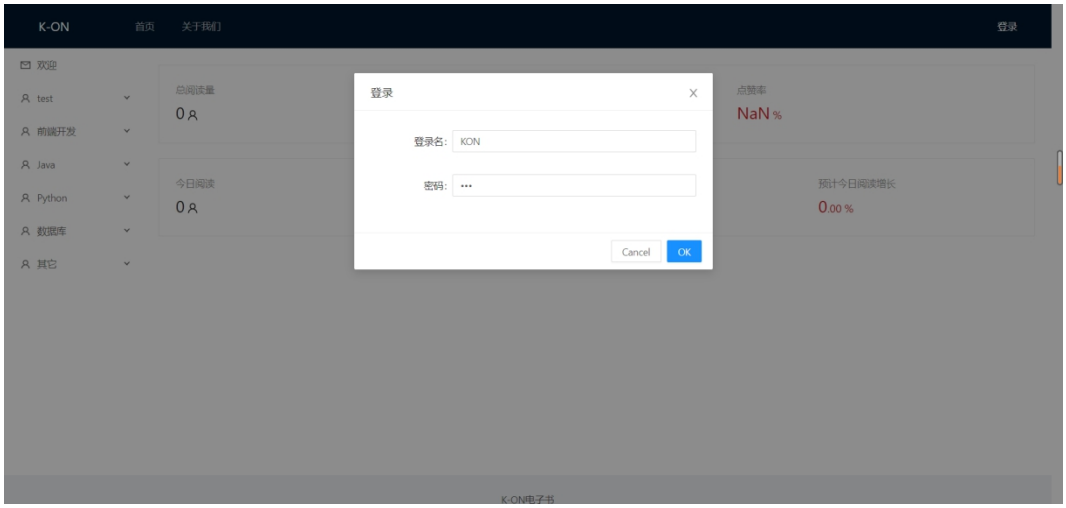

图 5.2 登录界面展示图

图 5.3 首页展示图（已登录）

5.1.2 实现细节

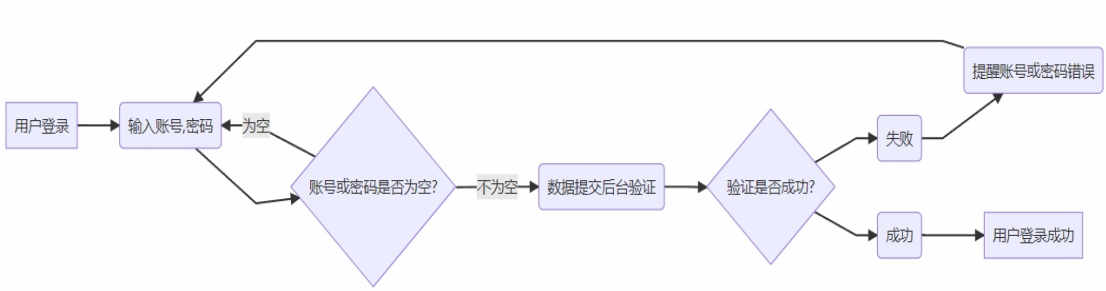

图 5.4 登录流程图

后台根据用户账号从数据库中查询对应用户信息，若比对成功，则生成 token（包含用户信息以的加密字符串），将 token 存入 redis 中，并返回给前端，用作用户身份校验。登录状态下，再次刷新首页，后台返回 总阅读数、总点赞数、今日阅读数、今日点赞数、今日预计阅读数、今日预计阅读增长以及近期生成的快照（一般为 30 天）。

核心代码如下：
登录逻辑处理

```
1.      /**
2.       * 登录
3.       */
4.      public UserLoginResp login(UserLoginReq req){
5.          User userDb = selectByLoginName(req.getLoginName());
6.          if (ObjectUtils.isEmpty(userDb)) {
7.              //用户名不存在
```

```
            LOG.info("用户名不存在，{} ", req.getLoginName());
            throw new BusinessException(BusinessExceptionCode.LOGIN_USER_ERROR);
        } else {
            if(userDb.getPassword().equals(req.getPassword())) {
                //登录成功
                UserLoginResp userLoginResp = CopyUtil.copy(userDb,UserLoginResp.class);
                return userLoginResp;
            } else {
                //密码不对
                                    LOG.info("秘密不对，输入密码：{}，数据库密码：{}，",req.getPassword(),userDb.getPassword());
                throw new BusinessException(BusinessExceptionCode.LOGIN_USER_ERROR);
            }
        }
    }
```

