

**《王东南互联网在线考试系统》**

**项目详细设计**

**V3.0**

**项目实训第12组 三元二次方程组**

目录

[第一部分 引言 4](#_Toc98943114)

[一、编写目的 4](#_Toc98943115)

[二、项目背景 4](#_Toc98943116)

[三、定义 4](#_Toc98943117)

[1、J2EE四层架构 4](#_Toc98943118)

[2、B/S架构 4](#_Toc98943119)

[3、前后端分离 4](#_Toc98943120)

[4、微服务架构 4](#_Toc98943121)

[四、参考资料 5](#_Toc98943122)

[第二部分 项目概述 5](#_Toc98943123)

[第三部分 架构设计 5](#_Toc98943124)

[一、前端开发 6](#_Toc98943125)

[二、后端开发 8](#_Toc98943126)

[1、SpringBoot 8](#_Toc98943127)

[2、MyBatis-plus 9](#_Toc98943128)

[3、JWT 10](#_Toc98943129)

[4、PageHelper 11](#_Toc98943130)

[三、网络层 12](#_Toc98943131)

[四、持久化层 13](#_Toc98943132)

[第四部分 界面和业务单设计 14](#_Toc98943133)

[一、界面设计 14](#_Toc98943134)

[1、注册界面设计 14](#_Toc98943135)

[2、登录界面设计 15](#_Toc98943136)

[3、前台主页界面设计 16](#_Toc98943137)

[4、习题练习界面设计 17](#_Toc98943138)

[5、错题回顾界面设计 17](#_Toc98943139)

[6、在线考试界面设计 18](#_Toc98943140)

[7、成绩查询界面设计 20](#_Toc98943141)

[8、试卷管理界面设计 20](#_Toc98943142)

[9、教师管理界面设计 21](#_Toc98943143)

[10、数据分析界面设计 22](#_Toc98943144)

[11、个人页面界面设计 22](#_Toc98943145)

[12、学生管理界面设计 23](#_Toc98943146)

[13、任务管理界面设计 24](#_Toc98943147)

[二、业务单设计 24](#_Toc98943148)

[成绩导出表 24](#_Toc98943149)

[第五部分 单元模块设计 25](#_Toc98943150)

[一、后台功能 25](#_Toc98943151)

[1、接口设计 25](#_Toc98943152)

[1.1、ExamQuestionsController 25](#_Toc98943153)

[1.2、TeacherExamPaperController 26](#_Toc98943154)

[1.3、TrainCourseController 26](#_Toc98943155)

[1.4、TrainCourseMaterialsController 27](#_Toc98943156)

[1.5、TrainSubjectController 28](#_Toc98943157)

[1.6、TrainUserController 29](#_Toc98943158)

[1.7、UserCenterController 30](#_Toc98943159)

[2、功能描述 30](#_Toc98943160)

[2.1、用户注册登录 30](#_Toc98943161)

[2.2、用户功能 32](#_Toc98943162)

[2.3、学科功能 35](#_Toc98943163)

[2.4、试卷功能 37](#_Toc98943164)

[2.5、任务功能 41](#_Toc98943165)

[二、前台功能 42](#_Toc98943166)

[1、接口设计 42](#_Toc98943167)

[1.1、ExamPaperScoreController 42](#_Toc98943168)

[1.2、ExamAnswerPaperController 43](#_Toc98943169)

[1.3、StuExamController 43](#_Toc98943170)

[1.4、TrainSelectCourseController 44](#_Toc98943171)

[2、功能描述 44](#_Toc98943172)

[2.1、答题功能 44](#_Toc98943173)

[2.1、错题本功能 47](#_Toc98943174)

[三、配置文件与工具类 51](#_Toc98943175)

[1、application.properties配置文件 51](#_Toc98943176)

[2、TrainSysException异常处理类 52](#_Toc98943177)

[3、MyMetaObjectHandler日期类型自动填充配置类 53](#_Toc98943178)

[4、JwtUtils工具类 54](#_Toc98943179)

[5、MD5工具类 55](#_Toc98943180)

[6、R和ResultCode统一返回类型 56](#_Toc98943181)

[第六部分 数据库设计 58](#_Toc98943182)

[一、数据库整体结构图 58](#_Toc98943183)

[二、数据库表设计 58](#_Toc98943184)

[1、数据库表清单 58](#_Toc98943185)

[（1）student表结构 59](#_Toc98943186)

[（2）student\_course表结构 59](#_Toc98943187)

[（3）teacher表结构 59](#_Toc98943188)

[（4）examSolving表结构 60](#_Toc98943189)

[（5）paper表结构 60](#_Toc98943190)

[（6）questionSolving表结构 60](#_Toc98943191)

[（6）question表结构 61](#_Toc98943192)

[2、外键清单 61](#_Toc98943193)

[第七部分 补充设计和说明 62](#_Toc98943194)

[一、编译运行环境设计 62](#_Toc98943195)

[1、服务器 62](#_Toc98943196)

[2、客户机器环境 62](#_Toc98943197)

[3、开发环境 62](#_Toc98943198)

[二、包路径与WEB目录结构设计 62](#_Toc98943199)

# 第一部分 引言

## 一、编写目的

编写本文的主要目的是把需求分析得到的用例模型转换为软件结构和数据结构。设计软件结构的具体任务是：将一个复杂系统按功能进行模块划分、建立模块的层次结构及调用关系、确定模块间的接口及人机界面等。数据结构设计包括数据特征的描述、确定数据的结构特性、以及数据库的设计。

本设计是指导详细设计和项目实施的重要指导性文件，也是进行系统集成测试和重要依据。

## 二、项目背景

* 系统名称：在线考试系统
* 任务提出者：略。
* 开发者：李依玲、戴广艳、季猛达、谢伟成、吴超
* 用户和运行该程序系统的计算中心：略。

## 三、定义

### 1、J2EE四层架构

从构件化与集成化的角度出发，采用J2EE四层架构的技术方案、构件化的开发模式，应用功能的实现在支撑构件的基础上完成。

通过J2EE标准体系将所有的应用构件在JVM上，实现和OS的无关性，保证系统的跨平台、扩展性和安全性。

### 2、B/S架构

在B/S架构中，用户只需安装浏览器作为客户端，而将应用逻辑集中在服务器上，这样可以有效提高数据处理性能，减少系统开发和维护的成本。由于在线培训和考试系统受众范围广，使用B/S架构不需要安装额外的客户端，降低了用户的使用门槛，也有利于提高平台的可维护性。

### 3、前后端分离

在前后端分离的应用模式中，后端只对数据进行处理和存储，向前端返回所需的数据，不再涉及页面的开发；前端仅负责页面内容和样式的编写，并从后端接收数据，显示在页面上。这种模式有助于提高开发效率，降低软件的耦合度。由于在线培训和考试系统功能较多，业务较为复杂，所以采用前后端分离的模式进行设计和开发。

### 4、微服务架构

微服务架构将整个应用划分成各种小的、互相连接的微服务，一个微服务完成一个比较单一的功能， 相互之间保持独立和解耦合，提高了应用的可扩展性和技术的异构性。在线培训和考试系统使用微服务架构有利于降低系统的耦合程度，方便未来对业务功能进行扩展，并提高平台的设计和开发效率。

## 四、参考资料

属于本项目的其他已发表的文件；

《需求说明》《概要设计》

本文件中各处引用到的文件资料，包括所要用到的软件开发标准。

《Java语言编码规范(Java Code Conventions)》

# 第二部分 项目概述

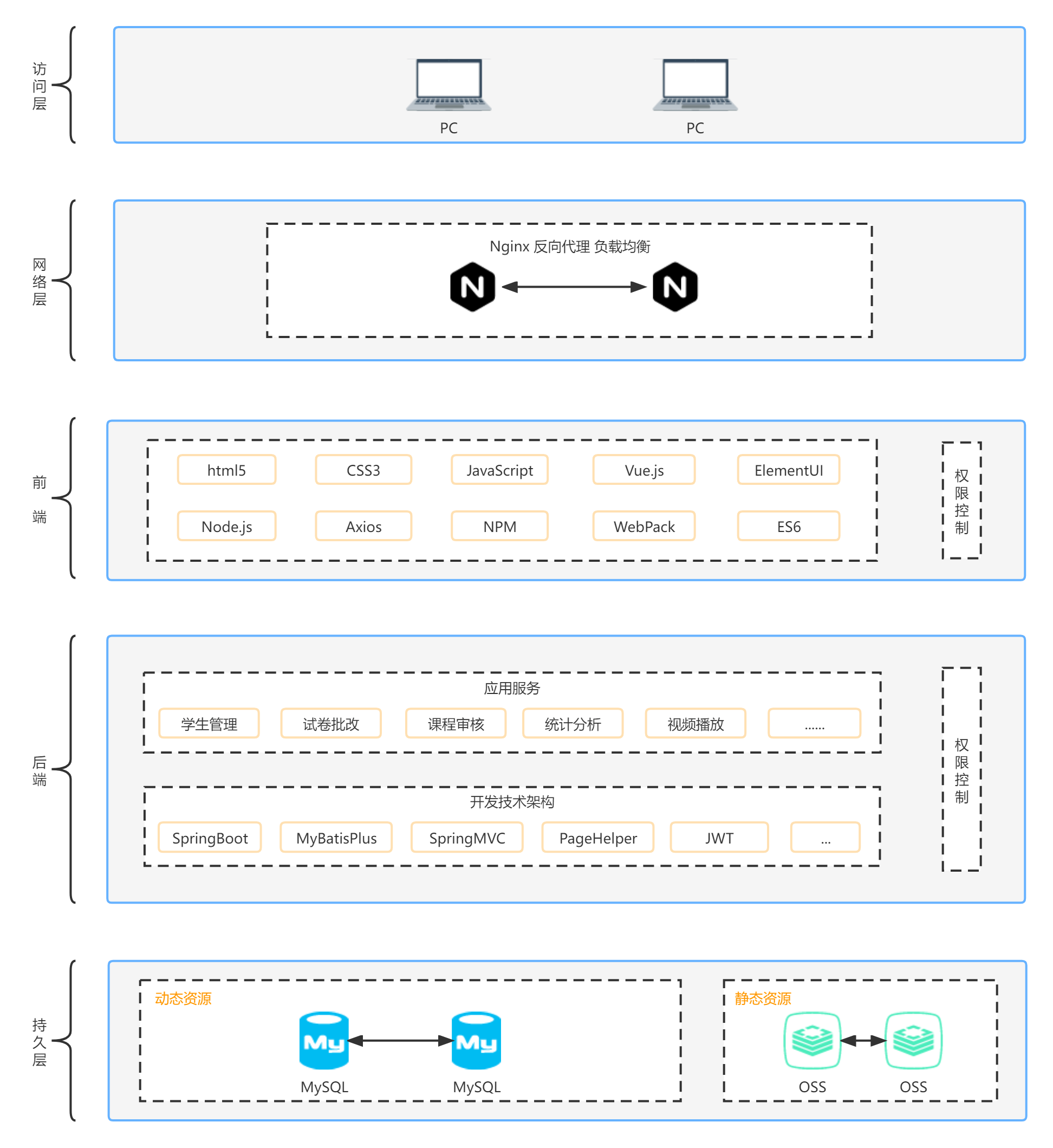
传统的学校教学存在老师固定，地点固定，人员固定等诸多缺陷，不方便知识的传播，交流以及沟通。传统的纸笔考试需要耗费大量的人力，物力资源，不利于生态文明建设。目前，随着IT行业的不断发展，越来越多的人来学习该行业相关课程，线下培训与考试灵活性较差，成本也比较高。新冠疫情的不断反复，也使得线下培训与考试愈发困难。

