

毕业设计（论文）

题目名称：古诗管理与生成系统

学院名称：软件学院

班 级：RB软工卓越161

学 号：201619150329

学生姓名：胡帅

指导教师：王尧

2020年 5月

**论文编号：201619150329**

**古诗管理与生成系统**

Ancient Poetry Management and Generation System

学院名称：软件学院

班 级：RB软工卓越161

学 号：201619150329

学生姓名：胡帅

指导教师：王尧

2020年 5月

# 摘 要

古诗是中国悠久历史和文化的重要组成部分，生动地反映了中国古代文化的内涵和基本精神。古诗是语言[艺术](https://baike.baidu.com/item/%E8%89%BA%E6%9C%AF)的凝练，是[民族](https://baike.baidu.com/item/%E6%B0%91%E6%97%8F/665)的精神与[心灵史](https://baike.baidu.com/item/%E5%BF%83%E7%81%B5%E5%8F%B2/6623407)，也是文化的主要形态之一，古诗生成与管理系统是基于近些年机器学习的蓬勃发展而实现计算机自动生成古诗。该系统利用神经循环网络算法Seq2Seq和各种软件资源，为使用者提供一个有趣的能够自动生成古诗的网页程序，同时也是对自己大学期间所学知识结合具体实践项目的重要体现。

本文主要实现的是古诗的一些的基本管理和生成一首古诗。在这个系统中综合应用了MongoDB、JSP、Python等知识。首先，在网页前台方面，采用的是Vue和iview模板实现。其次，对于SSM框架的搭建、Java程序编程、Vue编程的热加载以及MongoDB、Python深度学习进行学习和实践。本系统在学院内还没有开发过，但本文有信心完成本文工作。

**关键词**：古诗生成；Seq2Seq；Vue；

**Abstract**

Ancient poems are an important part of China's long history and culture, vividly reflecting the connotation and basic spirit of ancient Chinese culture. Ancient poetry literature is the art of language, the national spirit and spiritual history, and one of the main forms of culture. The ancient poetry generation and management system is based on the vigorous development of machine learning in recent years to realize the automatic generation of ancient poetry by computers. The system uses the neural cycle network algorithm Seq2Seq and various software resources to provide users with an interesting web program that can automatically generate ancient poems. It is also an important manifestation of the knowledge learned during the university combined with specific practical projects.

This article mainly implements some basic management of ancient poems and generates an ancient poem. In this system, MongoDB, JSP, Python and other knowledge are comprehensively applied. First of all, in the front page of the webpage, Vue and iview templates are used. Secondly, learn and practice the construction of SSM framework, hot loading of Java programming, Vue programming, and deep learning of MongoDB and Python. This system has not been developed in the college, but this article is confident to complete the work of this article.

**Keywords**: Ancient poem generation; Seq2Seq; Vue;

目录

[第1章 项目简介 1](#_Toc40638720)

[1.1 项目背景 1](#_Toc40638721)

[1.2 研究现状 1](#_Toc40638722)

[第2章 系统需求 3](#_Toc40638723)

[2.1 系统概述 3](#_Toc40638724)

[2.2 系统功能 3](#_Toc40638725)

[2.3 参与者列表 4](#_Toc40638726)

[2.4 系统运行 4](#_Toc40638727)

[2.4.1 运行环境 4](#_Toc40638728)

[2.4.2 开发工具 5](#_Toc40638729)

[2.5 用例规约 5](#_Toc40638730)

[2.5.1 古诗搜索 5](#_Toc40638731)

[2.5.2 古诗管理 7](#_Toc40638732)

[2.5.3 古诗问答 12](#_Toc40638733)

[2.5.4 古诗生成 13](#_Toc40638734)

[2.5.5 登录 15](#_Toc40638735)

[2.5.6 用户-注销 16](#_Toc40638736)

[2.5.7 用户管理 17](#_Toc40638737)

[第3章 系统分析 19](#_Toc40638738)

[3.1 项目流程 19](#_Toc40638739)

[3.2 静态模型 19](#_Toc40638740)

[3.2.1 实体类 19](#_Toc40638741)

[3.2.2 边界类 21](#_Toc40638742)

[3.2.3 控制类 21](#_Toc40638743)

[3.3 分析顺序图 22](#_Toc40638744)

[3.4 算法分析 32](#_Toc40638745)

[3.4.1 RNN算法 32](#_Toc40638746)

[3.4.2 LSTM算法 33](#_Toc40638747)

[3.4.3 Seq2Seq 36](#_Toc40638748)

[第4章 系统设计 37](#_Toc40638749)

[4.1 架构设计 37](#_Toc40638750)

[4.2 设计类 38](#_Toc40638751)

[4.3 数据库设计 38](#_Toc40638752)

[4.3.1 概念模型 39](#_Toc40638753)

[4.3.2 关系模式 39](#_Toc40638754)

[4.3.3 物理模型 39](#_Toc40638755)

[4.4 算法设计 41](#_Toc40638756)

[第5章 系统实现 42](#_Toc40638757)

[5.1 古诗生成 42](#_Toc40638758)

[5.2 古诗管理 43](#_Toc40638759)

[5.3 古诗搜索页面 45](#_Toc40638760)

[5.4 用户管理 46](#_Toc40638761)

[5.5 登陆页面 47](#_Toc40638762)

[第6章 系统测试 49](#_Toc40638763)

[6.1 古诗搜索测试 49](#_Toc40638764)

[6.2 古诗管理 49](#_Toc40638765)

[6.2.1 古诗操作列表显示 49](#_Toc40638766)

[6.2.2 添加古诗 50](#_Toc40638767)

[6.2.3 修改古诗信息 51](#_Toc40638768)

[6.2.4 古诗分类 52](#_Toc40638769)

[6.2.5 古诗标签 52](#_Toc40638770)

[6.3 古诗生成 53](#_Toc40638771)

[6.4 用户管理 54](#_Toc40638772)

[6.5 结束语 55](#_Toc40638773)

[致谢 57](#_Toc40638774)

[参考文献 58](#_Toc40638775)

[附录A 古诗管理与生成系统核心代码 59](#_Toc40638776)

[附录B: 软件使用说明书 64](#_Toc40638777)

# 第1章 项目简介

## 1.1 项目背景

古诗管理与生成系统是基于近些年深度学习领域自然语言处理的快速发展，以及对自然语言处理的理解，开发的一个能够对古诗进行管理、生成古诗的系统。本系统利用计算机的信息系统、算法研究和各种软件资源，为爱好古诗的用户和传承中国古诗文化，提供了一个有趣智能的古诗自动生成的程序，同时也是自己对算法领域应用的一个初步的探索。

古诗生成与管理系统根据社会需求调查的结果下，将整个系统模块分为：古诗列表展示、古诗的基本操作、古诗生成和用户管理四个模块。本系统使用Vue框架进行页面的设计和实现，在用户使用上充分体现了美观、简洁、方便性为一体，提高了用户的使用体验，系统页面可以满足用户对古诗操作的一些基本需求，使用者只需要按照系统页面提示进行一系列操作或者输入符合要求的正确的条件就能充分使用该系统所有功能。

古诗管理系统是为了在学习古诗的同时提高用户对古诗的兴趣，对传承古诗文化做出贡献的系统。用户在系统上能够及时的查看想要查看的古诗，同时能够上传古诗库中没有的古诗，能够对古诗的信息进行维护和修改，系统能够按照一定的策略对古诗进行分析，进而分类古诗的特征，当然最大的特点就是能够根据用户输入的关键字自动生成一首古诗，极大的提高用户对系统和古诗兴趣。古诗管理与生成系统运用完善的mongodb数据库保证了相关数据的安全性，系统项目流程结构严谨、安全，运行效率高。整个系统对于使用者来说操作简单，系统设计界面美观、友好，系统运行流畅，安全，可以有效完成古诗的生成。

该古诗生成系统使用计算机算法和不同编程软件资源，为用户提供了一个有趣的古诗生成和管理系统的网页程序。

## 1.2 研究现状

近年来，互联网的迅猛发展以及人们生产生活不断加速的前进，无数的文本数据和各式各样的信息随着人们的行为而产生，从一方面来讲人们希望快速并且高效的处理这些文本信息，从而提取其中重要的信息加以利用，另一方面，由于数据的庞大和处理的繁琐性，使用机器取代人工处理数据成为迫切的需要。而随着人类在自然语言处理领域的不断探究，以及计算机应用技术的不断发展和成熟，收获了很多的优秀成果。

中国古诗文化是文学邻域的精髓，对语言运用能力要求高，古诗作者需要拥有非常高的艺术构想力、文字掌控力以及主要的艺术创作灵感，古诗的创作也是人工智能最难解决的领域。但是创造诗词的能力并不是说天生就有的。它仍然是由人类语言逻辑方面的广泛培训和丰富经验组成的，是有迹可循的。因为诗词有一定的逻辑和大数据作为生成古诗的基础，所以人工智能具有相应的学习诗歌创作的方式。只要数据库足够大并且古诗编写程序符合古诗创作过程，那么利用机器人生成古诗就不会困难，困难的是写一首被人们认可的好诗。利用计算机来作诗是很有意思，也具有挑战性的任务。近年来，对自动作诗的研究也越来越多，方法从最初基于规则和模板的方法发展到基于神经网络序列到序列（Seq2Seq）的方法，这些方法大多都在关注生成诗词的质量上，比如连贯性，一致性等。

古诗管理与生成系统正式基于以上技术实现的一个具有完善性的系统，不仅包含古诗的自动生成，还包含古诗的一些基本操作，整个系统从用户角度出发，以简单、友好、有趣、灵活的页面保证用户要求。完成用户与古诗之间的交流，进一步的提高大家对诗的认识兴趣，能够发扬中华文化。系统采用前后端分离式、流程化的实现方案，能够有效的管理和解决系统运行过程中的每一个步骤，使整个系统拥有更好的完善性和运行效率性。

# 第2章 系统需求

## 2.1 系统概述

古诗生成与管理系统是一个关于古诗自动生成和古诗操作的系统化管理软件。

用户可以在系统中查看库内所有的古诗、按关键字查找古诗、添加古诗、对古诗进行修改、对机器进行提问古诗、同时能够对用户输入的关键字生成相应的古诗。

## 2.2 系统功能

管理员主要功能的用例图如图2.1所示，管理员拥有所有操作功能主页、古诗搜索、古诗列表查看、古诗修改、添加、删除等基本功能，同时拥有生成古诗、生成藏头诗、机器问答、用户管理功能。

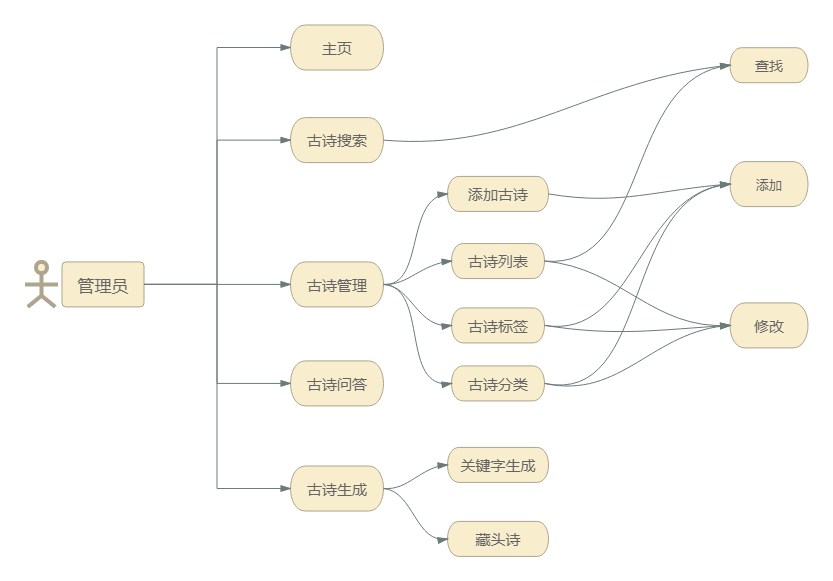


图2-1系统结构图

在分析了图2-1中系统结构图之后，抽象出了管理员的16个功能，具体用例列表如表2-1所示。

表2-1 系统功能用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 类别 | 功能描述 | 子功能描述 | 功能编号 |
| 管理员 | 古诗搜索 | 古诗信息列表显示 | FUN-1-1 |
| 查询古诗信息 | FUN-1-2 |
| 古诗管理 | 古诗信息列表显示、删除 | FUN-2-1 |
| 添加一首古诗 | FUN-2-2 |
| 古诗信息修改 | FUN-2-3 |
| 古诗包含的类别 | FUN-2-4 |
| 古诗包含的标签 | FUN-2-5 |
| 古诗问答 | 根据提问回答古诗 | FUN-3-1 |
|  | 按关键字生成一首古诗 | FUN-4-1 |
| 切换按钮并生成藏头诗 | FUN-4-2 |
| 将库内古诗信息对照 | FUN-4-3 |
| 基础 | 登陆 | 登陆 | FUN-5-1 |
| 注销 | 注销 | FUN-5-2 |
| 用户管理 | 对用户系统用户进行管理 | FUN-5-3 |

## 2.3 参与者列表

古诗生成与管理系统主要的参与者就是管理员，管理员登录成功后能够使用系统的全部功能，并在一定的限制条件下进行系统的具体操作。

## 2.4 系统运行

### 2.4.1 运行环境

1. JDK 1.8.0版本。
2. MongoDB 4.0版本。
3. Tomcat 9.0版本。
4. Python 3.6版本
5. Tensorflow 2.0版本

### 2.4.2 开发工具

1、IDE：IDEA、PyCharm

本课题因为系统需要的的运行环境不同，分别用了IDEA和PyCharm开发工具。IDEA 全称IntelliJ IDEA，是适用于java语言开发的一种高效、简单的集成环境（当然也可用于其编码语言），本系统使用IDEA编辑器的主要目的就是为了在开发中能够简化代码开发难度，加快代码开发的进程，其中包含基本的代码提示功能和更加智能的代码提示功能，该提示功能可用与上下文环境智能过滤，等一些强大的代码审查。同时它也支持各种常用环境的整合。

PyCharm是JetBrains于2010年开发的一个用于计算机编程的IDE。虽然这个IDE主要是为Python创建的，但是来自Team-Java的用户也可以使用PyCharm。一些使用户倾向于使用它的特性是，它提供代码分析，与图形调试器VCSes集成，并与Django、flask开发，与Anaconda开发数据科学，是用来学习算法的重要工具。

## 2.5 用例规约

### 2.5.1 古诗搜索

古诗搜索包含古诗信息的显示功能，按照古诗标题关键字搜索数据库内古诗。

1. 古诗搜索列表和关键字搜索的参与者是系统使用者，用于用户登录后进入页面时首要显示所有古诗信息，包含古诗正文内容、古诗的作者、古诗的标签和古诗的类别。用例规约如表2-2所示。

表2-2 古诗信息列表显示

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-1-1 | 功能描述 | 古诗信息列表显示 |
| 前置条件 | 使用账号密码进行登录古诗管理与生成系统，选择古诗搜索选项卡即可显示列表。 | | |
| 基本事件流 | 1.点击左侧菜单栏“古诗搜索”。  2.进入页面，搜索关键字为空，系统自动获取并展示数据库内所有古诗信息，同时进行页面的渲染。  3.显示内容包括古诗正文内容、古诗的作者、古诗的标签和古诗的类别。  4.古诗信息列表的展示不分先后。  5.页面提供输入关键字查询具体古诗、并显示在页面上。  6.页面提供首页和退出功能。 | | |
| 后置条件 | 跳转至相应的操作界面 | | |
| 原型设计图 | 见图2-2 | | |
| 操作员 | 管理员 | | |



图2-2 古诗搜索

1. 查询数据库内古诗信息的参与者是系统使用者，用于主动查询一条信息显示在页面中。用例规约如表2-3所示。

表2-3查询古诗信息

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-1-2 | 功能描述 | 查询古诗信息 |
| 前置条件 | 使用账号密码登录古诗生成与管理系统，选择古诗搜索菜单，在列表页面输入关键字并单击“搜索古诗”按钮。 | | |
| 基本事件流 | 1. 在古诗搜索页面的右侧控制台卡片中进行操作 2. 根据提示输入要查绚古诗的关键字，包含古诗作者、作品的名称。   3.输入正确，点击搜索古诗，想要查询的古诗就会覆盖默认的显示所有列表。输入错误点击提交会进行弹框提示，如果输入为空则默认显示所有古诗的列表。  4.页面提供首页和注销功能。 | | |
| 后置条件 | 点击搜索古诗会显示所有古诗信息列表 | | |
| 原型设计图 | 见图2-3 | | |
| 操作员 | 管理员 | | |



图2-3 古诗关键字搜索

### 2.5.2 古诗管理

古诗管理包括。对库内古诗的显示、修改、删除，添加一首古诗，古诗标签管理，古诗分类管理。

1. 古诗列表显示所有古诗，支持按古诗的类别进行查找，用于在刚进入古诗管理页面时显示所有古诗信息，同时提供管理员能够对古诗进行各种操作的按钮。用例规约如表2-4所示。