根据登录账号查询用户信息的 SQL

```
  <select id="selectByExample" parameterType="com.jiawa.wiki.domain.UserExample" resultMap="BaseResultMap">
    select
    <if test="distinct">
      distinct
    </if>
    <include refid="Base_Column_List" />
    from user
    <if test="_parameter != null">
      <include refid="Example_Where_Clause" />
    </if>
    <if test="orderByClause != null">
      order by ${orderByClause}
    </if>
  </select>
```

```
    <!-- 获取首页数值数据：总阅读数、总点赞数、今日阅读数、今日点赞数、今日预计阅读数、今日预计阅读增长 -->
    <select id="getStatistic" resultType="com.jiawa.wiki.resp.StatisticResp">
        select
            t1.`date` as `date`,
            sum(t1.view_count) as viewCount,
            sum(t1.vote_count) as voteCount,
            sum(t1.view_increase) as viewIncrease,
            sum(t1.vote_increase) as voteIncrease
        from
            ebook_snapshot t1
        where
            t1.`date` >= date_sub(curdate(), interval 1 day)
```

```
        group by
            t1.`date`
        order by
            t1.`date` asc;
    </select>
```

```
<!-- 查询近 30 天生成的快照 -->
<select id="get30Statistic" resultType="com.jiawa.wiki.resp.StatisticResp">
        select
            t1.`date` as `date`,
            sum(t1.view_increase) as viewIncrease,
            sum(t1.vote_increase) as voteIncrease
        from
            ebook_snapshot t1
        where
            t1.`date` between date_sub(curdate(), interval 30 day) and date_sub(curdate(), interval 1 day)
        group by
            t1.`date`
        order by
            t1.`date` asc;
    </select>
```

项目在启动后，会开启快照生成，
为所有的电子书生成一条今天的记录（如果还没有）；
更新总阅读数、总点赞数；
更新今日阅读数、今日点赞数

```
    /**
     * 自定义 cron 表达式跑批
     * 只有等上一次执行完成，下一次才会在下一个时间点执行，错过就错过
     */
    @Scheduled(cron = "0 0/1 * * * ?")
    public void doSnapshot() {
        // 增加日志流水号
        MDC.put("LOG_ID", String.valueOf(snowFlake.nextId()));
        LOG.info("生成今日电子书快照开始");
        Long start = System.currentTimeMillis();
        ebookSnapshotService.genSnapshot();
        LOG.info("生成今日电子书快照结束，耗时：{}毫秒", System.currentTimeMillis() - start);
    }
```

```
<update id="genSnapshot">
        insert into ebook_snapshot(ebook_id, `date`, view_count, vote_count, view_increase, vote_increase)
```

```
        select t1.id, curdate(), 0, 0, 0, 0
        from ebook t1
        where not exists(select 1
                         from ebook_snapshot t2
                         where t1.id = t2.ebook_id
                           and t2.`date` = curdate());

        update ebook_snapshot t1, ebook t2
        set t1.view_count = t2.view_count,
            t1.vote_count = t2.vote_count
        where t1.`date` = curdate()
          and t1.ebook_id = t2.id;

        update ebook_snapshot t1 left join (select ebook_id, view_count, vote_count
                                            from ebook_snapshot
                                              where `date` = date_sub(curdate(), interval 1 day)) t2
            on t1.ebook_id = t2.ebook_id
        set t1.view_increase = (t1.view_count - ifnull(t2.view_count, 0)),
            t1.vote_increase = (t1.vote_count - ifnull(t2.vote_count, 0))
        where t1.`date` = curdate();
    </update>
```