因此，人们对培训与考试的方式有了更高的要求。目前实现的针对IT行业实现的一套在线培训、视频学习、在线考试、在线闯关答题、员工积分等模块的综合性系统，可以拓展为全行业的培训、考试系统。

锁定智能化目标，利用软件开发技术改革教育行业，节约运行成本、提高服务质量、增加业务收益、拓展市场、更好地利用教育资源，立足建设规模化信息化在线培训考试的标杆。

# 第三部分 架构设计

在“王东南在线培训和考试系统”的开发中，将平台技术架构分为四个部分：网络层、前端、后端和持久化层。网络层负责连接用户操作和前端页面；前端负责网页页面布局和响应；后端负责逻辑处理和与存储层的交互；存储层负责对数据进行存储，并按照后端发来的命令进行数据操作。



## 一、前端开发

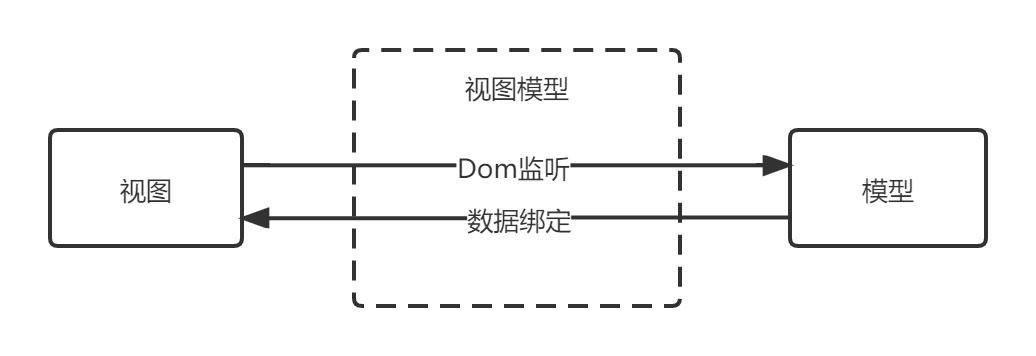
**（1）Vue定义**

Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

**（2）MVVM模式**

Vue框架基于MVVM模式进行设计，MVVM（Model-View-ViewModel）是一种软件架构设计模式，它是一种简化用户界面的事件驱动编程方式，它把软件系统分为三个部分，M是逻辑方法加上数据，V就是用户看到的界面，VM就是逻辑方法加上界面渲染的代码。

* View是视图层，也就是用户界面。前端主要由HTML和CSS来构成，为了更方便地展现ViewModel或者Model层的数据。
* Model是指数据模型，泛指后端进行的各种业务逻辑处理和数据操控，主要围绕数据库系统展开。这里的难点主要在于需要和前端约定统一的接口规则。
* ViewModel由前端开发人员组织生成和维护的视图数据层。在这一层，前端开发者从后端获取得到Model数据进行转换出来，做二次封装，以生成符合View层使用预期的视图数据模型。视图状态和行为都封装在ViewModel里。这样的封装使得ViewModel可以完整地去描述View层

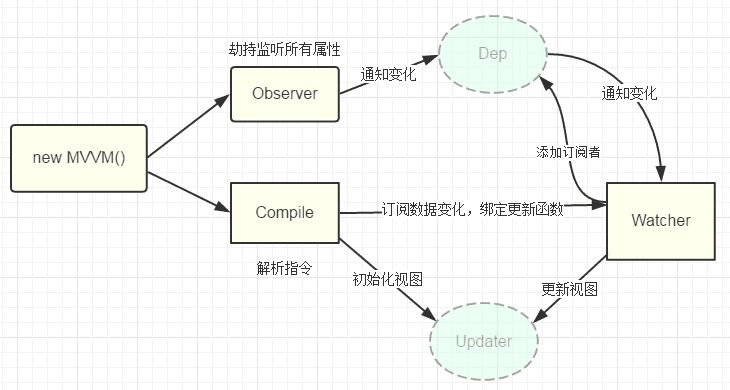


在MVVM架构中，是不允许数据和视图直接通信的，只能通过ViewModel来通信，而ViewModel就是定义了一个Observer观察者。ViewModel是连接View和Model的中间件。

* ViewModel能够观察到数据的变化，并对视图对应的内容进行更新。
* ViewModel能够监听到视图的变化，并能够通知数据发生变化。

**（3）双向绑定原理**

Vue.js基于MVVM模式的，核心为实现了DOM监听与数据绑定。Vue.js 是采用Object.defineProperty 的 getter 和 setter，并结合观察者模式来实现数据绑定的。当把一个普通 Javascript 对象传给 Vue 实例来作为它的 data 选项时，Vue 将遍历它的属性，用 Object.defineProperty 将它们转为 getter/setter。用户看不到 getter/setter，但是在内部它们让 Vue 追踪依赖，在属性被访问和修改时通知变化。



* **Observer** 数据监听器，能够对数据对象的所有属性进行监听，如有变动可拿到最新值并通知订阅者，内部采用Object.defineProperty的getter和setter来实现。
* **Compile** 指令解析器，它的作用对每个元素节点的指令进行扫描和解析，根据指令模板替换数据，以及绑定相应的更新函数。
* **Watcher** 订阅者， 作为连接 Observer 和 Compile 的桥梁，能够订阅并收到每个属性变动的通知，执行指令绑定的相应回调函数。
* **Dep** 消息订阅器，内部维护了一个数组，用来收集订阅者（Watcher），数据变动触发notify 函数，再调用订阅者的 update 方法。

## 二、后端开发

### 1、SpringBoot

Spring Boot是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式(继承starter，约定优先于配置)来进行配置，不再需要定义样板化的配置。Spring Boot并不是一个框架，从根本上将，它就是一些库的集合，maven或者gradle项目导入相应依赖即可使用Spring Boot，而且无需自行管理这些库的版本。

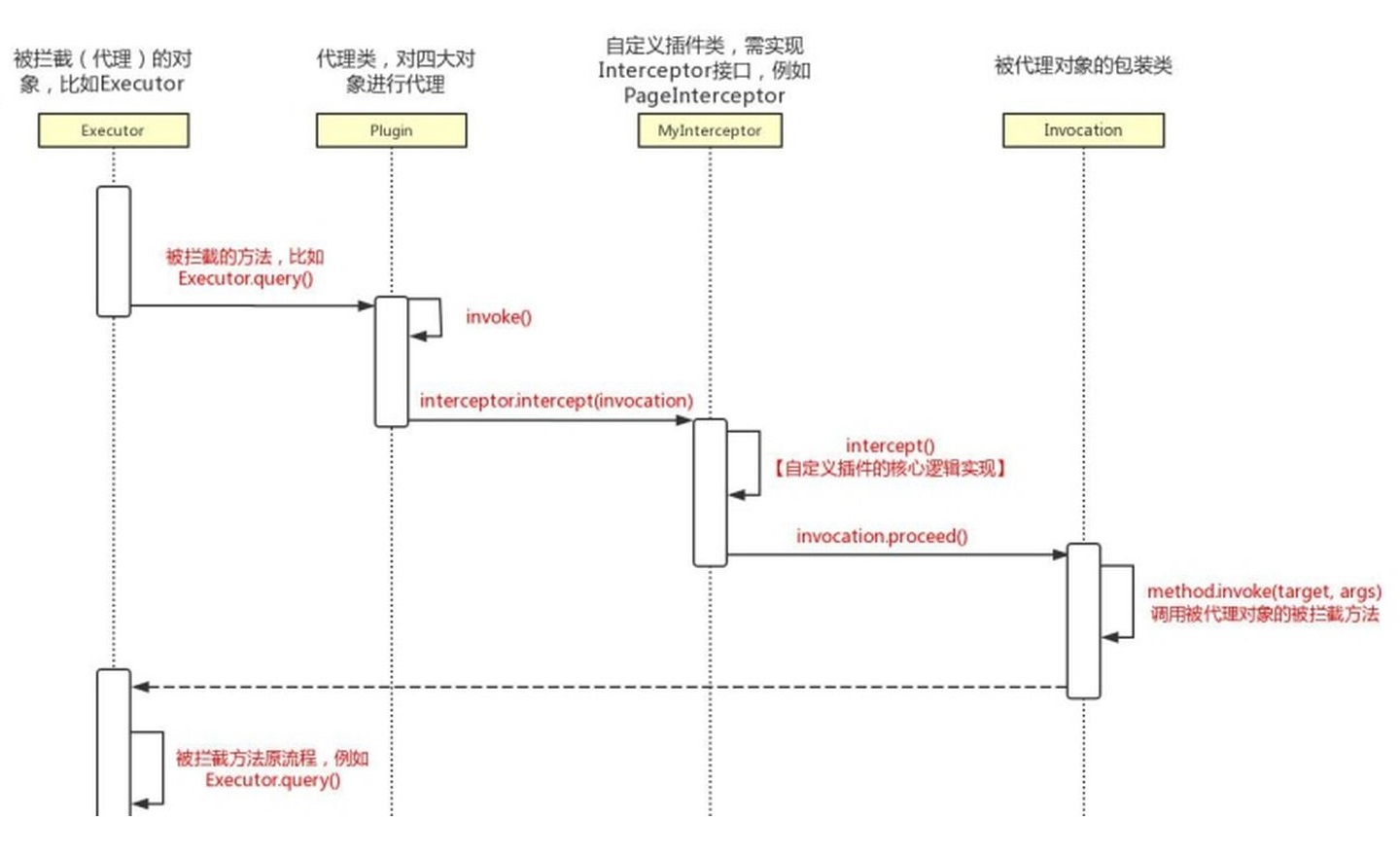
在线培训和考试系统设计为一个SpringBoot 的项目，基于Maven项目管理工具，快速构建Spring的配置和环境，专注于开发微服务后台接口。其中，SpringBoot每个微服务都包括四个层面：实体层Entity、持久层Mapper、业务层Service和控制层Controller。

* 实体层：包含数据库关系表在项目中的JavaBean类；
* 持久层：负责与数据库进行交互，调用实体层完成增删改查操作；
* 业务层：负责业务模块的逻辑应用设计，会调用持久化层接口，完成项目的基本功能；
* 控制层：对业务流程进行控制，调用业务层完成前后端的交互。



### 2、MyBatis-plus

MyBatis-plus是对MyBatis的功能增强，不仅能完成常规的增删改查操作，还能实现数据的自动填充、逻辑删除和自动分页等。在本项目，除了以上功能，还会使用MyBatis-plus的代码生成器快速生成Mapper、Entity、Service和Controller等模块的代码，简化系统的开发。



