**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**

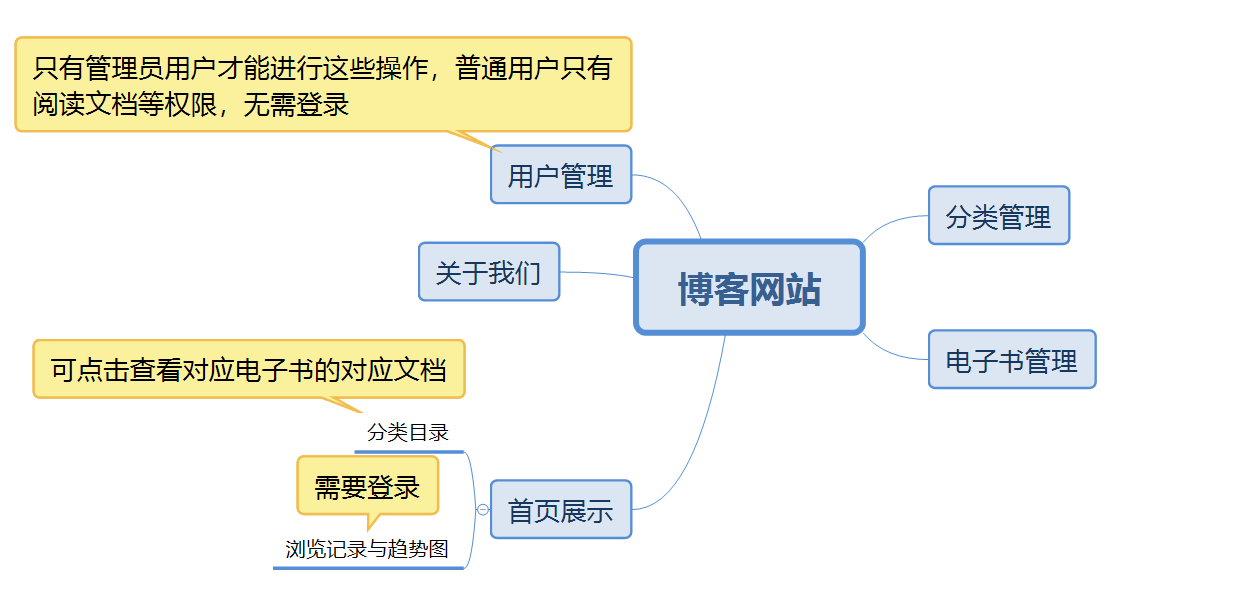


图1.1总体功能图

**1.2.1 用户管理**

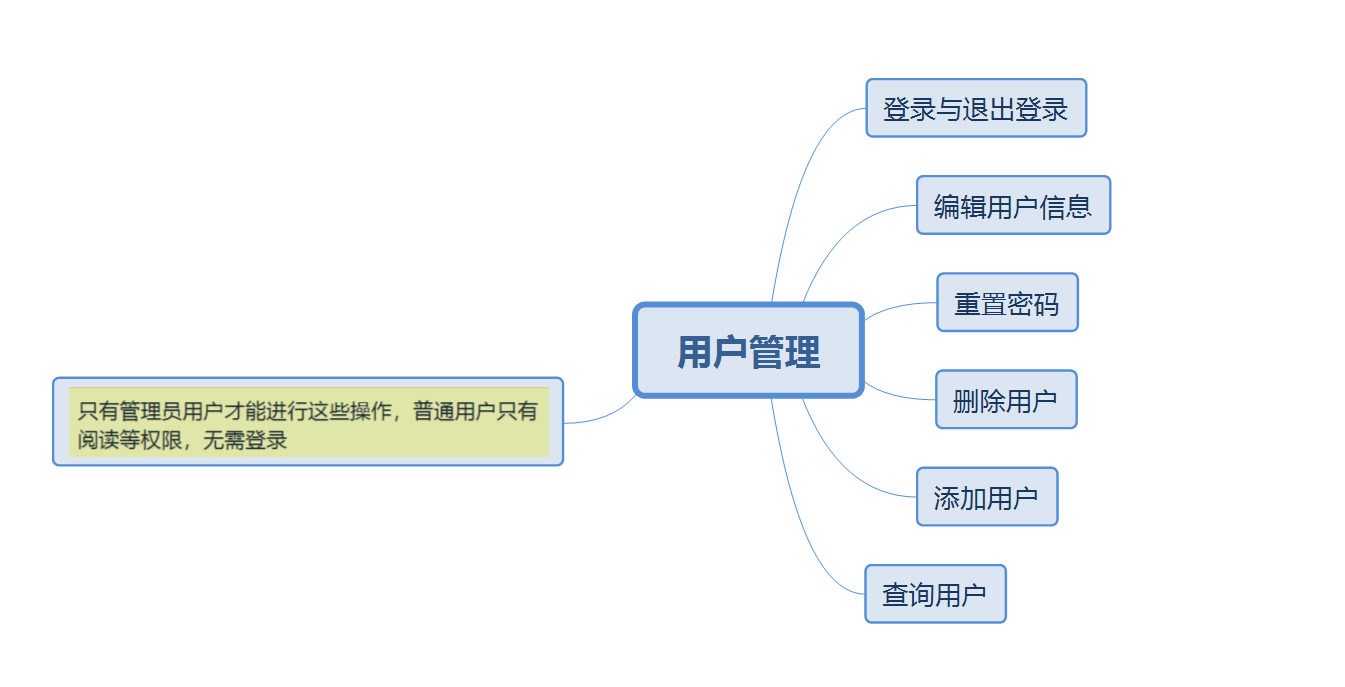


图1.1用户管理功能图

**1.2.2 分类管理**

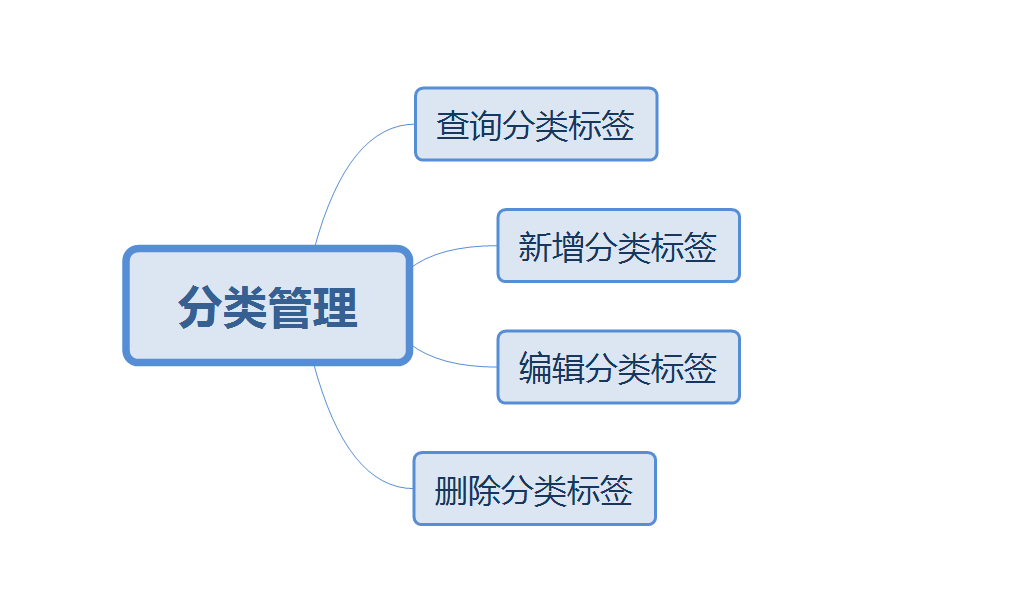


图1.2分类管理功能图

**1.2.3 电子书管理**

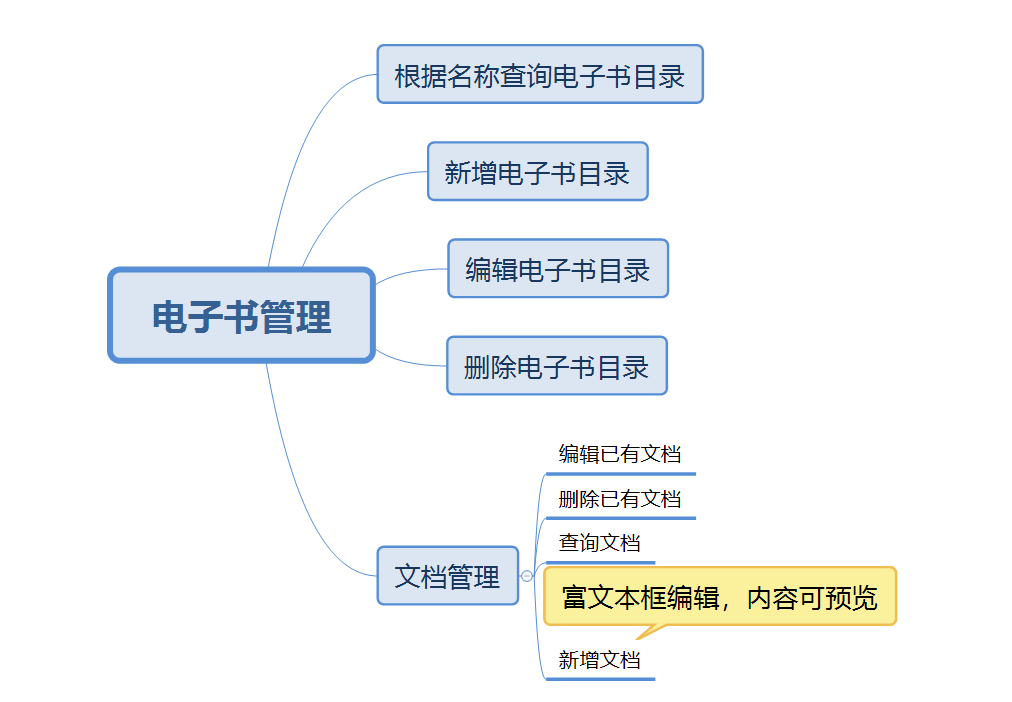


图1.3电子书管理功能图

## 2概念结构设计

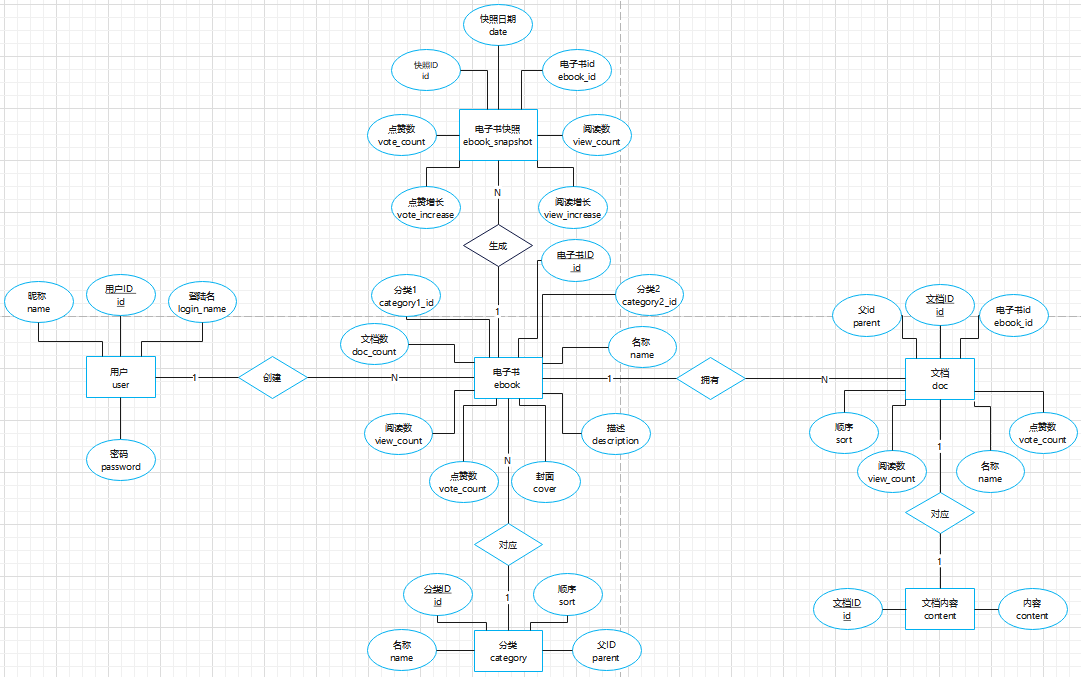


图1.4 ER图