5.2 用户管理功能展示

5.2.1 效果展示

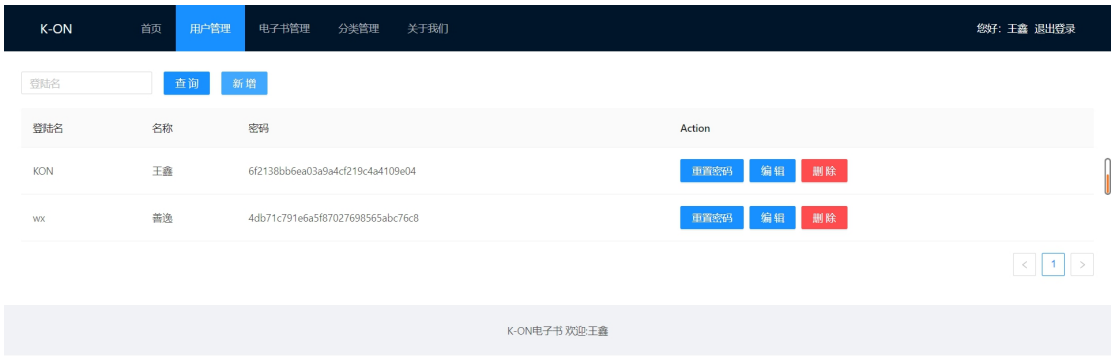

图 5.5 用户管理界面展示图

新增用户

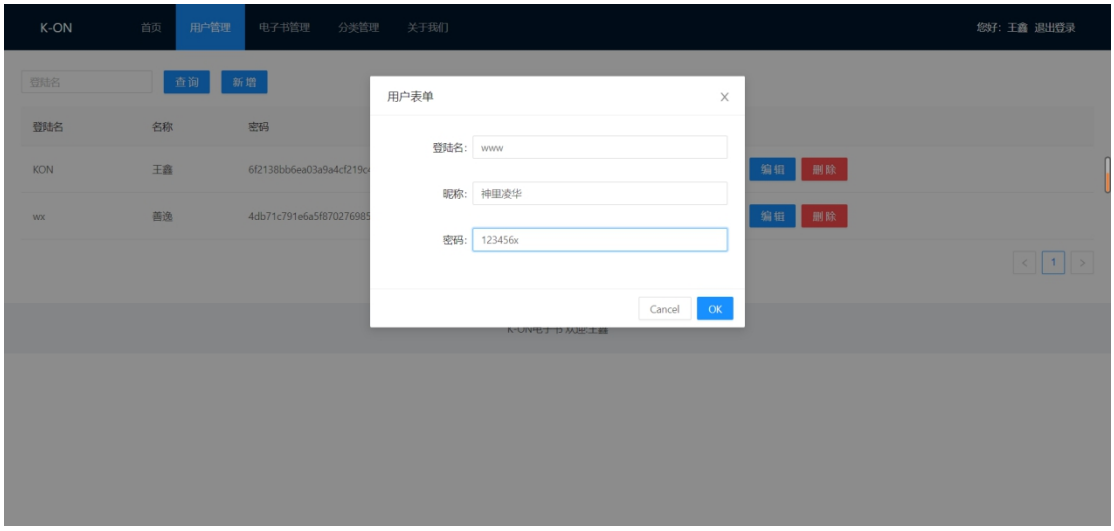

图 5.6 新增用户界面展示图

删除用户

图 5.7 删除用户界面展示图

查询用户

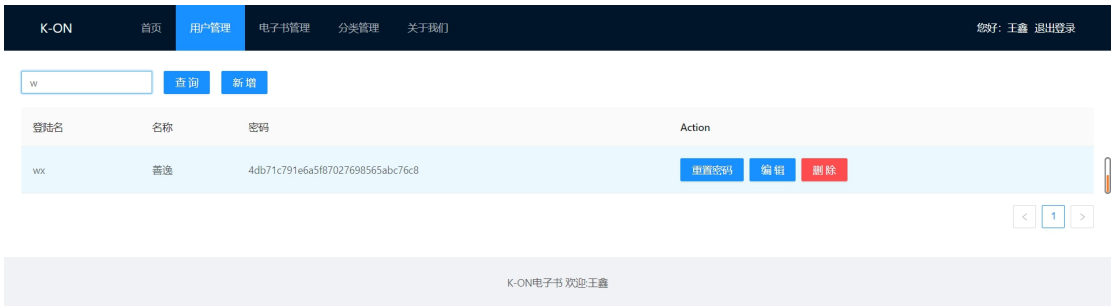

图 5.8 查询用户界面展示图

编辑用户

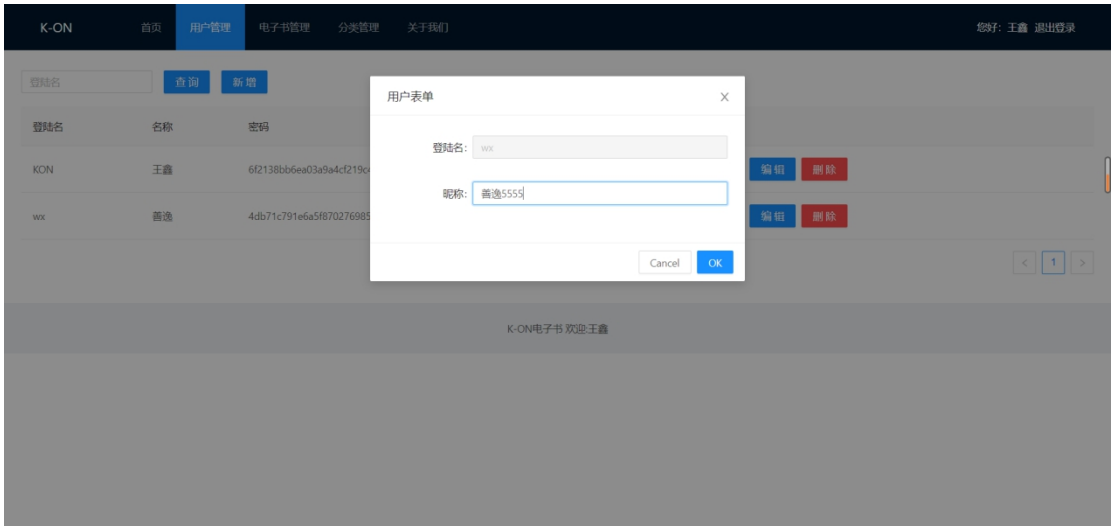

图 5.9 编辑用户界面展示图

5.2.2 实现细节

主键 ID 的值由雪花算法生成，用户密码使用 MD5 加密算法后进行存储。
查询用户核心代码

```
1.      /**
2.       * 分页查询所有用户
3.       * @param req
4.       * @return
5.       */
6.      public PageResp<UserQueryResp> list(UserQueryReq req){
7.          UserExample userExample = new UserExample();
8.          UserExample.Criteria criteria = userExample.createCriteria();
9.          if(!ObjectUtils.isEmpty(req.getLoginName())) {//判断账号是否为空
10.             criteria.andLoginNameEqualTo(req.getLoginName());
11.         }
12.         PageHelper.startPage(req.getPage(), req.getSize());//设置分页插件
13.         List<User> userList = userMapper.selectByExample(userExample);//数据查询
14.         PageInfo<User> pageInfo = new PageInfo<>(userList);
15.         LOG.info("总行数：{}",pageInfo.getTotal());
16.         LOG.info("总页数：{}",pageInfo.getPages());
17.          List<UserQueryResp> list = CopyUtil.copyList(userList, UserQueryResp.class);//数据转换，传递
    给前端
18.         PageResp<UserQueryResp> pageResp = new PageResp();
19.         pageResp.setTotal(pageInfo.getTotal());//获取数据条数
20.         pageResp.setList(list);
21.         return pageResp;
22.     }
```

```
1.    <select id="selectByExample" parameterType="com.jiawa.wiki.domain.UserExample" resultMap="BaseResu
    ltMap">
2.        select
3.        <if test="distinct">
4.          distinct
5.        </if>
6.        <include refid="Base_Column_List" />
7.        from user
8.        <if test="_parameter != null">
9.          <include refid="Example_Where_Clause" />
10.       </if>
11.       <if test="orderByClause != null">
12.         order by ${orderByClause}
13.       </if>
14.     </select>
```

