南開大學
Nankai University

计算机学院

软件工程

软件工程项目文档

姓名：程佳诺、马羽潇、王乐颖、汪晨、朱梓豪

学号：2012628、2014073、2011880、2013566、2011553

专业：计算机科学与技术、信息安全

2023 年 7 月 2 日

目录

1 小组个人信息

小组信息：
总第 41 组，班级 2 班，班级组别 2 班第 21 组，组长程佳诺 2012628
小组分工情况：
前端界面及 ppt、项目展示视频制作：马羽潇 2014073;
数据库设计、数据库具体内容填充：王乐颖 2011880;
前后端代码、前后端接口代码：程佳诺（组长）2012628; 汪晨 2013566; 朱梓豪 2011553;

2 项目概述

英语六级在线考试系统是一个基于 SpringBoot+Vue 开发的全面在线考试解决方案。该系统旨在提供一个方便、高效和全面的考试平台，用于英语六级考试的报考、在线考试、教师阅卷、成绩查询等功能。通过采用现代化的前后端分离架构和一系列强大的技术栈，该系统能够提供流畅的用户体验和可靠的性能。

管理员和学生是该系统的两类用户，他们登录后将进入不同的界面，根据各自的需求和权限进行操作。管理员拥有全面的管理权限，可以通过控制台获取项目的基本信息，并在在线考试窗口中查看和选择各种考试。管理员还可以管理题库，包括试题的上传和删除，以及管理考试的各项设置。此外，系统管理和用户管理功能可让管理员轻松管理系统配置、用户信息和权限。

学生用户通过登录后可以在控制台中查看相关信息，并通过在线考试窗口选择参加考试。学生还可以查询以往的考试成绩，方便进行自我评估和学习计划的制定。

该系统具备一系列强大的功能和特点。首先，系统拥有完善的前后端分离架构，提供规范的接口文档，确保系统的稳定性和可扩展性。其次，权限控制是该系统的关键功能，采用了 Shiro 和 JWT 技术，确保用户身份验证和权限管理的安全性。基础功能模块包括系统配置、用户管理、部门管理和角色管理等，提供了完善的系统管理工具。题库管理功能支持多种题型，包括单选题、多选题和判断题，并支持试题的批量导入导出，方便管理和使用。此外，考试权限功能可以根据需求指定考试范围，包括完全公开和指定部门人员考试等灵活设置。在线考试功能允许学员进行在线考试，查看分数，并提供考试错题训练，帮助学生加强对知识点的掌握和理解。

该系统是一个功能全面、操作便捷的考试解决方案。通过其先进的技术栈和完善的功能，用户可以轻松进行报考、在线考试和成绩查询等操作。这个系统将极大地提升英语六级考试的管理效率和学习体验，为用户提供高质量的考试服务。

3 需求分析与系统设计

3.1 需求分析

3.1.1 管理员需求

- 登录和注册账号：管理员需要通过登录系统来管理各项功能，同时可以注册新的管理员账号。
- 控制台信息展示：管理员可以查看系统的基本信息和技术栈，以及其他相关信息。
- 在线考试窗口：管理员可以查看可用的考试列表，并选择进入指定考试进行管理和监控。
- 成绩窗口：管理员可以查看考试成绩和考试记录。

- 题库管理：管理员可以管理题库，包括试题的上传和删除。
- 试题管理：管理员可以上传、删除试题，对试题进行管理和维护。
- 考试管理：管理员可以管理考试，包括考试的设置、时间安排和权限控制。
- 系统管理：管理员可以进行系统配置，管理系统的基本设置和参数。
- 用户管理：管理员可以管理用户，包括添加、编辑和删除用户账号，以及分配用户角色和权限。

3.1.2 学生需求

- 登录和注册账号：学生需要通过登录系统来参加在线考试，并可以注册新的学生账号。
- 控制台信息展示：学生可以查看系统的基本信息和相关提示。
- 在线考试窗口：学生可以查看可用的考试列表，并选择参加指定考试。
- 查询成绩：学生可以查询以往的考试成绩和相关统计信息。
- 考试错题训练：学生可以通过系统提供的考试错题训练功能来巩固和复习知识点。

3.2 系统设计

- 前端设计：采用 Vue.js 和 Element UI 等技术栈进行前端开发，实现用户友好的界面和交互体验。
- 后端设计：采用 Spring Boot 框架进行后端开发，使用 Shiro 进行用户身份认证和权限管理，使用 MyBatis Plus 进行数据库操作，利用 Redis 进行缓存优化，同时使用 JWT 实现安全的身份验证和授权。
- 数据库设计：设计适当的数据库表结构，包括用户表、考试表、成绩表和试题表等，以支持系统的功能需求。
- 接口设计：设计规范的接口文档，定义前后端之间的数据交互格式和接口规范。
- 题库管理：实现支持不同类型试题（单选题、多选题、判断题）的管理功能，包括试题的批量导入和导出。
- 考试管理：实现灵活的考试设置，包括考试时间、权限控制、考试范围等。
- 在线考试功能：设计在线考试的页面和流程，包括考试界面的展示、答题、交卷和计分等功能。
- 成绩查询：实现学生查询以往考试成绩和统计信息的功能，提供清晰的结果展示。
- 错题训练：为学生提供考试错题的训练功能，帮助他们加强对知识点的掌握和理解。

通过以上的需求分析与系统设计，我们可以确保该英语六级在线考试系统具备完善的功能和良好的用户体验。系统将提供高效、可靠的在线考试解决方案，满足管理员和学生的不同需求，并提供灵活的题库管理、考试管理和成绩查询等功能，以优化英语六级考试的管理和学习过程。

4 项目管理

接口项目为 SpringBoot 后端项目、提供完整的业务逻辑和接口访问，实现前后端分离，接口提供给管理后台访问。

管理后台采用 Vue / Element 框架，主要实现后台管理内容，WEB 学员端考试功能。

程序运行时，使用 vue cli 脚手架程序打包前端 vue 框架，将 dist 目录下的内容拷贝到/exam-api/src/main/resources/static 目录下，这样就可以直接运行后端运行整个程序，无需单独部署前端。

项目开发过程中,使用 github 和飞书群聊进行项目开发和跟进,通过 git commit、merge、push/pull 来共同开发维护代码。

5 用户手册

5.1 引言

5.1.1 编写目的

本手册是南开大学 2023 年软件工程编程大作业英语六级考试系统（以下简称考试系统）面对用户编写的使用说明手册，在本文档中对项目的相关操作进行了详细的描述，通过本文档读者可以了解系统的全部功能。