## 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）

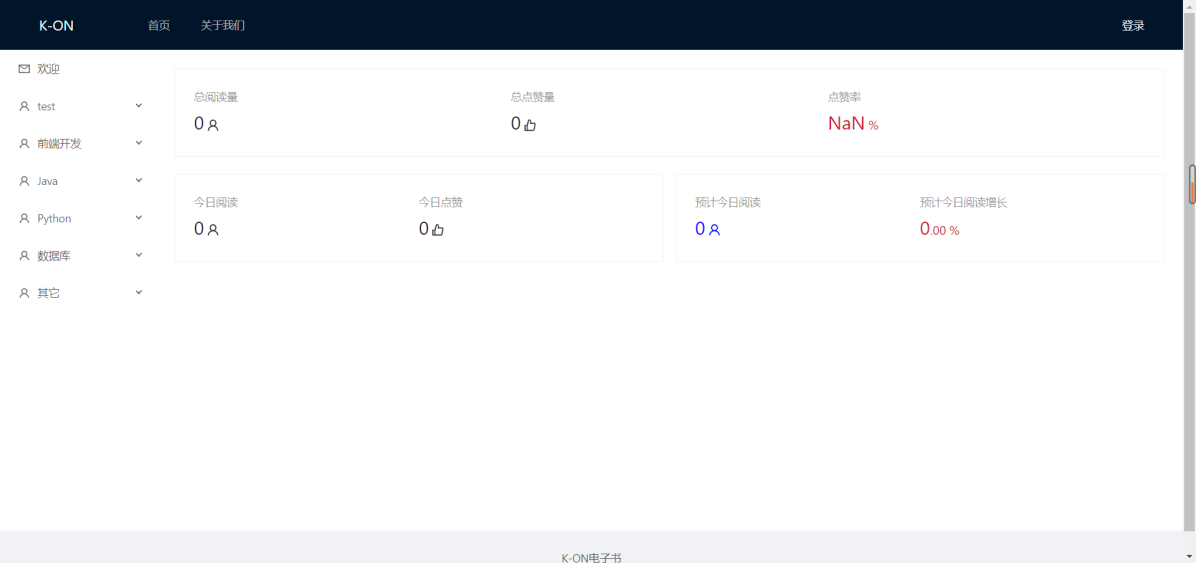


图5.1首页界面展示图（未登录）

点击右上角登录，输入账号和密码

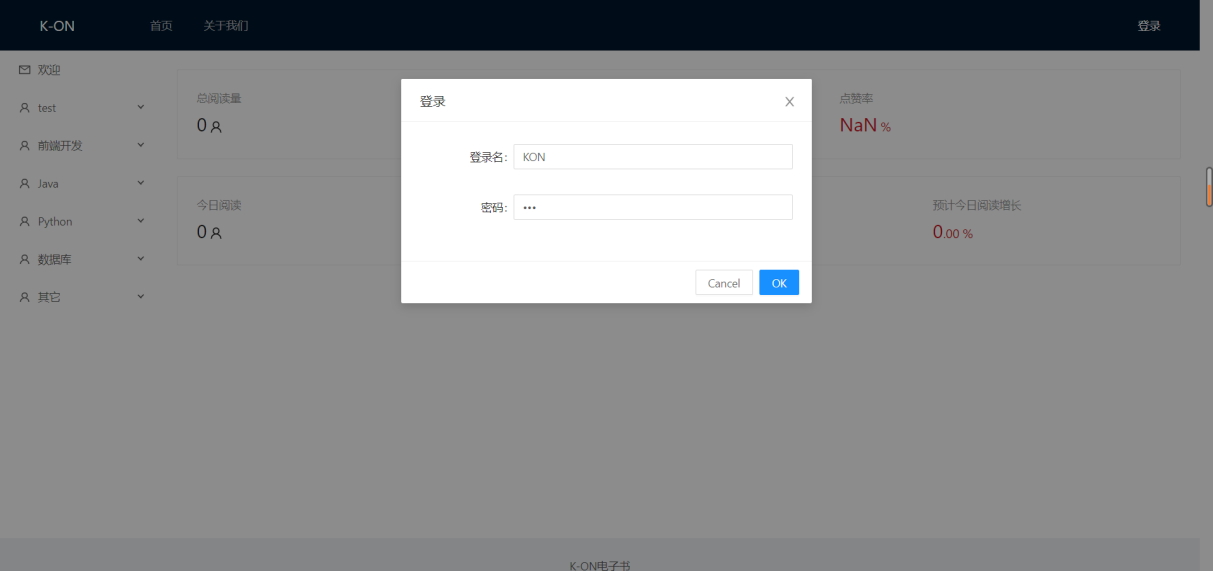


图5.2登录界面展示图



图5.3首页展示图（已登录）

**5.1.2 实现细节**

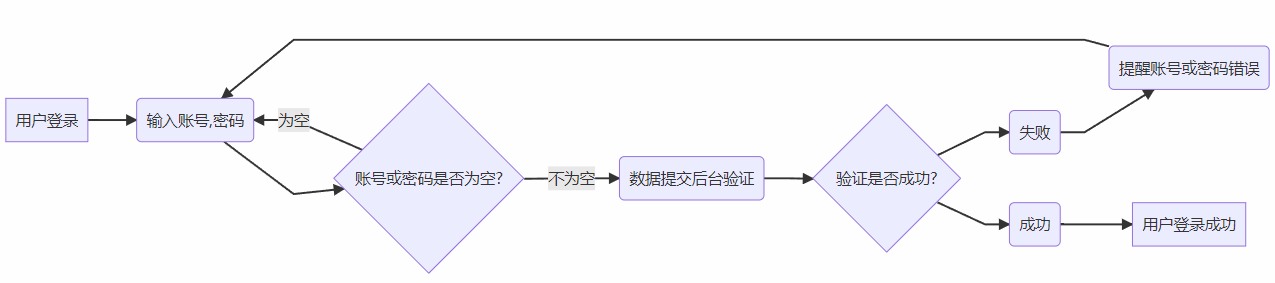


图5.4登录流程图

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
8. LOG.info("用户名不存在，{} ", req.getLoginName());
9. **throw** **new** BusinessException(BusinessExceptionCode.LOGIN\_USER\_ERROR);
10. } **else** {
11. **if**(userDb.getPassword().equals(req.getPassword())) {
12. //登录成功
13. UserLoginResp userLoginResp = CopyUtil.copy(userDb,UserLoginResp.**class**);
14. **return** userLoginResp;
15. } **else** {
16. //密码不对
17. LOG.info("秘密不对, 输入密码：{}, 数据库密码：{}, ",req.getPassword(),userDb.getPassword());
18. **throw** **new** BusinessException(BusinessExceptionCode.LOGIN\_USER\_ERROR);
19. }
20. }
21. }