其它功能的代码类似

5.3 分类管理功能展示

5.3.1 效果展示

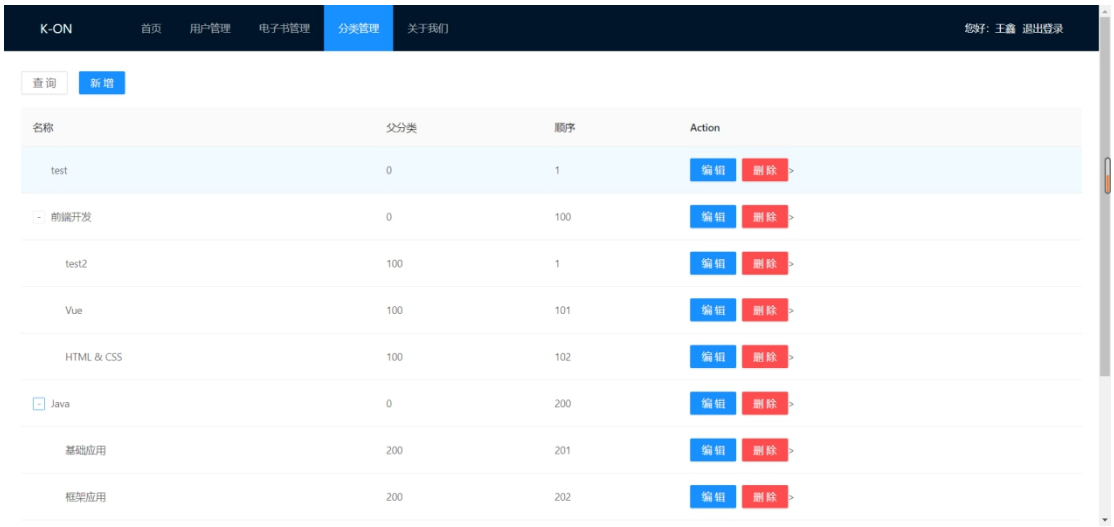

图 5.10 分类管理界面展示图

编辑分类

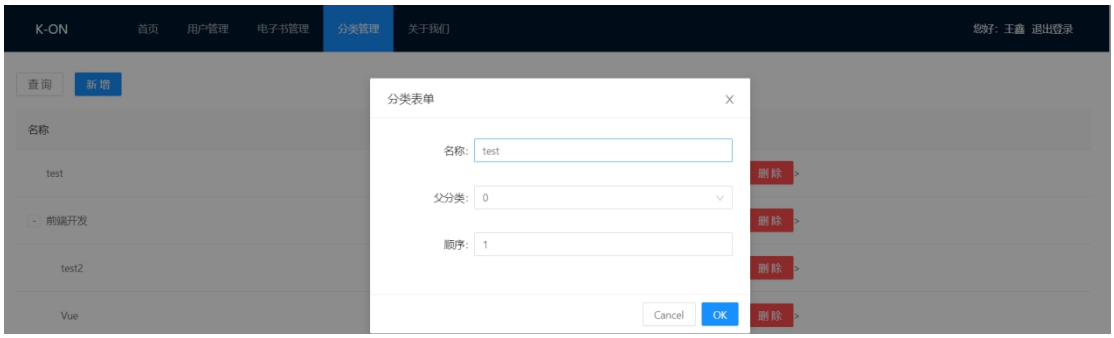

图 5.11 编辑分类界面展示图

新增分类

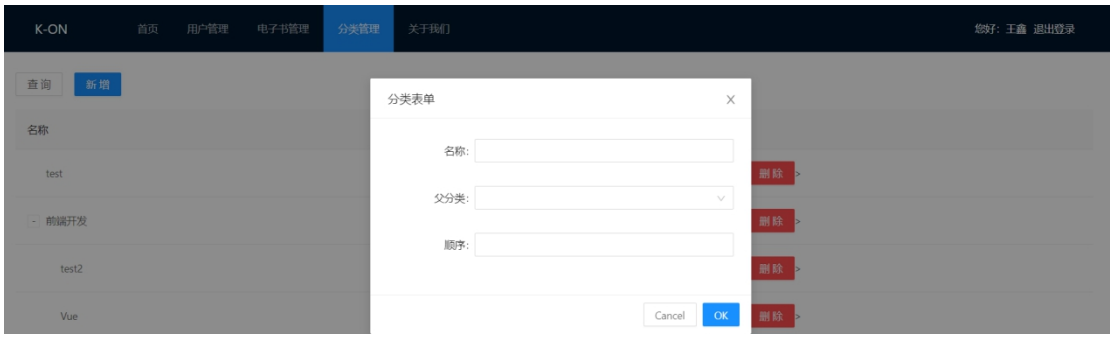

图 5.12 新增分类界面展示图

删除分类

图 5.13 新增分类界面展示图

5.3.2 实现细节

以分类新增与编辑为例

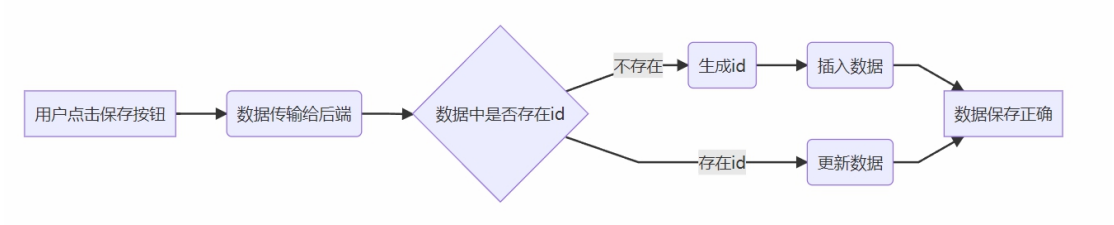

图 5.14 保存分类流程图

核心代码：

```
1.    /**
2.       * 保存分类
3.       * @param req
4.       */
5.      public void save(CategorySaveReq req){
```

```
        Category category = CopyUtil.copy(req,Category.class);
        if(ObjectUtils.isEmpty(req.getId())){
            //新增
            category.setId(snowFlake.nextId());
            categoryMapper.insertSelective(category);
        } else{
            //更新
            categoryMapper.updateByPrimaryKey(category);
        }
    }
```

```
<!--插入数据-->
  <insert id="insertSelective" parameterType="com.jiawa.wiki.domain.Category">
    insert into category
    <trim prefix="(" suffix=")" suffixOverrides=",">
      <if test="id != null">
        id,
      </if>
      <if test="parent != null">
        parent,
      </if>
      <if test="name != null">
        `name`,
      </if>
      <if test="sort != null">
        sort,
      </if>
    </trim>
    <trim prefix="values (" suffix=")" suffixOverrides=",">
      <if test="id != null">
        #{id,jdbcType=BIGINT},
      </if>
      <if test="parent != null">
        #{parent,jdbcType=BIGINT},
      </if>
      <if test="name != null">
        #{name,jdbcType=VARCHAR},
      </if>
      <if test="sort != null">
        #{sort,jdbcType=INTEGER},
      </if>
    </trim>
  </insert>
```

```
<!--  更新数据-->
  <update id="updateByPrimaryKey" parameterType="com.jiawa.wiki.domain.Category">
    update category
    set parent = #{parent,jdbcType=BIGINT},
      `name` = #{name,jdbcType=VARCHAR},
      sort = #{sort,jdbcType=INTEGER}
    where id = #{id,jdbcType=BIGINT}
  </update>
```