表2-4 古诗列表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-2-1 | 功能描述 | 古诗列表 |
| 前置条件 | 使用账号密码登录古诗生成与管理系统，在右侧选项卡中选择古诗管理。 | | |
| 基本事件流 | 1. 系统默认展示所有的古诗信息。  2. 显示内容包括古诗编号、古诗标题、古诗创建时间、古诗作者、古诗类别、操作按钮等。  3. 古诗列表展示按上传时间倒序排列。  4. 页面提供修改古诗信息入口、删除古诗操作、修改古诗状态等操作按钮，不同按钮提示对应的信息。  5. 古诗列表展示提供分页显示功能。  6. 页面提供按照古诗分类进行古诗的筛选，提高刷新列表的功能。  7.页面提供首页和退出功能。 | | |
| 后置条件 | 跳转至相应的操作界面 | | |
| 原型设计图 | 见图2-4 | | |
| 操作员 | 管理员 | | |



图2-4 古诗列表显示

1. 添加古诗参与者是系统使用者，用于在系统中添加一首古诗并保存到数据库中，其中古诗的分类和标签是数据库中已经存在的，如果数据库中没有对应的标签或分类则需要提前添加。用例规约如表2-5所示。

表2-5 添加古诗

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-2-2 | 功能描述 | 添加古诗 |
| 前置条件 | 以管理员身份登录古诗生成与管理系统，在右侧的选项卡中选择古诗管理—>添加古诗，进入添加古诗页面。 | | |
| 基本事件流 | 1．添加古诗包含古诗标题、古诗作者、古诗状态、选择古诗标签和古诗分类、添加古诗正文。  2．以上内容都是必须添加不能为空，否则会显示红色错误信息，古诗韵律可以选填。  3．古诗标签和古诗分类都是手动选择，必须为数据库中已经存在的，否则就需要在古诗分类和古诗标签中提前添加对应信息。  5．点击右侧的发布按钮会提示是否保存成功。  4．页面提供首页和注销功能。 | | |
| 后置条件 | 点击后提示是否保存成功 | | |
| 原型设计图 | 见图2-5 | | |
| 操作员 | 管理员 | | |



图2-5 添加古诗

1. 修改具体古诗的参与者是系统使用者，用于管理员对一首古诗根据需要修改其古诗信息，添加古诗韵律，其中古诗的分类和标签是数据库中已经存在的，如果数据库中没有对应的标签或分类则需要提前添加。用例规约如表2-6所示。

表2-6 修改古诗信息

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-2-3 | 功能描述 | 修改古诗信息 |
| 前置条件 | 以管理员身份登录古诗生成与管理系统，在右侧的选项卡中选择古诗管理—>古诗列表，点击“修改”按钮，跳转修改古诗页面。 | | |
| 基本事件流 | 1．古诗包含古诗标题、古诗作者、古诗状态、选择古诗标签和古诗分类、添加古诗正文。  2．以上内容都是修改不能为空，否则会显示红色错误信息，古诗韵律可以选填。  3．古诗标签和古诗分类都是手动选择，必须为数据库中已经存在的，否则就需要在古诗分类和古诗标签中提前添加对应信息。  5．点击右侧的“修改”按钮会提示是否保存成功。  4．页面提供首页和注销功能。 | | |
| 后置条件 | 修改成功后自动跳转到古诗列表显示页面 | | |
| 原型设计图 | 见图2-6 | | |
| 操作员 | 管理员 | | |



图2-9 修改古诗信息

1. 古诗分类的参与者是系统使用者，用于对添加和修改古诗时提供选择的先置条件，并对古诗简单分类。用例规约如表2-10所示。

表2-10 古诗分类

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-2-4 | 功能描述 | 古诗分类 |
| 前置条件 | 使用账号密码登录古诗生成与管理系统，在右侧的选项卡中选择古诗管理—>古诗分类 | | |
| 基本事件流 | 1. 显示当前古诗的类别、每个类别下分别有哪些子类别。  2. 显示标签分类管理的说明。  3. 每个分类父类都是不可删除和修改的对象。  4. 分类父类下可以对分类类别进行管理，能够修改、增加和删除。  5. 提供刷新的按钮功能。 | | |
| 后置条件 | 系统中其它类别都是基于古诗分类存在而存在的。 | | |
| 原型设计图 | 见图2-10 | | |
| 操作员 | 管理员 | | |

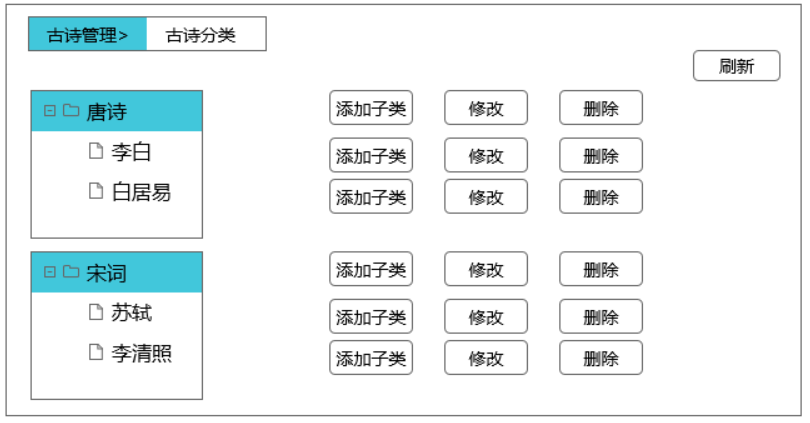


图2-10 古诗分类

1. 古诗标签的参与者是系统使用者，用于对添加和修改古诗时提供标签选择的先置条件，并对古诗进行简单的标签处理。用例规约如表2-11所示。

表2-11 古诗标签

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-2-5 | 功能描述 | 古诗标签 |
| 前置条件 | 以管理员身份登录古诗生成与管理系统，在右侧的菜单选项卡中选择古诗管理—>古诗标签 | | |
| 基本事件流 | 1. 显示当前古诗的所有标签、每个标签下分别有哪些子标签。  2. 显示标签分类管理的说明。  3. 每个标签父类都是不可修改的对象。  4. 分类标签下可以对标签类型进行管理，能够修改标签、增加和删除标签。  5. 提供刷新、添加一级标签按钮。 | | |
| 后置条件 | 系统中其它标签都是基于古诗标签存在而存在的。 | | |
| 原型设计图 | 见图2-11 | | |
| 操作员 | 管理员 | | |



图2-11 古诗标签

### 2.5.3 古诗问答

用户输入指定格式的语句格式，自动回复一首与用户输入有关的一首古诗。

1. 古诗问答用例的参与者是系统使用者，用于根据用户的需要回复对应内容。用例规约如表2-12所示。

表2-12 古诗问答

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-3-1 | 功能描述 | 古诗问答 |
| 前置条件 | 使用账号密码登录古诗管理与生成系统，在左侧选项卡中选择-古诗问答。 | | |
| 基本事件流 | 1. 以对话的形式显示回复内容。  2. 使用者按照固定的格式进行提问，机器自动回复一首古诗，或者相对应得语句。  3. 回复内容基于库内古诗的精确查找。  4. 页面提供首页和注销功能。 | | |
| 后置条件 |  | | |
| 原型设计图 | 见图2-11 | | |
| 操作员 | 管理员 | | |

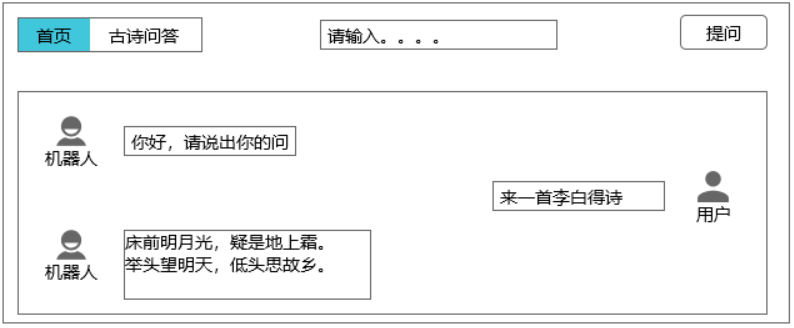


图2-11 古诗问答

### 2.5.4 古诗生成

古诗生成主要包括使用者输入一个关键字或者关键词，系统根据关键字、关键词生成一首古诗，或者输入四字成语系统自动回复一首以四个字为开头的古诗。回复内容通过对话的形式展现

1. 关键词古诗生成用例的参与者是系统使用者，用于使用者在输入关键词后自动生成一首古诗，并及时的显示在页面上。用例规约如表2-12所示。

表2-12 关键词古诗生成

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-4-2 | 功能描述 | 关键词古诗生成 |
| 前置条件 | 使用账号密码登录古诗生成与管理系统，右侧选项卡中选择古诗生成。 | | |
| 基本事件流 | 1. 系统默认展示对话框，系统自动打招呼。  2. 显示内容包括系统的回答、用户的输入问题，系统图标、代表用户的图标。  3. 在下方的输入框中输入关键字，或者关键词，输入太多可能读取关键字出错，正确输入，系统自动回复一首古诗。  4. 页面提供首页和注销功能。 | | |
| 后置条件 | 输出对应的古诗 | | |
| 原型设计图 | 见图2-12 | | |
| 操作员 | 管理员 | | |

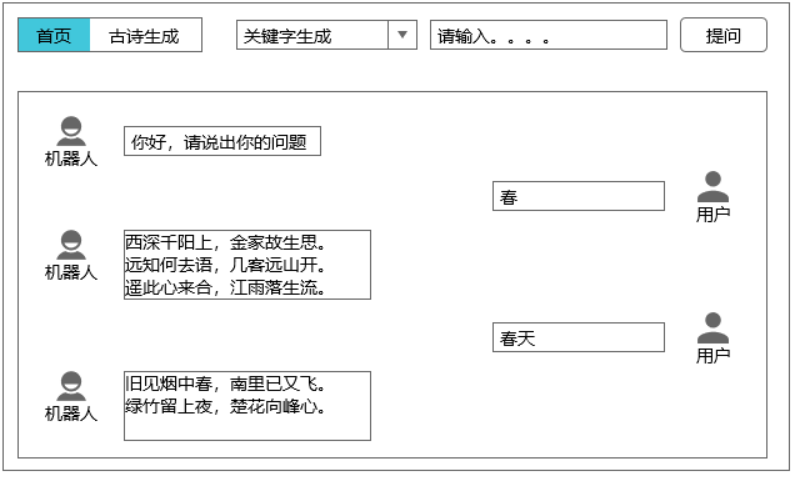


图2-12 关键词生成古诗

1. 藏头诗生成用例的参与者是系统使用者，用于使用者在输入四字成语或者四字内容后自动生成一首古诗，并及时的显示在页面上。用例规约如表2-13所示。

表2-12 藏头古诗生成

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-4-3 | 功能描述 | 藏头古诗生成 |
| 前置条件 | 以使用者身份登录古诗生成与管理系统，右侧选项卡中选择古诗生成。 | | |
| 基本事件流 | 1. 系统默认展示对话框，系统自动打招呼。  2. 显示内容包括系统的回答、用户的输入问题，系统图标代表用户的图标。  3. 在下方的输入框中输入四字内容，系统自动回复一首古诗，输入太多生成可能出错。  4. 页面提供首页和注销功能4.页面提供首页和注销功能。 | | |
| 后置条件 | 输出对应的古诗 | | |
| 原型设计图 | 见图2-13 | | |
| 操作员 | 管理员 | | |

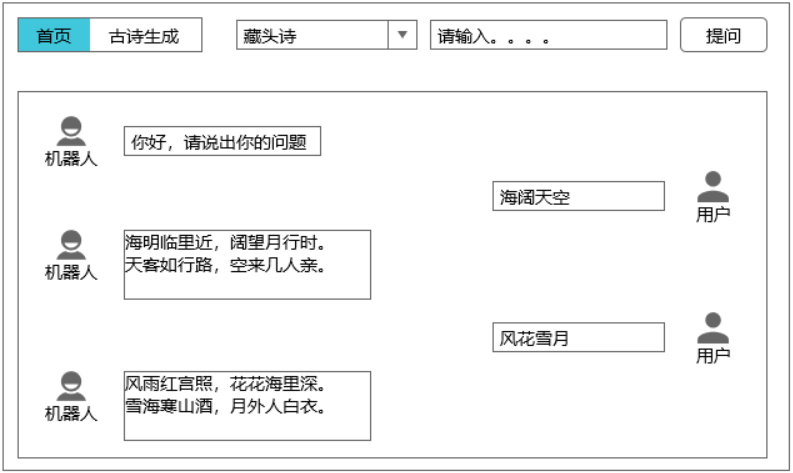


图2-13藏头古诗生成

### 2.5.5 登录

1）登录系统用例的参与者是数据库中的所有成员，用于系统使用者使用登录名和密码进行系统登录。用例规约如表2-14所示。

表2-14登录

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-5-1 | 功能描述 | 登录 |
| 前置条件 | 运行系统进入登录页面。 | | |
| 基本事件流 | 1.系统使用者进行登录时需要进行验证。  2. 使用系统开发过程中数据库中的账号和密码。  3. 用户名和密码输入后，点击“登录”按钮。系统将会检查登录账号名和登录密码，是否与数据库已存在账号、密码匹配，如果不匹配，系统将会提示错误信息。如果它们匹配，将转到系统主页。 | | |
| 后置条件 |  | | |
| 原型设计图 | 见图2-14 | | |
| 操作员 | 系统使用者 | | |



图2-14 登录

### 2.5.6 用户-注销

（1）注销用例的参与者是系统的所有使用者，用于当前登录用户点击注销后返回到登录界面，进行用户的切换。用例规约如表2-15所示。

表2-15 注销

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-5-2 | 功能描述 | 注销 |
| 前置条件 | 在页面右上角选择“退出” | | |
| 基本事件流 | 1. 用户返回到登陆界面。 | | |
| 后置条件 |  | | |
| 原型设计图 | 见图2-15 | | |
| 操作员 | 系统使用者 | | |



图2-15 用户注销

### 2.5.7 用户管理

（1）用户管理用例的参与者是系统使用者，用于当前管理员根据需要，对系统使用者添加用户、删除用户、修改登录账号信息。用例规约如表2-16所示。

表2-16 用户管理

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-5-3 | 功能描述 | 用户管理 |
| 前置条件 | 管理员登录系统后，在右侧选项卡中选择用户管理，在页面中选择想要修改账号信息的用户。 | | |
| 基本事件流 | 1. 系统显示当前所有用户信息。 2. 系统管理员账号信息无法修改。 3. 系统使用者能够添加新的用户，以及添加用户的基本信息、删除用户，对用户登录有效性进行修改等。 4. 页面提供输入关键字筛选用户、并显示在页面上。 5. 当用户数量过多时，提供页面的分页功能。 | | |
| 后置条件 | 修改成功后能够及时显示 | | |
| 原型设计 | 见图2-16 | | |
| 操作员 | 系统使用者 | | |



图2-16 用户管理

（2）使用者密码修改参与者是系统使用者，用于当前管理员根据需要，对系统使用者进行修改密码。用例规约如表2-17所示。

表2-17 修改密码

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FUN-5-4 | 功能描述 | 修改密码 |
| 前置条件 | 管理员登录系统后，在右侧选项卡中选择用户管理，在页面中选择想要修改账号信息的用户。 | | |
| 基本事件流 | 1. 新密码由字符构成，由字母和数字组成的字符串，若不符合，系统提示：请输入正确格式的密码。  2. 确认密码必须与新密码一致。  3. 已存在的账号可以选择直接进行登录。 | | |
| 后置条件 | 修改成功后能够使用新密码进行登陆 | | |
| 原型设计 | 见图2-17 | | |
| 操作员 | 系统使用者 | | |



图2-17 修改密码

# 第3章 系统分析

## 3.1 项目流程

整体项目流程如图3-1，其中古诗生成模块，用户在前端页面中输入古诗关键字，通过接口在后端获取输入数据，然后通过后端调用（Python）服务接口，将输入数据传给（Python）模型，同时后端获取输出的结果返回给前端，并展示出来。

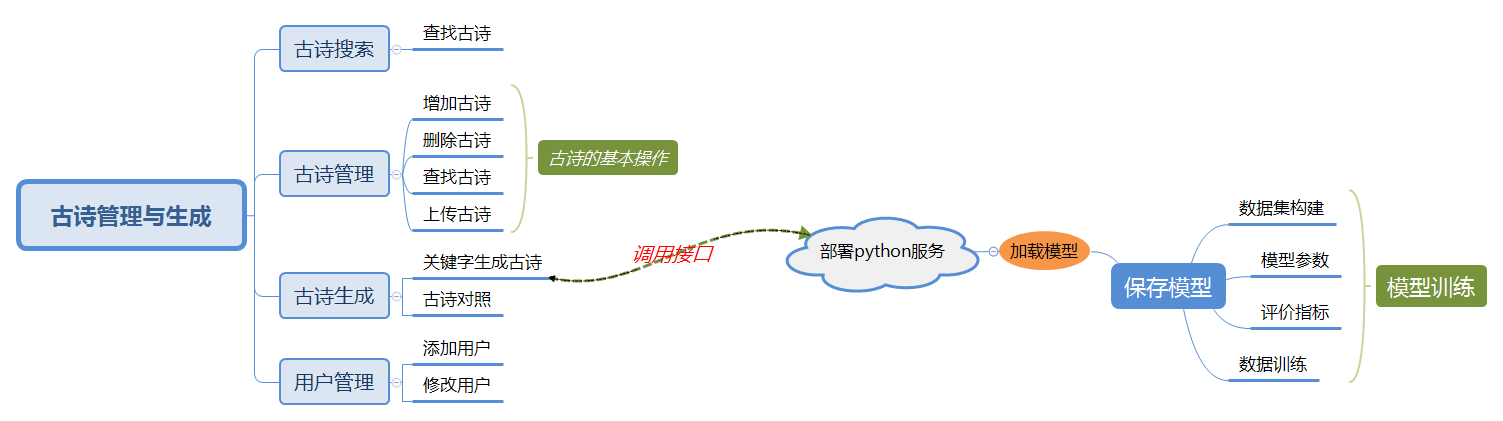


图3-1项目流程

## 3.2 静态模型

在进行系统简单分析并创建好用例后，接下来系统实现的主要目标就是找出系统中的拥有的静态模型，和对应的实体类，然后根据系统原理分析出相应的系统边界类和系统控制类，以及建立系统的动态模型。