MyBatis Plus提供了分页插件PaginationInterceptor、执行分析插件SqlExplainInterceptor、性能分析插件PerformanceInterceptor以及乐观锁插件OptimisticLockerInterceptor。Mybatis 通过插件 (Interceptor) 可以做到拦截四大对象（Executor、StatementHandler、ParameterHandler和ResultSetHandler）相关方法的执行 ,根据需求完成相关数据的动态改变。

四大对象的每个对象在创建时，都会执行interceptorChain.pluginAll()，会经过每个插件对象的 plugin()方法，目的是为当前的四大对象创建代理。代理对象就可以拦截到四大对象相关方法的执行，因为要执行四大对象的方法需要经过代理 。

### 3、JWT

JWT是用于身份验证的token的一种具体实现方式，其本质为一个字符串，将用户信息保存到一个Json字符串中，然后进行编码后得到一个JWT token，并且这个JWT token带有签名信息，接收后可以校验是否被篡改，用于在各方之间安全地将信息作为Json对象传输。

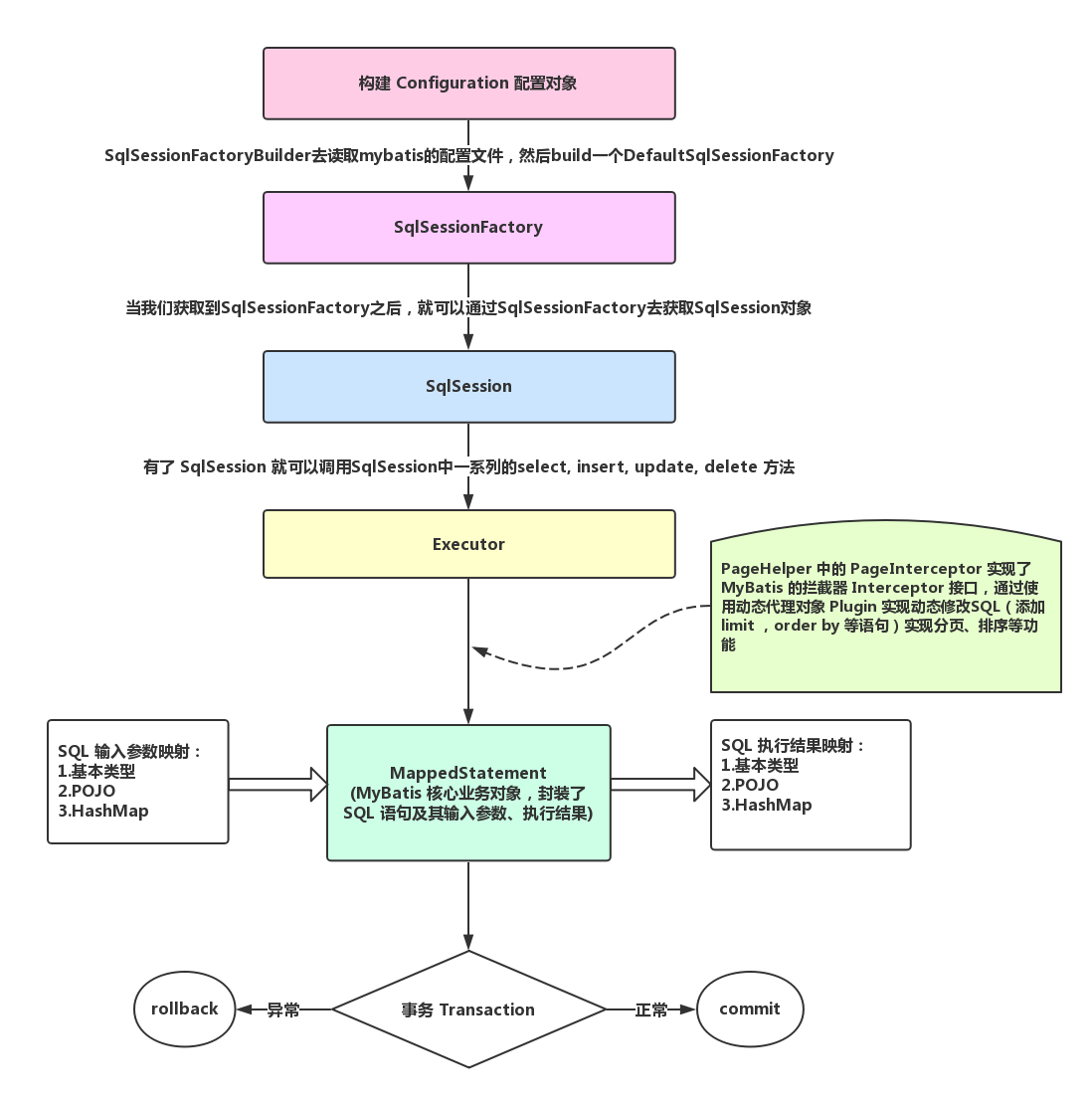
JWT的认证流程如下：

1. 首先，前端通过Web表单将自己的用户名和密码发送到后端的接口，这个过程一般是一个POST请求。建议的方式是通过SSL加密的传输(HTTPS)，从而避免敏感信息被嗅探；
2. 后端核对用户名和密码成功后，将包含用户信息的数据作为JWT的Payload，将其与JWT Header分别进行Base64编码拼接后签名，形成一个JWT Token；
3. 后端将JWT Token字符串作为登录成功的结果返回给前端。前端可以将返回的结果保存在浏览器中，退出登录时删除保存的JWT Token即可；
4. 前端在每次请求时将JWT Token放入HTTP请求头中的Authorization属性中；
5. 后端检查前端传过来的JWT Token，验证其有效性，比如检查签名是否正确、是否过期、token的接收方是否是自己等；
6. 验证通过后，后端解析出JWT Token中包含的用户信息，进行其他逻辑操作(一般是根据用户信息得到权限等)，返回结果。

### 4、PageHelper

PageHelper是MyBatis的一个插件，其作用是更加方便地进行分页查询，简化如总数、每页数量、当前页数、上一页或下一页等查询操作，简化开发效率。

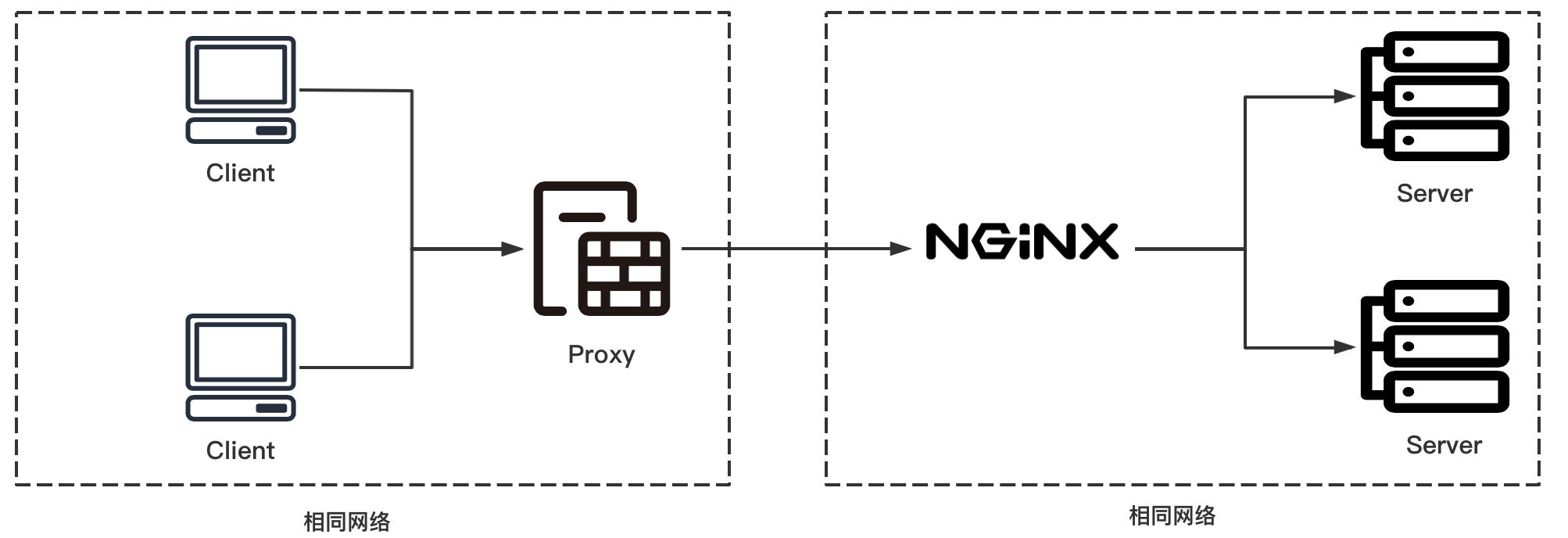
PageHelper方法使用了静态的ThreadLocal参数，分页参数和线程是绑定的。内部流程是ThreadLocal中设置了分页参数（pageIndex，pageSize），之后在查询执行的时候，获取当线程中的分页参数，执行查询的时候通过拦截器在sql语句中添加分页参数，之后实现分页查询，查询结束后在 finally 语句中清除ThreadLocal中的查询参数。



## 三、网络层

Nginx是一个高性能的HTTP和反向代理web服务器，其特点是占有内存少，并发能力强，具有反向代理、负载均衡等功能。

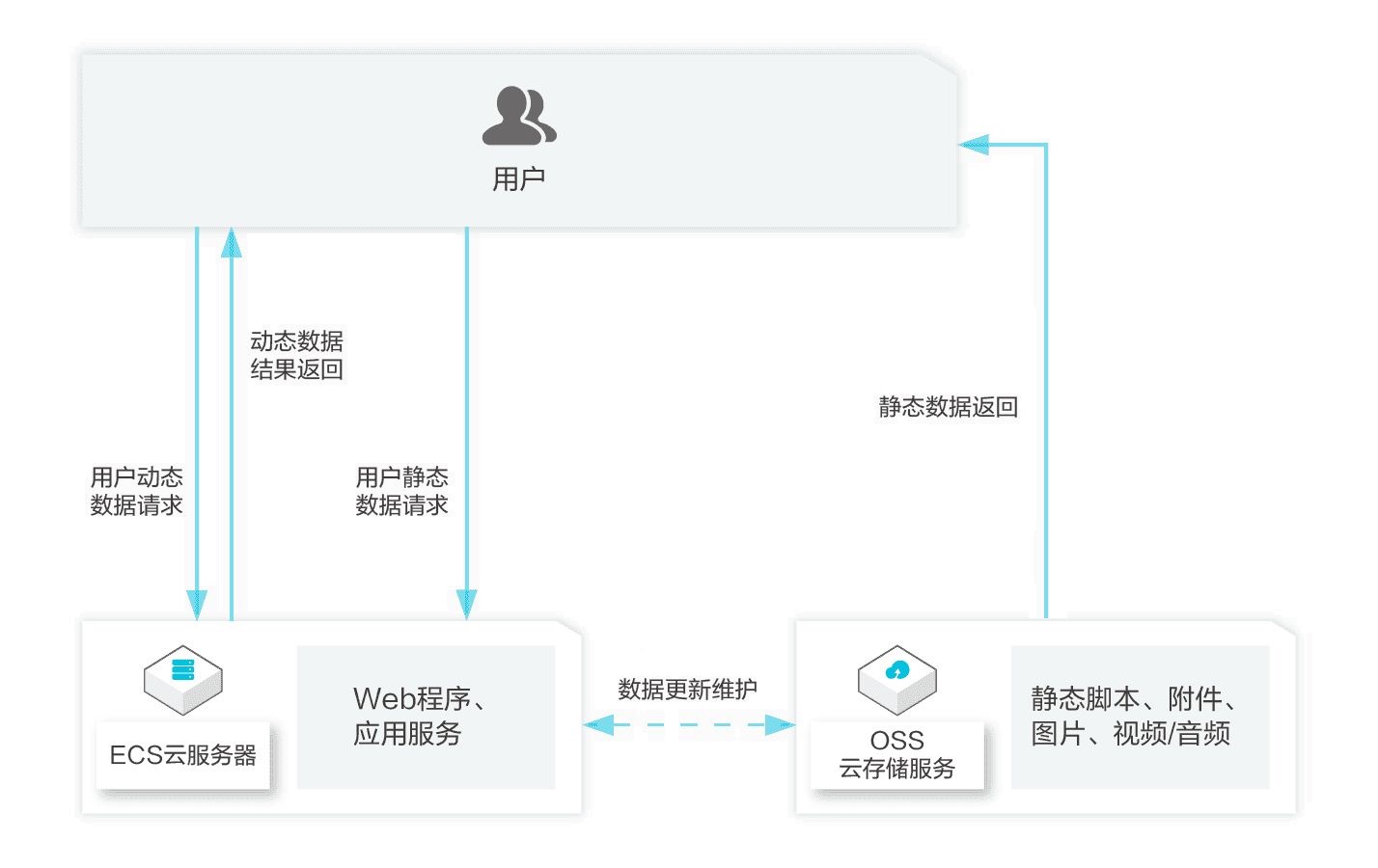
* 反向代理：将多台服务器代理成一台服务器，客户端不需要任何配置就可以访问，只需将请求发送到反向代理服务器，由反向代理服务器去选择目标服务器获取数据后，在返回给客户端；
* 负载均衡：将多个请求均匀的分配到多台服务器上，减轻每台服务器的压力，提高服务的吞吐量；