5.1.2 背景

在线考试系统的目的是创办一种新的考试模式，通过这种新的模式，弥补传统线下考试的缺点，为学生提供一种新的学习、考试环境，提高组织考试工作的效率和标准化水平。使学校管理者、教师和学生可以随时随地通过网络进行考试。在线考试系统取代了传统考试的应用型软件，完全实现电脑自动化。

传统考试从出题、组卷、印刷，到试卷的分发，答题、收卷，再到判卷，公布成绩，统计分析考试结果整个过程都需要人工参与，周期长，工作量大，容易出错，还要有适当的保密工作，使得整个学习考试成本较大。在线考试学习系统可以完全实现无纸化、网络化、自动化的计算机在线学习考试，对单位的信息化建设具有深远的现实意义和实用价值。

本系统设计的目的即搭建一套简易的在线考试模式用于实际应用的参考。

5.2 软件概述

5.2.1 目标

使用户通过阅读本手册能够掌握程序的功能，并且能够做到使用本系统进行在线考试的相关步骤。

5.2.2 功能概述

程序的具体功能主要包括以下四点：

1. 不同权限级别账号的注册登录
2. 线上考试和结果查询
3. 系统设置的设定和变更
4. 考试安排和试题管理

5.3 运行环境

5.3.1 硬件环境

- 内存：建议 16GB 及以上
- 硬盘：建议 128GB 及以上

5.3.2 软件环境

- 操作系统：建议 64 位 windows 10 及以上
- 其余内容可参照部署环境部分

5.4 使用说明

运行 ExamApplication.java 即可启动项目，当命令行输出提示信息”ClusterManager: Check-in complete.” 时即可通过浏览器访问系统。

图 5.1: 控制台输出

5.4.1 注册登录

若未登录，则访问系统时会先跳转至以下登录页面，可使用默认的用户名及密码登录，也可自行注册一个新的普通用户。普通用户的默认账号密码为 person/person，管理员的默认账号密码为 admin/admin。普通用户仅能使用在线考试功能，管理员能够使用在线考试、考试管理、系统设置功能。登录成功后，点击右上角下拉框，可查看用户资料或退出登录。

图 5.2: 注册登录

5.4.2 在线考试

在线考试包括在线考试、我的成绩两部分。

图 5.3: 在线考试

各部分大体功能如下：

- 在线考试可选定列表中的考试参加，点击“去考试”进入准备考试页面，点击“开始考试”正式开始考试。考试时间用尽或手动点击“交卷”可提交试卷，随后显示试题作答相关信息。
- 点击“我的成绩”即可查询已参加的考试结果，并可进一步查询考试详细和错题。点击“错题”，可以看到错题的详细信息，并进行错题训练。

5.4.3 系统设置

系统设置包括系统配置学校管理、角色管理及用户管理四部分。

图 5.4: 系统设置

各部分大体功能如下：

- 系统配置部分可更新系统名称、上传系统 LOGO 以及修改版权信息。
- 学校管理部分可添加、修改、删除、查询学校。
- 角色管理和用户管理部分用于展示和设定不同用户的权限信息。点击用户管理界面的“添加用户”，可添加普通用户或管理员。

5.4.4 考试管理

考试管理包括题库管理、试题管理及考试管理三部分。

图 5.5: 考试管理

各部分大体功能如下：

- 题库管理部分可添加题库、修改题库名称以及查看已有题库信息。选定题库，点击下拉框，可删除选定题库。
- 试题管理部分可添加、修改具体试题内容，按字段查找指定试题。点击右上角“导入”/”导出”可上传导入、下载导入模板以及导出试题（试题文件为.xlsx 格式）。
- 考试管理部分用于安排考试和查看考试内容。点击“查看详细”，可查看参加考试人员的姓名、考试次数、最高分、是否通过、考试时间等信息。

具体各界面样式可参照演示视频和本报告中的截图部分。

6 项目部署环境具体参数

JDK 1.8+ https://www.runoob.com/java/java-environment-setup.html
maven https://www.runoob.com/maven/maven-setup.html
nodejs https://www.runoob.com/nodejs/nodejs-install-setup.html
redis https://www.runoob.com/redis/redis-install.html
mysql 5.7+ https://cdn.yfhl.net/java-win/mysql-installer-community-5.7.31.0.msi
上述环境安装完成后，克隆项目代码，进入/exam-vue 目录，运行 npm install 安装依赖。

在 mysql 中导入仓库中的数据库脚本，mysql 用户名为 root，密码为 root。如果本地数据库不同，请修改/exam-api/src/main/resources 目录下的 yaml 配置文件 (application-dev.yml, application-local.yml)，将数据库字段改为本地数据库的用户名密码即可。

完成上述环境部署后，build 整体项目，编译运行
/exam-api/src/main/java/com/yf/exam/ExamApplication.java，
即可通过 http://localhost:8101/访问系统网站。

