开发文档——WordKeeper(单词管家)

厦门大学第十三届软件设计大赛决赛材料

DAOO 参赛组

组员：周兴、林联辉、陈丰铧

目录

第 1 章 绪 论

1.1 选题背景和意义

随着国际化的发展，英语的学习也越发地重要。所以针对此现状，并以移动学习的核心参与群体——大学生为对象，对新型大学英语词汇学习网站的设计提出设想。

但是，近些年，英语学习类的软件应用特别词汇类数量不断增加，但是能够脱颖而出却很难，项目死亡率也在不断上升，整个应用领域的竞争越来越激烈。所以，进行软件的优化设计和深入创新是重中之重。而英语学习软件在大学生的学习生活中扮演着重要的角色，因为大学生的生活自由度比较高，一款能符合大学生英语学习特点的英语软件能够更好地帮助大学生提高自身英语水平。因此值得我们进行在该领域进行创新，所以我们想到利用人工智能技术结合传统的英语词汇学习网站的模式，进行合理创新，实现英语学习类软件的经济效益和最大程度发挥其学习作用的共赢。

1.2 文档组织结构

第一章绪论。介绍本项目的选题背景和意义。

第二章相关技术与工具。介绍开发此系统的相关技术与开发工具，以及开发环境。

第三章可行性研究与需求分析。介绍系统开发可行性，以及系统所具有的的功能需求和性能需求。

第四章概要设计。介绍本系统的架构设计，系统功能设计，数据库设计，以及物理结构设计。

第五章详细设计和实现。详细介绍本系统各个功能的类实现和核心实现，以及各大功能模块的实现效果。

第六章系统测试。进行系统测试工作，发现系统中潜在的漏洞，并加以说明。

第七章总结和展望。对毕业设计的制作过程进行总结和思考，从中认识到该系统做的好的地方和做的不好的地方，并对系统做的不好的地方提出优化方案。

第 2 章 相关技术与工具

2.1 关键技术

2.1.1 SpringBoot：

SpringBoot 是 Spring 框架的一个子项目，用于创建 Spring 4.0 项目。Spring 拥有轻量级的组件代码，但配置却十分麻烦，并且项目的依赖管理也十分耗费精力。SpringBoot 对上述 Spring 的缺点进行优化，基于约定优于配置的思想，使得开发人员可以专心致志地编写逻辑业务的代码，不需要因为配置问题而分神，大大提高了开发效率，开发项目时间得以缩短。

2.1.2 MyBatis：

MyBatis 是一个优秀的持久层框架，是一个开源免费的轻量级框架，学习成本低，开发者可快速上手。它对 JDBC 操作数据库的过程进行封装，使开发者只需要关注 SQL 本身。

2.1.3 B/S 架构：

Browser/Server 结构即浏览器/服务器”模式，它是随着 Internet 的兴起，在 Web 兴起后的一种网络结构模式，这一模式统一了客户端，让核心的业务处理在服务端完成，用户只需要在个人的手机或者电脑中安装一个浏览器就可以通过 Web Server 与数据库进行数据交互。B/S 结构能够在无需安装专门对应的软件客户端情况下对系统进行操作，浏览器便是客户端，维护成本更低和升级方式更为简单。

2.1.4 Maven

Maven 项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的项目管理工具软件。

2.1.5 深度学习

深度学习(DL, Deep Learning)是机器学习(ML, Machine Learning)领域中一个新的研究方向，它被引入机器学习使其更接近于最初的目标——人工智能(AI, Artificial Intelligence)。

2.2 开发工具

集成开发工具：Intellij IDEA：Intellij IDEA 是一个 Java 开发集成环境。Intellij 是一个非常好的 Java 开发工具，能够提供给开发人员许多便利，例如：代码自动提示，动态语法检测，代码检查等。

2.3 开发环境

硬件：Intel Core i5、内存容量：8GB，硬盘容量 500GB
软件：操作系统:windows10 64 位
开发平台:IDEA 2019
web 服务器:Tomcat 8.5.3
jdk:JDK 1.8
数据库:mysql 8.0

第 3 章 可行性研究与需求分析

3.1 可行性研究

3.1.1 技术可行性

系统所涉及 SpringBoot 框架、数据库、动态网页等各项技术，全部都是目前应用广泛且经受住考验不断发展的可靠技术，这足以保证网站建成后的性能以及运行的安全可靠性。

3.1.2 经济可行性

开发网站的主要支出在于软硬件成本、通信成本、推介成本等，现阶段的开发成本较以前降低。本项目最主要的资金用途在于维护网站数据的稳定性，以及购买部分优质英文刊物、其他材料，并以文章推送的形式为用户提供良好的使用体验。

3.1.3 市场可行性

随着国际化的发展以及教育水平的不断提高，人们（特别是各类学历的学生）对英语学习的需求也在不断增加。通过网络进行单词记忆、文章阅读等也成为了常见的英语学习手段，通过这种成本低、人群覆盖面广的渠道方式可以创造更大的市场空间。