根据登录账号查询用户信息的SQL

1. **<select** id="selectByExample" parameterType="com.jiawa.wiki.domain.UserExample" resultMap="BaseResultMap"**>**
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>**
15. <!-- 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 -->
16. **<select** id="getStatistic" resultType="com.jiawa.wiki.resp.StatisticResp"**>**
17. select
18. t1.`date` as `date`,
19. sum(t1.view\_count) as viewCount,
20. sum(t1.vote\_count) as voteCount,
21. sum(t1.view\_increase) as viewIncrease,
22. sum(t1.vote\_increase) as voteIncrease
23. from
24. ebook\_snapshot t1
25. where
26. t1.`date` **>**= date\_sub(curdate(), interval 1 day)
27. group by
28. t1.`date`
29. order by
30. t1.`date` asc;
31. **</select>**
32. <!-- 查询近30天生成的快照 -->
33. **<select** id="get30Statistic" resultType="com.jiawa.wiki.resp.StatisticResp"**>**
34. select
35. t1.`date` as `date`,
36. sum(t1.view\_increase) as viewIncrease,
37. sum(t1.vote\_increase) as voteIncrease
38. from
39. ebook\_snapshot t1
40. where
41. t1.`date` between date\_sub(curdate(), interval 30 day) and date\_sub(curdate(), interval 1 day)
42. group by
43. t1.`date`
44. order by
45. t1.`date` asc;
46. **</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

1. /\*\*
2. \* 自定义cron表达式跑批
3. \* 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
4. \*/
5. @Scheduled(cron = "0 0/1 \* \* \* ?")
6. **public** **void** doSnapshot() {
7. // 增加日志流水号
8. MDC.put("LOG\_ID", String.valueOf(snowFlake.nextId()));
9. LOG.info("生成今日电子书快照开始");
10. Long start = System.currentTimeMillis();
11. ebookSnapshotService.genSnapshot();
12. LOG.info("生成今日电子书快照结束，耗时：{}毫秒", System.currentTimeMillis() - start);
13. }
14. **<update** id="genSnapshot"**>**
15. insert into ebook\_snapshot(ebook\_id, `date`, view\_count, vote\_count, view\_increase, vote\_increase)
16. select t1.id, curdate(), 0, 0, 0, 0
17. from ebook t1
18. where not exists(select 1
19. from ebook\_snapshot t2
20. where t1.id = t2.ebook\_id
21. and t2.`date` = curdate());
23. update ebook\_snapshot t1, ebook t2
24. set t1.view\_count = t2.view\_count,
25. t1.vote\_count = t2.vote\_count
26. where t1.`date` = curdate()
27. and t1.ebook\_id = t2.id;
29. update ebook\_snapshot t1 left join (select ebook\_id, view\_count, vote\_count
30. from ebook\_snapshot
31. where `date` = date\_sub(curdate(), interval 1 day)) t2
32. on t1.ebook\_id = t2.ebook\_id
33. set t1.view\_increase = (t1.view\_count - ifnull(t2.view\_count, 0)),
34. t1.vote\_increase = (t1.vote\_count - ifnull(t2.vote\_count, 0))
35. where t1.`date` = curdate();
36. **</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