7 项目 github 地址

https://github.com/MrChengjn/CET6-application-system

8 项目演示截图

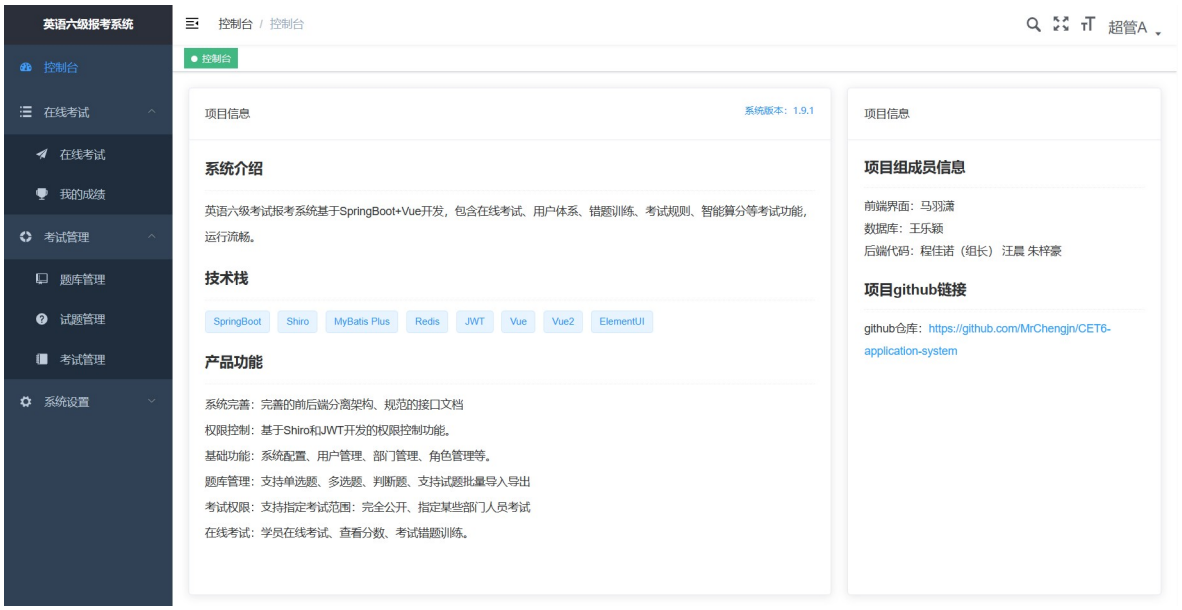

图 8.6: 首页展示

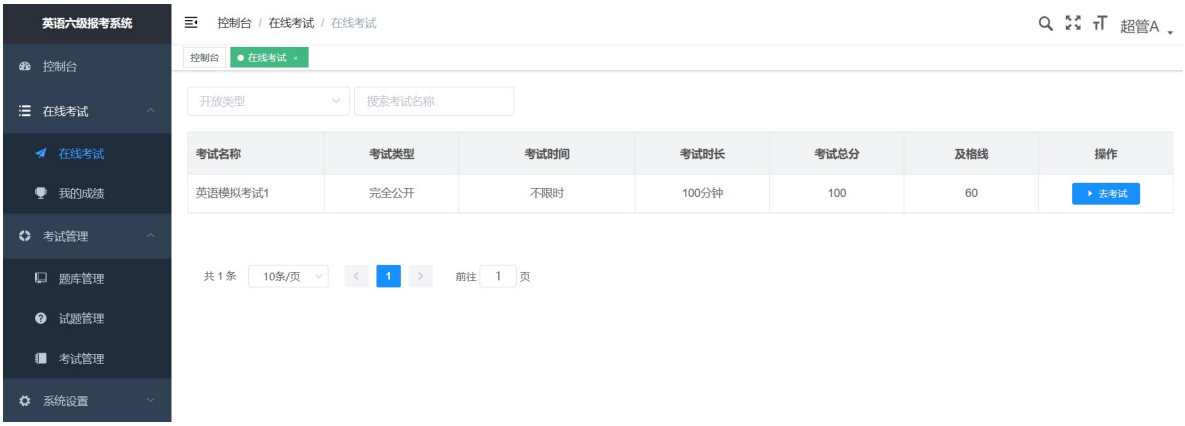

图 8.7: 考试选择界面

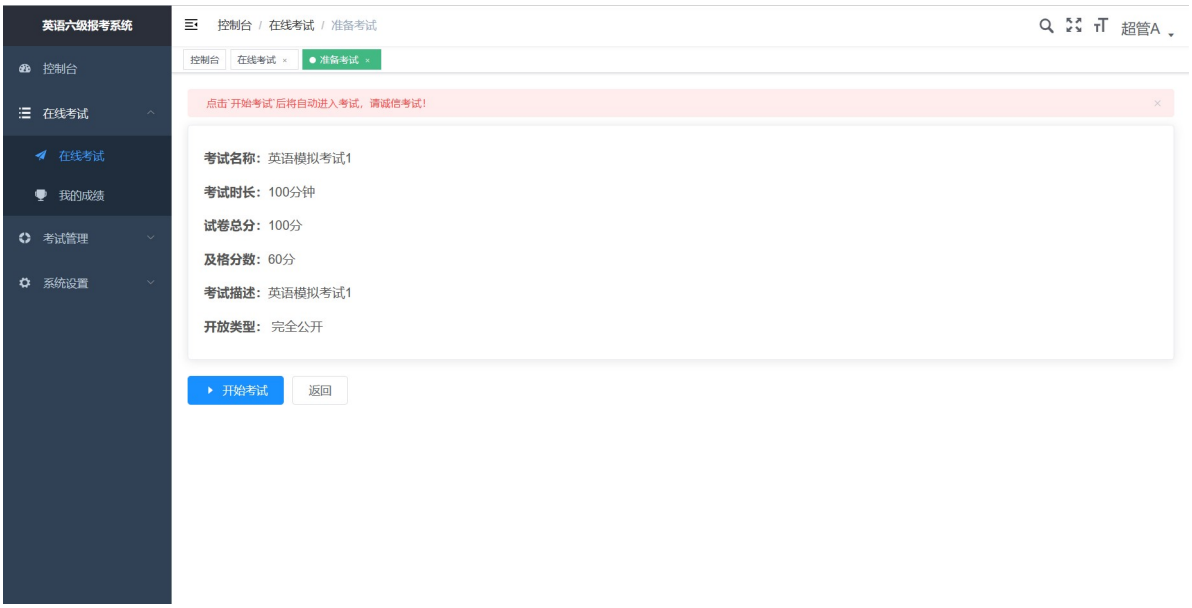

图 8.8: 考试准备界面

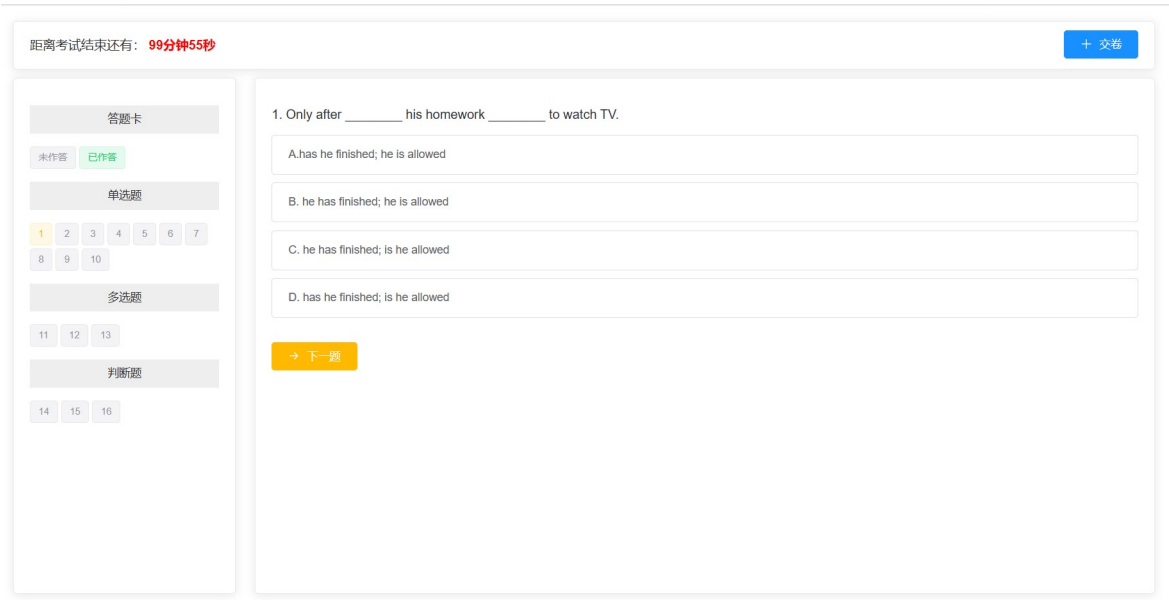

图 8.9: 在线考试界面

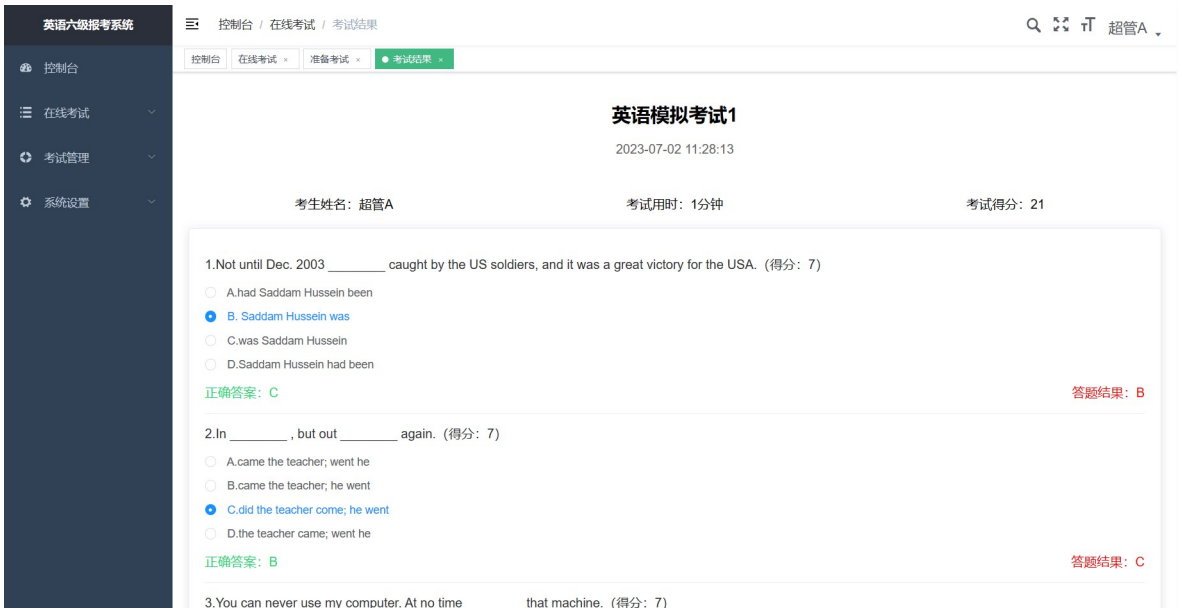

图 8.10: 考试结果界面

图 8.11: 成绩查询界面

图 8.12: 题库管理界面

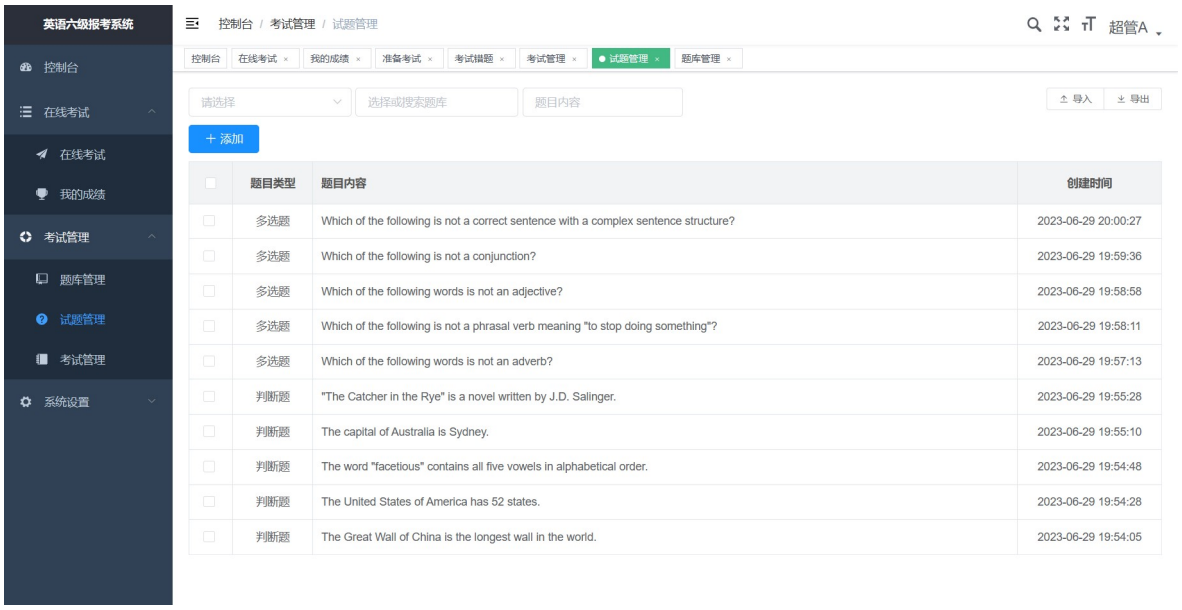

图 8.13: 试题管理界面

图 8.14: 考试管理界面

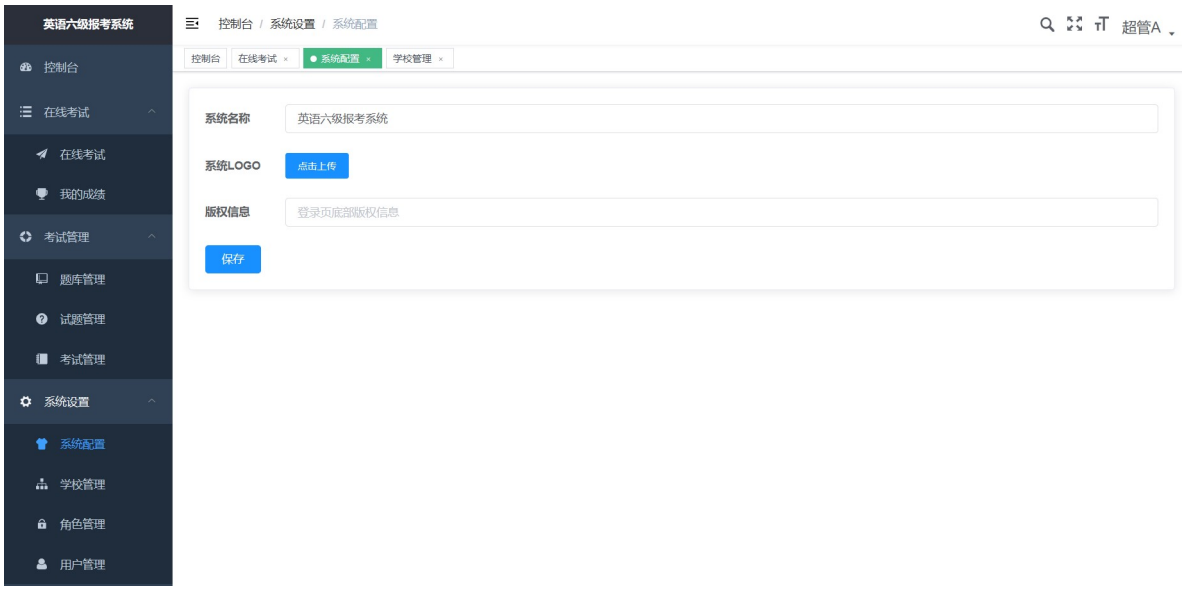

图 8.15: 系统配置界面

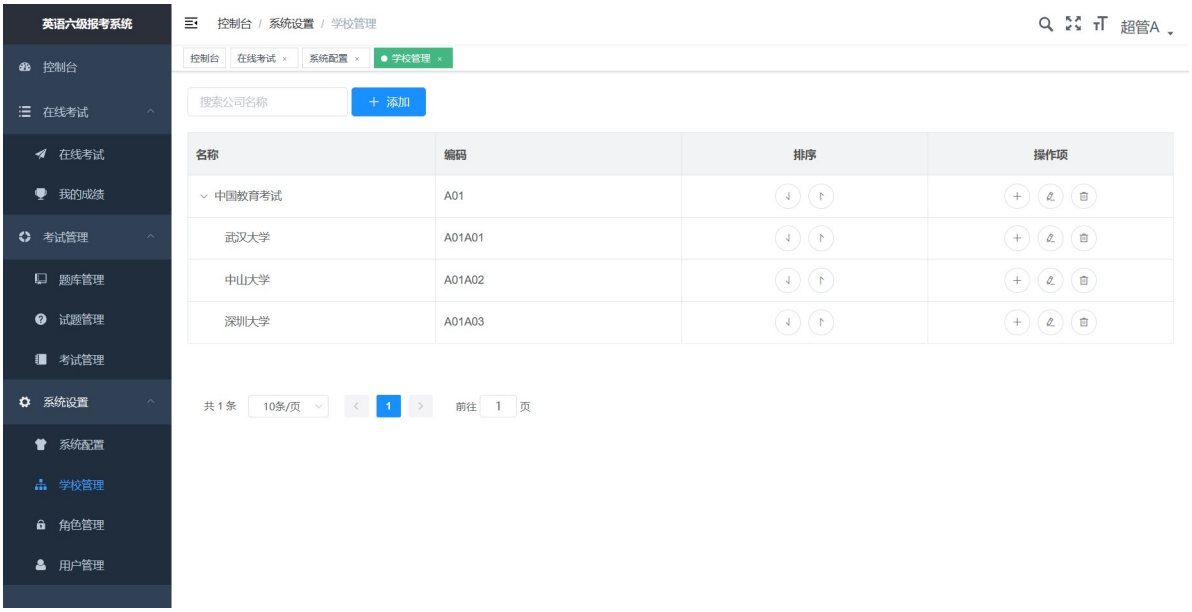

图 8.16: 学校管理界面

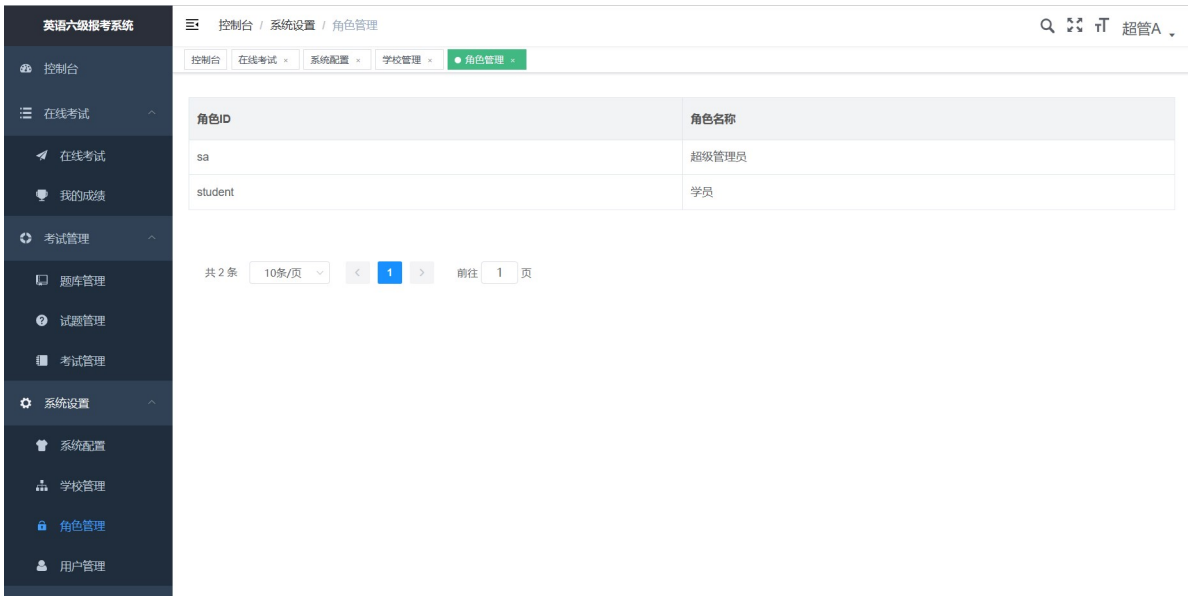

图 8.17: 角色管理界面

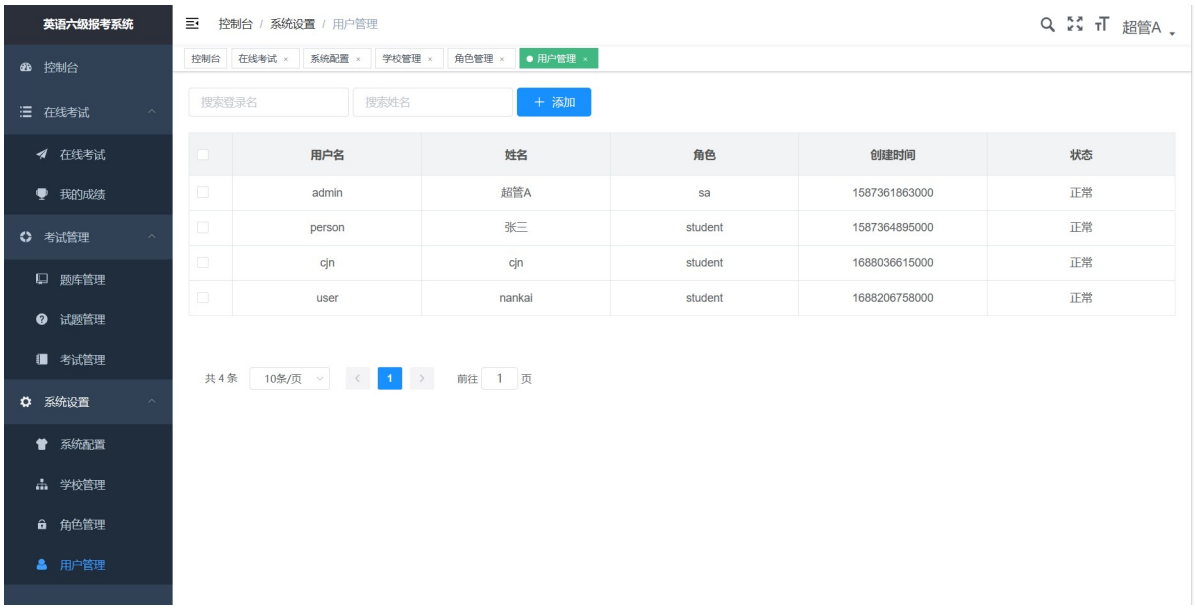

图 8.18: 用户管理界面

9 代码功能介绍

9.1 数据库设计

数据库部分的设计整体上分为两个核心板块：用户信息的存储以及考试安排相关信息的设计。

具体地，在用户相关信息的存储方面，我们将用户分为管理员用户和普通用户，以此区分登录后所拥有的不同权限。在考试安排和成绩管理方面，我们设计了试题信息表，答案记录表，题库信息表，考试安排表，成绩管理表等来辅助这部分功能的实现。

9.1.1 用户信息存储设计

我们首先一个**用户角色管理表**来汇总整体系统中所涉及的用户角色，目前包含两种角色：管理员角色和用户角色，并通过不同的角色 id 来进行区分。

参数	类型	描述