### 3.2.1 实体类

首先需要从系统需求中提取名称，然后进行简要分析和提炼，共提取出古诗列表中的一些名词，古诗标题、古诗内容、古诗作者、古诗标签、古诗分类、创建日期、用户数据等。分析精炼出的系统实体类列表如表3-1所示。

1. 古诗实体类：主要拥有标题、id、内容、作者、状态、类别、创建日期、标签、韵律等属性。
2. 标签实体类：主要拥有标签id、权重、父类标签、子标签（子标签继承父标签）等。
3. 类别实体类：主要拥有分类id、古诗父类别、子类别（子类别继承父类别）等属性。
4. 用户管理类：主要拥有用户id、登录账号、登录密码、用户描述、用户创建日期、是由有效等一些属性。

表3-1 系统实体类描述表

|  |  |  |
| --- | --- | --- |
| **实体名称** | **实体属性** | **备注** |
| 古诗 | 题目、id、古诗内容、作者、状态、古诗分类、创建日期、古诗标签、古诗韵律 | 其中类别、标签都是数据库中古诗类别、古诗标签中已经存在定义的。 |
| 古诗标签 | 标签id、权重、父类标签、子标签 | 子标签存在于父标签下，子标签不能重复。 |
| 古诗类别 | 分类id、古诗父类别、子类别 | 子类别存在于父类别下，子标签不能重复。 |
| 用户管理 | 用户id、用户账号、用户密码、用户描述、用户创建日期、是否有效 |  |

实体类及其彼此的关联关系如图3-2所示。

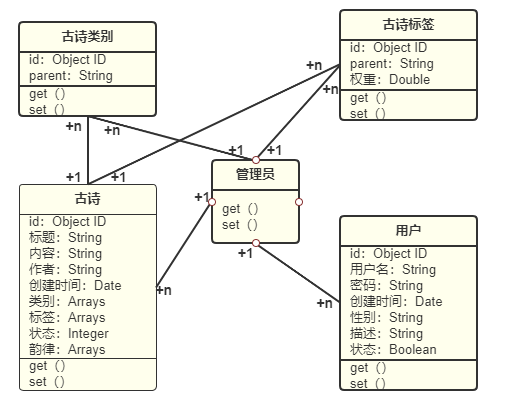


图3-2 实体类关系图

实体之间的关系主要有两种，一种是泛化关系，一种是关联关系。比如，古诗实体和用户之间是泛化关系，两者之间没有直接的关系，管理员需要继承用户的用户名和密码属性，用来进行系统登录。古诗实体和古诗类别、古诗标签之间的关系是关联关系，古诗中的类别和标签必须继承于古诗类别和古诗标签，一首古诗能够拥有多个标签，多个类别，所以是古诗和标签、类别是一对多的关系。一个管理员能够对多个古诗进行操作，同时也能够对古诗的类别和古诗的标签进行添加修改操作，所以管理员和古诗标签、古诗类别之间是一对多的关系。

### 3.2.2 边界类

系统参与者通过系统边界类和系统进行数据交互，并完成数据传输。参与者启动在对边界类的接口请求，而系统边界类会将系统参与者的请求申请（例如界面接口，对话框等）发送给系统控制类，然后由控制类调用系统实体类，完成对数据库的的操作。系统的边界类列表如表3-2所示。

表3-2 系统使用者边界类列表

|  |  |  |
| --- | --- | --- |
| **序号** | **边界类名称** | **简要** |
| BC-01 | 系统登录 |  |
| BC-02 | 古诗列表 | 同时也是系统的主界面 |
| BC-03 | 关键字搜索 | 按关键字搜索古诗信息 |
| BC-04 | 古诗的问答 | 根据指定格式回复内容 |
| BC-05 | 添加古诗 | 标签和分类数据库读取 |
| BC-06 | 古诗操作列表 | 根据时间排序 |
| BC-07 | 古诗信息修改 |  |
| BC-08 | 古诗的删除 |  |
| BC-09 | 古诗状态修改 |  |
| BC-10 | 古诗列表筛选 | 按类别筛选 |
| BC-11 | 古诗分类的信息维护 |  |
| BC-12 | 古诗标签的信息维护 |  |
| BC-13 | 古诗的关键字生成 | 生成一首古诗 |
| BC-14 | 古诗的藏头诗生成 | 生成一首藏头诗 |
| BC-15 | 用户的信息维护 | 修改、添加用户等 |

### 3.2.3 控制类

系统边界类通过控制类从而访问系统实体类。系统中所需要的控制类如表3-3所示。

表3-3 控制类列表

|  |  |  |
| --- | --- | --- |
| 序号 | 控制类名称 | 简要 |
| CONC-01 | 查询古诗列表 | 古诗搜索显示显示 |
| CONC-02 | 关键字查询古诗 | 搜索查询后的古诗 |
| CONC-03 | 古诗问答 | 回复指定的古诗数据 |
| CONC-04 | 添加古诗 | 古诗管理页面添加古诗 |
| CONC-05 | 古诗列表 | 古诗列表操作页面 |
| CONC-06 | 古诗信息修改 | 古诗管理页面、修改 |
| CONC-07 | 古诗删除 | 古诗管理页面、删除 |
| CONC-08 | 古诗分类 | 古诗管理页面、分类 |
| CONC-09 | 古诗标签 | 古诗管理页面、标签 |
| CONC-10 | 古诗生成 | 古诗生成页面 |
| CONC-11 | 用户管理 | 用户管理页面 |
| CONC-12 | 古诗生成对接 | 古诗生成页面 |

## 3.3 分析顺序图

分析顺序图可以描述对象在交互过程中的状态，以及分析其如何调用方法和怎样运行的，从而显示活动或行为展开的顺序。在本系统中每个用例都拥有自己独立的时序图，以便更加清楚地了解系统功能的运行过程。

本项目的体系结构采用MVC基础架构规范和项目管理工具Maven。其中：

M-模型层：主要是对系统的一种抽象的对象模型，主要是实体类，它包含了对数据库中数据的一些基本操作和处理。

V-表示层：用于系统、用户之间进行交互的主要服务层，为用户提供数据的呈现。

C-控制层：用于负责视图和具体模型之间的相互交互，调用业务逻辑产生合适的数据，实现把客服端用户的请求发送到对应的模型，然后决定使用哪个模型进行去处理请求返回数据。

**1.古诗搜索顺序图**

古诗搜索顺序图主要展示的是显示古诗信息列表用例。参与者通过使用登陆账号和登录密码登录系统后，默认主页进行古诗列表信息的请求，古诗搜索边界类将会把请求数据发送给控制类，控制类从数据库中查询数据并通过创建古诗实体类实现整个实例，返回请求数据。古诗搜索列表顺序图如图3-3所示。

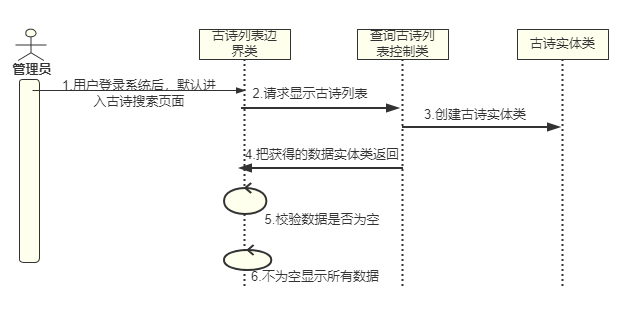


图3-3 古诗列表顺序图

**2.关键字搜索古诗顺序图**

古诗关键字搜索顺序图主要展示的是显示一首古诗信息列表用例。使用者在控制台输入框中输入要查询的古诗标题，点击发送按钮发出请求，边界类把请求发送给控制类，控制类查询数据并最终通过古诗实体类实现用例，返回请求数据。关键搜索古诗信息顺序图如图3-4所示。

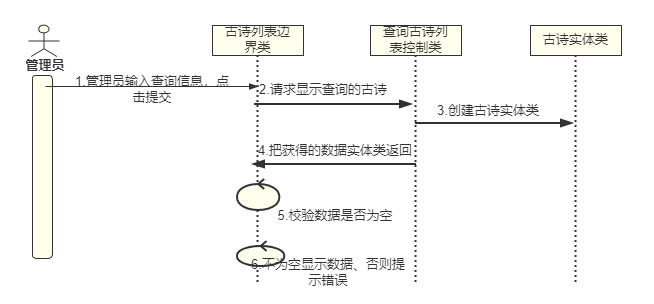


图3-4 关键字搜索古诗顺序图

1. **古诗问答顺序图**

古诗问答顺序图主要展示的是自动回复一条信息。参与者在古诗问答边界类输入限定的请求条件，请求条件无误后控制类通过处理后调用古诗问答实体类实现用例，返回问答结果。古诗问答顺序图如图3-5所示。

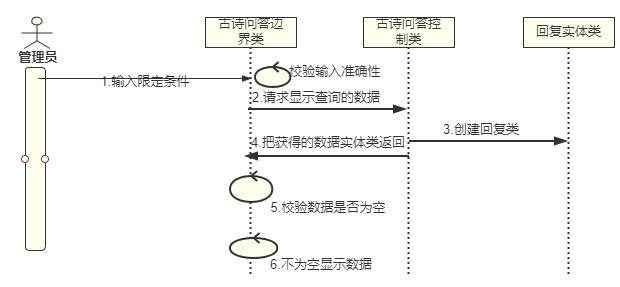


图3-5 古诗问答顺序图

1. **添加古诗顺序图**

添加一首古诗信息顺序图主要展示的是添加古诗信息用例。用户在主边界类中提出添加一首新古诗的请求，主边界类将会调用添加古诗的边界类，首先此边界类会显示数据库中已存在的古诗标签、古诗类别，在添加古诗时提供用户选择标签和分类，在古诗添加边界类中填写古诗信息，系统边界类会把添加古诗请求交给添加古诗控制类，最后由古诗控制类通过创建古诗实体类操作数据库实现添加古诗的用例。添加古诗顺序图如图3-6所示。

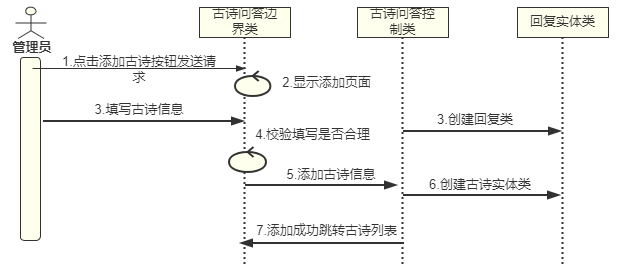


图3-6 添加古诗顺序图

**5.古诗修改顺序图**

修改古诗信息顺序图主要展示的是修改古诗信息用例。参与者在修改古诗边界类中提出请求修改确定古诗的请求，在修改古诗信息边界类中边界类会通过控制类调用实体类查询数据库并返回当前古诗的数据信息，确定要修改的信息，并通过修改古诗控制类进行调用古诗实体类进行修改数据库，最后返回结果实现用例。古诗修改顺序图如图3-7所示。

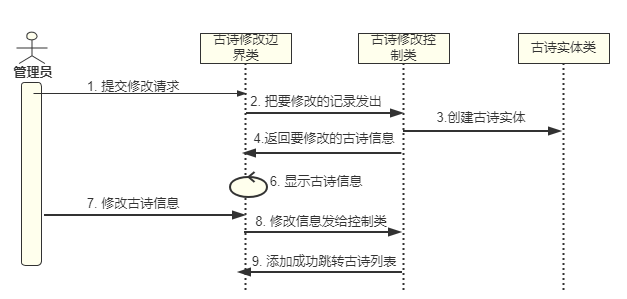


图3-7 古诗修改顺序图

**6.古诗删除顺序图**

古诗删除顺序图是主要展示删除一首古诗的用例。在古诗操作的列表中，参与者在边界类中发出删除指定的一首古诗的请求，主边界类获取要删除的古诗标识，通过古删除控制类调用古诗实体类修改数据库，返回结果实现用例。删除古诗顺序图如图3-8所示。

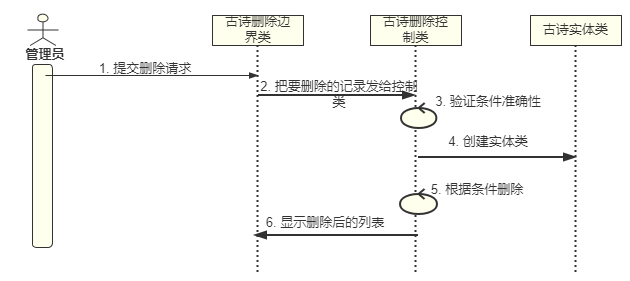


图3-8 古诗删除顺序图

1. **古诗分类列表显示顺序图**

古诗分类列表显示顺序图主要展示的是显示古诗分类信息列表用例。系统使用者通过点击页面左侧选项卡的古诗分类发送显示所有古诗分类的请求，系统主边界类将会把请求发送给控制类，控制类将会通过调用古诗分类实体类查询分类信息并返回数据实现实例。古诗信息分类列表显示顺序图如图3-9所示。

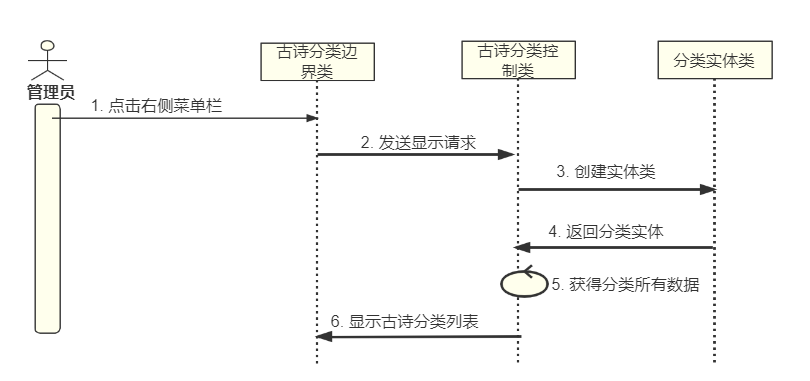


图3-9 古诗分类列表显示顺序图

**8.添加古诗分类顺序图**

添加古诗分类顺序图主要展示的是添加古诗类别用例。使用者在古诗分类边界类中发出添加古诗子类别的请求，古诗分类边界类将会调用添加古诗类别边界类，此边界类将会展示数据库中古诗类别的基本信息，提供对古诗类别的添加操作，在添加古诗类别边界类中添加想要添加的古诗类别，然后边界类将添加请求发送给控制类，然后由古诗分类控制类调用创建古诗分类实体类操作数据库实现用例。增加古诗分类顺序图如图3-10所示。

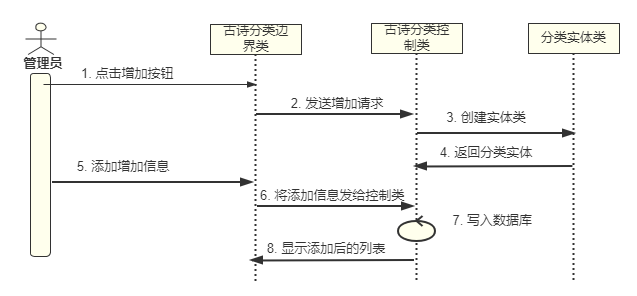


图3-10 增加古诗分类顺序图

1. **修改古诗分类顺序图**

修改古诗分类顺序图主要展示的是修改古诗分类用例。在修改古诗边界类中使用者请求修改具体的古诗分类信息，并判断是否是父类，首先古诗分类边界类会通过调用古诗分类实体类从数据库中返回当前指定分类信息，请求发出者在修改古诗分类信息后，古诗分类边界类创建分类实体类修改数据库实现用例。修改古诗分类信息顺序图如图3-11所示。

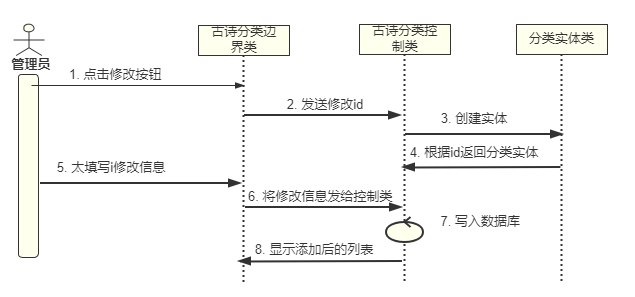


图3-11修改古诗分类信息顺序图

1. **古诗标签显示顺序图**

古诗标签列表显示顺序图主要展示的是显示古诗标签信息列表用例。使用者通过点击页面左侧选项卡古诗标签发送获取古诗标签数据的请求，古诗标签边界类将会把请求发送给标签控制类，标签控制类将会调用标签实体类查询数据并返回数据实现实例。古诗信息标签列表显示顺序图如图3-12所示。