5.4 电子书管理功能展示

5.4.1 效果展示

图 5.15 电子书管理页面展示图

新增电子书

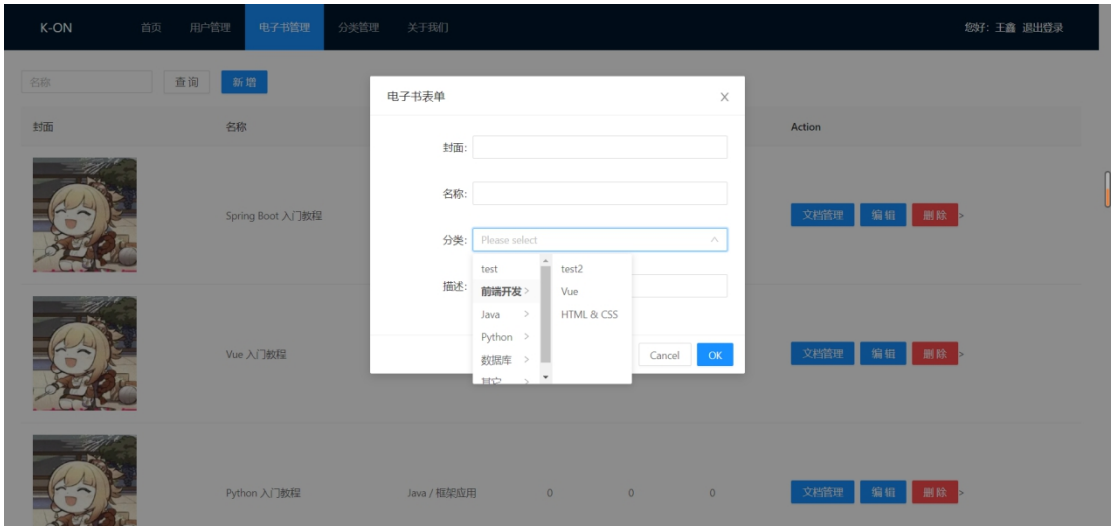

图 5.16 新增电子书展示图

编辑电子书

图 5.17 编辑电子书展示图

查询功能（下图查询的是名称包含字母 s 的电子书）

图 5.18 查询电子书展示图

文档编辑

图 5.19 文档编辑及内容预览展示图

新增文档

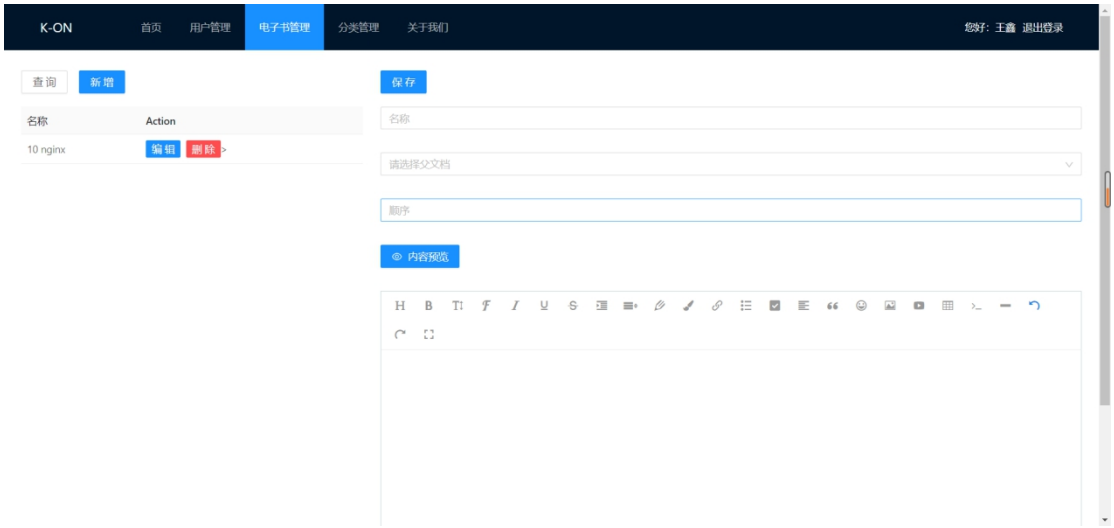

图 5.20 新增文档展示图

5.4.2 实现细节

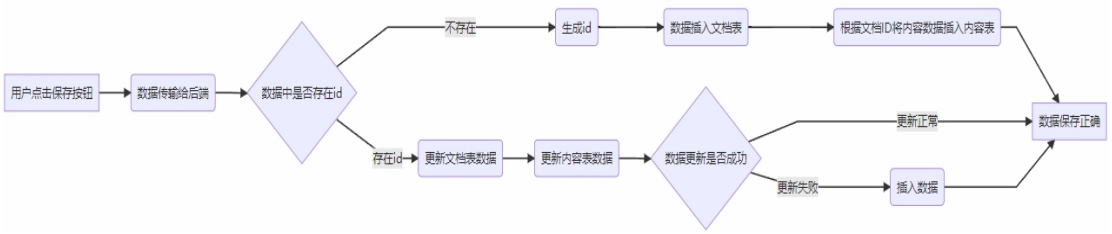

图 5.21 保存文档流程图

前端使用 wangEditor 来实现富文本框的编辑，wangEditor 是开源 Web 的 富文本编辑器，简洁易用，功能强大。

核心代码：

```
1.    /**
2.         * 保存
3.         * @param req
4.         */
5.        @Transactional
6.        public void save(DocSaveReq req){
7.            Doc doc = CopyUtil.copy(req,Doc.class);
8.            Content content = CopyUtil.copy(req,Content.class);
9.            if(ObjectUtils.isEmpty(req.getId())){
10.               //新增
11.               doc.setId(snowFlake.nextId());
12.               doc.setViewCount(0);
13.               doc.setVoteCount(0);
14.               docMapper.insertSelective(doc);
15.               //添加内容
```

```
            content.setId(doc.getId());
            contentMapper.insertSelective(content);
        } else{
            //更新
            docMapper.updateByPrimaryKey(doc);
            int count = contentMapper.updateByPrimaryKeyWithBLOBs(content);
            if(count == 0){  //更新失败则插入数据
                contentMapper.insert(content);
            }
        }
    }
```

```
<!--  插入文档内容-->
  <insert id="insertSelective" parameterType="com.jiawa.wiki.domain.Doc">
    insert into doc
    <trim prefix="(" suffix=")" suffixOverrides=",">
      <if test="id != null">
        id,
      </if>
      <if test="ebookId != null">
        ebook_id,
      </if>
      <if test="parent != null">
        parent,
      </if>
      <if test="name != null">
        `name`,