id	varchar(30)	角色 id, 主键
role_name	varchar(255)	角色名称

表 1: 用户角色表 role

本系统的目标用户主要是各大高校的大学生，因此这里单独设计一张**高校信息表**，为每个高校分配一个 id，方便后面对学生用户信息表的设计和管理。

参数	类型	描述
id	varchar(32)	高校 id, 主键
university_name	varchar(255)	高校名称

表 2: 高校信息表 university

接下来就是本部分设计的核心——**用户信息表 user**，主要包含了用户 id, 该用户的角色 id 和用户的一些个人信息和登录密码。

参数	类型	描述
id	varchar(32)	用户 id, 主键
role_id	varchar(30)	用户角色 id, 外键
university_id	varchar(32)	用户所属高校 id, 外键
user_name	varchar(32)	用户姓名
phone	varchar(20)	联系方式
mail	varchar(32)	用户邮箱
doctype	varchar(32)	证件类型
docnum	varchar(40)	证件号
password	varchar(30)	登陆密码
salt	varchar(255)	密码盐
create_time	datetime	注册时间
update_time	datetime	更新时间

表 3: 用户信息表 user

9.1.2 考试相关部分设计

首先是针对**考试安排表 exam** 的设计，包含考试编号 id, 考试名称，考试时间，考试时长，分数设置等细节信息。

每一次考试都需要为本次考试创建一个题库，由一个 id 进行标识，每个题库中会包含各种类型的试题，如选择题，作文题等，设计**题库表 exam_repo** 如下：

以下为**试题表 qu** 的设计，每一道试题由一个 id 标识，在每一条试题记录中，还应包含该试题的题目，难度，题目解析等信息.

每一个选择题都对应四个选项，因此需要一张表记录选择题的答案选项表，并通过一个标识来记录该选项是否为正确答案

参数	类型	描述
id	varchar(64)	考试 id, 主键
title	varchar(255)	考试名称