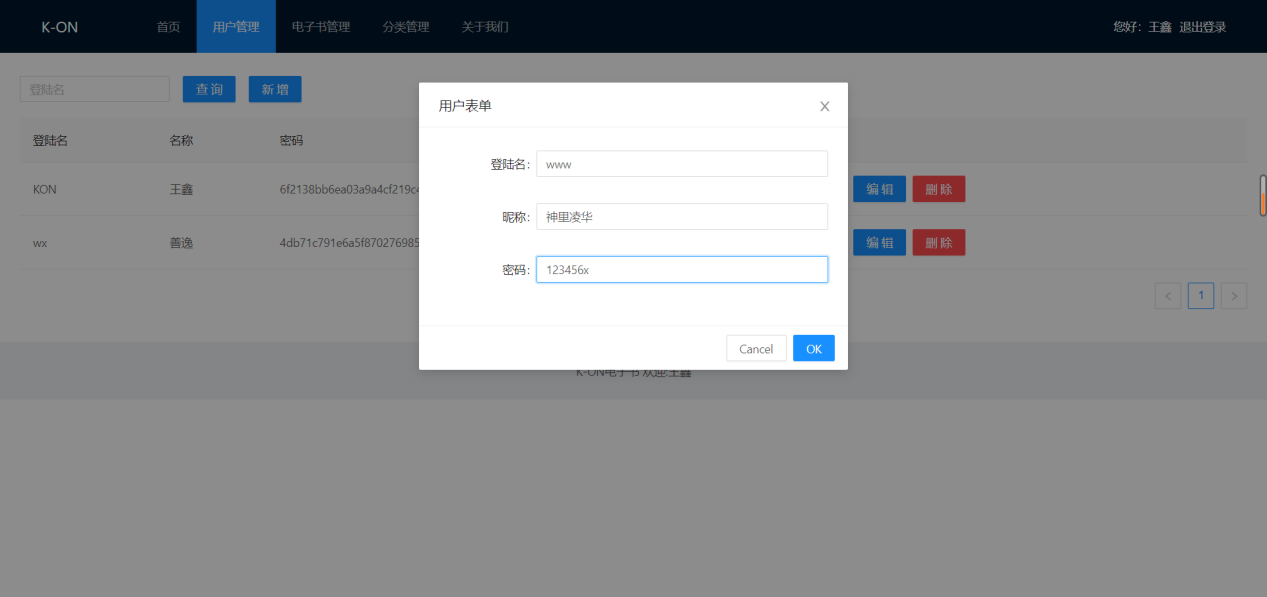


图5.6新增用户界面展示图

删除用户

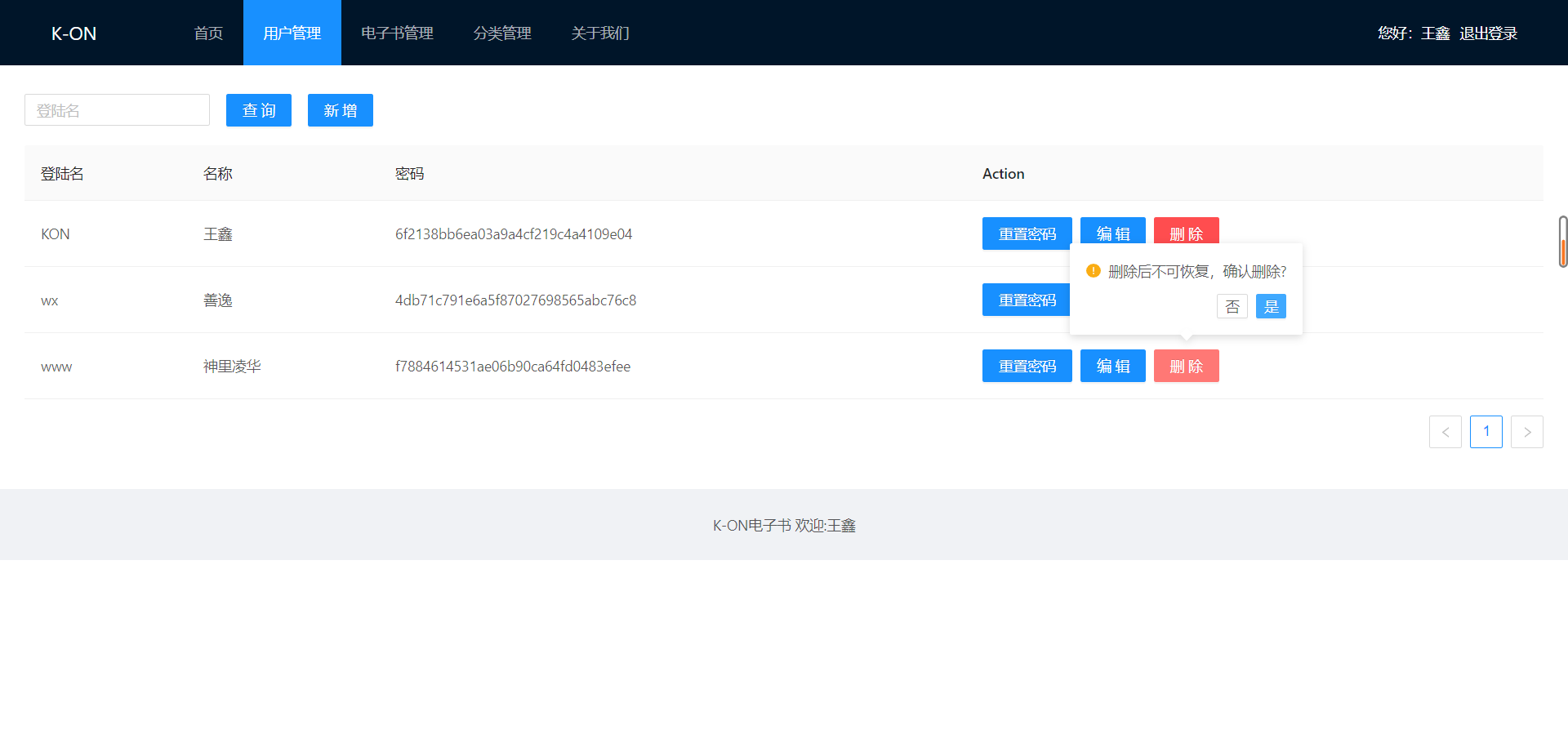


图5.7删除用户界面展示图

查询用户



图5.8查询用户界面展示图

编辑用户

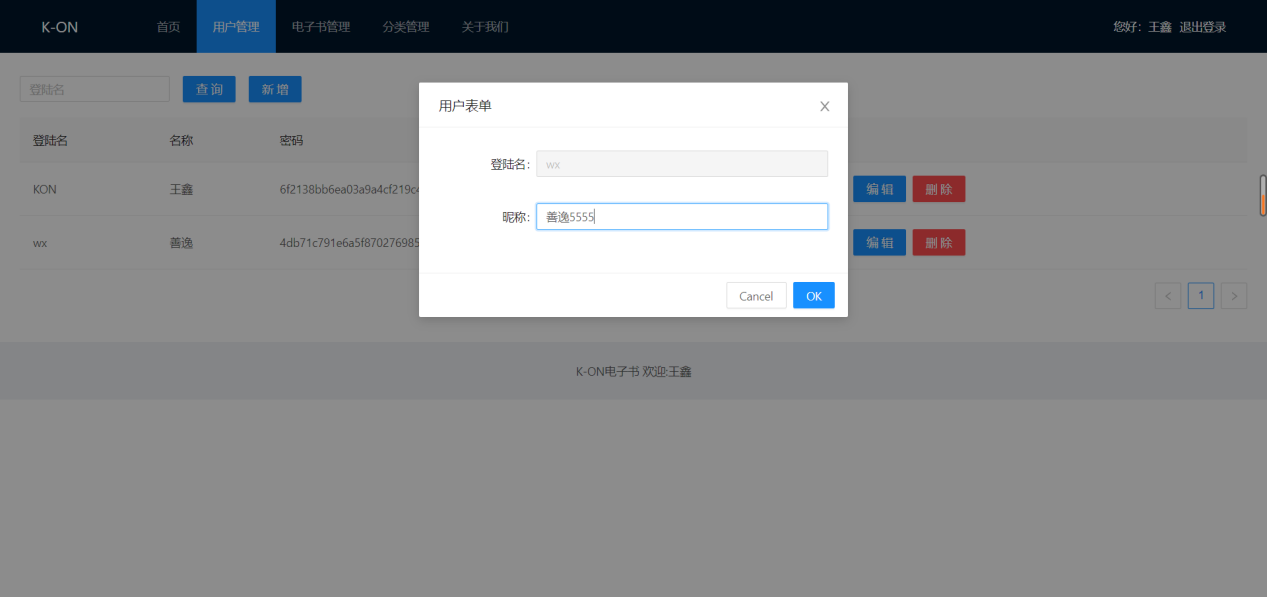


图5.9编辑用户界面展示图

**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. }
23. **<select** id="selectByExample" parameterType="com.jiawa.wiki.domain.UserExample" resultMap="BaseResultMap"**>**
24. select
25. **<if** test="distinct"**>**
26. distinct
27. **</if>**
28. **<include** refid="Base\_Column\_List" **/>**
29. from user
30. **<if** test="\_parameter != null"**>**
31. **<include** refid="Example\_Where\_Clause" **/>**
32. **</if>**
33. **<if** test="orderByClause != null"**>**
34. order by ${orderByClause}
35. **</if>**
36. **</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**