## 四、持久化层

在“王东南在线培训和考试系统”中使用MySQL对普通数据进行存储，使用阿里云OSS数据库对课程的视频资料和文档资料进行存储。

用户发送动态数据请求时，Web网页与应用接口交互，访问MySQL数据库，将结果直接返回；用户发送静态数据请求时，应用服务与云存储服务进行通讯，完成数据的更新维护，OSS云存储服务将用户请求的静态资源进行返回。



阿里云对象存储OSS是一款海量、安全、低成本、高可靠的云存储服务，OSS具有与平台无关的RESTful API接口，您可以在任何应用、任何时间、任何地点存储和访问任意类型的数据。可以使用阿里云提供的API、SDK接口或者OSS迁移工具轻松地将海量数据移入或移出阿里云OSS。

# 第四部分 界面和业务单设计

## 一、界面设计

### 1、注册界面设计

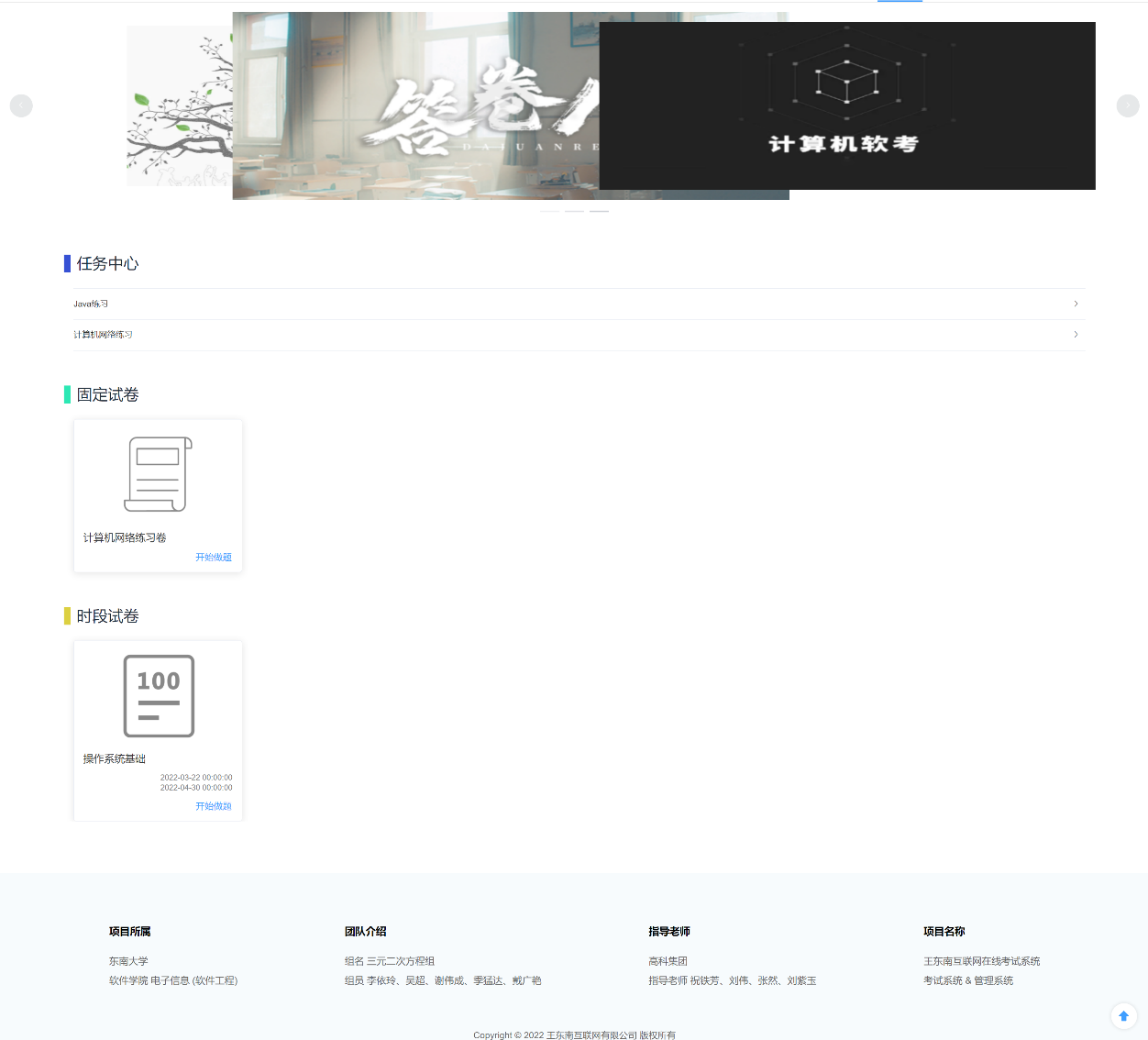


### 2、登录界面设计

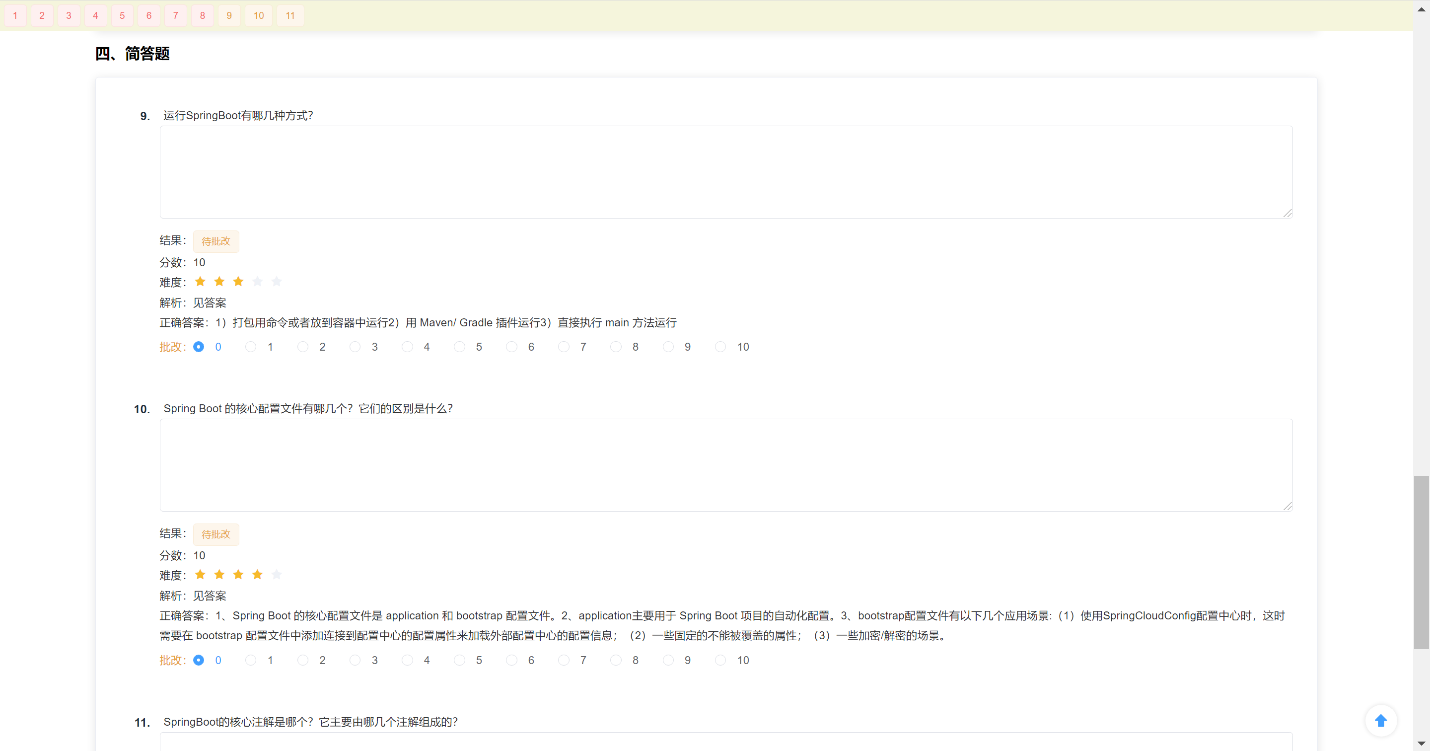




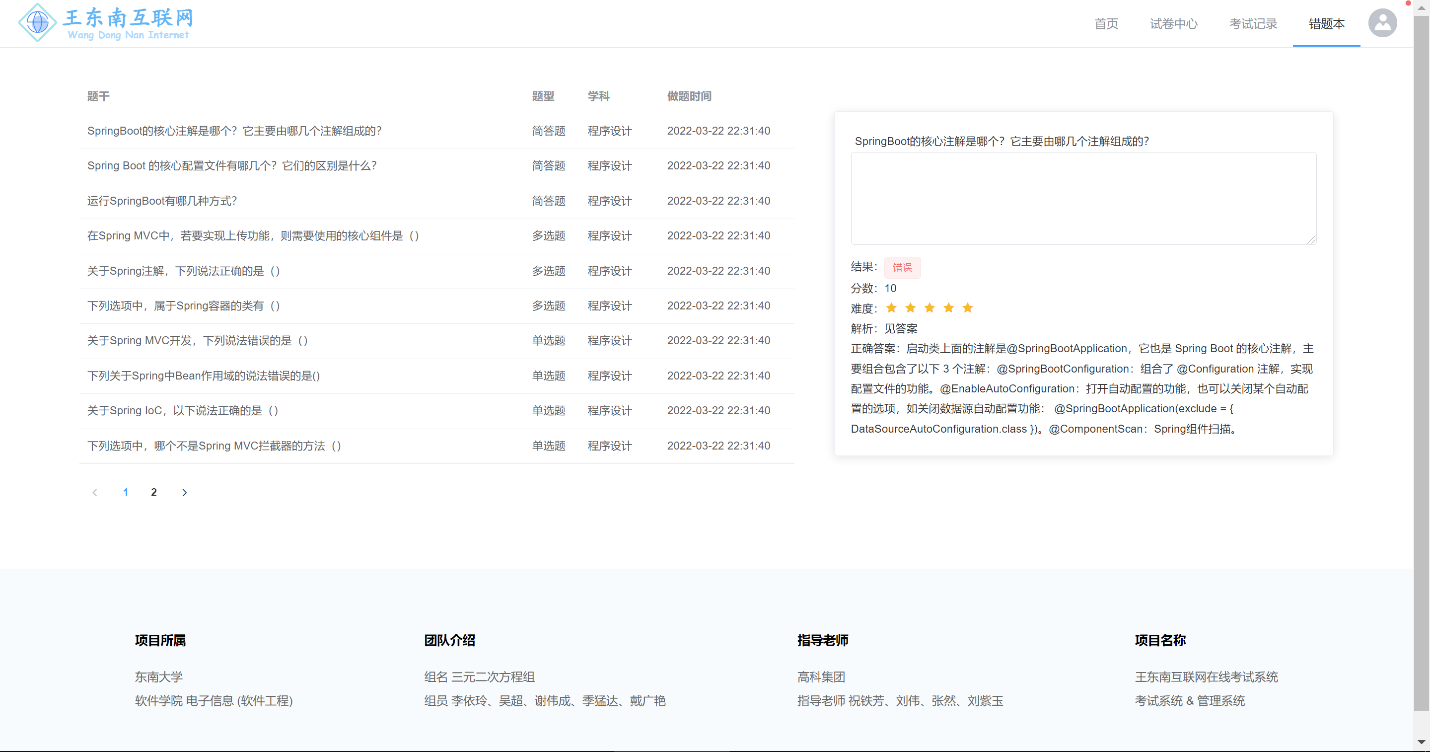
### 3、前台主页界面设计



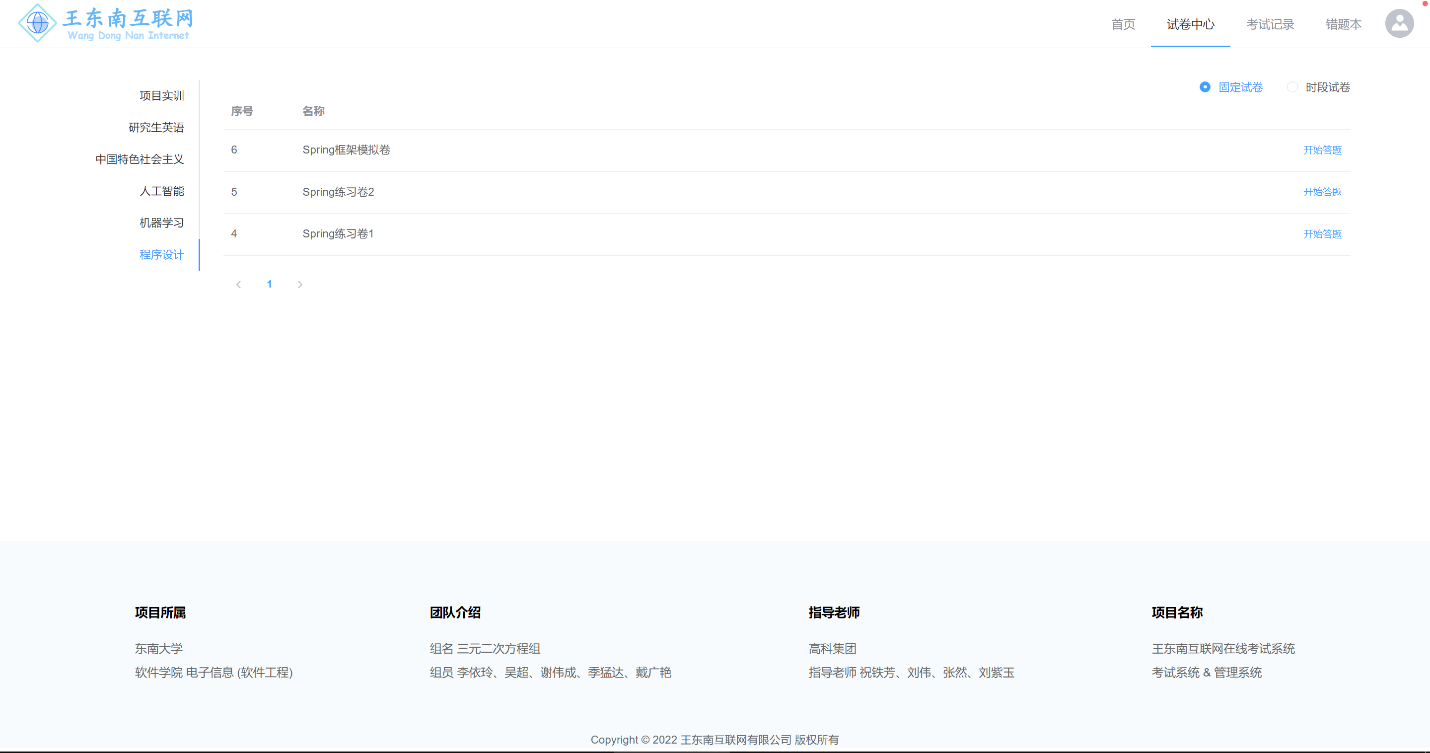
### 4、习题练习界面设计



### 5、错题回顾界面设计

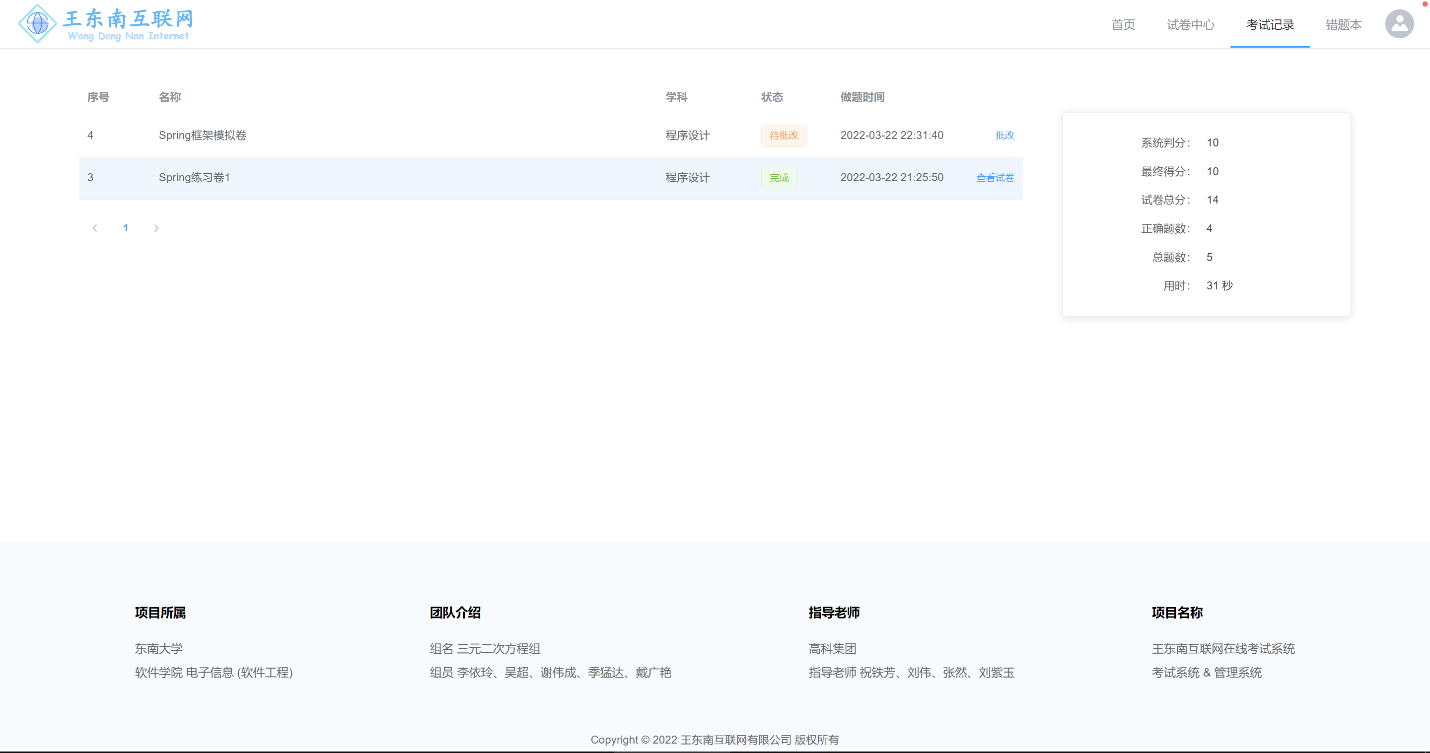


### 6、在线考试界面设计



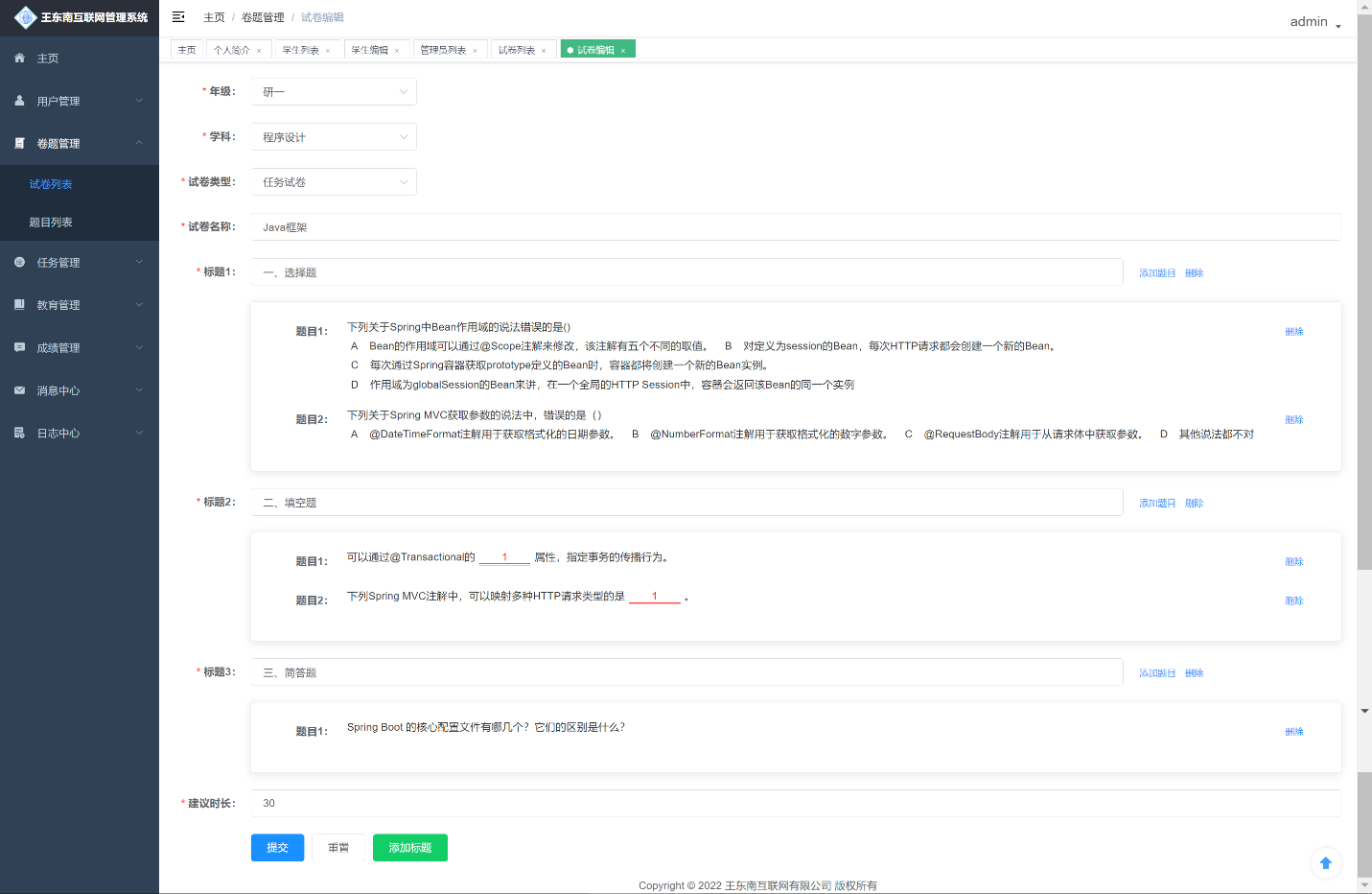


### 7、成绩查询界面设计

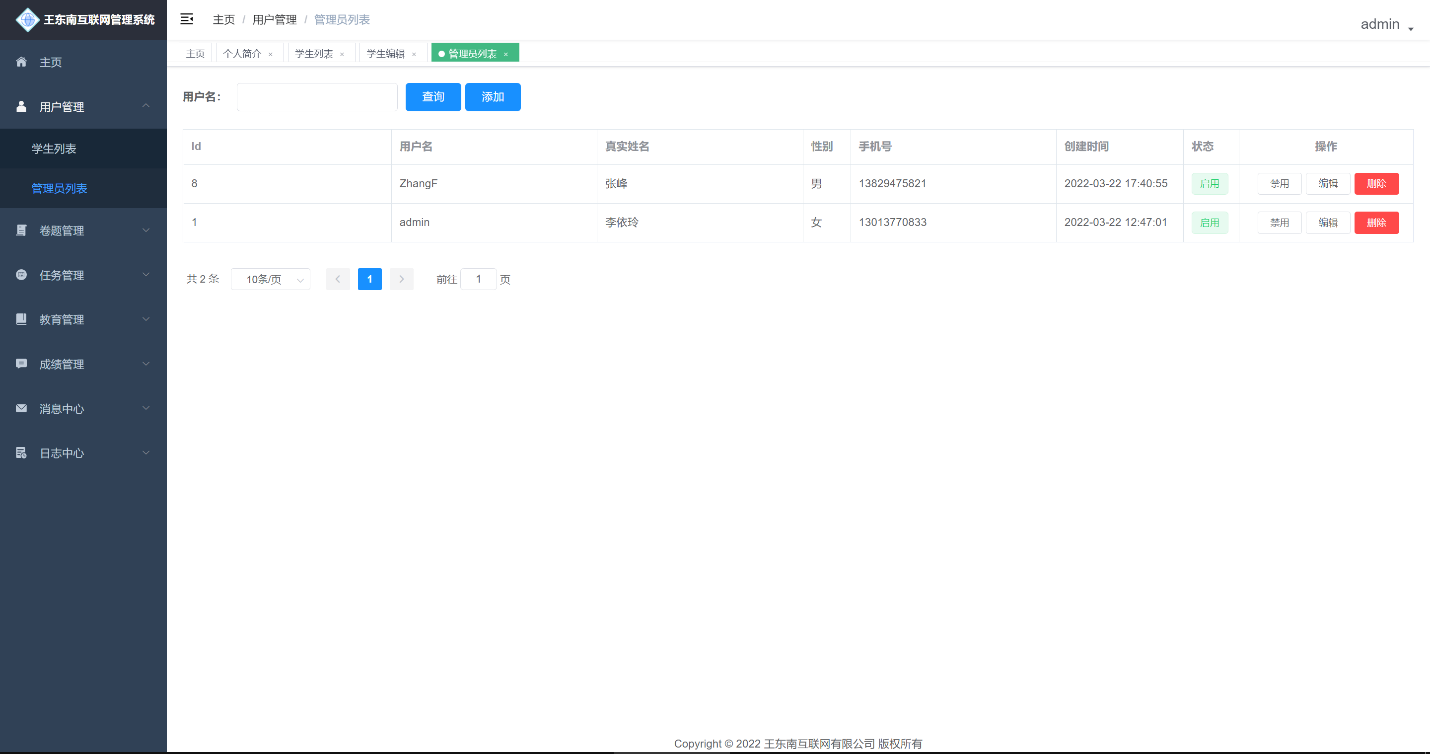


### 8、试卷管理界面设计





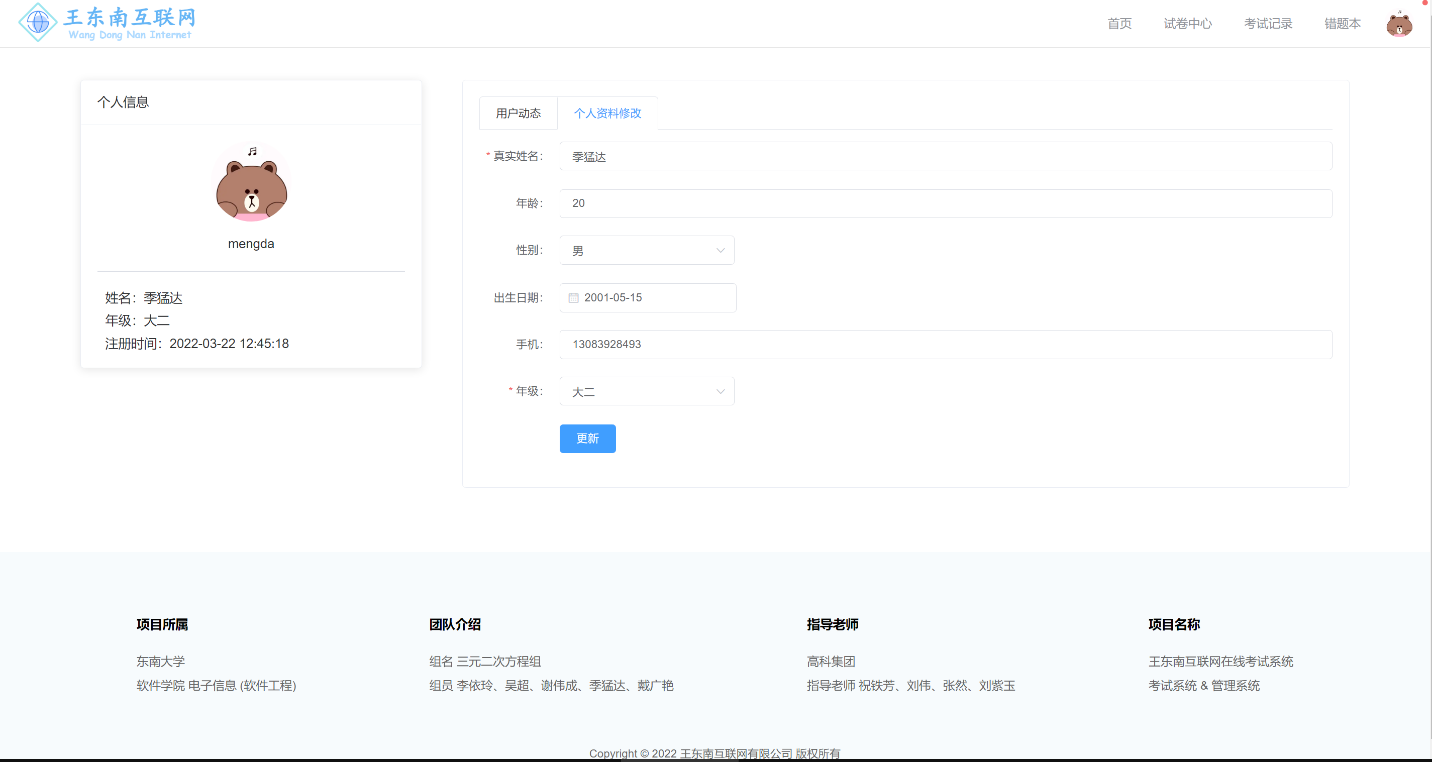
### 9、教师管理界面设计



### 10、数据分析界面设计

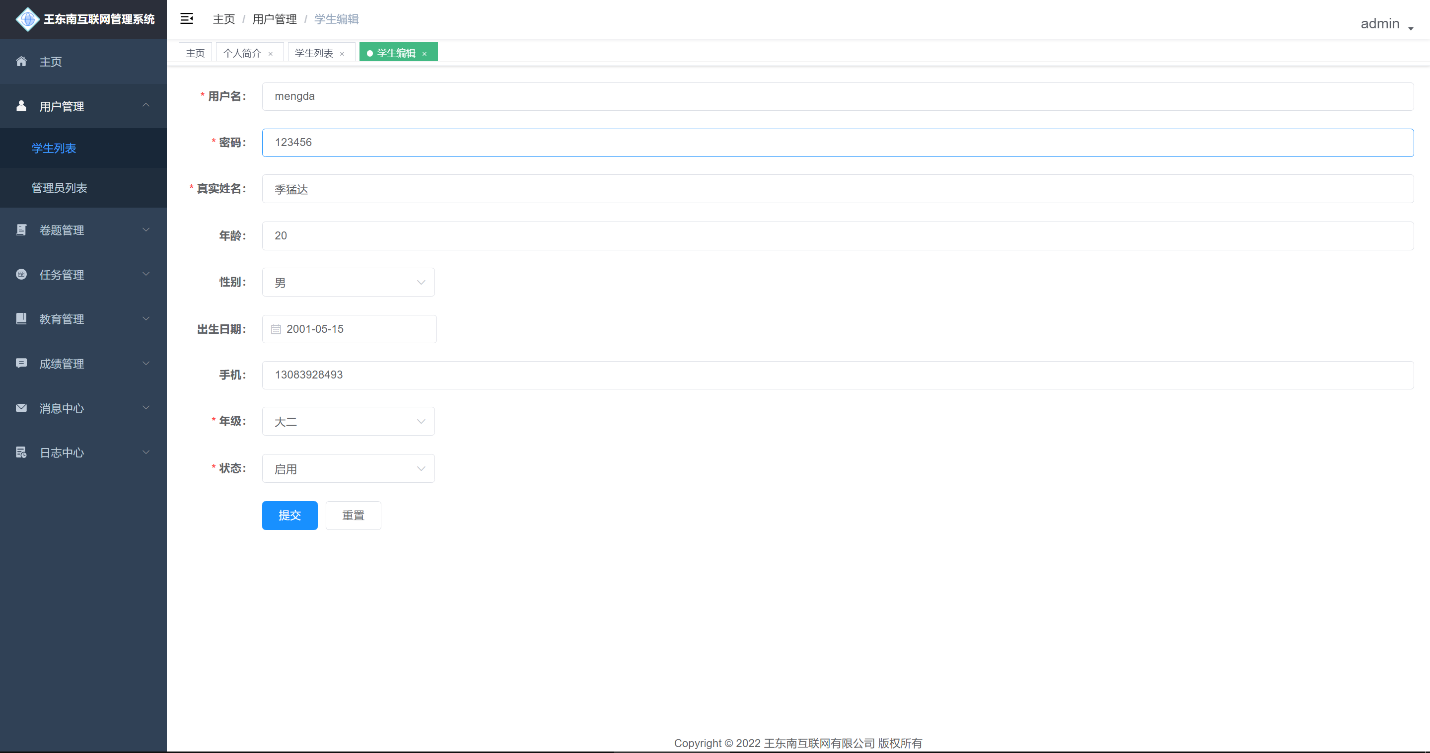


### 11、个人页面界面设计

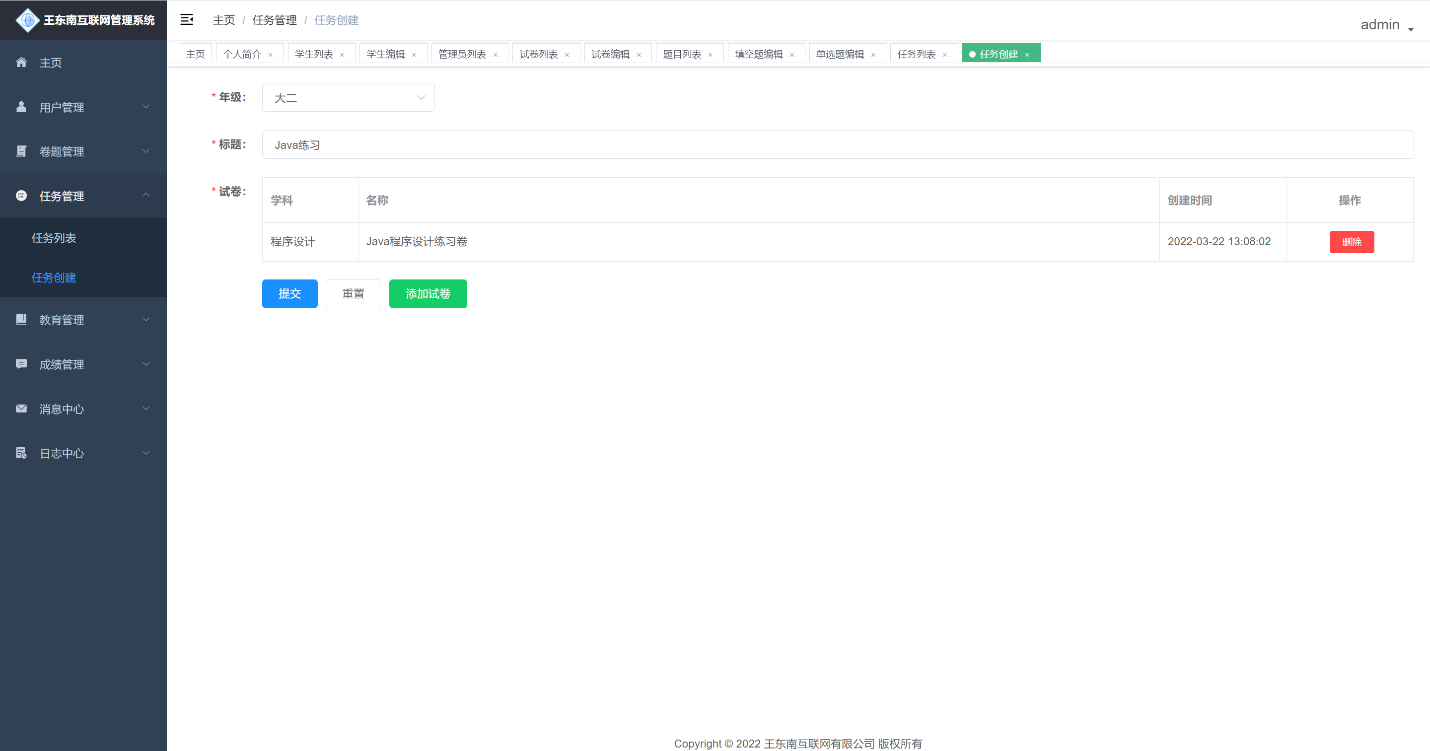




### 12、学生管理界面设计



### 13、任务管理界面设计



## 二、业务单设计

### 成绩导出表



# 第五部分 单元模块设计

# 一、后台功能

## 1、接口设计

### 1.1、ExamQuestionsController

|  |
| --- |
| ExamQuestionsController |
| * addQuestion() * selectQuestion() * deleteQuestion() * updateQuestion() * pageQuestionCondition() |
| ExamQuestionsController教师试题管理接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R addQuestion(@RequestBody ExamQuestions examQuestions);  添加试题 |
| public R selectQuestion(@PathVariable Integer questionid);  查询试题 |