content	varchar(255)	考试描述
state	int	考试状态
time_limit	tinyint	是否限时
start_time	datetime	开始时间
end_time	datetime	结束时间
create_time	datetime	创建时间
update_time	datetime	更新时间
total_score	int	考试总分
total_time	int	考试总时长（分钟）
qualify_score	int	及格分数

表 4: 考试安排表 exam

参数	类型	描述
id	varchar(64)	考试题库 id, 主键
exam_id	varchar(64)	使用这个题库的考试 id，外键
radio_count	int	单选题数量
radio_score	int	单选题分数
writing_count	int	作文题数量
writing_score	int	作文题分数
translation_count	int	翻译题数量
translation_score	int	翻译题分数

表 5: 题库表 exam_repo

参数	类型	描述
id	varchar(64)	试题 id, 主键
qu_type	int	题目类型
level	int	题目难度，1 普通 2 较难
content	varchar(2000)	题目内容
create_time	datetime	创建时间
update_time	datetime	更新时间
analysis	varchar(2000)	题目解析

表 6: 试题表 qu

参数	类型	描述
id	varchar(64)	答案 id, 主键
qu_id	varchar(64)	对应试题 id, 外键
is_right	tinyint	该选项是否为正确选项
content	varchar(5000)	选项内容
create_time	datetime	创建时间
update_time	datetime	更新时间
analysis	varchar(3000)	答案解析

表 7: 试题答案选项表 answer

正如上述所说，每一次考试需指定一个题库，每个题库包含若干试题，考虑到某一道试题可以出现在多个题库当中，因此我们并没有在试题表中直接指定每个试题所属的题库，而是单独设计了一个