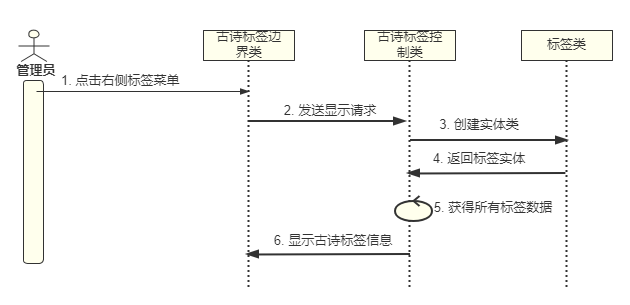


图3-12古诗标签列表显示顺序图

1. **添加古诗标签顺序图**

添加古诗标签顺序图主要展示的是添加古诗标签用例。使用者在古诗标签边界类中发出添加古诗标签或子标签的请求，古诗标签边界类将会调用添加古诗标签边界类，此边界类将会展示数据库中古诗标签的基本信息，提供对古诗标签的添加操作，在添加古诗标签边界类中添加想要添加的古诗标签，然后边界类将添加请求发送给标签控制类，然后由古诗标签控制类调用创建古诗标签实体类操作数据库实现用例。增加古诗标签顺序图如图3-13所示。

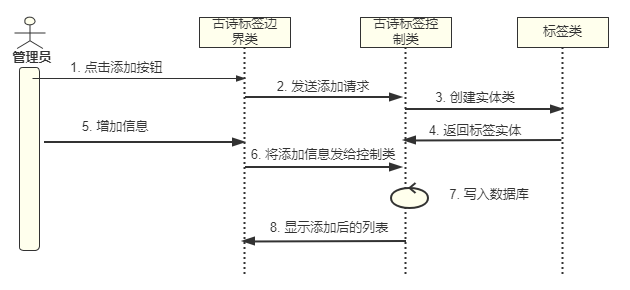


图3-13增加古诗标签顺序图

1. **修改古诗标签顺序图**

修改古诗标签顺序图主要展示的是修改古诗标签用例。在主边界类使用者请求修改具体的古诗标签信息，并判断是否为父标签，首先古诗标签边界类会通过调用古诗标签实体类从数据库中返回指定标签信息，使用者在修改古诗标签信息后，标签控制类调用古诗标签实体类修改数据库最终实现用例。修改古诗标签信息顺序图如图3-14所示。

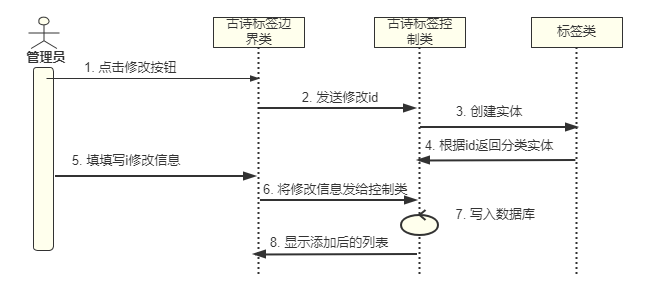


图3-14修改古诗标签顺序图

1. **古诗生成顺序图**

古诗生成位顺序图主要展示的是生成古诗、生成藏头诗用例。使用者在主边界类中请求生成古诗并给出生成古诗的固定条件，古诗生成主边界类通过控制类调用python边界类请求数据，然后返回数据给控制类，最后调用古诗生成实体类最终实现古诗生成用例。古诗生成顺序图如图3-15所示。

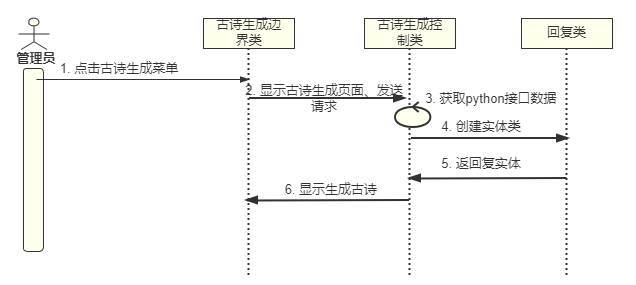


图3-15古诗生成顺序图

1. **用户管理列表显示顺序图**

用户信息列表顺序图主要展示的是显示用户信息列表用例。使用者通过点击左侧页面选项卡用户管理，系统默认发送显示数据库内所有用户信息请求，然后通过用户管理边界类将用户请求发送给控制类，最后由控制类调用用户实体类查询数据库返回用户信息实现用例。用户信息列表顺序图如图3-16所示。

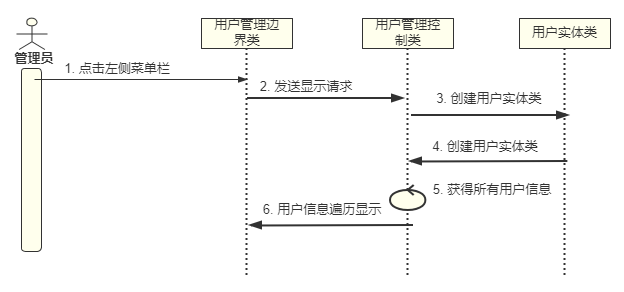


图3-16用户信息列表显示顺序图

1. **修改用户信息顺序图**

修改用户信息序列图主要展示了用于修改用户信息的用例。系统使用者需要在用户管理边界类别中发出修改某一用户信息的请求，在修改用户信息的边界类中通过调用用户信息控制类，最后控制类调用用户实体类进行用户数据查询并返回，请求发出者在修改完用户信息后，用户控制类调用创建用户实体类修改数据库，最终实现用户信息修改用例。修改用户信息顺序图如图3-17所示。

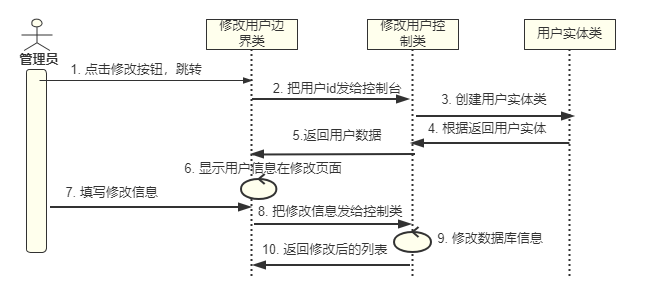


图3-17修改用户信息顺序图

1. **添加用户顺序图**

添加用户顺序图主要展示的是添加用户用例。使用者通过页面发送添加用户信息的请求给边界类，并在添加用户边界类中进行正确的用户信息的填写，然后将边界类请求发送给控制类，由控制类调用用户实体类进行用户信息数据的保存实现实例。添加用户顺序图如图3-18所示。

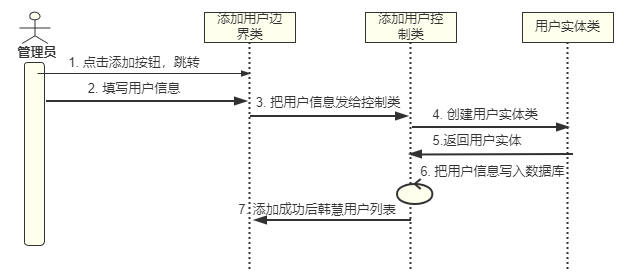


图3-18添加用户顺序图

## 3.4 算法分析

### 3.4.1 RNN算法

RNN是一类处理序列数据的神经网络，而这里说的序列数据是能够代表某种事物一些特点的数据。在自然语言处理（NLP），语音图像识别等多个领域均有非常广泛的应用。RNN网络和其他网络相比最大的不同就在于RNN能够实现某种“记忆功能”，是进行时间序列分析时最好的选择。就好比人类能够凭借自己对以前事物的记忆从而更加清楚地认识这个世界一样。RNN也实现了类似于人脑的这一机制，对训练所处理过的数据信息保留有一定的记忆。

**1.RNN模型结构**

一个简单的RNN网络包含一个输入x，一个输出o和一个神经网络单元A。RNN神经网络单元A不仅仅与输入和输出存在一定的联系，其与自身也存在一个循环回路。这种网络结构在一定程度上揭示了RNN的实质：上一个时刻的网络状态信息将会作用于下一个时刻的网络状态。RNN网络按照时间序列展开成如图3-19所示：

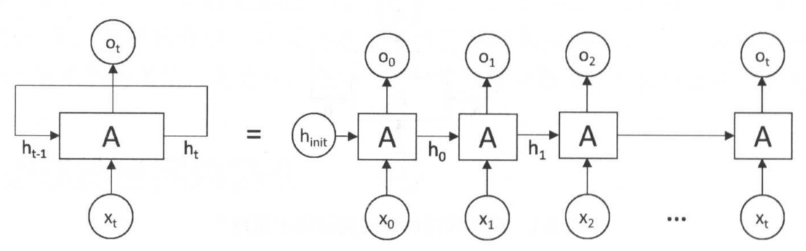


图3-19 RNN展开图

图3.19中等号左边是RNN模型没有按时间展开的部分，右边是RNN按时间序列展开式的部分，其中：代表在序列在时间t时训练样本的输入数据。同样的，和代表在序列t−1和t+1时训练样本的数据输入。代表在序列t时模型的隐藏状态，是由输入和上一层隐藏状态共同决定的，代表在序列t时模型的输出，由模型当前的隐藏状态和决定，A代表RNN模型。其数学定义为：输入为对应的隐状态为，则输出就为。计算公式如3-20：

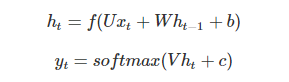


图3-20 RNN运算过程

其中，U,W,V,b,c均为权值参数，而表示激活函数，一般为tanh函数。

### 3.4.2 LSTM算法

由于RNN网络在求解长序列训练过程中，激活函数复杂且容易出现梯度消失或者梯度爆炸的问题，因此得到了RNN的一种特殊类型LSTM（Long Short-Term Memory），它通过门控制的方式将加法运算带入网络中，在一定程度上避免了RNN的梯度消失，因此在神经网络模型中得到了广泛的应用。LSTM结构图如3-21

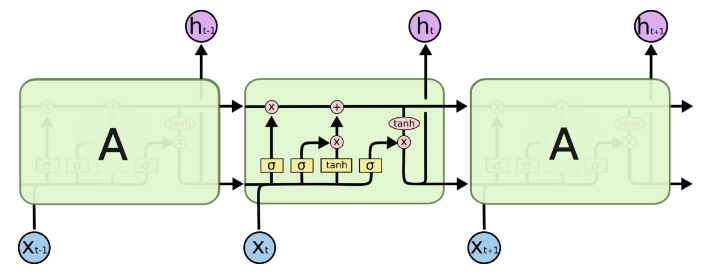


图3-21 LSTM结构图

1. **细胞状态**

LSTM算法最重要的部分就是细胞状态的更新，如图3-22中上面的长横线。在每个训练的t时刻除了拥有和RNN一样的隐藏状态，还拥有另一个不同的隐藏状态，这个隐藏状态我们叫它细胞状态(Cell State)也叫门结构，即表示为Ct。而且也正是通过这种能够控制数据信息是否通过的结构，来达到控制信息删除或添加到下一时刻的目的，能够选择性地决定信息是否通过，使LSTM算法能够很好的选择对训练更为有用的数据。

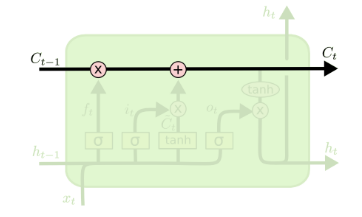


图3-22 细胞状态图

**2. 遗忘门**

遗忘门（forget gate）是一个决定什么信息可以通过细胞状态的设置，在LSTM中会进行计算得出一定的概率，来控制是否让上一刻的Ct通过还是部分通过。遗忘门子结构如图3-23所示：

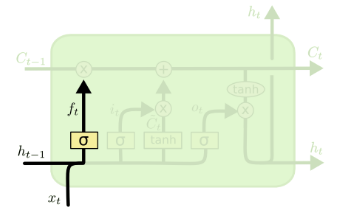


图3-23 遗忘门

图3-22中会读取有上一层序列的隐藏状态和，通过一个激活函sigmoid，得到遗忘门的输出，其中输出的值为0到1之间，表示每个部分有多少变量能够通过，‘0’代表不允许任何变量通过，‘1’代表全部通过。由于sigmoid的输出在[0,1]之间，因此这里的输出代表了遗忘上一层隐藏细胞状态的概率。用数学表达式即为：

其中Wf,Uf,bf为关系系数，和RNN类似。σ为sigmoid的激活函数。

**3. 输入门**

输入门（input gate）主要是用来处理当前时刻计算后的输入数据和tanh函数的值是否要更新到细胞状态中。输入门子结构如下图3-24：

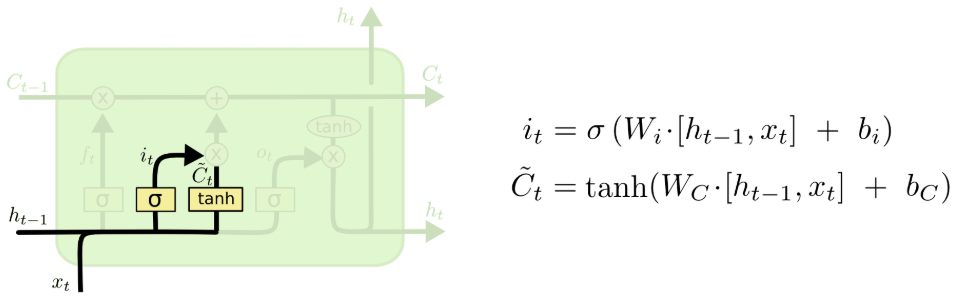


图3-24 输入门

从上图中可以看出输入门的结构主要包含两个部分，首先就是σ(sigmoid激活函数)层来决定输入数据和上层隐藏状态中有哪些值需要我们进行更新到细胞中，计算结果输出为it。第二部分为tanh函数层，创建输出候选值将会被加入到状态中，这两部分产生的值进行结合，来对状态进行更新。

状态更新时，先将旧的cell state乘以来遗忘掉不需要或者与内容关系不大的信息，然后再与\*相加，得到了候选值，结合输入输出门进行更新cell状态，去掉不需要的信息，添加新信息。如图3-25

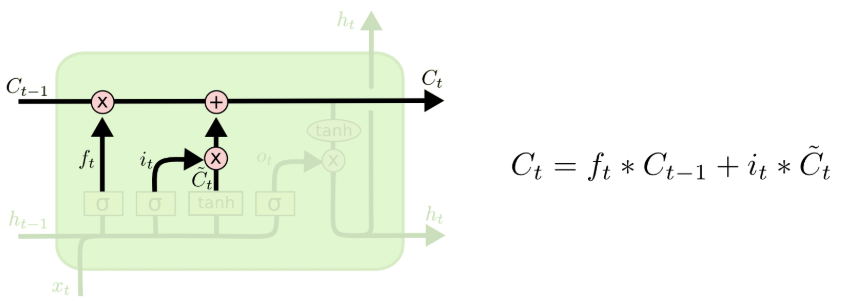


图3-25 细胞状态更新

**4.输出门**

输出门（output gate）决定模型的输出，而这个输出的结果是在Ct的基础上得出的，由图可以看出隐藏状态的输出主要有两部分组成，首先就是当前时刻序列数据和上一层得出的隐藏状态，通过σ(sigmoid激活函数)计算得到的输出值，另外一部分是由细胞更新后的Ct和tanh函数计算得出，两部分共同作用最终得出输出。如图3-26输出门：

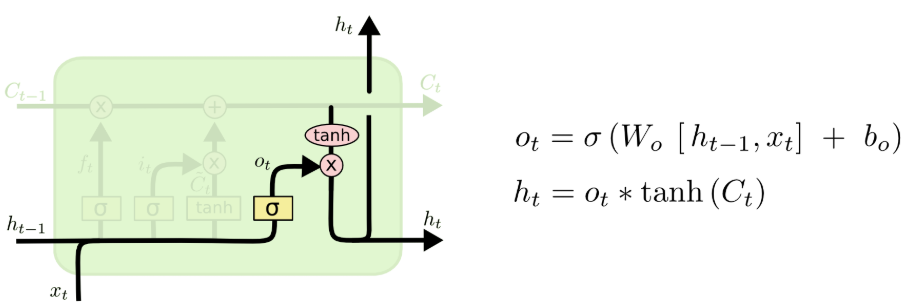


图3-26 输出门图

### 3.4.3 Seq2Seq

Seq2Seq (序列到序列模型)，主要是完成序列到序列的转换，也是一种循环神经网络，包括编码器 (**Encoder**) 和解码器 (**Decoder**) 两部分，是自然语言处理中的一种重要模型，下面只介绍其基本的模型。基本思想就是利用两个RNN，一个RNN作为encoder，另一个RNN作decoder，这种encoder-decoder模型对输入和输出序列长度没有太多要求。**Encoder主要功能是将训练输入的数据通过一定的方式压缩成固定长度的向量，通常这种方式是使用多个LSTM单元进行处理，即将输入的数据编码成一个上下文的语义向量C，这就是编码。**如图3-27 Encoder。