| public R deleteQuestion(@PathVariable Integer questionid);  删除试题 |
| public R updateQuestion(@RequestBody ExamQuestions examQuestions);  更新试题 |
| public R pageQuestionCondition(@PathVariable Integer userid, @PathVariable long current, @PathVariable long limit,@RequestBody(required = false) QuestionQuery questionQuery);  题目条件分页 |

### 1.2、TeacherExamPaperController

|  |
| --- |
| TeacherExamPaperController |
| * selectPaper() * addPaper() * updatePaper() * deletePaper() * pagePaperCondition() |
| TeacherExamPaperController教师试卷管理接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R selectPaper(@PathVariable Integer id);  查询试卷 |
| public R addPaper(@RequestBody ExamPaper examPaper);  添加试卷 |
| public R updatePaper(@RequestBody ExamPaper examPaper);  更新试卷 |
| public R deletePaper(@PathVariable Integer id);  删除试卷 |
| public R pagePaperCondition(@PathVariable Integer userid,@PathVariable long current,@PathVariable long limit,@RequestBody(required = false) ExamPaperQuery examPaperQuery);  试卷条件分页 |

### 1.3、TrainCourseController

|  |
| --- |
| TrainCourseController |
| * pageCourseCondition() * deleteCourse() * getCourseById() * addCourse() * updateCourse() |
| TrainCourseController教师课程管理接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R pageCourseCondition(@PathVariable Integer current, @PathVariable Integer limit, @RequestBody(required = false) CourseQuery CourseQuery);  课程条件分页 |
| public R deleteCourse(@PathVariable Integer id);  删除课程 |
| public R getCourseById(@PathVariable Integer id);  查询课程 |