试题和题库的关联表 qu_repo, 从而提高了试题关联的灵活性。

参数	类型	描述
repo_id	varchar(64)	试题归属题库 id, 外键
qu_id	varchar(64)	试题 id, 外键
remark	varchar(1000)	备注

表 8: 试题题库关联表 qu_repo

在完成了考试安排和考试试题相关的设计后，接下来我们考虑关于用户的答题记录和成绩统计相关的问题。

首先我们先设计一个**用户考试记录表 paper**，用来记录用户的考试时长，考试得分（客观题得分和主观题得分）等信息, 其中加入本来的考试时长，考试总分，及格分等信息是为了方便对照。

参数	类型	描述
id	varchar(64)	用户答卷记录 id, 主键
user_id	varchar(64)	用户 id, 外键
exam_id	varchar(64)	考试 id, 外键
total_time	int	考试时长
user_time	int	用户考试用时
total_score	int	试卷总分
qualify_score	int	及格分
obj_score	int	客观分数
subj_score	int	主观分数
user_score	int	用户得分
state	int	试卷状态

表 9: 用户考试记录表 paper

我们也需要设计出用户对每一道题的答题记录表，方便统计分数和考后复盘。具体需要包括该道试题是否已答，该题是否答对以及该题的得分情况。

参数	类型	描述
paper_id	varchar(64)	用户答题记录 id, 外键
qu_id	varchar(64)	试题 id, 外键
qu_type	int	题目类型
answered	tinyint	是否已答
answer	varchar(5000)	主观题答案
score	int	该题分值
get_score	int	用户该题得分
is_right	tinyint	该题是否答对

表 10: 用户单题得分记录表 paper_qu

对于选择题来说，我们需要记录用户选择的选项和该题正确的选项，依次来判断用户在这道选择题中是否选择了正确的选项，因此这里额外设计了一张用户选择题记录表 choose_record.