3.2 需求分析

3.2.1 需求描述

需求分析以可行性分析出发，可以让开发者明白系统需要实现的功能，确定系统功能需求，非功能需求等。本文所设计的单词管家主要涉及了：

（1）用户对单词记忆的需求，包括查询词典的需求，获取单词发音及关联图片的需求，生词本管理的需求；

（2）用户对单词进行拓展应用的需求，包括精品文章推送的需求，口语评分的需求，作文打分的需求；

（3）用户登录注册该网站的需求，个人资料管理的需求，打卡签到的需求；

（4）管理员对网站用户资料管理的需求，包括用户的新增、删除、修改、具体信息查询的需求；

（5）管理员对网站学习资料管理的需求，包括单词的新增、删除、修改、查询的需求，文章的新增、删除、修改、查询的需求。

3.2.2 功能需求

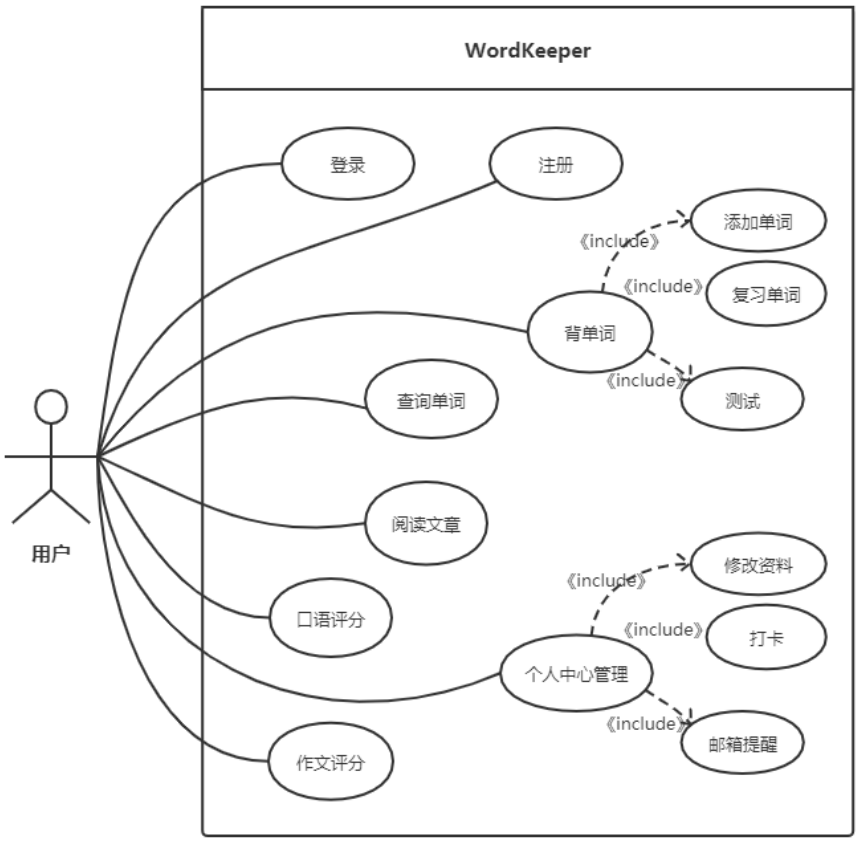

图 3.1 功能需求总图

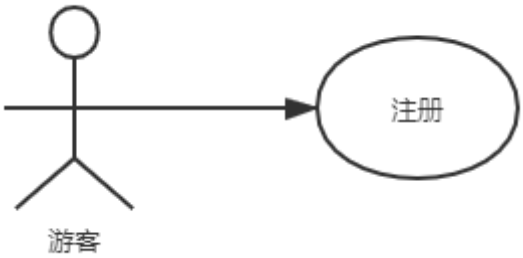

图 3.2 注册

表 3.2 注册功能用例规约

用例名称：	注册	
用例 ID：	1	
角色：	游客	
用例说明：	该用例描述的是游客注册成为用户	
前置条件：	游客进入注册界面	
基本事件流：	参与者动作	系统响应
	1. 游客输入待注册的用户名、所用邮箱、密码以及确认密码，点击注册。	2.系统查找数据库，查看用户名是否已被占用。若已被占用则进入 2.1，若未被占用，则继续检查两次密码是否一致，一致则将注册信息存入数据库，完成注册；不一致则进入 2.3。
异常事件流：	参与者动作	系统响应
	2.1.1 用户名为空 2.2.1 邮箱为空 2.2.2 邮箱格式错误 2.3.1 两次密码输入不一致	2.1 提示更改用户名 2.2 提示邮箱不可用 2.3 提示两次密码输入不一致，请重新输入
后置条件：	游客成功注册用户	

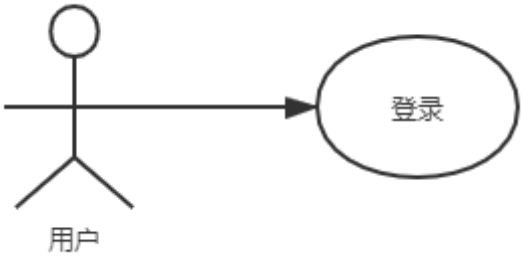

图 3.3 登录

表 3.3 登录功能用例规约

用例名称：	登录	
用例 ID：	2	
角色：	用户	
用例说明：	该用例描述的是用户实现登录	
前置条件：	已注册用户进入登录界面	
基本事件流：	参与者动作	系统响应
	1.用户输入先前已注册成功的用户名，输入相应密码，点击登录按钮。	2.系统查找数据库，看该用户名密码对是否在数据库中。若存在则进入主界面，若不存在则进入 2.2
异常事件流：	参与者动作	系统响应
	2.1.1 用户名为空 2.1.2 密码为空 2.2.1 用户名不存在 2.2.1 输入密码错误	2.1 提示用户名或密码不能为空 2.2 提示用户名或密码输入错误
后置条件：	用户成功登录	

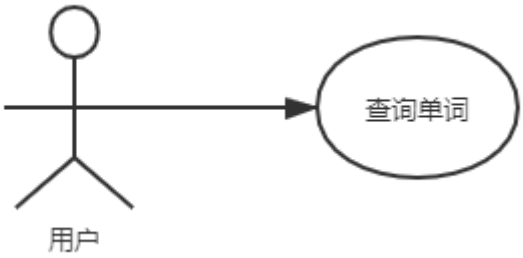

图 3.4 查询单词

表 3.4 查询单词功能用例规约

用例名称：	查询单词	
用例 ID：	3	
角色：	用户	
用例说明：	该用例描述的是用户搜索单词信息	
前置条件：	用户进入单词查询界面	
基本事件流：	参与者动作	系统响应
	1.用户在单词查询界面输入框中输入单词，点击搜索按钮对进行查找	2.系统接受到用户所提供的单词，并调用接口获取单词释义、发音、关联图片信息，返回给前台
后置条件：	用户成功查询单词信息，获取相关图片	

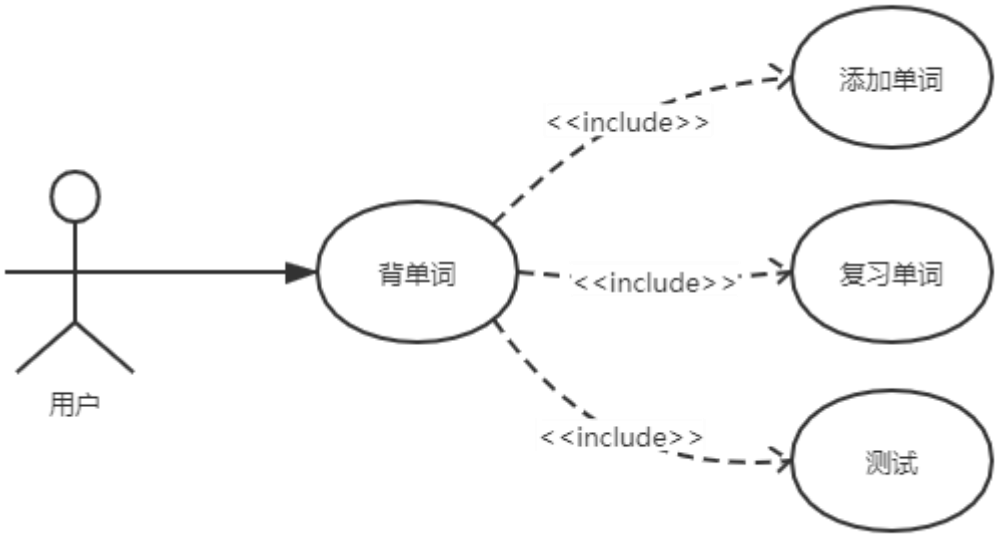

图 3.5 单词学习

表 3.5 添加生词功能用例规约

用例名称：	添加生词	
用例 ID：	4	
角色：	用户	
用例说明：	该用例描述的是用户学习并记忆单词	
前置条件：	用户进入单词学习界面	
基本事件流：	参与者动作	系统响应
	1.用户输入一段文字，点击提交 3.用户勾选需要学习的生词，点击提交	2.系统分析段落中存在的生词，将列表返回给前端 4.系统将用户选择的生词加入生词本，并更新复习计划
后置条件：	进入复习单词界面	

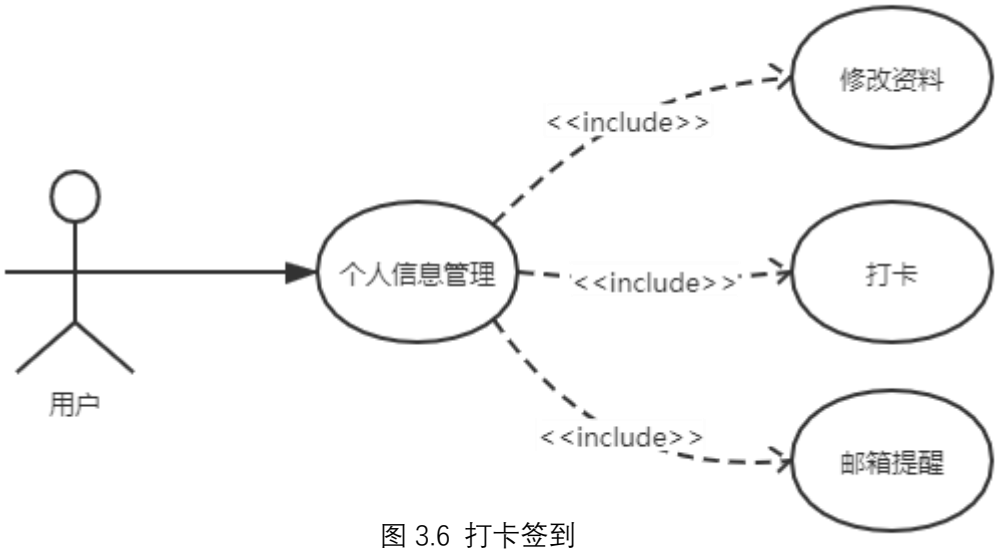

图 3.6 打卡签到

表 3.6 打卡签到用例规约

用例名称：	打卡签到	
用例 ID：	5	
角色：	用户	
用例说明：	该用例描述的是用户进行每日打卡	
前置条件：	用户进入首页	
基本事件流：	参与者动作	系统响应
	1.用户点击打卡按钮 3.用户进行答题	2.系统生成每日打卡选择题，返回给前台 4.系统将打卡结果记录到用户打卡历史中，并将该次结果返回给前端
后置条件：	/	