| public R addCourse(@RequestBody TrainCourse trainCourse);  添加课程 |
| public R updateCourse(@RequestBody TrainCourse trainCourse);  修改课程 |

### 1.4、TrainCourseMaterialsController

|  |
| --- |
| TrainCourseMaterialsController |
| * getCourseMaterials() * addCourseMaterials() * deleteCourseMaterials() |
| TrainCourseMaterialsController课程附加信息管理控制接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R getCourseMaterials(@PathVariable Integer id);  根据id查询课程附加信息 |
| public R addCourseMaterials(@RequestBody TrainCourseMaterials trainCourseMaterials);  添加课程附加信息 |
| public R deleteCourseMaterials(@PathVariable Integer id);  删除课程附加信息 |

### 1.5、TrainSubjectController

|  |
| --- |
| TrainSubjectController |
| * findAllCondition() * deleteSubject() * getSubjectById) * addSubject() * updateSubject() |
| TrainSubjectController教师学科管理接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R findAllCondition(@PathVariable Integer current, @PathVariable Integer limit, @RequestBody(required = false) SubjectQuery subjectQuery);  分页查询 |
| public R deleteSubject(@PathVariable Integer id);  删除学科 |
| public R getSubjectById(@PathVariable Integer id);  查询学科 |
| public R addSubject(@RequestBody TrainSubject trainSubject);  添加学科 |
| public R updateSubject(@RequestBody TrainSubject trainSubject);  更新学科 |

### 1.6、TrainUserController

|  |
| --- |
| TrainUserController |
| * pageUserCondition() * getUser() * addUser() * removeUser() * updateUser() |