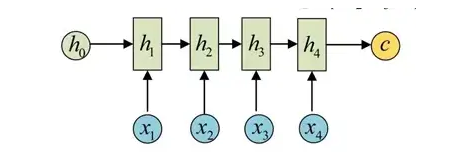


图3-27 Encoder

而**decoder主要功能是根据编码生成的语义向量C，从而得出指定的序列**，这个由语义向量到输出序列得过程也是解码过程，如下图3-28Decoder，最简单的一种方式就是在拿到语义变量C后，直接使用C作为初始值输入到解码序列中，然后对其进行解码，从而得到输出序列。

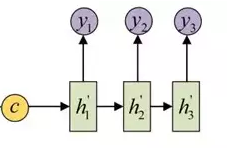


图3-28 Decoder

# 第4章 系统设计

## 4.1 架构设计

MVC的结构设计是软件开发中最常见的一种设计典范，也是使用最广的结构，一般为表示层、业务逻辑层、数据访问层以及数据库。如图4-1所示。



图4-1 MVC设计模式

本系统在MVC设计模式的基础上，采用前后端分离的方式进行开发，客户端通过api接口与服务端进行数据的交互，这样更加有利于系统的开发和维护。前端页面主要采用Vue框架进行实现，通过Axios从服务端接口获取数据，并将得到的数据进行渲染，展示在页面上，服务端采用SpringBooot框架进行编写，主要负责向客户端提供接口服务，并与数据库实现数据的存储，主要分为Controller层接收客户端的数据请求，然后调用相应的Service层完成请求中具体业务逻辑的处理，而Service层调用Dao层通过实体类进行数据库具体的访问操作。其中古诗生成采用python进行算法的编写和古诗生成模型的训练，利用flask框架搭建简易的服务接口，用来提供数据的获取。详情见图4-2所示。

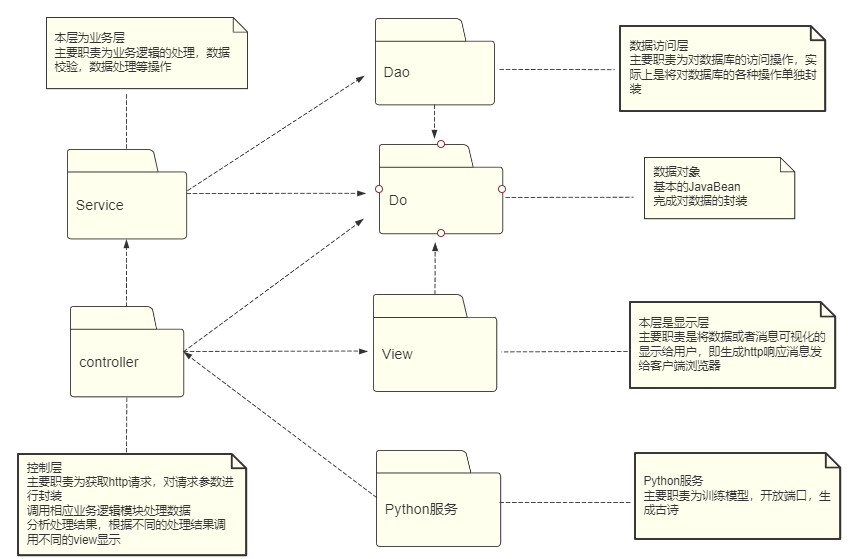


图4-2 软件架构图

## 4.2 设计类

基本的实体类已通过顺序图体现出来，顺序图中的消息实际上对应于此类的方法并通过不断地抽取共同点获得设计类图。如图4-3所示。图中只是列出了设计实体类以及实体类相互之间的关系。

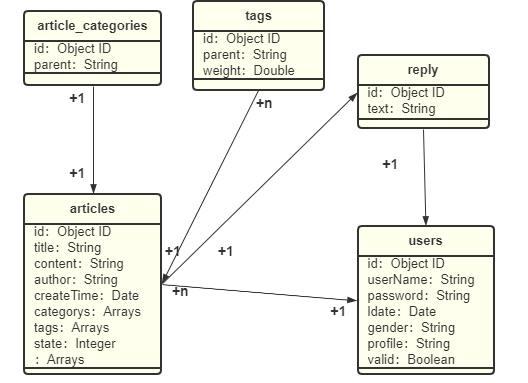


图4-3 设计类图

Article(古诗）实体类中主要有id (编号)、title(标题)、content(内容)、author(作者)、createTime(创建时间)、categories(类别)、tags(标签)、state (状态)等属性。

Users(用户)实体类中主要有id(用户id)、userName(登录名)、password(登录密码)、gender(性别)、profile(用户描述)等属性。

tags(标签）实体类中主要有id (编号)、parent(root)、weight(权重)等属性

## 4.3 数据库设计

数据库是系统的基本，无论是数据的处理还是数据的显示，都需要在完善的数据库基础之上进行的，数据库的创建需要完成从概念模型的设计到关系模型之间的映射，以及从关系模型到实际的物理模型的映射，最后完成创建。

### 4.3.1 概念模型

概念模式是对系统信息的一个抽象的建模，并提取出有关的实体。如图4-4所示的系统E-R图。

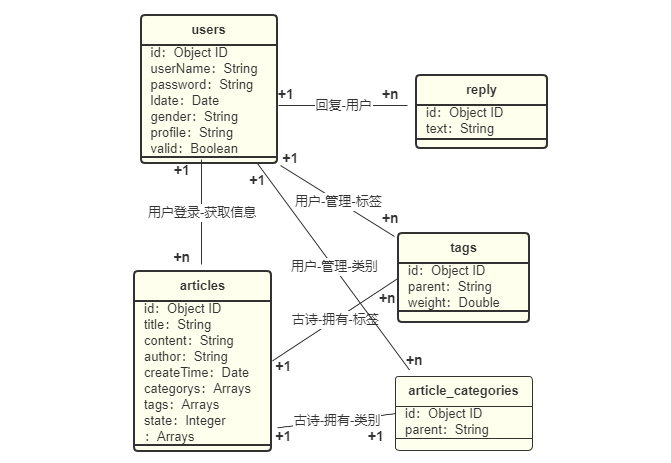


图4-4概念模型

本系统拥有users（用户）、articles（古诗）、tags（标签）、article\_categories（类别）、reply（回复）5个实体表。一个用户可以查看所有古诗信息，能够对多个古诗进行操作，所以是一对多的关系。

### 4.3.2 关系模式

1. Articles(id，title，content，author，createTime，tags，categorys，state)；
2. Users(id，userName，password，date，gender，profile，valid);
3. Reply(id，text);
4. Tags(id，parent，weight);
5. Article\_categories(id，parent);

### 4.3.3 物理模型

本系统采用MongoDB数据库管理系统。articles表用于存放古诗信息，其中标签和分类与古诗分类和古诗标签表进行关联。如表4-1所示。

表4-1 articles表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 列名 | 字段说明 | 数据类型 | 主键 | 允许空 |
| 1 | id | Id | ObjectID | 是 | 否 |
| 2 | title | 标题 | String | 否 | 是 |
| 3 | content | 内容 | String | 否 | 是 |
| 4 | author | 作者 | String | 否 | 是 |
| 5 | createTime | 创建日期 | Date | 否 | 是 |
| 6 | tags | 标签 | Arrays | 否 | 是 |
| 7 | state | 状态 | Inreger | 否 | 是 |
| 8 | categorys | 类别 | Arrays | 否 | 是 |

users表用于存放用户信息。如表4-2所示。

表4-2 Users表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 列名 | 字段说明 | 数据类型 | 主键 | 允许空 |
| 1 | id | id | ObjectID | 是 | 否 |
| 2 | username | 登录账号 | String | 否 | 是 |
| 3 | password | 登录密码 | String | 否 | 是 |
| 4 | date | 创建日期 | Date | 否 | 是 |
| 5 | profile | 用户描述 | String | 否 | 是 |
| 6 | valid | 是否有效 | Boolean | 否 | 否 |

tags表用于存放古诗标签信息，其中子标签继承父类标签，父类删除，子类标签也将被删除。如表4-3所示。

表4-3 tags表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 列名 | 字段说明 | 数据类型 | 主键 | 允许空 |
| 1 | id | id | ObjectID | 是 | 否 |
| 2 | parent | 父属性 | String | 否 | 是 |
| 3 | weight | 权重 | Double | 否 | 是 |

article\_categories表用于存放古诗分类信息，其中子分类继承父类分类，父类删除，子类也将被删除。如表4-4所示。

表4-4 article\_categories表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 列名 | 字段说明 | 数据类型 | 主键 | 允许空 |
| 1 | id | id | ObjectID | 是 | 否 |
| 2 | parent | 父属性 | String | 否 | 是 |

## 4.4 算法设计

古诗管理与生成系统的重要功能就是古诗生成，构建古诗生成的模型是关键中关键，构建古诗生成模型主要分为一下几步，如图4-5算法逻辑

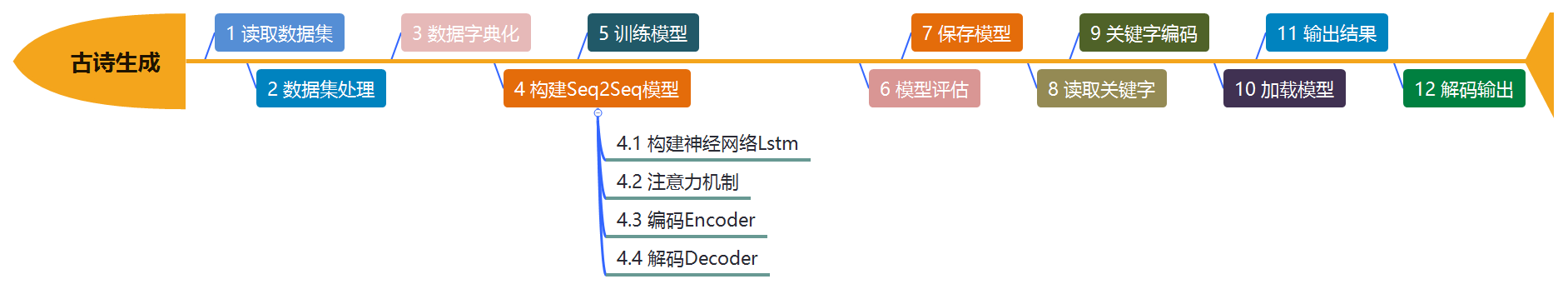


图4-5 算法逻辑

数据集处理包含数据的读取、数据的处理、数据字典化等操作。对数据处理的大致流程首先是根据文本的格式内容，读取文本数据，按行切分，根据冒号分离出标题和正文，这里只使用正文，构成古诗列表，考虑到训练模型的大小，在处理数据时需要对过长的古诗分割进行保留，过短的数据进行数据填充，统计保留古诗中的词频构建词汇表，加入开始和结束标志。如图4-6

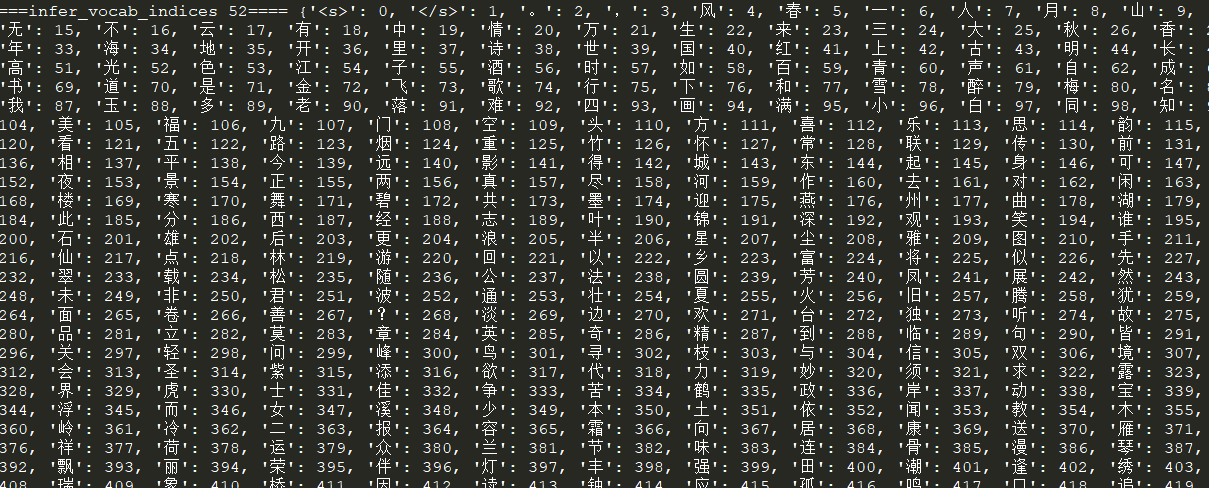


图4-6 数据字典

模型训练主要是使用Seq2Seq算法对文本进行计算、训练最终将训练的模型进行保存，并提供使用。具体算法结构和第三章算法分析基本一致，不同的是为了提高模型的输出效果，加入了注意力机制（Attention Mechanism），在训练时为了生成古诗的内容效果与输入的关键字内容符合程度更高。

用户在输入古诗生成的关键字后，系统读取到关键字然后对关键字进行编码，转化为数据向量，通过加载训练好的模型传并入模型参数，系统会根据关键字自动生成对应的数据向量，通过解码生成一首古诗，输出到页面中。

# 第5章 系统实现

## 5.1 古诗生成

古诗生成的主要负责自动生成一首古诗。主要分为关键字生成古诗和生成藏头诗两个模块，在古诗生成页面中默认情况下是根据用户输入的关键词点击发送后，自动生成一首古诗，随后以聊天框的形式显示在页面中，如图5-1所示。