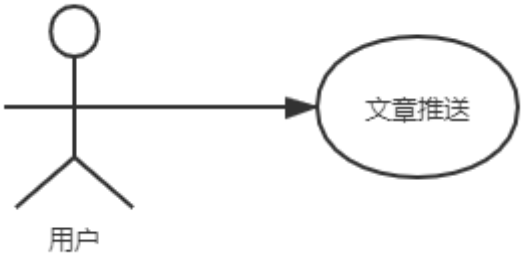

图 3.7 文章推送

表 3.7 查看文章推送用例规约

用例名称：	查看文章推送	
用例 ID：	6	
角色：	用户	
用例说明：	该用例描述的是用户查看每周文章推送	
前置条件：	用户进入首页	
基本事件流：	参与者动作	系统响应
	1.用户下拉页面至文章推送版块 2.用户点击一篇文章标题	3.系统返回文章内容
后置条件：	用户加入文章阅读界面	

3.2.3 非功能需求

信息完整性：倘若用户输入的数据与系统的要求不匹配时，系统应及时做出提示信息。
运行环境：系统运行在现在最为流行的 Windows 系统上，便于日后系统的升级。
响应速度：要求系统能够快速响应，在用户可忍受的时间内范围给予提示。
界面：系统的每个界面以及操作模式一致，且风格简洁，让用户一目了然，容易快速上手。

第 4 章 概要设计

4.1 系统架构设计

系统是基于 B/S 架构和 MVC 模式进行框架设计，采用 B/S 架构用户只需要在个人的手机或者电脑中安装一个浏览器就可以通过 Web Server 与数据库进行数据交互。MVC 模式能够将各个业务功能解耦，使各个功能模块达到高内聚、低耦合。系统分为三层，分别为数据库层、表现层、业务逻辑层。表现层是系统与用户交互信息的窗口，用户通过 Web 浏览器操作系统，实现与后台服务的对话，实现局部刷新功能；业务逻辑层处理系统图业务功能逻辑；数据库层是对用户的请求实现数据操作，系统采用 Mysql 数据库并引入 Mybatis 持久化框架。