| TrainUserController用户管理控制接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R pageUserCondition(@PathVariable long current,@PathVariable int limit,@RequestBody(required = false) UserQuery userQuery);  条件查询分页 |
| public R getUser(@PathVariable String id);  根据用户id进行查询 |
| public R addUser(@RequestBody TrainUser trainuser);  添加用户 |
| public R removeUser(  @ApiParam(name = "id", value = "用户ID", required = true)  @PathVariable String id);  逻辑删除用户 |
| public R updateUser(@RequestBody TrainUser trainuser);  更新用户 |

### 1.7、UserCenterController

|  |
| --- |
| UserCenterController |
| * loginUser() * registerUser() * getUserInfo() |
| UserCenterController用户登录管理接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R loginUser(@RequestBody TrainUser trainUser);  用户登录管理 |
| public R registerUser(@RequestBody TrainUser trainUser);  用户登录管理 |
| public R getUserInfo(@RequestParam(value="token") String token);  获取用户信息 |

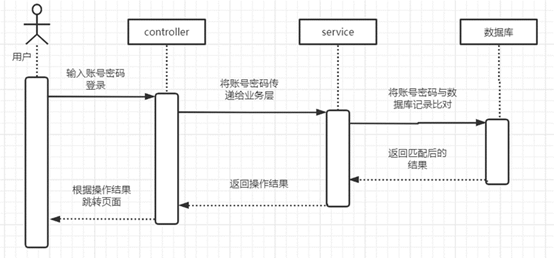
## 2、功能描述

### 2.1、用户注册登录

**（1）功能汇总**

|  |  |
| --- | --- |
| 功能 | 描述 |
| 登录 | 判断用户名和密码是否与数据库一致；若一致，生成token返回 |
| 获取用户信息 | 根据传入的token获取用户信息 |

**（2）时序图**



**（3）类详细描述**

**UserCenterController登录流程的控制类**

url路径：/train/login

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | userCenterService | @Autowired | UserCenterService类 |
| 方法 | loginUser | @PostMapping | 调用userCenterService中的login方法，成功则返回token |
| 方法 | getUserInfo | @PostMapping | 调用JwtUtils中getUserIdByJwtToken获取用户id，调用userCenterService的getById方法，获取TrainUser信息并返回 |