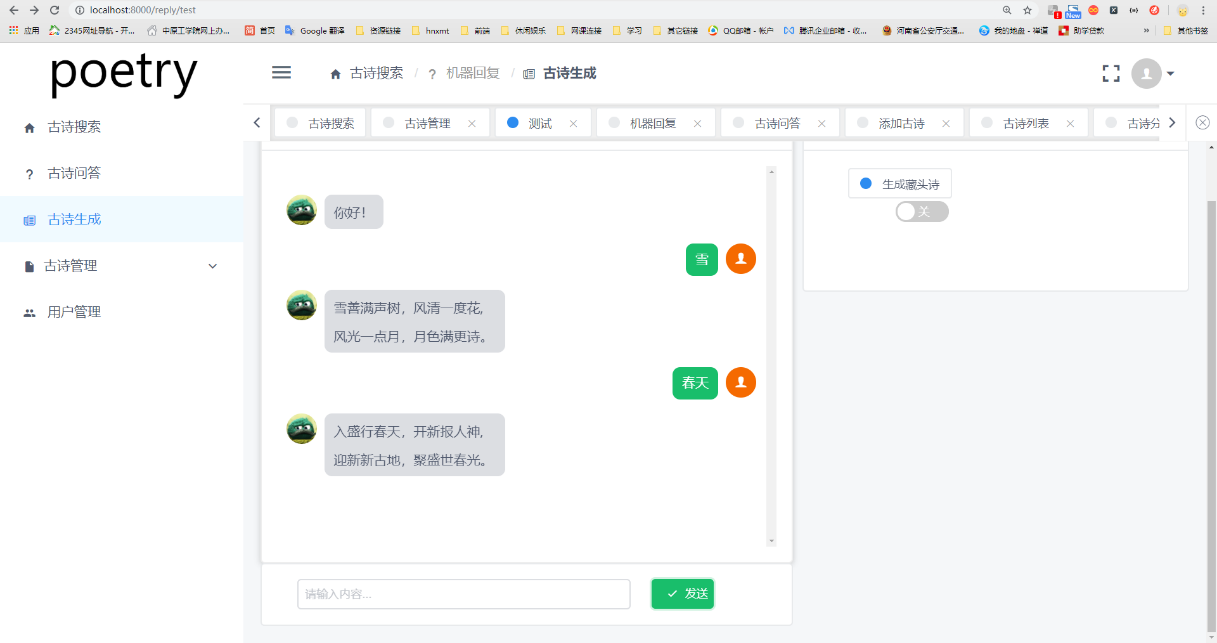


图5-1 古诗生成页面

当用户打开右侧生成藏头诗开关时，在输入指定的关键词后点击发送，系统自动生成一首藏头诗以对话的方式显示在页面中，如图5-2所示。

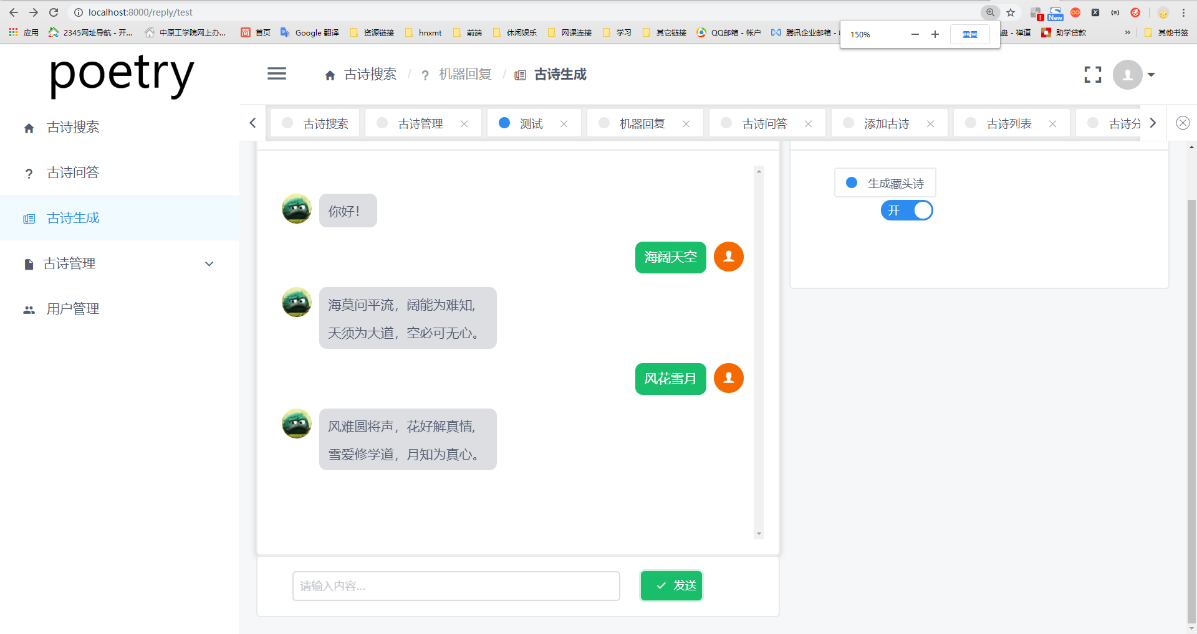


图5-2 藏头古诗页面

## 5.2 古诗管理

古诗管理主要负责维护古诗信息的各项数据，并能够对古诗进行一系列的操作。主要分为添加古诗、古诗列表、古诗标签、古诗分类四个模块。

在古诗管理的子模块中，古诗列表显示数据库中所有能够进行操作的古诗信息，主要有删除、修改、刷新古诗列表等功能，在右侧的古诗分类选项卡下，可通

过选择对应的分类进行古诗的筛选。点击删除按钮会，会提示是否删除成功，这里不进行页面的展示。古诗列表如图5-3所示

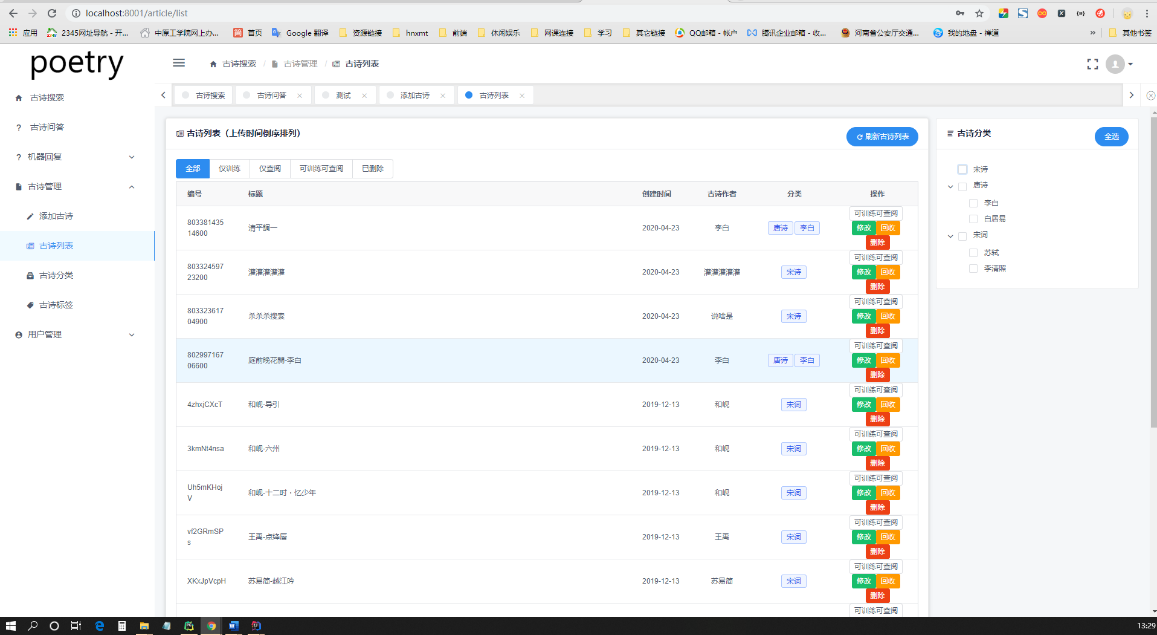


图5-3 古诗列表

在古诗列表页面中用户单击修改古诗按钮，系统会自动弹出古诗修改框，在修改弹窗中用户可以对当前古诗得一系列信息进行修改，也可以点击关闭弹窗取消修改操作，在信息修改完成后，点击发布按钮，修改古诗信息弹窗自动关闭，古诗列表发生变化，如图5-4所示。

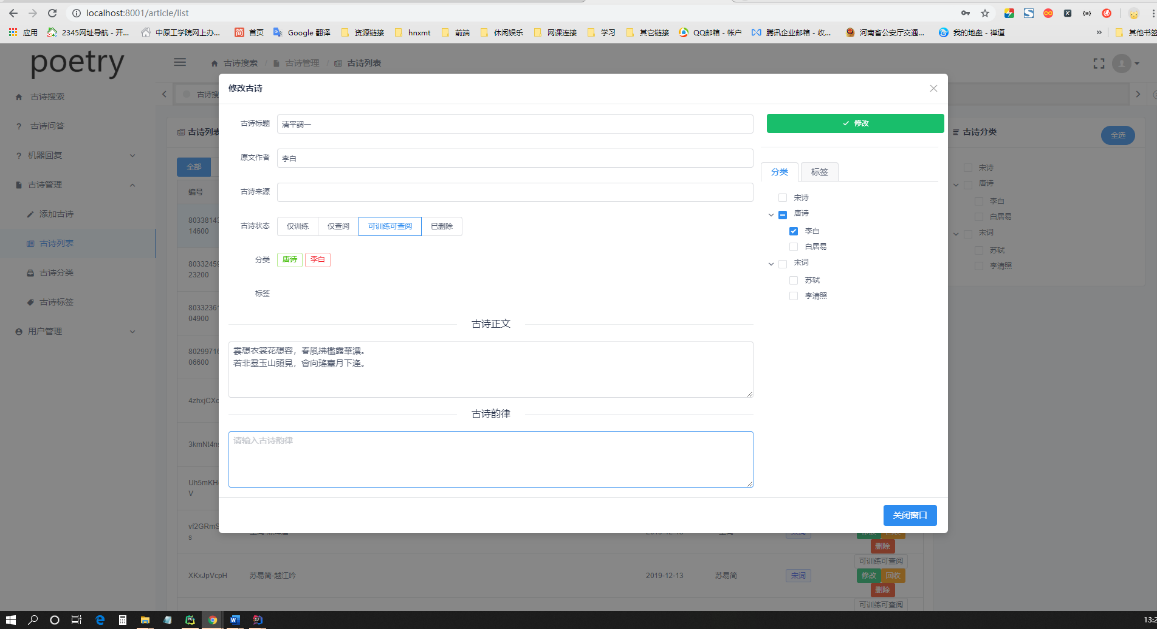


图5-4 古诗修改

在添加古诗页面中，使用者根据界面提示填写古诗的基本信息，按照页面指示进行操作，其中古诗标签和古诗分类是手动选择，其数据是从数据库中查询而来，用户只能选择数据中已经存在的分类和标签，否则需要提前在古诗分类和古诗标签中添加，在填写信息准确后，点击发布按钮能够添加一首古诗，如图5-5所示。

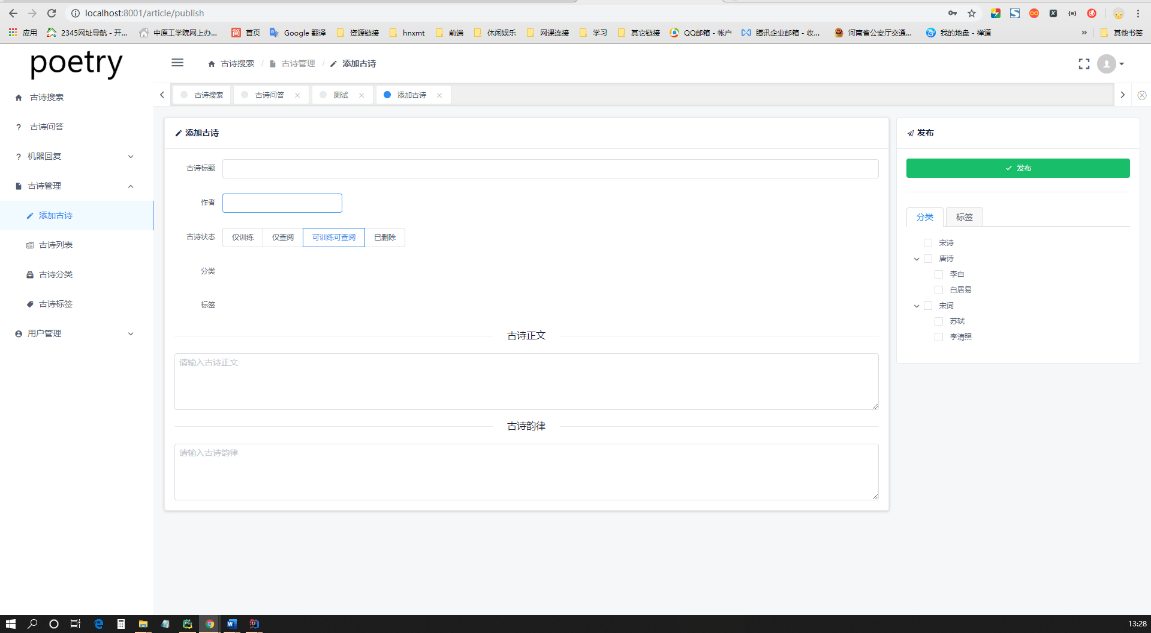


图5-5 添加古诗页面页面

在古诗分类中，页面默认显示数据库中所有分类信息，以树结构呈现。用户在页面中可以修改分类、删除分类、添加分类。其中分类父类不能修改或者删除。用户需按照古诗分类管理说明进行操作如图5-6所示

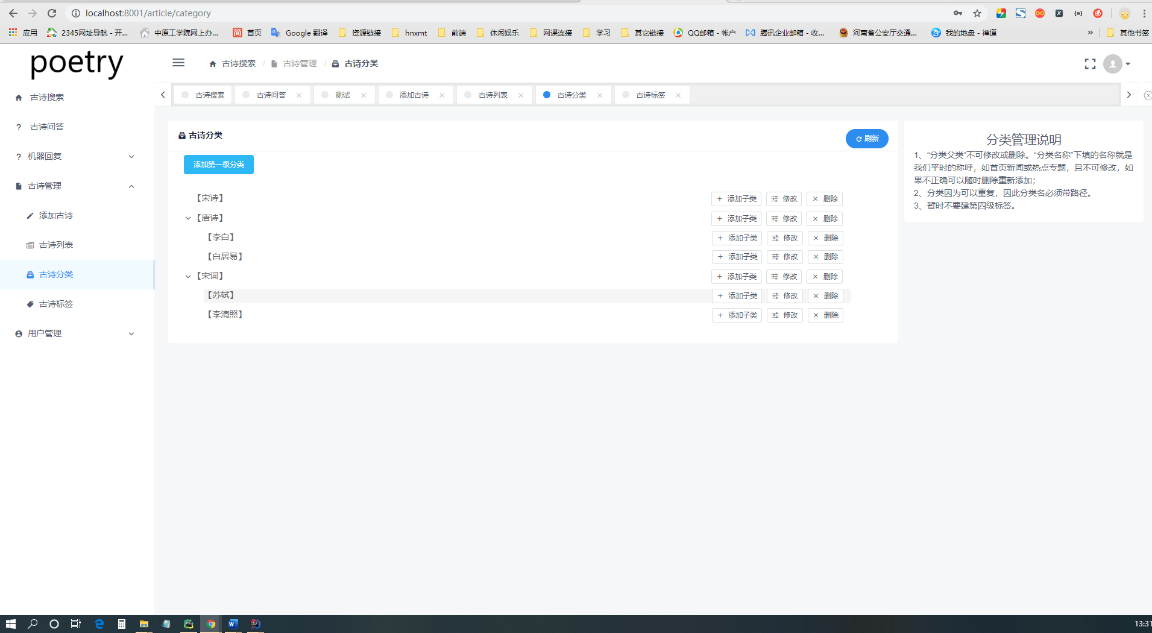


图5-6 古诗分类

在古诗标签中，页面默认显示数据库中所有标签信息能够刷新标签，以树结构呈现。用户在页面中可以修改标签、删除标签、添加标签，点击对应的按钮会弹出相应的对话框从而进行操作。其中最低级标签不支持修改，左侧显示古诗标签的管理说明，提供用户按照固定格式进行操作。如图5-7所示

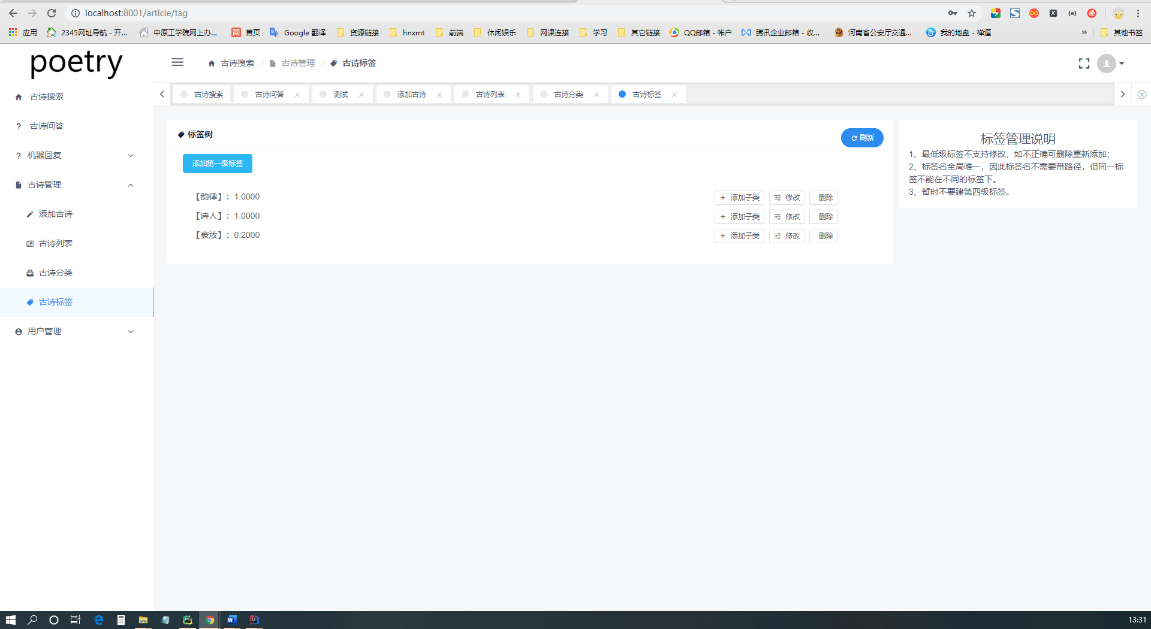


图5-7古诗标签

## 5.3 古诗搜索页面

古诗搜索页面也是系统的主页面，默认显示所有古诗的信息，使用者可以在左侧控制台中输入要查绚的关键字，例如古诗标题、作者等信息，在点击查找按钮后显示根据关键字查找的古诗。如图5-8所示。

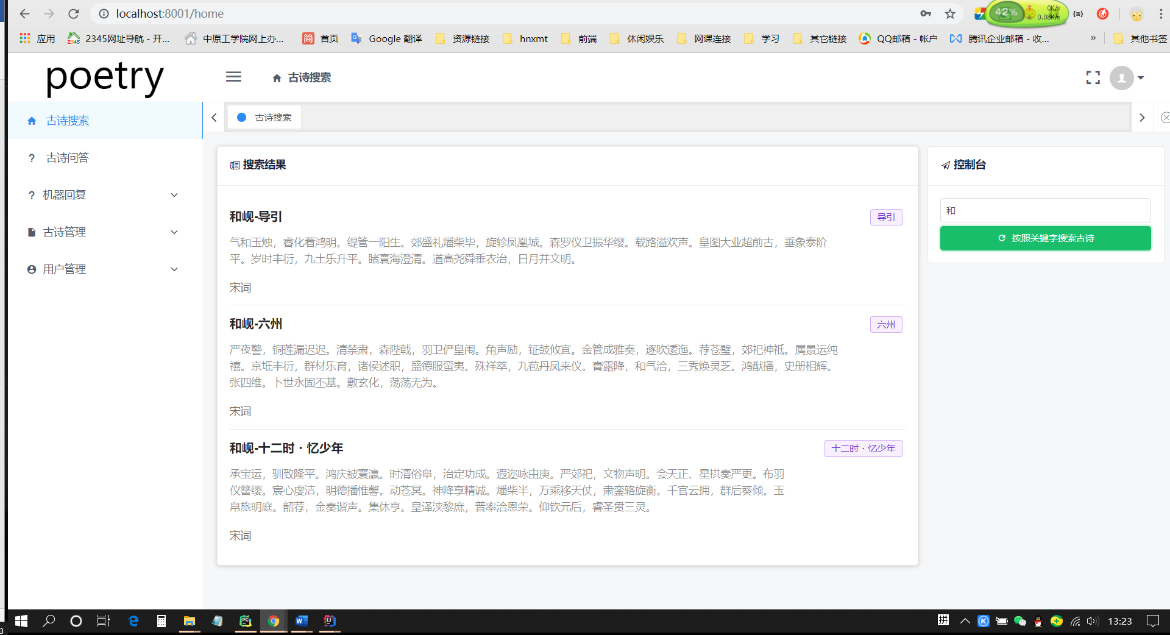


图5-8学生实训信息维护

## 5.4 用户管理

用户管理的主要功能是对用户信息进行管理。主要功能有显示所有用户信息、添加用户、对用户进行编辑、查找用户等。系统使用者点击用户管理，默认显示所有用户的基本信息和操作，同时能够通过用户id、登录账号、性别进行条件查询，从而某个用户，如图5-9所示。