针对用户本次考试中的错题，我们设计了错题记录表 wrong_record, 方便用户加入错题本进行不定期复盘，具体包括做错的题目 id，最近错误时间等信息。

系统与数据库之间进行交互通过 Mapper 来实现，通过 Mapper 来建立数据库中的字段和实体类之间的映射，从而完成对数据库表的增删改查操作。

参数	类型	描述
paper_id	varchar(64)	用户答题记录 id, 外键
answer_id	varchar(64)	用户单题回答记录 id, 外键
qu_id	varchar(64)	试题 id, 外键
is_right	tinyint	是否正确项
choosed	tinyint	该选项是否选中
title	vatchar(3)	选项标签

表 11: 用户选择题答题记录表 choose_record

参数	类型	描述
exam_id	varchar(64)	考试 id, 外键
user_id	varchar(64)	用户 id, 外键
qu_id	varchar(64)	试题 id, 外键
wrong_time	datetime	做错时间

表 12: 错题记录表 wrong_record

9.2 账号登录与注册

在现代软件应用中，账号登录和注册是非常重要的功能之一。无论是电子商务平台、社交媒体还是企业内部系统，用户都需要通过账号登录来访问个人信息、享受特定权限或执行特定操作。

在这个系统中，我们注重了账号登录和注册功能的设计与实现。通过精心选择和使用 Shiro 框架，我们确保了系统的安全性和可靠性。Shiro 提供了强大的身份验证和权限控制机制，使用户只有在提供正确的凭据后才能获得访问权限。

账号登录功能的实现主要涉及到以下代码片段：

```
@Override
public SysUserLoginDTO login(String userName, String password) {

    QueryWrapper<SysUser> wrapper = new QueryWrapper<>();
    wrapper.lambda().eq(SysUser::getUserName, userName);

    SysUser user = this.getOne(wrapper, false);
    if(user == null){
        throw new ServiceException(ApiError.ERROR_90010001);
    }

    // 被禁用
    if(user.getState().equals(CommonState.ABNORMAL)){
        throw new ServiceException(ApiError.ERROR_90010005);
    }

    boolean check = PassHandler.checkPass(password,user.getSalt(), user.getPassword());
    if(!check){
        throw new ServiceException(ApiError.ERROR_90010002);
    }

    return this.setToken(user);
```

```
}
```

上述代码片段展示了用户登录功能的实现过程。我们首先根据用户名查询数据库中的用户信息，如果用户不存在，则抛出相应的异常。接着，我们判断用户的状态是否正常，如果用户被禁用，则抛出异常，禁止登录。然后，我们使用 PassHandler.checkPass() 方法对用户输入的密码进行校验，确保输入的密码与数据库中存储的密码匹配。如果校验不通过，则抛出异常，拒绝登录。最后，我们调用 this.setToken() 方法为用户生成并设置访问令牌（Token），并将生成的登录信息返回给用户。

```
public SysUserLoginDTO reg(SysUserDTO reqDTO) {

    QueryWrapper<SysUser> wrapper = new QueryWrapper<>();
    wrapper.lambda().eq(SysUser::getUserName, reqDTO.getUserName());

    int count = this.count(wrapper);

    if(count > 0){
        throw new ServiceException(1, "用户名已存在，换一个吧！");
    }


    // 保存用户
    SysUser user = new SysUser();
    user.setId(IdWorker.getIdStr());
    user.setUserName(reqDTO.getUserName());
    user.setRealName(reqDTO.getRealName());
    PassInfo passInfo = PassHandler.buildPassword(reqDTO.getPassword());
    user.setPassword(passInfo.getPassword());
    user.setSalt(passInfo.getSalt());

    // 保存角色
    List<String> roles = new ArrayList<>();
    roles.add("student");
    String roleIds = sysUserRoleService.saveRoles(user.getId(), roles);
    user.setRoleIds(roleIds);
    this.save(user);

    return this.setToken(user);
}
```

在上述代码中，我们首先查询数据库中是否已存在相同用户名的用户，如果存在，则抛出相应的异常，要求用户使用其他用户名进行注册。接着，我们创建一个新的用户对象，并设置相应的属性，包括用户名、密码等。在设置密码时，我们使用 PassHandler.buildPassword() 方法生成加密密码和 Salt 值，并将其存储在用户对象中。接下来，我们保存用户角色信息，并将用户信息保存到数据库中。最后，我们调用 this.setToken() 方法为用户生成并设置访问令牌（Token），并将生成的登录信息返回给用户。

通过以上代码片段的说明，我们可以看到账号登录和注册功能的实现细节，包括数据库查询、密码校验、异常处理和令牌生成等步骤。这些代码片段的执行过程保证了系统的安全性和可靠性，为用