      </if>
      <if test="sort != null">
        sort,
      </if>
      <if test="viewCount != null">
        view_count,
      </if>
      <if test="voteCount != null">
        vote_count,
      </if>
    </trim>
    <trim prefix="values (" suffix=")" suffixOverrides=",">
      <if test="id != null">
        #{id,jdbcType=BIGINT},
      </if>
      <if test="ebookId != null">
        #{ebookId,jdbcType=BIGINT},
```

```
        </if>
        <if test="parent != null">
          #{parent,jdbcType=BIGINT},
        </if>
        <if test="name != null">
          #{name,jdbcType=VARCHAR},
        </if>
        <if test="sort != null">
          #{sort,jdbcType=INTEGER},
        </if>
        <if test="viewCount != null">
          #{viewCount,jdbcType=INTEGER},
        </if>
        <if test="voteCount != null">
          #{voteCount,jdbcType=INTEGER},
        </if>
      </trim>
    </insert>
```

6 总结

1. 在保存信息的 service 层方法上需要添加注解@Transactional，加上这个注解，如果发生 unchecked exception，就会发生 rollback，它会在事务开始时，通过 AOP 机制，生成一个代理 connection 对象，并将其放入 DataSource 实例的某个与 DataSourceTransactionManager 相关的某处容器中。在接下来的整个事务中，客户代码都应该使用该 connection 连接数据库执行所有数据库命令。事务结束时，回滚在第 1 步骤中得到的代理 connection 对象上执行的数据库命令，然后关闭该代理 connection 对象。

2. 生成今日电子书快照在数据库层有两种方案：

方案一（ID 不连续）：

删除今天的数据

为所有的电子书生成一条今天的记录

更新总阅读数、总点赞数

更新今日阅读数、今日点赞数

方案二（ID 连续）：

为所有的电子书生成一条今天的记录（如果还没有）

更新总阅读数、总点赞数

更新今日阅读数、今日点赞数

在查询当天阅读数据与近 30 天阅读数据都需要用到函数 date_sub(date,interval expr type)，从日期减去指定的时间间隔。

t1.`date` between date_sub(curdate(), interval 30 day) and date_sub(curdate(), interval 1 day)就表示近 30 天。

3. 使用 WebSocket 进行消息推送，如点赞消息，一个用户对一篇文章点赞后 24 小时内不能再对该文章点赞(没有做点赞状态持久化)。因为使用的用单点 token

来验证用户身份, token 在创建时需要设置过期, 过期后用户就需要重新登录, 没有做到自动刷新登录状态。