4.2 系统功能设计

系统的主要功能模块前台有：首页展示、词汇学习、口语打分、文章推送、用户登录注册等模块。后台有：用户管理、词汇管理、文章管理等模块。具体功能结构图如下：

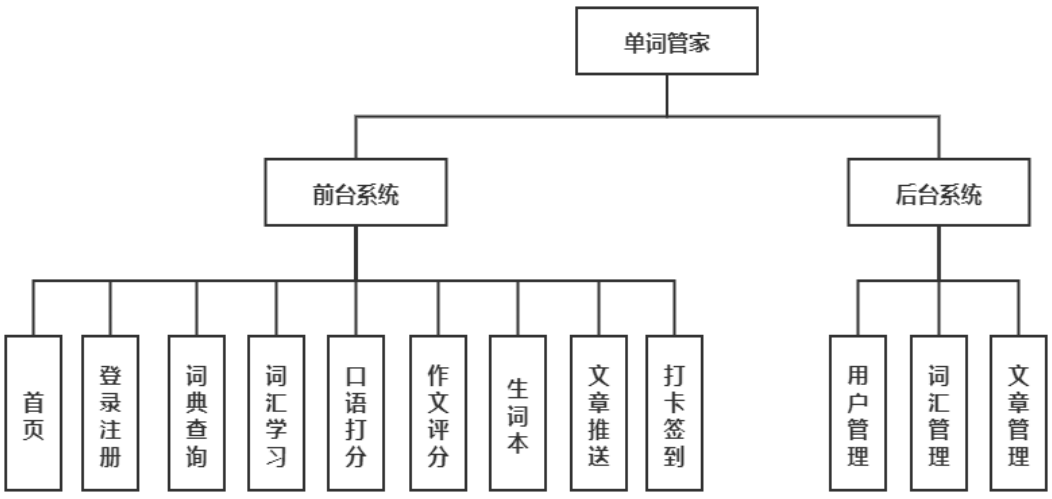

图 4.1 功能结构图

4.3 数据库设计

4.3.1 概念结构设计

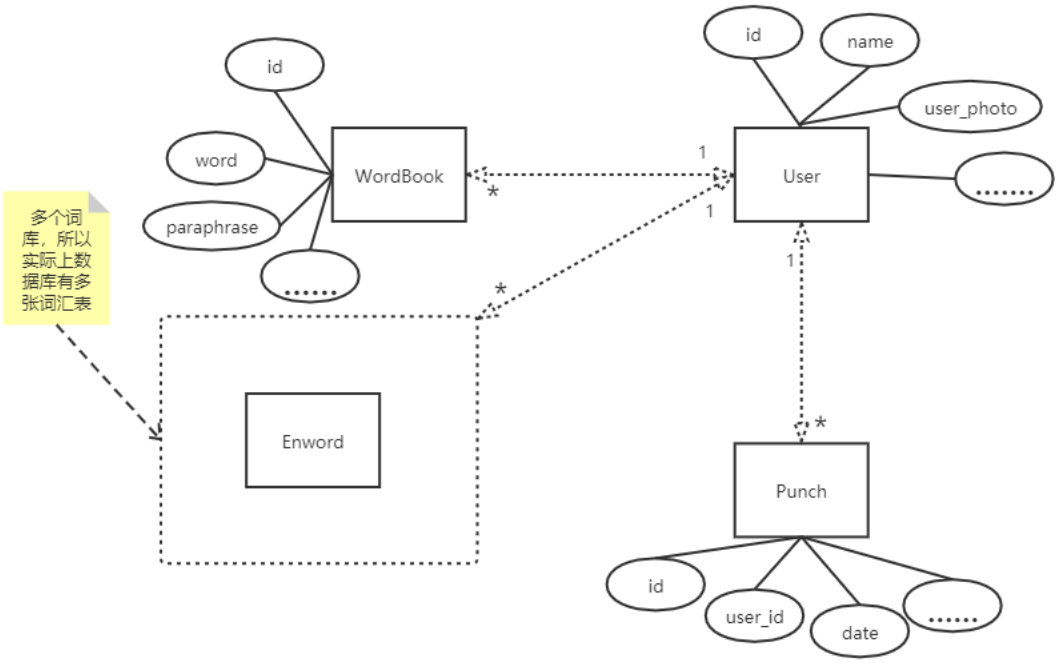

图 4.2 概念结构设计图

4.3.2 逻辑结构设计

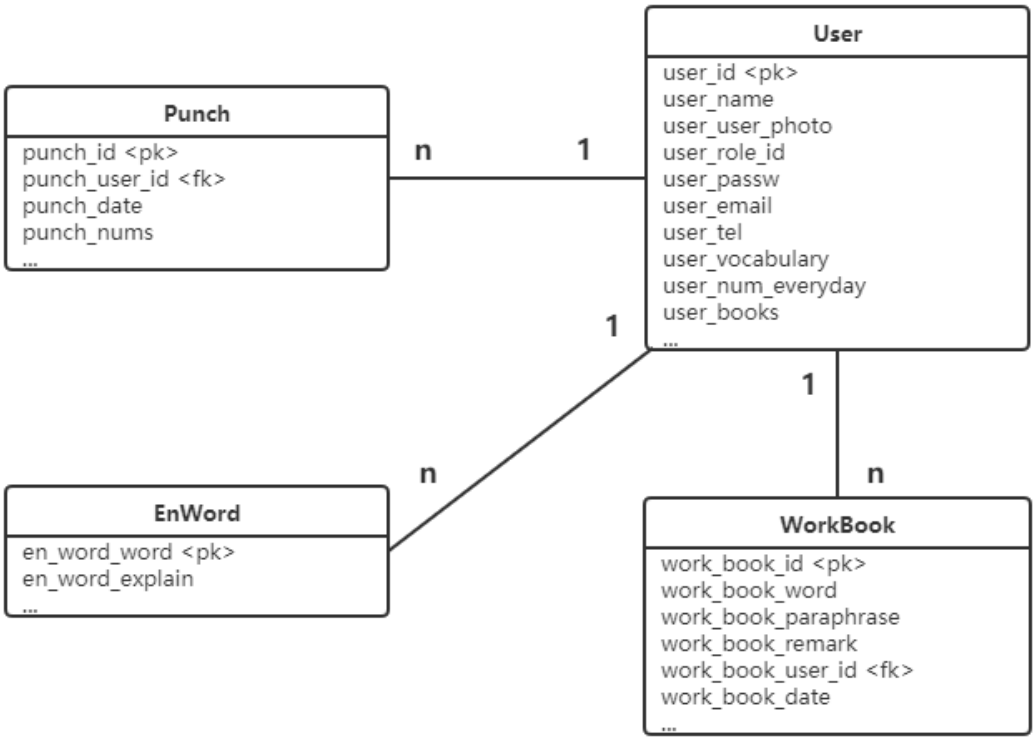

图 4.3 逻辑结构设计图

1. 用户信息

字段名	类型	描述	其他约束
id	bigint(9)	用户 id（自增得来）	PRIMARY KEY AUTO_INCREMENT
name	varchar(31)	用户名	NOT NULL
user_photo	varchar(255)	头像 url	DEFAULT NULL
role_id	tinyint(1)	用户权限	0、1、
passw	varchar(255)	用户密码	密文存储 NOT NULL
email	varchar(255)	用户邮箱	NOT NULL
tel	varchar(11)	用户电话号码	NOT NULL
vocabulary	bigint(9)	词汇量	DEFAULT NULL
num_everyday	bigint(5)	每天单词学习数	DEFAULT NULL
books	varchar(255)	所选单词书	DEFAULT NULL
gmt_create	datetime(2)	表项创建时间	DEFAULT NULL
gmt_modified	datetime(2)	表项最后一次的修改时间	DEFAULT NULL
is_deleted	tinyint(1)	是否被逻辑删除	unsigned DEFAULT '0'

2. 打卡记录

字段名	类型	描述	其他约束
id	bigint(9)	记录 id（自增得来）	PRIMARY KEY AUTO_INCREMENT
user_id	bigint(9)	用户 id	NOT NULL
date	datetime(2)	打卡时间	NOT NULL
nums	bigint(5)	连续打卡的次数	NOT NULL
gmt_create	datetime(2)	表项创建时间	DEFAULT NULL
gmt_modified	datetime(2)	表项最后一次的修改时间	DEFAULT NULL
is_deleted	tinyint(1)	是否被逻辑删除	unsigned DEFAULT '0'

3. 自定义生词本

字段名	类型	描述	其他约束
id	bigint(9)	生词 id（自增得来）	PRIMARY KEY AUTO_INCREMENT
word	varchar(255)	单词或者短语	NOT NULL
paraphrase	varchar(255)	释意	NOT NULL
remark	varchar(255)	备注	DEFAULT NULL
user_id	bigint(9)	用户 id	NOT NULL
date	datetime(2)	创建时间	NOT NULL
gmt_create	datetime(2)	表项创建时间	DEFAULT NULL
gmt_modified	datetime(2)	表项最后一次的修改时间	DEFAULT NULL
is_deleted	tinyint(1)	是否被逻辑删除	unsigned DEFAULT '0'

4.3.3 物理结构设计

本文的系统采用 mysql 8.0 软件进行数据库物理结构设计。根据本系统的概论结构设计和逻辑结构设计创建数据库，数据库名为:“wordkeeper”，在该数据库中定义表、字段和数据的命名规范，每个字段选择合适的数据类型，设计合理的表与表之间的关系，创建所需要数据表。

第5章 详细设计和实现

5.1 登录注册模块的详细设计与实现

5.1.1 用户类设计

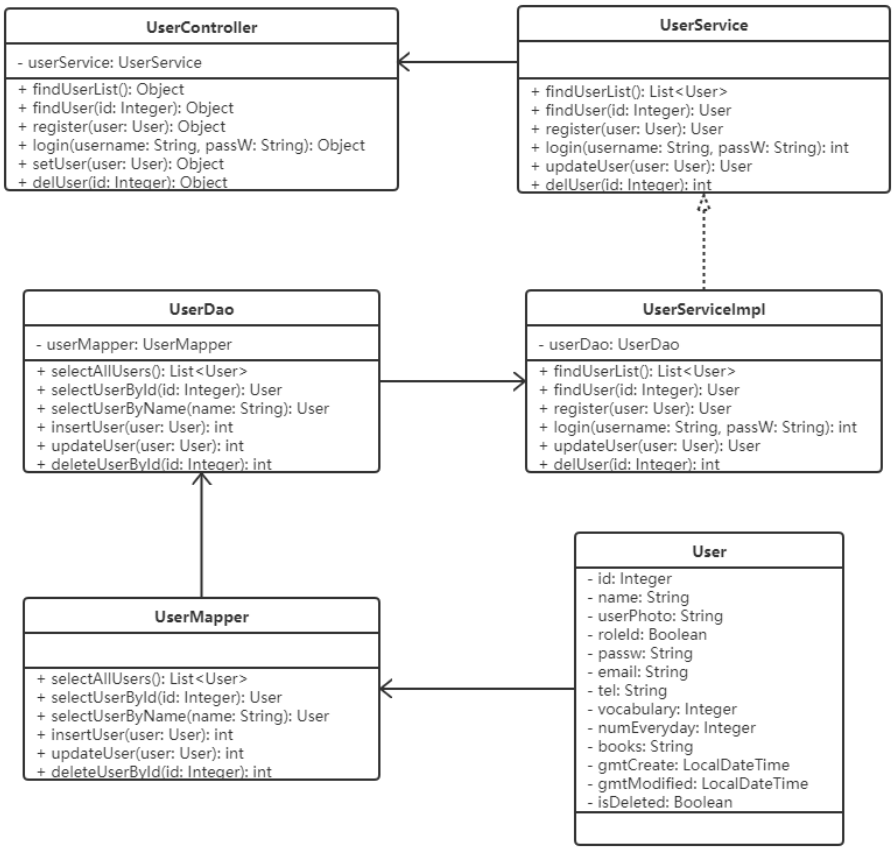

图 5.1 用户类设计

登录注册功能是由 Spring MVC Dispatcher 作为请求的总处理入口，Spring MVC UserController 将请求分发给 UserController 作为前端处理接口。UserController 将核心处理过程交给 UserService 实现，UserServiceImpl 调用 userDao 对象操作数据库。