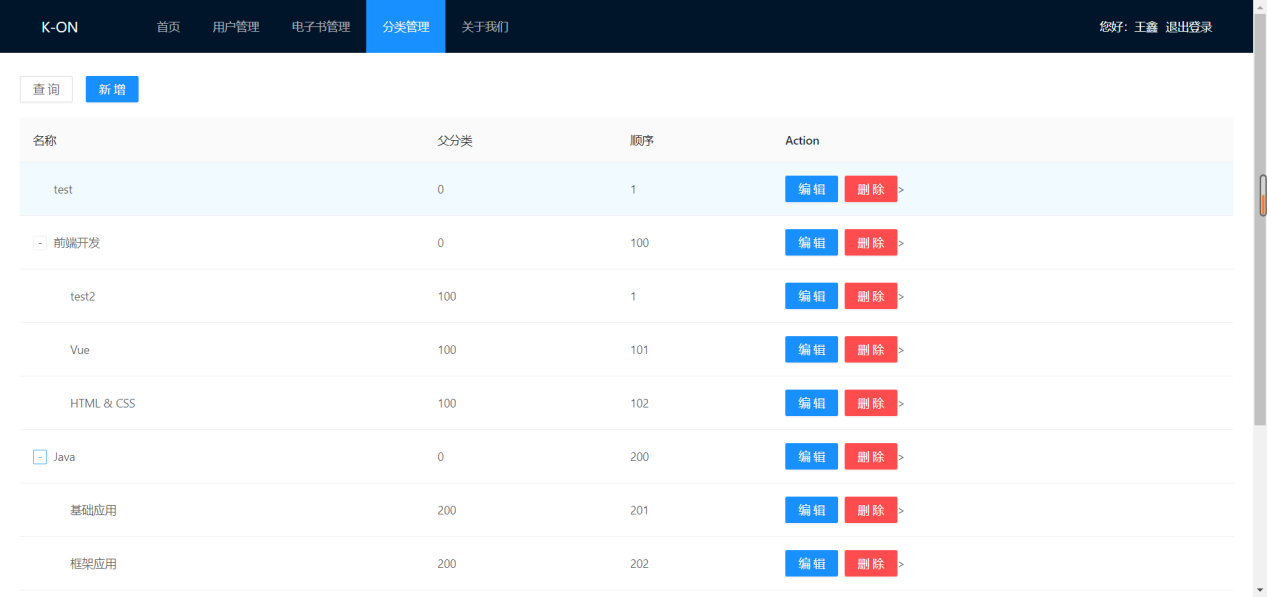


图5.10分类管理界面展示图

编辑分类

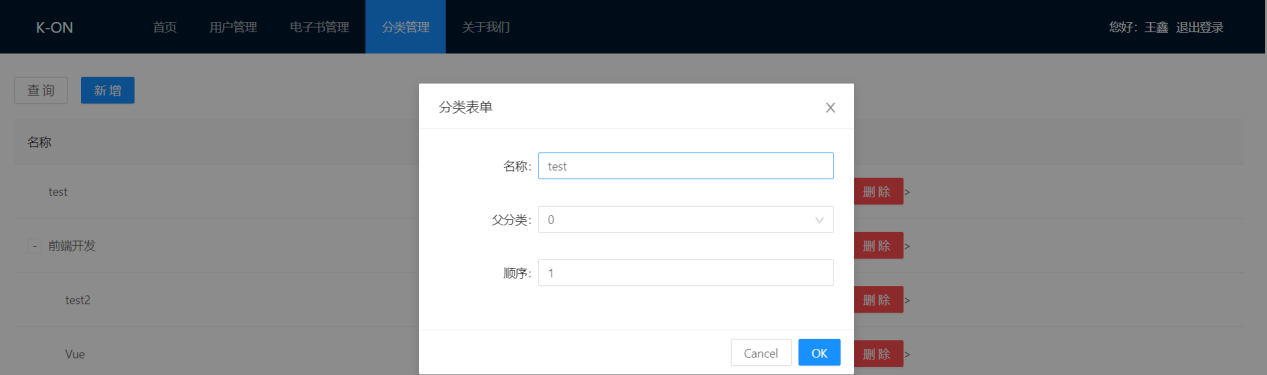


图5.11编辑分类界面展示图

新增分类

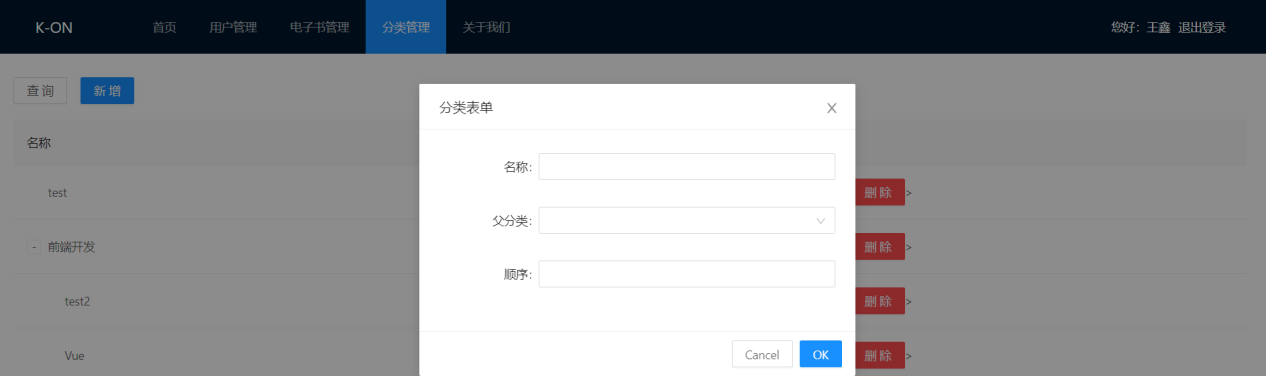


图5.12新增分类界面展示图

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

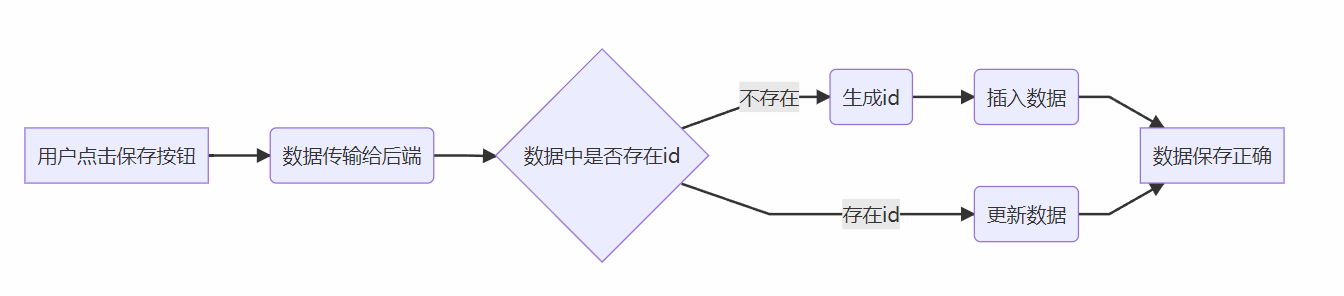


图5.14保存分类流程图

核心代码：

1. /\*\*
2. \* 保存分类
3. \* @param req
4. \*/
5. **public** **void** save(CategorySaveReq req){
6. Category category = CopyUtil.copy(req,Category.**class**);
7. **if**(ObjectUtils.isEmpty(req.getId())){
8. //新增
9. category.setId(snowFlake.nextId());
10. categoryMapper.insertSelective(category);
11. } **else**{
12. //更新
13. categoryMapper.updateByPrimaryKey(category);
14. }
15. }
16. <!--插入数据-->
17. **<insert** id="insertSelective" parameterType="com.jiawa.wiki.domain.Category"**>**
18. insert into category
19. **<trim** prefix="(" suffix=")" suffixOverrides=","**>**
20. **<if** test="id != null"**>**
21. id,
22. **</if>**
23. **<if** test="parent != null"**>**
24. parent,
25. **</if>**
26. **<if** test="name != null"**>**
27. `name`,
28. **</if>**
29. **<if** test="sort != null"**>**
30. sort,
31. **</if>**
32. **</trim>**
33. **<trim** prefix="values (" suffix=")" suffixOverrides=","**>**
34. **<if** test="id != null"**>**
35. #{id,jdbcType=BIGINT},
36. **</if>**
37. **<if** test="parent != null"**>**
38. #{parent,jdbcType=BIGINT},
39. **</if>**
40. **<if** test="name != null"**>**
41. #{name,jdbcType=VARCHAR},
42. **</if>**
43. **<if** test="sort != null"**>**
44. #{sort,jdbcType=INTEGER},
45. **</if>**
46. **</trim>**
47. **</insert>**
48. <!--  更新数据-->
49. **<update** id="updateByPrimaryKey" parameterType="com.jiawa.wiki.domain.Category"**>**
50. update category
51. set parent = #{parent,jdbcType=BIGINT},
52. `name` = #{name,jdbcType=VARCHAR},
53. sort = #{sort,jdbcType=INTEGER}
54. where id = #{id,jdbcType=BIGINT}
55. **</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