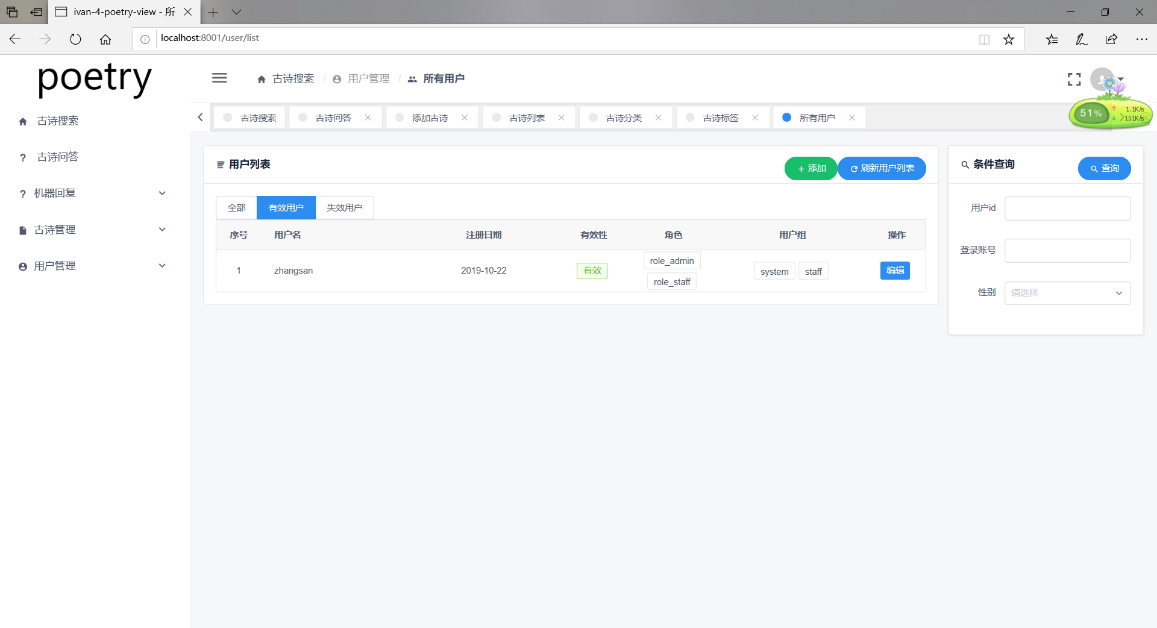


图5-9用户列表显示

当系统使用者在页面上点击添加按钮时，系统会弹出添加登录用户的弹框，从使用者从而进行登录用户账号和密码的添加，如图5-10所示。同时使用者点击编辑按钮后，系统弹出编辑弹框，对用户一些基本的信息进行修改，这里不进行展示。详情以系统页面为主。

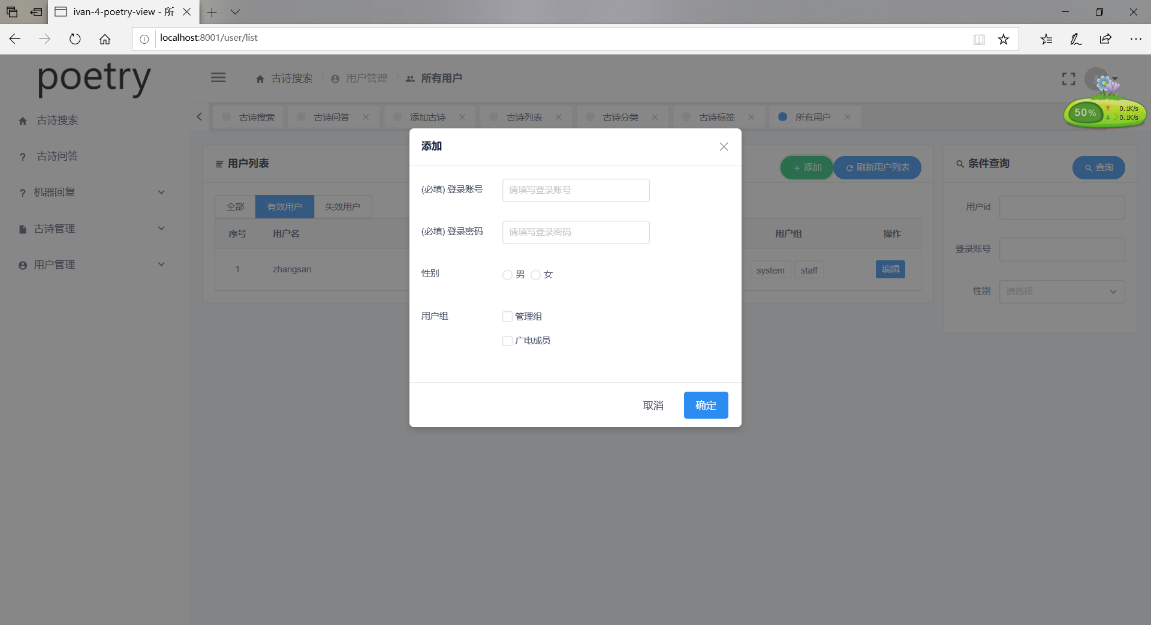


图5-10用户添加

## 5.5 登陆页面

古诗管理与生成系统成功运行之后，使用者输入指定网址进入登录页面，如图5-10。使用者使用正确的登录账号和登录密码进行登录操作，如果输入的登录名或者登录密码不正确，系统将会提示错误信息，如果用户名和密码正确会进入系统主页，如图5-11。同时系统主页也是系统的古诗搜索功能。



图5-11 系统登录页面

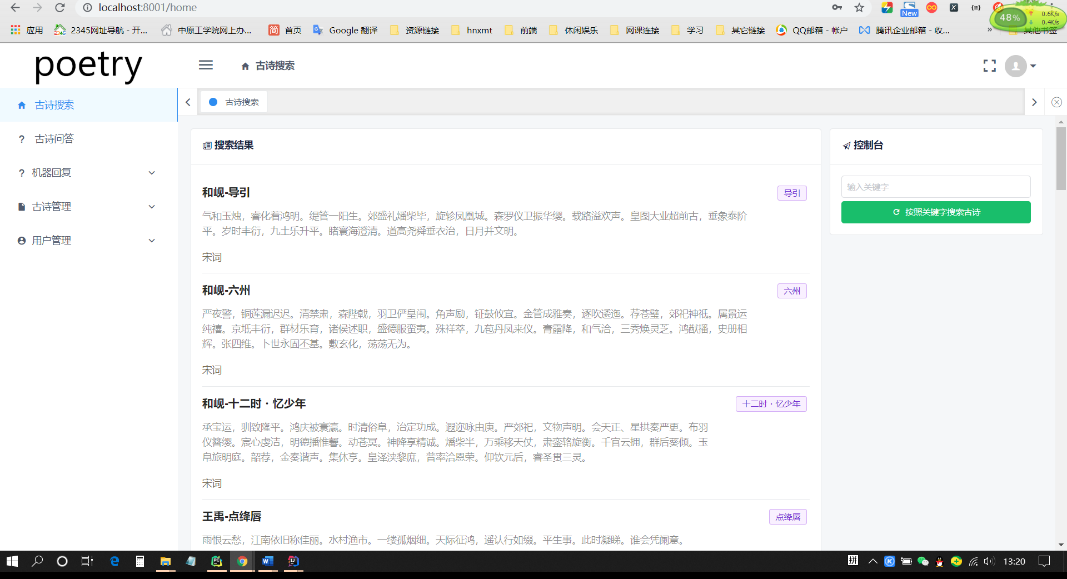


图5-12 系统主页面

# 第6章 系统测试

## 6.1 古诗搜索测试

古诗信息列表显示测试用例，主要用来测试古诗搜索页面是否正常显示，古诗信息是否正常显示，古诗标签、古诗分类是否能够显示在页面上，关键字搜索功能能否实现。如表6-1所示。

表6-1 古诗搜索测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1 | | 编写者 | 胡帅 | |
| 测试项 | 古诗信息列表显示、搜索功能 | | | | |
| 前提条件 | 使用用户名、密码登录系统 | | | | |
| 操作步骤 | 默认显示所有古诗信息列表 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 1.页面中显示有30条古诗信息 | 无 | 1.古诗信息列表按照接口传输数据全部显示。  2.显示内容包含要显示的所有信息。 | | | 实际结果同预期结果相同 |
| 2.在搜索框中输入内容 | 点击“搜索”按钮 | 1.根据输入的内容关键字查找对应的古诗数据。  2.结果页面中显示查询的结果，覆盖所有列表。  3.数据返回数目不定，根据返回结果变化。  4.输入为空的情况下显示所有古诗列表 | | | 实际结果同预期结果相同 |

## 6.2 古诗管理

### **6.2.1** 古诗操作列表显示

古诗列表的测试用例，主要用来测试古诗列表页面是否正常显示，古诗信息是否正常显示，古诗修改、古诗删除、古诗回收等功能是否正常，相应的弹框是否正常显示。如表6-2所示。

表6-2 古诗操作列表测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-2-1 | | 编写者 | 胡帅 | |
| 测试项 | 古诗操作列表显示，页面功能是否正常运行 | | | | |
| 前提条件 | 使用用户名、密码登录系统，点击古诗管理🡪古诗列表 | | | | |
| 操作步骤 | 点击古诗列表显示所有信息 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 1.在页面中默认显示10条数据 | 点击古诗列表菜单栏 | 1.古诗列表按照每页20条数据显示。  2.提供分页功能，点击分页相对应的功能，页面有不同的变化。 | | | 实际结果与预期相同 |
| 点击“删除”按钮 | 1.页面以弹框的形式二次确认是否确定删除数据  2. 点击取消按钮系统返回古诗列表界面，点击确定删除此条信息，系统提示删除成功  3删除后列表是否刷新  4.数据库中是否删除 | | | 删除功能正常，数据库中数据正常删除，页面提示删除成功，列表数据更新 |

### 6.2.2 添加古诗

添加古诗测试用例，主要用来测试添加页面是否正常显示，添加后的古诗是否保存到数据库，添加成功后能否自动跳转到古诗列表页面。如表6-2所示。

表6-3 添加古诗测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | FUN-2-2 | 编写者 | 胡帅 | |
| 测试项 | 添加古诗信息 | | | |
| 前提条件 | 使用用户名、密码登录系统，点击古诗管理🡪添加古诗 | | | |
| 操作步骤 | 点击添加古诗，进入添加页面 | | | |
|  | 输入数据/动作 | 预期结果 | | 实际结果 |
| 1.根目录下添加古诗信息，输入正常数据 | 按照页面提示进行填写数据信息。例如：  古诗标题：静夜思  作者：李白  古诗状态：可查询  分类：唐诗  标签：浪漫  内容：自己随意输入，满足长度限制即可。  韵律：可选  点击“发布”按钮 | 1.填写信息中的分类、标签是从数据库读取的，显示数据是否与数据库保持一致。  2.选择分类和标签时，添加古诗中的分类和标签会自动添加。  2.数据填写正确的情况下，点击发布按钮数据会保存到数据库，页面会提示保存成功的消息。 | | 分类和标签显示正常，与数据库保持一致，选择分类和标签时会自动添加到古诗信息中，数据保存成功 |
| 2. 在根目录下添加古诗信息，输入异常数据 | 输入数据时，非空选项不填写数据，例如：标题、作者或者内容其中一个或多个不填写 | 1.非空选项不填写数据，点击发布按钮时，按钮无效，并且以红色字体提示必填数据。  2.非空信息填写完成后，发布按钮功能正常，数据保存成功 | | 与预期结果相同 |

### 6.2.3 修改古诗信息

修改古诗信息测试用例，主要用来测试修改古诗弹窗页面是否正常，修改过的古诗信息能否成功保存。如表6-4所示。

表6-4 修改古诗测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | FUN-2-3 | 编写者 | 胡帅 | |
| 测试项 | 修改古诗信息 | | | |
| 前提条件 | 以用户名、密码登录系统，点击古诗管理🡪古诗列表信息显示 | | | |
| 操作步骤 | 点击指定的古诗的修改按钮，页面弹出修改页面弹框 | | | |
|  | 输入数据/动作 | 预期结果 | | 实际结果 |
| 1.不做任何动作 | 直接点击修改按钮 | 提示：修改成功，点击发布后，返回古诗信息列表显示页面的第1页？列表的显示顺序有变化吗？ | | 页面修改弹框自动关闭并提示修改成功，同时返回到古诗列表页面，数据刷新。 |
| 2.在修改古诗页面输入异常数据 | 输入数据时，非空选项不填写数据，例如：标题、作者或者内容其中一个或多个修改为空 | 1.非空选项修改时数据为空，点击发布按钮时，按钮无效，并且以红色字体提示必填数据。  2.信息修改完成后，发布按钮功能正常，数据保存成功 | | 与预期结果相同 |

### 6.2.4 古诗分类

古诗分类测试用例，主要用来测试古诗分类页面显示是否正常，用户使用添加分类、修改分类功能是否正常。如表6-5所示。

表6-5 古诗分类测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-2-4 | | 编写者 | 胡帅 | |
| 测试项 | 古诗分类页面功能 | | | | |
| 前提条件 | 以用户名、密码登录系统，点击古诗管理🡪古诗分类 | | | | |
| 操作步骤 | 点击指定功能按钮，系统做出相应反应 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 1.根目录下无数据 | 无 | 古诗分类列表为空 | | | 与预期结果相同 |
| 点击添加按钮 | 页面弹出添加对话框，列表增加一条数据 | | | 与预期结果相同 |
| 2.根目录下有古诗分类信息 | 无 | 1.显示数据库中说有的分类数据 | | | 实际结果同预期结果相同 |
| 点击增加按钮 | 页面弹出添加对话框，列表增加一条数据 | | | 实际结果同预期结果相同 |
| 点击删除按钮 | 页面提示是否确认删除数据信息，确认则删除数据，否则关闭对话框 | | | 实际结果同预期结果相同 |
| 点击修改按钮 | 页面弹出到修改对话框 | | | 页面弹出修改对话框 |

### 6.2.5 古诗标签

古诗标签测试用例，主要用来测试古诗标签页面显示是否正常，用户使用添加标签、修改标签功能是否正常。如表6-6所示。

表6-6 古诗标签测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-2-5 | | 编写者 | 胡帅 | |
| 测试项 | 古诗标签页面功能 | | | | |
| 前提条件 | 以用户名、密码登录系统，点击古诗管理🡪古诗标签 | | | | |
| 操作步骤 | 点击指定功能按钮，系统做出相应反应 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 1.根目录下无数据 | 无 | 古诗标签列表为空 | | | 与预期结果相同 |
| 点击添加按钮 | 页面弹出添加对话框，列表增加一条数据 | | | 与预期结果相同 |
| 2.根目录下有古诗标签信息 | 无 | 1.显示数据库中所有的标签 | | | 实际结果同预期结果相同 |
| 点击增加按钮 | 页面弹出添加对话框，列表增加一条数据 | | | 实际结果同预期结果相同 |
| 点击删除按钮 | 页面提示是否确认删除数据信息，确认则删除数据，否则关闭对话框 | | | 实际结果同预期结果相同 |
| 点击修改按钮 | 页面弹出到修改对话框 | | | 页面弹出修改对话框 |

## 6.3 古诗生成

古诗生成测试用例，主要用来测试古诗生成页面是否正常显示，生成古诗功能是否正常。如表6-7所示。

表6-7 古诗生成测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-3 | | 编写者 | 胡帅 | |
| 测试项 | 古诗生成页面功能 | | | | |
| 前提条件 | 以用户名、密码登录系统，点击古诗生成 | | | | |
| 操作步骤 | 点击指定功能按钮，系统做出相应反应 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 1.点击古诗生成页面显示聊天界面 | 无 | 显示聊天界面，系统自动发出一条语句。 | | | 实际结果同预期结果相同 |
| 2.点击古诗生成页面 | 默认情况下使用者输入关键词或字，点击发送按钮 | 1.用户输入的内容显示在聊天页面中，用户头像和系统头像正常显示  2.用户在输入框中输入关关键词，页面自动回复一首古诗。  3.回复内容合理的显示在聊天界面中 | | | 用户和系统头像显示正常，回复古诗正常，显示内容正常 |
| 3.输入异常数据 | 输入关键词太长，输入内容有无法识别的符号或者输入为空 | 1.提示输入关键词不符合要求  2.输入为空时发送按钮无效 | | | 实际结果同预期结果相同 |
| 4.点击古诗生成页面 | 打开右侧生成藏头诗开关，使用者输入四字关键词，点击发送按钮 | 1.用户输入的内容显示在聊天页面中，用户头像和系统头像正常显示  2.系统自动回复一首古诗，古诗每句话开头为用户输入的内容  3.回复内容合理的显示在聊天界面中 | | | 用户和系统头像显示正常，回复古诗正常，显示内容正常 |
| 3.输入异常数据 | 打开右侧生成藏头诗开关，然后用户输入关键词不满足四字要求，输入内容有无法识别的符号或者输入为空 | 1.提示输入关键词不符合要求  2.输入为空时发送按钮无效 | | | 实际结果同预期结果相同 |