5.1.2 用户登录注册的核心实现

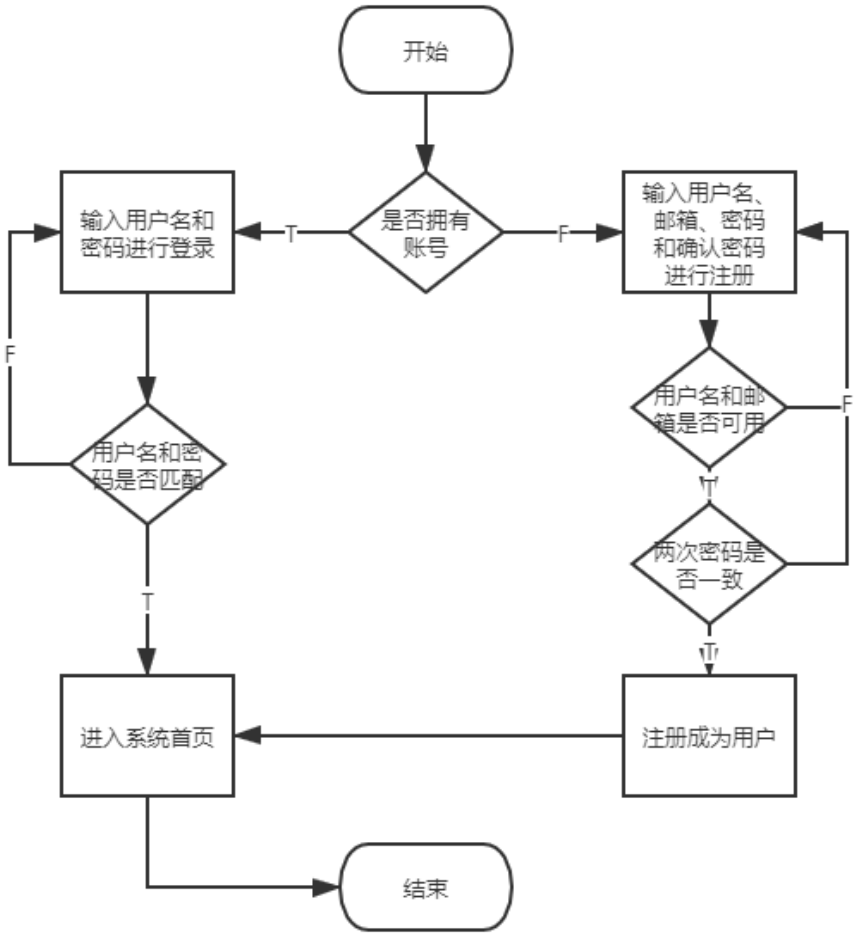

图 5.2 用户注册与登录的流程图

用户注册与登录的主要代码如下：

```
1. public int login(String userName, String passW) {
2.     User result=userDao.selectUserByName(userName);
3.     if(result==null) {
4.         return 0;
5.     } else
6.     {
7.         if(MD5Util.encrypt(passW).equals(result.getPassw())) {
8.             return 1;
```

```
            } else {
                return 0;
            }
        }
    }
public User register(User user) {
        User info=user;
        User temp1=userDao.selectUserByName(user.getName());
        if(temp==null)
            //该用户名未注册过
        {
            String passw=MD5Util.encrypt(info.getPassw());
            info.setPassw(passw);
            //密码加密存储
            int result=userDao.insertUser(info);
            if(result==1) {
                return user;
            } else {
                return null;
            }
        }
        else {
            return null;
        }
    }
```

5.1.3 用户登录与注册的实现效果

图 5.4 用户登录界面

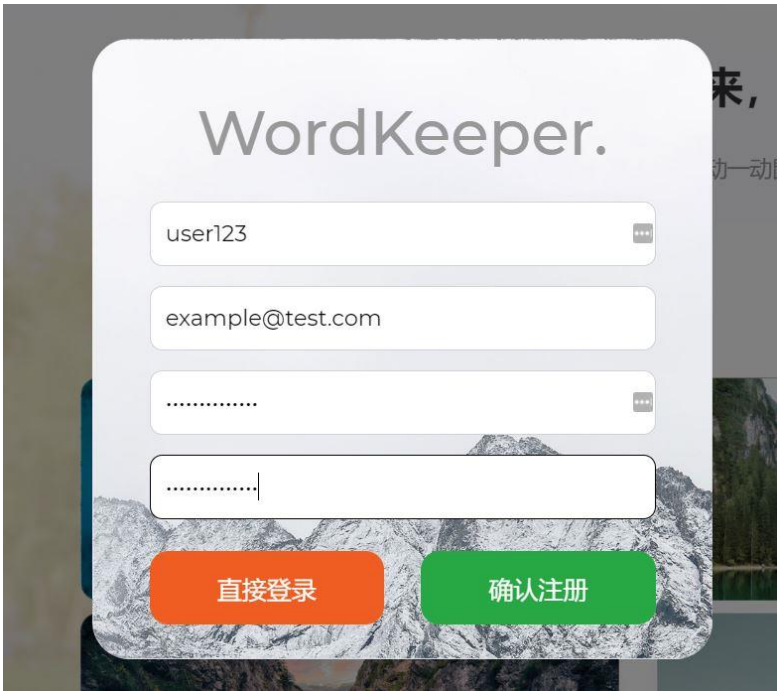

图 5.5 用户注册界面

5.2 生词本管理的详细设计与实现

5.2.1 生词本的类设计

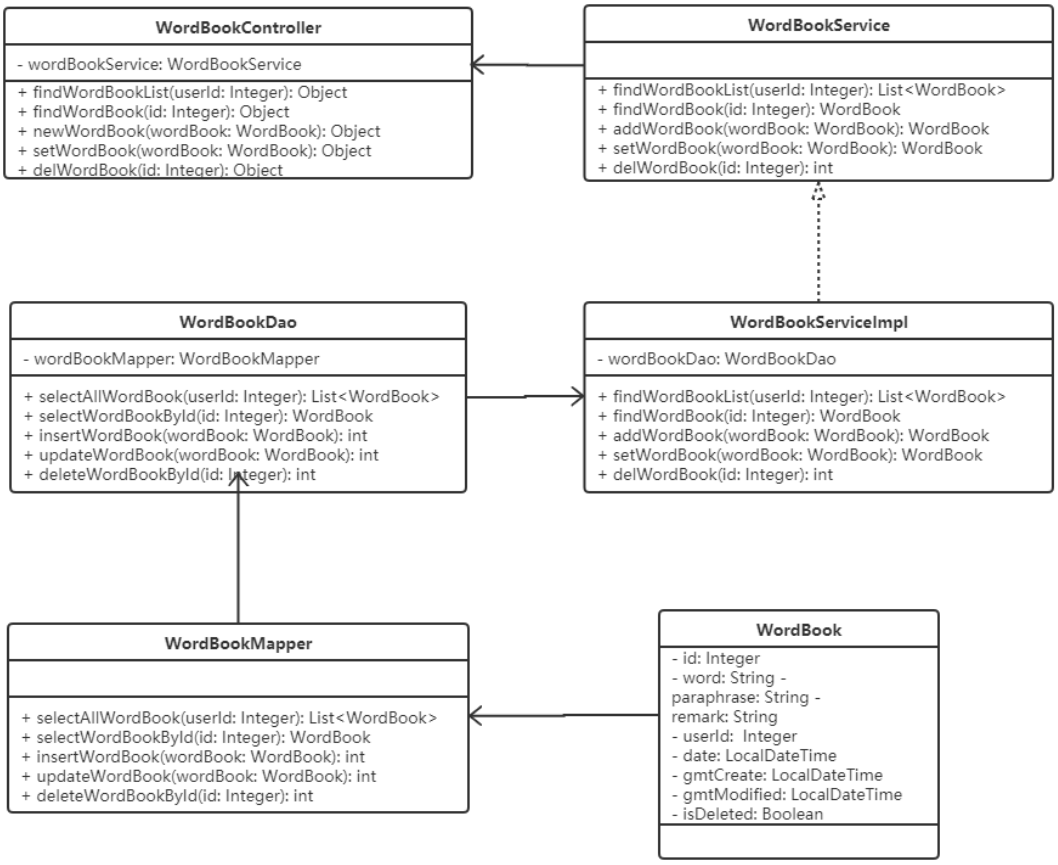

图 5.6 用户登录注册模块的类设计

生词本模块，由 Spring MVC Dispatcher 作为请求的总处理入口，Spring MVC WordBookController 将请求分发给 WordBookController 作为前端处理接口。WordBookController 将核心处理过程交给 WordBookService 实现，WordBookServiceImpl 调用 WordBookDao 对象操作数据库。