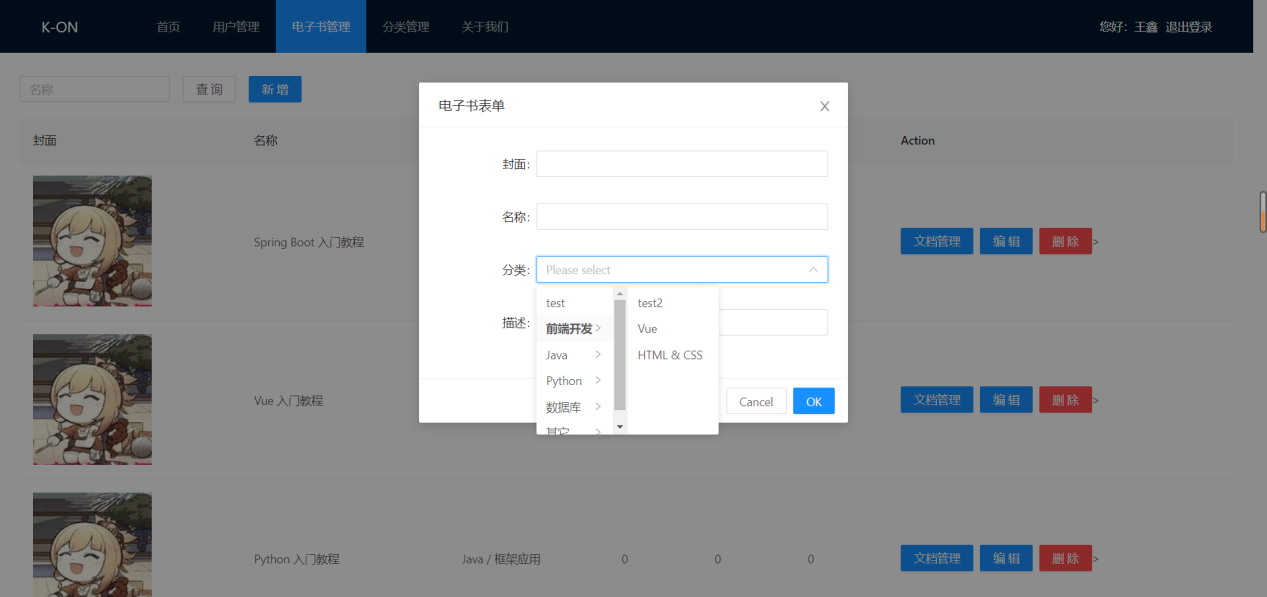


图5.16新增电子书展示图

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

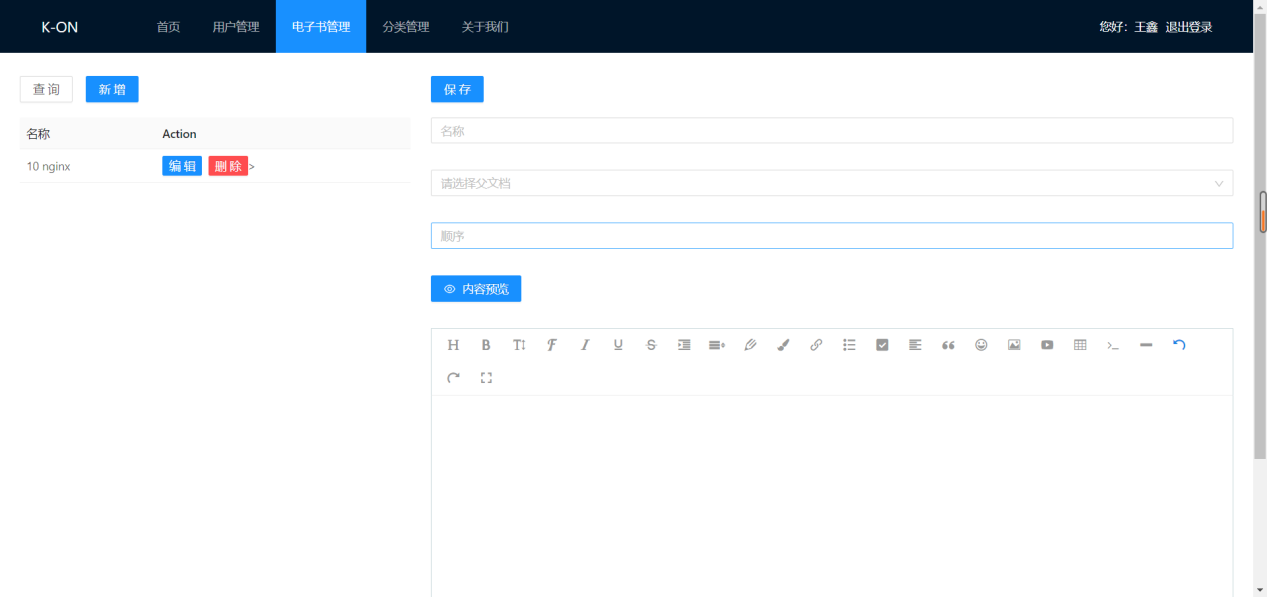


图5.20新增文档展示图

**5.4.2 实现细节**

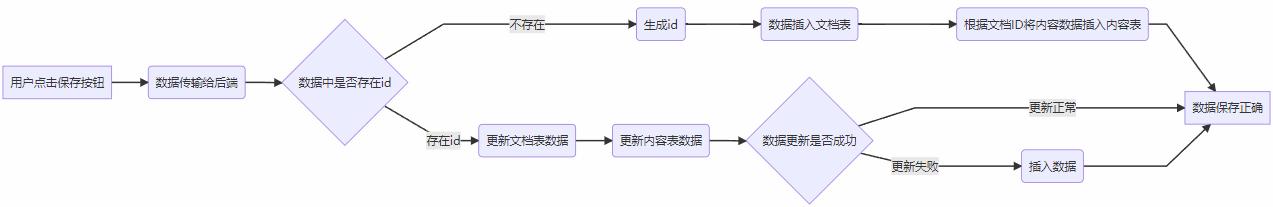


图5.21保存文档流程图

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
16. content.setId(doc.getId());
17. contentMapper.insertSelective(content);
18. } **else**{
19. //更新
20. docMapper.updateByPrimaryKey(doc);
21. **int** count = contentMapper.updateByPrimaryKeyWithBLOBs(content);
22. **if**(count == 0){  //更新失败则插入数据
23. contentMapper.insert(content);
24. }
25. }
26. }
27. <!--  插入文档内容-->
28. **<insert** id="insertSelective" parameterType="com.jiawa.wiki.domain.Doc"**>**
29. insert into doc
30. **<trim** prefix="(" suffix=")" suffixOverrides=","**>**
31. **<if** test="id != null"**>**
32. id,
33. **</if>**
34. **<if** test="ebookId != null"**>**
35. ebook\_id,
36. **</if>**
37. **<if** test="parent != null"**>**
38. parent,
39. **</if>**
40. **<if** test="name != null"**>**
41. `name`,
42. **</if>**
43. **<if** test="sort != null"**>**
44. sort,
45. **</if>**
46. **<if** test="viewCount != null"**>**
47. view\_count,
48. **</if>**
49. **<if** test="voteCount != null"**>**
50. vote\_count,
51. **</if>**
52. **</trim>**
53. **<trim** prefix="values (" suffix=")" suffixOverrides=","**>**
54. **<if** test="id != null"**>**
55. #{id,jdbcType=BIGINT},
56. **</if>**
57. **<if** test="ebookId != null"**>**
58. #{ebookId,jdbcType=BIGINT},
59. **</if>**
60. **<if** test="parent != null"**>**
61. #{parent,jdbcType=BIGINT},
62. **</if>**
63. **<if** test="name != null"**>**
64. #{name,jdbcType=VARCHAR},
65. **</if>**
66. **<if** test="sort != null"**>**
67. #{sort,jdbcType=INTEGER},
68. **</if>**
69. **<if** test="viewCount != null"**>**
70. #{viewCount,jdbcType=INTEGER},
71. **</if>**
72. **<if** test="voteCount != null"**>**
73. #{voteCount,jdbcType=INTEGER},
74. **</if>**
75. **</trim>**
76. **</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。

1. 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。