## 6.4 用户管理

用户管理测试用例，主要用来测试用户管理页面各种功能是否正常。如表6-8所示。

表6-8 用户管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-5 | | 编写者 | 胡帅 | |
| 测试项 | 用户管理各项功能 | | | | |
| 前提条件 | 以用户名、密码登录系统，点击用户管理 | | | | |
| 操作步骤 | 点击指定功能按钮，系统做出相应反应 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 1.点击用户管理菜单栏 | 无 | 默认显示数据库内所有用户信息， | | | 实际结果同预期结果相同 |
| 点击添加按钮 | 系统弹出添加用户信息对话框，点击取消返回列表 | | | 实际结果同预期结果相同 |
| 点击编辑按钮 | 系统弹出编辑用户信息对话框，点击取消返回列表 | | | 实际结果同预期结果相同 |
| 2.添加用户弹框 | 根据页面提示填写用户信息，所有信息正确填写完成。 | 1.使用者点击提交按钮，在数据填写正确并通过验证。  2.添加弹框自动关闭，显示保存成功的提示  3.返回到用户列表，列表刷新，新添加用户在列表中。 | | | 与预期结果相同 |
| 输入数据时，非空选项不填写数据，其中一个或多个不填写 | 1.非空选项不填写数据，点击确定按钮时，按钮无效，并且以红色字体提示必填数据。  2.非空信息填写完成后，发布按钮功能正常，数据保存成功 | | | 实际结果同预期结果相同 |
| 3.编辑用户弹框 | 根据需要修改用户信息，所有信息正确修改完成。 | 1.点击确定按钮，在填写无误的情况下，用户信息保存到数据库  2页面提示修改成功  3.返回到用户列表，列表刷新，修改用户在列表中信息发生变化。 | | | 与预期结果相同 |
| 输入数据时，非空选项，其中一个或多个数据为空时 | 1.非空选项不填写数据，点击确定按钮时，按钮无效，并且以红色字体提示必填数据。  2.非空信息填写完成后，发布按钮功能正常，数据保存成功 | | | 实际结果同预期结果相同 |

## 6.5 结束语

古诗是中国历史文化中的一个精髓，使用计算机自动生成古诗不仅是科学的发展的结果，还是对古诗文化的传承。古诗管理与生成系统不仅是自己所学知识的检验，还是对新兴算法的一个学习和实践的过程。在该系统开发的工程中本文主要分析、设计和实现了古诗的管理和根据关键字生成一首古诗。主要功能如下：

古诗搜索：系统使用者可以对数据库内所有古诗进行查看，根据关键字进行拆查询古诗。

古诗管理：系统使用者可以对古诗进行添加、修改、删除等一系列的操作，其中古诗分类、古诗标签能够对添加的古诗和数据库内古诗进行一个简单的分类和定义标签，以便以后能够更简单的搜索，在后续的功能中能够根据分类和标签进行再次的训练，用以生成古诗，改善生成效果。

古诗生成：在此功能下系统能够根据用户输入的关键词生成一首与关键词相关的古诗（例如：输入‘春’，系统输出：‘旧见烟中春，南里已又飞。绿竹留上夜，楚花向峰心。’）。

用户管理：系统使用者根据需要能够对登录用户进行添加、编辑和修改等一系列的操作。

# 致谢

首先感谢中工44亩地，它给我提供了这个平台和这次机会。同时我要感谢王尧导师在这几年里，不论是授课还是大学四年的实训中、毕业设计、又或平时的思想指导上都给我提供了巨大的帮助，很感谢老师对我的各种帮助和对我问题耐心的讲解，让我在思路方面得到了扩展，我想对老师说：“老师，您辛苦了！”。同时感谢那些帮助过我的老师和同学们，感谢你们这几年的殷勤教会和无私的帮助，请允许我为你们鞠个躬来表达我心中最诚挚的谢意。我相信在跟随老师学习的过程中，这些知识和经验会一直影响着我。

# 参考文献

1. 周志华. 机器学习. 北京：清华大学出版社 2018: 53
2. 欧二强. Java编程手记例. 北京： 清华大学出版社. 2016.6
3. 疯狂软件.SpringBoot2企业应用实战[M]. 北京：电子工业出版社. 2018.6
4. 软件工程 (英)萨默维尔著，程成等译. 北京：机械工业出版社. 2011
5. 张雯杰.MongoDB从入门到商业实战. 北京：电子工业出版社. 2019.9
6. 阿里巴巴数据技术及产品部.大数据之路[M].北京：电子工业出版社，2017.7
7. Python编程从入门到实践.(美)Eric Matthes著/袁国忠译. 北京：人民邮电出版社，2017.9
8. Python机器学习基础教程（德）Andreas C. Müller、（美）Sarah Guido.张亮译. 北京：人民邮电出版社.2018.01
9. 利用Python进行数据分析.(美) Wes McKinney著,徐敬一译. 北京：机械工业出版社.2018.07
10. 张帆. Vue.js项目开发实战. 北京：机械工业出版社出版 2018.07
11. 邱锡鹏. 神经网络与机器学习. 北京：机械工业出版社. 2020:65
12. Vue.js <https://cn.vuejs.org/v2/guide/>
13. Mika Hamalainen and Khalid Alnajjar. Generating Modern Poetry Automatically in Finnish. In Proceedings of EMNLP, 2019.
14. 附录

## 附录A 古诗管理与生成系统核心代码

# 将给定数据填充到相同长度

def padding\_seq(seq):

results = []

max\_len = 0

for s in seq:

if max\_len < len(s):

max\_len = len(s)

for i in range(0, len(seq)):

l = max\_len - len(seq[i])

results.append(seq[i] + [0 for j in range(l)])

return results

def encode\_text(words, vocab\_indices):

return [vocab\_indices[word] for word in words if word in vocab\_indices]

def decode\_text(labels, vocabs, end\_token = '</s>'):

results = []

for idx in labels:

word = vocabs[idx]

if word == end\_token:

return ' '.join(results)

results.append(word)

return ' '.join(results)

def read\_vocab(vocab\_file):

f = open(vocab\_file, 'rb')

vocabs = [line.decode('utf8')[:-1] for line in f]

f.close()

return vocabs

class SeqReader():

def \_\_init\_\_(self, input\_file, target\_file, vocab\_file, batch\_size,

queue\_size = 2048, worker\_size = 2, end\_token = '</s>',

padding = True, max\_len = 50):

self.input\_file = input\_file

self.target\_file = target\_file

self.end\_token = end\_token

self.batch\_size = batch\_size

self.padding = padding

self.max\_len = max\_len

# self.vocabs = read\_vocab(vocab\_file) + [end\_token]

# 读取vocabas中的数据 并转化为对应的数据字典

self.vocabs = read\_vocab(vocab\_file)

self.vocab\_indices = dict((c, i) for i, c in enumerate(self.vocabs))

self.data\_queue = Queue(queue\_size)

self.worker\_size = worker\_size

with open(self.input\_file, encoding='utf-8') as f:

for i, l in enumerate(f):

pass

f.close()

self.single\_lines = i+1

self.data\_size = int(self.single\_lines / batch\_size)#总行数/迭代大小（32）

self.data\_pos = 0

self.\_init\_reader()

def start(self):

return

'''

for i in range(self.worker\_size):

t = Thread(target=self.\_init\_reader())

t.daemon = True

t.start()

'''

# 将序列的所有元素随机排序

def read\_single\_data(self):

if self.data\_pos >= len(self.data):

random.shuffle(self.data)

self.data\_pos = 0

result = self.data[self.data\_pos]

self.data\_pos += 1

return result

def read(self):

while True:

batch = {'in\_seq': [],

'in\_seq\_len': [],

'target\_seq': [],

'target\_seq\_len': []}

for i in range(0, self.batch\_size):

item = self.read\_single\_data()

batch['in\_seq'].append(item['in\_seq'])

batch['in\_seq\_len'].append(item['in\_seq\_len'])

batch['target\_seq'].append(item['target\_seq'])

batch['target\_seq\_len'].append(item['target\_seq\_len'])

if self.padding:

batch['in\_seq'] = padding\_seq(batch['in\_seq'])

batch['target\_seq'] = padding\_seq(batch['target\_seq'])

yield batch

def \_init\_reader(self):

self.data = []

input\_f = open(self.input\_file, 'rb')

target\_f = open(self.target\_file, 'rb')

for input\_line in input\_f:

input\_line = input\_line.decode('utf-8')[:-1]

target\_line = target\_f.readline().decode('utf-8')[:-1]

input\_words = [x for x in input\_line.split(' ') if x != '']

# 如果一行数据太长 取不超过最大长度的部分

if len(input\_words) >= self.max\_len:

input\_words = input\_words[:self.max\_len-1]

# 加入结束标志</s>

input\_words.append(self.end\_token)

target\_words = [x for x in target\_line.split(' ') if x != '']

if len(target\_words) >= self.max\_len:

target\_words = target\_words[:self.max\_len-1]

# 加入开始<s> 和结束</s>

target\_words = ['<s>',] + target\_words

target\_words.append(self.end\_token)

in\_seq = encode\_text(input\_words, self.vocab\_indices)

target\_seq = encode\_text(target\_words, self.vocab\_indices)

self.data.append({

'in\_seq': in\_seq,

'in\_seq\_len': len(in\_seq),

'target\_seq': target\_seq,

'target\_seq\_len': len(target\_seq) - 1

})

input\_f.close()

target\_f.close()

self.data\_pos = len(self.data)

import tensorflow as tf

from tensorflow.contrib import rnn

from tensorflow.python.layers import core as layers\_core

#获取多隐藏层神经网络单元

def getLayeredCell(layer\_size, num\_units, input\_keep\_prob,

output\_keep\_prob=1.0):

# 构建神经网络，

#防止过拟合使用的函数

return rnn.MultiRNNCell([rnn.DropoutWrapper(rnn.BasicLSTMCell(num\_units),

input\_keep\_prob, output\_keep\_prob) for i in range(layer\_size)])

# 构建编码encoder

def bi\_encoder(embed\_input, in\_seq\_len, num\_units, layer\_size, input\_keep\_prob):

# 将输入编码为向量

bi\_layer\_size = int(layer\_size / 2)

# 定义了正向的lstm和反向的lstm

encode\_cell\_fw = getLayeredCell(bi\_layer\_size, num\_units, input\_keep\_prob)

encode\_cell\_bw = getLayeredCell(bi\_layer\_size, num\_units, input\_keep\_prob)

# 拼接编码输出和状态

encoder\_output = tf.concat(bi\_encoder\_output, -1)

encoder\_state = []

for layer\_id in range(bi\_layer\_size):

encoder\_state = tuple(encoder\_state)

return encoder\_output, encoder\_state

# 注意力机制

# 定义decoder的rnn神经网络 带有attention机制的多层rnn。

def ss(encoder\_output, in\_seq\_len, num\_units, layer\_size,

input\_keep\_prob):

attention\_mechanim = tf.contrib.seq2seq.BahdanauAttention(num\_units,

encoder\_output, in\_seq\_len, normalize = True)

# attention\_mechanim = tf.contrib.seq2seq.LuongAttention(num\_units,

# encoder\_output, in\_seq\_len, scale = True)

cell = getLayeredCell(layer\_size, num\_units, input\_keep\_prob)

cell = tf.contrib.seq2seq.AttentionWrapper(cell, attention\_mechanim,

attention\_layer\_size=num\_units)

return cell

def seq2seq(in\_seq, in\_seq\_len, target\_seq, target\_seq\_len, vocab\_size,

num\_units, layers, dropout):

#获取维度信息

in\_shape = tf.shape(in\_seq)

batch\_size = in\_shape[0]

if target\_seq != None:

input\_keep\_prob = 1 - dropout

else:

input\_keep\_prob = 1

projection\_layer=layers\_core.Dense(vocab\_size, use\_bias=False)

# embedding input and target sequence

# 嵌入输入和目标序列

with tf.device('/cpu:0'):

embedding = tf.get\_variable(

name = 'embedding',

shape = [vocab\_size, num\_units])

embed\_input = tf.nn.embedding\_lookup(embedding, in\_seq, name='embed\_input')

# encode and decode

encoder\_output, encoder\_state = bi\_encoder(embed\_input, in\_seq\_len,

num\_units, layers, input\_keep\_prob)

batch\_size = tf.shape(in\_seq\_len)[0]

if target\_seq != None:

embed\_target = tf.nn.embedding\_lookup(embedding, target\_seq,

name='embed\_target')

helper = tf.contrib.seq2seq.TrainingHelper(

embed\_target, target\_seq\_len, time\_major=False)

else:

# TODO: start tokens and end tokens are hard code

helper = tf.contrib.seq2seq.GreedyEmbeddingHelper(

embedding, tf.fill([batch\_size], 0), 1)

decoder = tf.contrib.seq2seq.BasicDecoder(decoder\_cell, helper,

init\_state, output\_layer=projection\_layer)

# dynamic\_decode()是一个Decoder类，主要功能是解码序列生成

# 依据Encoder进行解码，实现序列的生成（映射)

outputs, \_, \_ = tf.contrib.seq2seq.dynamic\_decode(decoder,

maximum\_iterations=100)

if target\_seq != None:

return outputs.rnn\_output

else:

return outputs.sample\_id

def seq\_loss(output, target, seq\_len):

target = target[:, 1:]

batch\_size = tf.shape(target)[0]

loss\_mask = tf.sequence\_mask(seq\_len, tf.shape(output)[1])

cost = cost \* tf.to\_float(loss\_mask)

return tf.reduce\_sum(cost) / tf.to\_float(batch\_size)

## 附录B: 软件使用说明书

1. 软件概述

本系统为古诗管理与生成系统，此系统可以对古诗进行管理，并且能够生成古诗。鉴于以上因素系统选择的开发框架是以springboot为基础采用ssm框架搭建和MongoDB数据库来开发本系统，古诗生成使用python进行编写。

1.1 功能

本课题实现的是古诗管理与生成系统，设计界面友好、明确、人性化。主要功能如下：

1.古诗搜索；

2.管理管理；

3.古诗生成；

4.用户管理；

2. 软件安装

首先在官方网站下载了JDK1.8.0、IDEA和Tomcat9.0、MongoDB4.0、Python3.6，然后把下载的压缩包根据安装向导安装到指定盘下，在python中安装tensorflow1.8。

2.1 系统要求

系统要求配置见表1-3：

表B-3网站运行硬件配置表

|  |  |  |
| --- | --- | --- |
|  | **最低配置** | **推荐配置** |
| CPU | P/500 | P8/6.0G |
| 内存 | 4MB | 16G |
| 硬盘 | 10 G | 100G |
| 显示器 | VGA | LCD |

系统软件需求：数据库选用MongoDB，操作平台为Windows XP以上操作系统。

2.2 安装前的准备

右键我的电脑在弹出菜单中选择"属性"→"高级系统设置"→"环境变量"，会弹出编辑环境对话框，就可以编辑系统的环境变量了，同时配置好JDK、Python环境变量，安装Tensorflow。

2.3 安装

将程序上传至服务器中，前端Vue提前加载模型，设置MongoDB的数据库访问目录，配置python运行训练古诗生成模型，然后运行服务。

2.4 安装后

启动Tomcat后，打开浏览器，在地址栏中输入<http://localhost:8080（Tomcat>的端口可以修改，一般默认端口为8080），如果在浏览器中看到Tomcat的欢迎界面，表示Tomcat能够工作正常。

3. 测试项目

首先编写一个test.jsp文件，使其在页面显示“Hello World!”。然后把test.jsp放在C:\Tomcat 6.0\webapps\ROOT目录下，在地址栏输入<http://localhost:8080/test.jsp>,如果浏览器显示“Hello World!”，则说明环境变量配置成功了。在键盘同时按住win+R键，在弹出的界面中输入cmd进入管理员控制台，输入python回车，显示当前安装的python版本，证明安装成功。

4. 使用过程

1.启动Vue前端项目、SpringBoot后端项目、Python算法服务和服务接口。

1.在地址栏中输入<http://localhost:8081>/login便可以进入网站首页。

2.输入正确的用户名、密码之后提示登录成功，进入系统主页面，系统右侧菜单栏中显示可以操作的选项。用户可以进行相应的操作。管理员的用户名默认为zhangsan，密码默认123456。

4. 疑难解答

本系统采用的是前后端分离，如果提示有错误，确定是页面错误还是后台服务错误，根据相应提示确定错误位置并进行修改。