5.2.2 生词本模块的核心实现

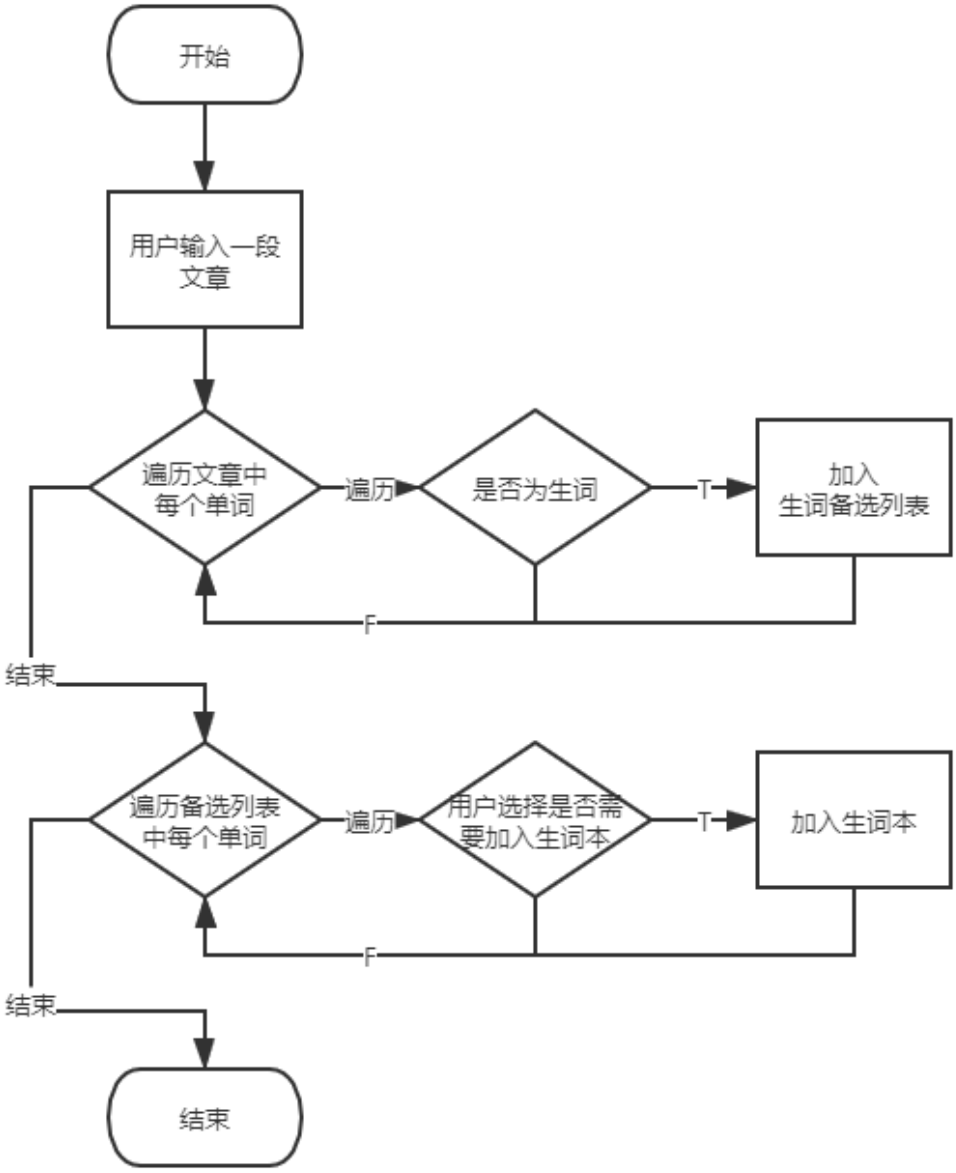

图 5.7 添加生词的流程图

```
1. public WordBook addWordBook(WordBook wordBook) {
2.     int result=wordBookDao.insertWordBook(wordBook);
3.     if(result==1) {
4.         return wordBook;
5.     } else {
6.         return null;
7.     }
8. }
```

5.2.3 生词本模块的实现效果

图 5.8 生词本界面

5.3 打卡签到模块的详细设计与实现

5.3.1 打卡记录的类设计

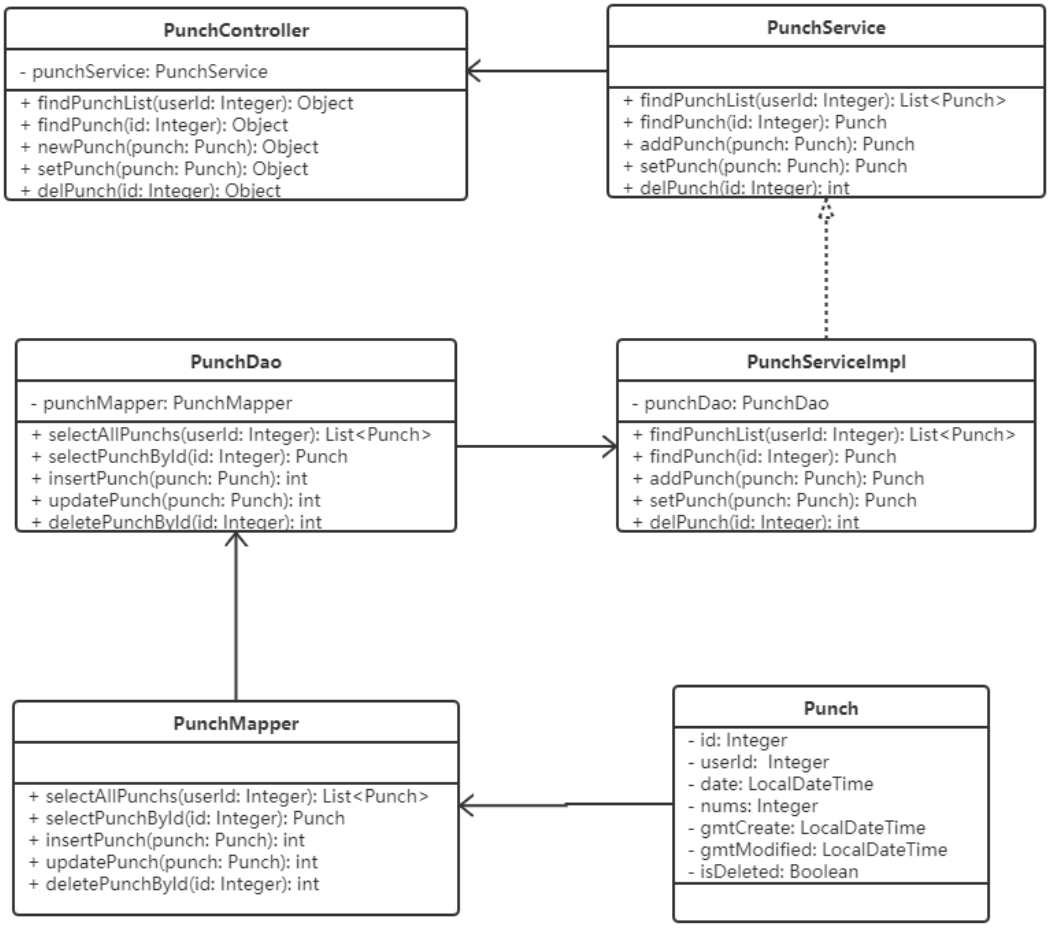

图 5.9 打卡签到模块的类设计

打卡签到模块，由 Spring MVC Dispatcher 作为请求的总处理入口，Spring MVC PunchController 将请求分发给 PunchController 作为前端处理接口。PunchController 将核心处理过程交给 PunchService 实现，PunchServiceImpl 调用 PunchDao 对象操作数据库。