户提供了稳定和可信赖的账号登录和注册体验。

9.3 系统设置

当设计系统时，我们特别关注通用配置的功能和实现。通用配置用于存储系统的一些通用属性，例如系统名称、LOGO 图标和版权信息等。以下是相关代码示例，以更好地理解通用配置模块的设计和实现方式。

```
@Data
@ApiModel(value="通用配置", description="通用配置")
public class SysConfigDTO implements Serializable {
    private static final long serialVersionUID = 1L;
    @ApiModelProperty(value = "ID", required=true)
    private String id;
    @ApiModelProperty(value = "系统名称")
    private String siteName;
    @ApiModelProperty(value = "前端LOGO")
    private String frontLogo;
    @ApiModelProperty(value = "后台LOGO")
    private String backLogo;
    @ApiModelProperty(value = "版权信息")
    private String copyRight;
}

// SysConfigController.java
@Api(tags={"通用配置"})
@RestController
@RequestMapping("/exam/api/sys/config")
public class SysConfigController extends BaseController {
    @Autowired
    private SysConfigService baseService;

    // 添加或修改通用配置信息
    @ApiOperation(value = "添加或修改")
    @RequestMapping(value = "/save", method = { RequestMethod.POST})
    public ApiRest<BaseIdRespDTO> save(@RequestBody SysConfigDTO reqDTO) {
        // 复制参数并保存或更新通用配置
        SysConfig entity = new SysConfig();
        BeanMapper.copy(reqDTO, entity);
        baseService.saveOrUpdate(entity);
        return super.success(new BaseIdRespDTO(entity.getId()));
    }

    // 查找通用配置的详情
    @ApiOperation(value = "查找详情")
    @RequestMapping(value = "/detail", method = { RequestMethod.POST})
    public ApiRest<SysConfigDTO> find() {
        SysConfigDTO dto = baseService.find();
        return super.success(dto);
```

```
    }
}

// SysConfig.java
@Data
@TableName("sys_config")
public class SysConfig extends Model<SysConfig> {
    private static final long serialVersionUID = 1L;
    @TableId(value = "id", type = IdType.ASSIGN_ID)
    private String id;
    @TableField("site_name")
    private String siteName;
    @TableField("front_logo")
    private String frontLogo;
    @TableField("back_logo")
    private String backLogo;
    @TableField("copy_right")
    private String copyRight;
}

// SysConfigMapper.java
public interface SysConfigMapper extends BaseMapper<SysConfig> {
    // 方法省略
}

// SysConfigService.java
public interface SysConfigService extends IService<SysConfig> {
    SysConfigDTO find();
}

// SysConfigServiceImpl.java
@Service
public class SysConfigServiceImpl extends ServiceImpl<SysConfigMapper, SysConfig> implements
     SysConfigService {
    @Override
    public SysConfigDTO find() {
        QueryWrapper<SysConfig> wrapper = new QueryWrapper<>();
        wrapper.last(" LIMIT 1");
        SysConfig entity = this.getOne(wrapper, false);
        SysConfigDTO dto = new SysConfigDTO();
        BeanMapper.copy(entity, dto);
        return dto;
    }
}
```

通过以上代码示例，我们可以看到：

- SysConfigDTO 类用于表示通用配置的数据模型，其中包含了系统名称、LOGO 图标和版权信息等属性。

- SysConfigController 控制器负责处理通用配置的请求，并提供了添加或修改通用配置信息的接口 (save) 和查找通用配置详情的接口 (find)。
- SysConfigService 接口定义了对通用配置的操作方法，SysConfigServiceImpl 实现类实现了这些方法的具体逻辑。在 SysConfigServiceImpl 中，通过 SysConfigMapper 进行数据库交互，使用查询条件和限制来获取通用配置信息，并将结果映射到 SysConfigDTO 对象中返回。

通过这些核心代码，我们可以更好地理解通用配置模块的设计和实现方式。它提供了灵活的接口和数据模型，使得我们可以方便地管理和修改系统的通用属性。无论是添加新的配置项还是更新现有的配置信息，这个通用配置模块都为我们提供了便利和安全性。

9.4 试题管理

该部分是本系统的核心功能之一，设计时需要考虑对试卷以及答题的具体操作，主要包括修改试题、显示试卷相关内容、监控考试状态、更新考试状态等条目。

```
//PaperDTO
@Data
@ApiModel(value="试卷", description="试卷")
public class PaperDTO implements Serializable {
    private static final long serialVersionUID = 1L;
    @ApiModelProperty(value = "试卷ID", required=true)
    private String id;
    //此处略去部分
    @ApiModelProperty(value = "截止时间")
    private Date limitTime;
}
//PaperController.java
public class PaperController extends BaseController {
    @Autowired
    private PaperService baseService;
    @ApiOperation(value = "添加或修改")
    @RequestMapping(value = "/save", method = { RequestMethod.POST})
    public ApiRest<BaseIdRespDTO> save(@RequestBody PaperDTO reqDTO) {
        //方法略
    }
    @ApiOperation(value = "批量删除")
    @RequestMapping(value = "/delete", method = { RequestMethod.POST})
    public ApiRest edit(@RequestBody BaseIdsReqDTO reqDTO) {
        //方法略
    }
    @ApiOperation(value = "查找详情")
    @RequestMapping(value = "/detail", method = { RequestMethod.POST})
    public ApiRest<PaperDTO> find(@RequestBody BaseIdReqDTO reqDTO) {
        //方法略
    }
    @ApiOperation(value = "分页查找")
    @RequestMapping(value = "/paging", method = { RequestMethod.POST})
```

```
    public ApiRest<IPage<PaperListRespDTO>> paging(@RequestBody PagingReqDTO<PaperListReqDTO>
        reqDTO) {
        //方法略
    }
    @ApiOperation(value = "创建试卷")
    @RequestMapping(value = "/create-paper", method = { RequestMethod.POST})
    public ApiRest<BaseIdRespDTO> save(@RequestBody PaperCreateReqDTO reqDTO) {
        String paperId = baseService.createPaper(UserUtils.getUserId(), reqDTO.getExamId());
        return super.success(new BaseIdRespDTO(paperId));
    }
    @ApiOperation(value = "试卷详情")
    @RequestMapping(value = "/paper-detail", method = { RequestMethod.POST})
    public ApiRest<ExamDetailRespDTO> paperDetail(@RequestBody BaseIdReqDTO reqDTO) {
        ExamDetailRespDTO respDTO = baseService.paperDetail(reqDTO.getId());
        return super.success(respDTO);
    }
    //略去部分方法
    @ApiOperation(value = "检测进行中的考试")
    @RequestMapping(value = "/check-process", method = { RequestMethod.POST})
    public ApiRest<PaperDTO> checkProcess() {
        //方法略
    }
}
//PaperService.java
public interface PaperService extends IService<Paper> {
    String createPaper(String userId, String examId);
    ExamDetailRespDTO paperDetail(String paperId);
    ExamResultRespDTO paperResult(String paperId);
    PaperQuDetailDTO findQuDetail(String paperId, String quId);
    void fillAnswer(PaperAnswerDTO reqDTO);
    void handExam(String paperId);
    IPage<PaperListRespDTO> paging(PagingReqDTO<PaperListReqDTO> reqDTO);
    PaperDTO checkProcess(String userId);
}
```

通过以上代码示例，我们可以了解到试题管理部分具体的实现：

- PaperController 负责包装试题相关方法，为其它模块提供了操作试题的接口。
- PaperService 中则是定义了根据试题 ID、考试 ID 等参数进行试题管理的操作方法。具体的逻辑则由 PaperServiceImpl 实现，由于代码量过大故不在此处展示。