5.3.2 打卡签到模块的核心实现

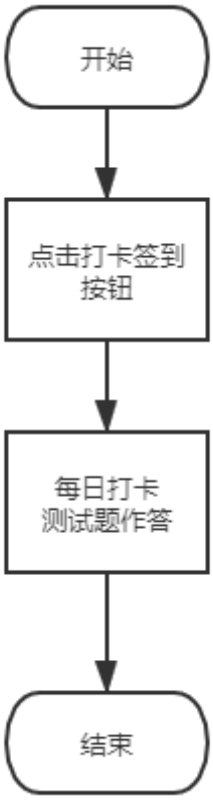

图 5.10 打卡签到流程

```
public Punch addPunch(Punch punch) {
    List<Punch> temp = punchDao.selectAllPunches(punch.getUserId()
);
    punch.setNums(temp.size()+1);
    int result=punchDao.insertPunch(punch);
    if(result==1) {
        return punch;
    } else {
        return null;
    }
}
```

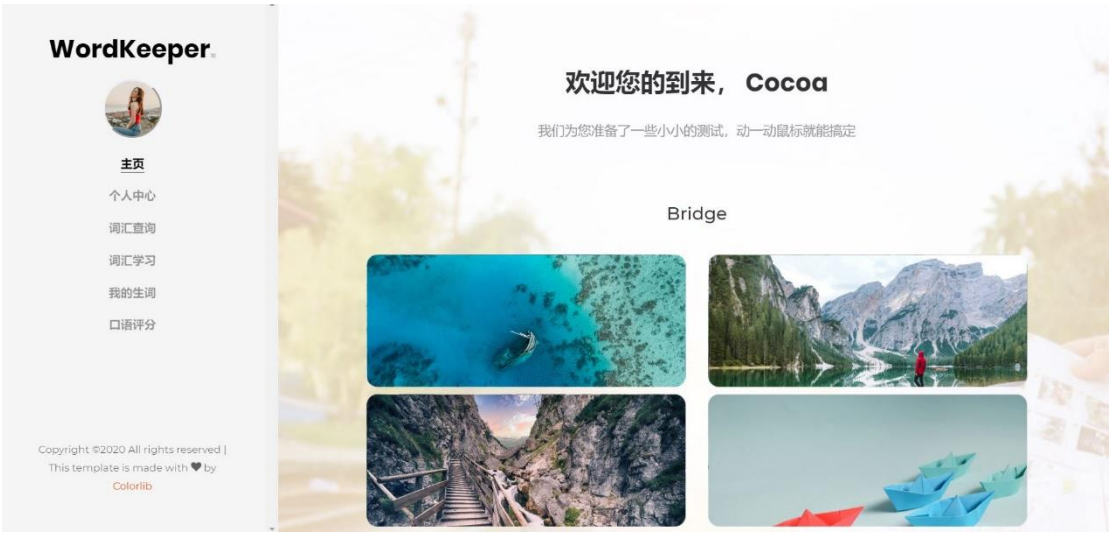
WordKeeper.
主页
个人中心
词汇查询
词汇学习
我的生词
口语评分
Copyright ©2020 All rights reserved |
This template is made with ♥ by
Colorlib
欢迎您的到来，Cocoa
我们为您准备了一些小小的测试，动一动鼠标就能搞定
Bridge

图 5.11 打卡签到界面

5.4 其它功能模块的设计与实现

5.4.1 部分功能模块的类设计

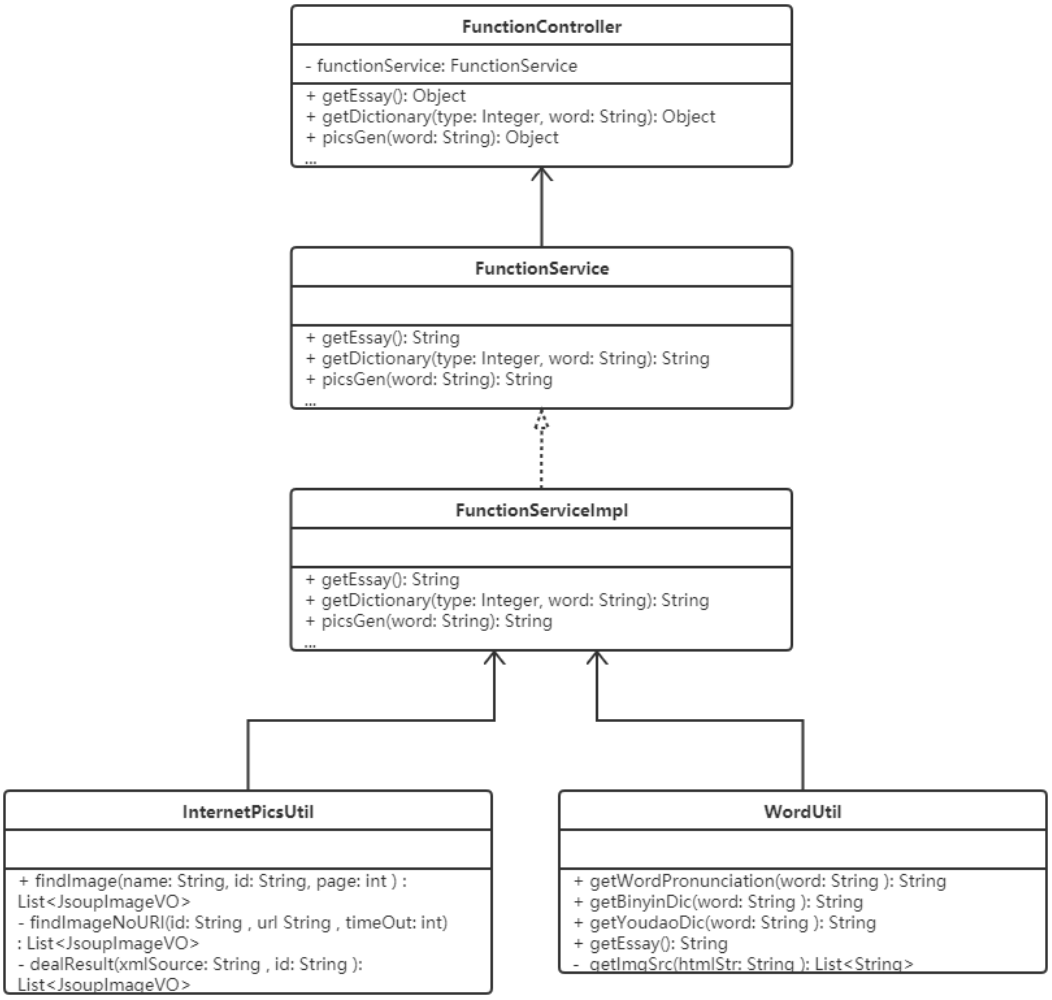

图 5.12 其它功能模块类设计

5.4.2 文章推送的核心实现

```
1.  public static String getEssay()
2.  {
3.      int max=179604,min=1;
4.      String r = "";
5.    while (true) {
6.      int ran2 = (int) (Math.random() * (max - min) + min);
7.      int a = ran2 + 1000000;
8.      String url = "https://www.globaltimes.cn/content/" + a + ".sht
    ml";
9.      String p = null;
10.     String result = HttpRequestUtil.sendGet(url, p);
11.     Matcher m = compile("<br><br>.*<br><br>").matcher(result);
12.     if (!m.find()) {
13.         Matcher m1 = compile("<br /><br />.*<br /><br />").matcher
    (result);
14.         try { r = m1.group(0); break; }
15.         catch (Exception e) {continue; }
16.     } else {
17.         try { r = m.group(0); break; }
18.         catch (Exception e) { continue; }
19.     }
20.   }
21.   return r;
22. }
```

图 5.13 文章推送界面

5.4.3 关联图片搜索的核心实现

```
1. private static List<JsoupImageVO> findImageNoURl(String id, String
    url, int timeOut) {
2.     List<JsoupImageVO> result = new ArrayList<JsoupImageVO>();
3.      Document document = null;
4.      try {
5.          document = Jsoup.connect(url).data("query", "Java")
              .userAgent("Mozilla/4.0 (compatible; MSIE 9.0; Windows
    NT 6.1; Trident/5.0)").timeout(timeOut).get();
6.          String xmlSource = document.toString();
7.          result = dealResult(xmlSource, id);
8.      } catch (Exception e) {
9.          String defaultURL = "http://qnimg.zowoyoo.com/img/15463/15
   09533934407.jpg";
10.         result = dealResult(defaultURL,id);
11.     }
12.     return result;
13. }
14. public static List<JsoupImageVO> findImage(String name, String id,
    int page) {
15.     int number=5;
16.     String url = "http://image.baidu.com/search/avatarjson?tn=resu
   ltjsonavatarnew&ie=utf-
   8&word=" + name + "&cg=star&pn=" + page * 30 + "&rn="+number+"&itg
   =0&z=0&fr=&width=&height=&lm=-1&ic=0&s=0&st=-
   1&gsm=" + Integer.toHexString(page * 30);
17.     int timeOut = 5000;
18.     return findImageNoURl(id, url, timeOut);
19. }
20. private static List<JsoupImageVO> dealResult(String xmlSource, Str
   ing id) {
21.     List<JsoupImageVO> result = new ArrayList<JsoupImageVO>();
22.     xmlSource = StringEscapeUtils.unescapeHtml3(xmlSource);
23.     String reg ="objURL\":\"http://.+?\\.(gif|jpeg|png|jpg|bmp)";
24.     Pattern pattern = Pattern.compile(reg);
25.     Matcher m = pattern.matcher(xmlSource);
26.     while (m.find()) {
27.         JsoupImageVO jsoupImageVO = new JsoupImageVO();
28.         String imageURL = m.group().substring(9);
29.         if(imageURL==null || "".equals(imageURL)){
```

```
            String defaultURL = "http://qnimg.zowoyoo.com/img/15463/1509533934407.jpg";
            jsoupImageVO.setUrl(defaultURL);
        }else{
            jsoupImageVO.setUrl(imageURL);
        }
        jsoupImageVO.setName(id);
        result.add(jsoupImageVO);
    }
    return result;
}
```

5.4.3 单词查询的核心实现

```
public static String getBinyinDic(String word)
{
    String r="";
    String url="http://cn.bing.com/dict/search";
    String p="q="+word;
    String result = HttpRequestUtil.sendGet(url,p);
    r =WordUtil.getImgSrc(result).get(0);
    Matcher m = compile("\".*\"").matcher(r);
    m.find();
    r=m.group(0).substring(1,m.group(0).length()-1);
    return r;
}

public static String getYoudaoDic(String word)
{
    String r="";
    String url="http://fanyi.youdao.com/translate";
    String p="&doctype=json&type=AUTO&i="+word;
    String result = HttpRequestUtil.sendGet(url,p);
    Matcher m = compile("tgt\":\".*\"").matcher(result);
    m.find();
    r=m.group(0).substring(6,m.group(0).length()-1);
    return r;
}
```

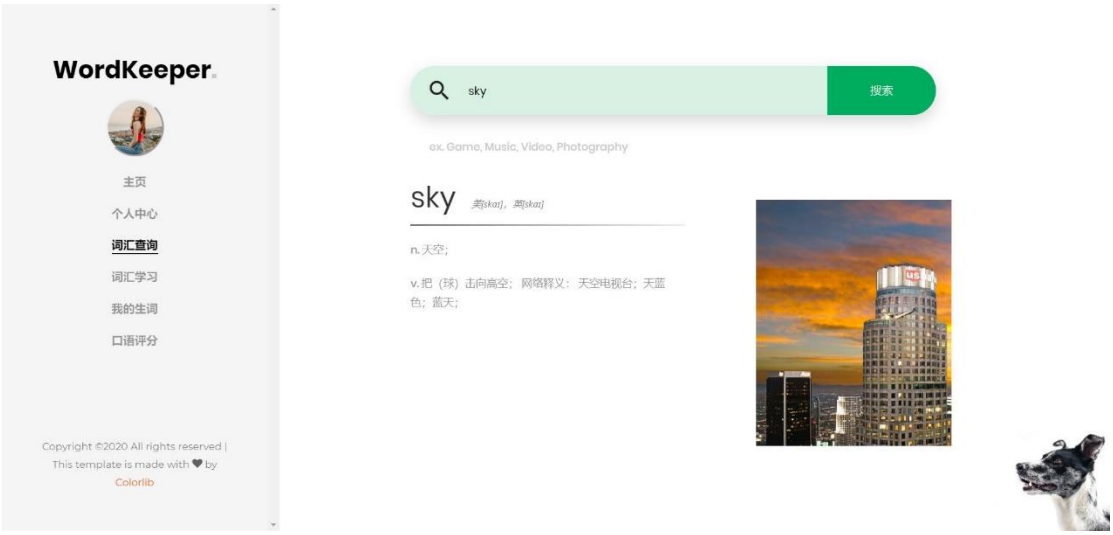
WordKeeper
主页
个人中心
词汇查询
词汇学习
我的生词
口语评分
Copyright ©2020 All rights reserved |
This template is made with ♥ by
Colorlib
sky
搜索
ex. Game, Music, Video, Photography
sky
n.天空;
v.把（球）击向高空；网络释义：天空电视台；天蓝色；蓝天;

图 5.14 单词查询界面

第 6 章 系统测试

6.1 测试方法

6.1.1 单元测试

单元测试采用 IDEA 编译器，基于白盒测试的方法设计测试用例，以求达到 100% 的代码覆盖率。

6.1.2 功能测试

功能测试使用 QTP 自动测试化工具为项目录制脚本，采用自动化测试的好处在于之后的回归测试只需要做少量的修改就能够继续完成了，从而节省了人力成本，在功能测试中对于单词量的测试，主要与词汇大纲进行单词数量的对比；非标准考试则根据考试涉及的范围进行词汇量评估后与网站内该词汇量进行对比；记忆效果的测试参考艾宾浩斯记忆曲线对其单词记忆的分布情况进行分析。

6.1.3 性能测试

性能测试则使用当前较为热门的 JMeter 工具通过增加网站的并发数，分析网站对于不同并发用户数量的响应情况。

6.2 测试用例

详见《测试报告》。

6.3 测试结果

详见《测试报告》。

第 7 章 总结和展望

总 结

本文所设计并实现的单词管家 wordkeeper 是参考了市面上的单词学习系统，和结合自身生活学习的实际需求和痛点后进行的，并在此基础上融入了深度学习以及数据挖掘等技术。在整个设计和开发的过程自己感觉受益良多。以下是本系统在设计与开发过程中的总结：

本项目在疫情之初就开始计划进行，但是由于疫情关系的赛制调整，停滞了很长时间，本项目的完成度和当初预计的还是有差距，主要原因就是因为进度计划的容错能力不行，没有考虑到疫情的影响。

展望

由于自身开发经验的不足，系统还有许多需要改进的地方，如下：

1.项目分析阶段应尽可能预先想到系统可能存在的问题和不足，这样才能避免开发进行到一半时又回过头来改进之前的开发，造成无效开发和时间的浪费。

2.整个系统的开发过程中以实现功能为紧要目标，并没有考虑代码冗余的情况，以及代码执行效率。

3.当今互联网世界，网络安全问题越来越得到人们的重视，本系统有存在安全隐患的风险，因在系统安全性方面没有深入研究，需要日后不断升级改进。