**UserCenterService接口**

继承IService方法，定义login方法，返回String类型

**UserCenterServiceImpl类**

继承ServiceImpl方法，实现UserCenterService接口

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | userCenterMapper | @Autowired | UserCenterMapper类 |
| 方法 | loginUser | @Override | 判断传入的phone和密码是否与数据库一致；若一致，则调用JwtUtils中getJwtToken方法获取Token信息并返回 |

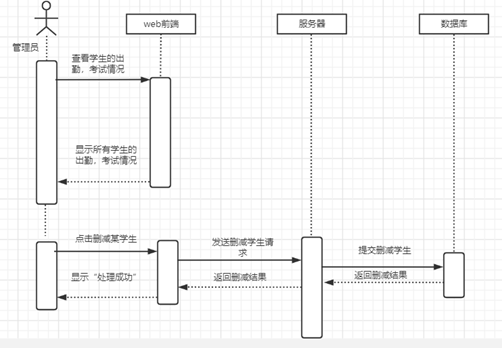
### 2.2、用户功能

**（1）功能汇总**

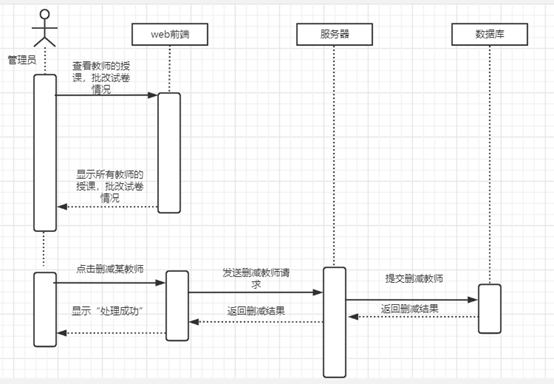
|  |  |
| --- | --- |
| 功能 | 描述 |
| 根据id查询用户信息 | 根据传入的id获取用户信息并返回 |
| 根据条件查询并分页 | 根据传入的查询条件（无条件、姓名、角色、创建时间、更新时间），获取用户记录，并根据当前页和每页记录数进行分页并返回 |
| 添加用户 | 根据传入的信息，新增用户记录 |
| 修改用户 | 根据传入的信息，修改用户记录 |
| 删除用户 | 根据传入的用户id，删除对应的用户记录 |

**（2）时序图**

学生管理：



教师管理：



**（3）类详细描述**

**TrainUserController用户管理控制类**

url路径：/train/user

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | trainUserService | @Autowired | trainUserService类 |
| 方法 | pageUser-Condition | @PostMapping | trainUserService调用trainUserService中的page方法，传入查询条件，返回分页后的数据和总页数 |
| 方法 | getUser | @GetMapping | 调用trainUserService的getById方法，传入id值，获取用户信息并返回 |
| 方法 | addUser | @PostMapping | 调用trainUserService的save方法，传入用户信息，返回方法结果的布尔值 |
| 方法 | updateUser | @PostMapping | 调用trainUserService的updateById方法，传入用户信息，返回方法结果的布尔值 |
| 方法 | deleteUser | @DeleteMapping | 调用trainUserService的removeById方法，传入用户信息，返回方法结果的布尔值 |

**TrainUserService接口**

继承IService方法，定义getTeacherList方法，传入Page<TrainUser>对象，返回Map<String, Object>对象

**TrainUserServiceImpl类**

继承ServiceImpl方法，实现TrainUserService接口

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | trainUserMapper | @Autowired | TrainUserMapper类 |
| 方法 | getTeacherList | @Override | 调用baseMapper的selectPage方法，传入筛选条件，获取分页信息（数据、当前页码、每页记录数、上一页、下一页、总页数等）并返回 |

### 2.3、学科功能

**（1）功能汇总**

|  |  |
| --- | --- |
| 功能 | 描述 |
| 根据id查询学科信息 | 根据传入的id获取学科信息并返回 |
| 根据条件查询并分页 | 根据传入的查询条件（无条件、一级学科），获取学科记录，并根据当前页和每页记录数进行分页并返回 |
| 添加学科 | 根据传入的信息，新增学科记录 |
| 修改学科 | 根据传入的信息，修改学科记录 |
| 删除学科 | 根据传入的学科id，删除对应的学科记录 |

**（2）类详细描述**

**TrainSubjectController用户管理控制类**

url路径：/train/subject

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | trainSubjectService | @Autowired | trainSubjectService类 |
| 方法 | pageSubject-Condition | @PostMapping | trainSubjectService调用trainSubjectService中的page方法，传入查询条件，返回分页后的数据和总页数 |
| 方法 | getSubject | @GetMapping | 调用trainSubjectService的getById方法，传入id值，获取学科信息并返回 |
| 方法 | addSubject | @PostMapping | 调用trainSubjectService的save方法，传入学科信息，返回方法结果的布尔值 |
| 方法 | updateSubject | @PostMapping | 调用trainSubjectService的updateById方法，传入学科信息，返回方法结果的布尔值 |
| 方法 | deleteSubject | @DeleteMapping | 调用trainSubjectService的removeById方法，传入用学科信息，返回方法结果的布尔值 |

**TrainSubjectService接口**

继承IService方法，定义getSubjectList方法，传入Page<TrainSubject>对象，返回Map<String, Object>对象

**TrainSubjectServiceImpl类**

继承ServiceImpl方法，实现TrainSubjectService接口

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | trainSubject-Mapper | @Autowired | TrainSubjectMapper类 |
| 方法 | getSubjectList | @Override | 调用baseMapper的selectPage方法，传入筛选条件，获取分页信息（数据、当前页码、每页记录数、上一页、下一页、总页数等）并返回 |

### 2.4、试卷功能

**（1）功能汇总**

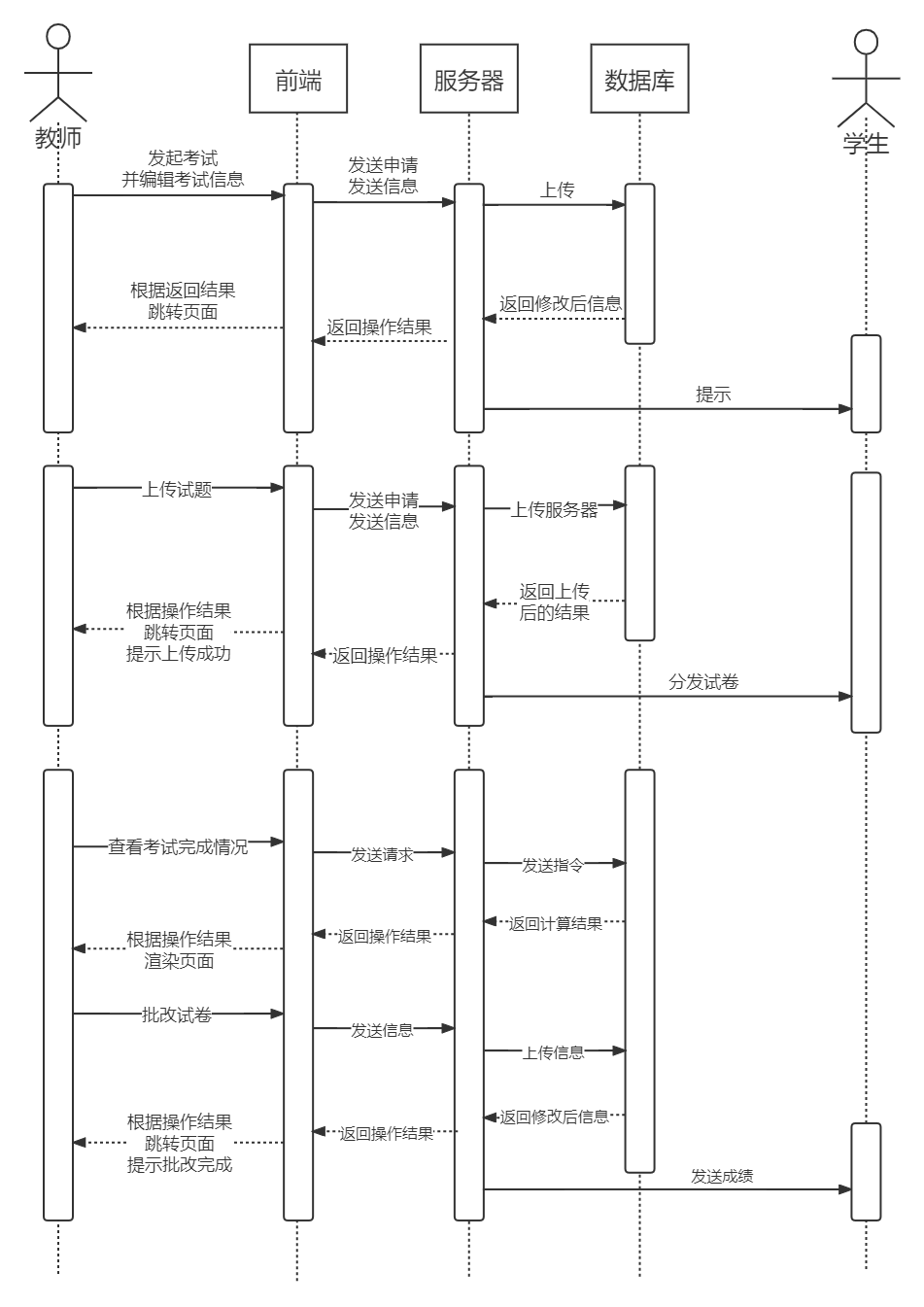
试卷功能：

|  |  |
| --- | --- |
| 功能 | 描述 |
| 根据id查询试卷信息 | 根据传入的id获取试卷信息，根据试卷的题目号字符串查询试卷包括的所有题目，返回试卷和题目信息 |
| 根据条件查询并分页 | 根据传入的查询条件（无条件、所属课程、试卷类型、试卷名称），获取试卷记录，并根据当前页和每页记录数进行分页并返回 |
| 添加试卷 | 根据传入的信息，新增试卷记录 |
| 修改试卷 | 根据传入的信息，修改试卷记录 |
| 删除试卷 | 根据传入的时间id，删除对应的试卷记录 |

题目功能：

|  |  |
| --- | --- |
| 功能 | 描述 |
| 根据id查询题目信息 | 根据传入的id获取题目信息并返回 |
| 根据条件查询并分页 | 根据传入的查询条件（无条件、所属课程、题目类型、开始时间、截止时间），获取题目记录，并根据当前页和每页记录数分页并返回 |
| 添加题目 | 根据传入的信息，新增题目记录 |
| 修改题目 | 根据传入的信息，修改题目记录 |
| 删除题目 | 根据传入的题目id，删除对应的题目记录 |

**（2）时序图**



**（3）类详细描述**

**ExamPaperController试卷管理控制类**

url路径：/train/paper

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | examPaper-Service | @Autowired | ExamPaperService类 |
| 属性 | examQuestionService | @Autowired | ExamQuestionService类 |
| 方法 | pagePaper-Condition | @PostMapping | examPaperService调用examPaperService中的page方法，传入查询条件，返回分页后的数据和总页数 |
| 方法 | getPaper | @GetMapping | 调用examPaperService的getById方法，传入id值，获取课程信，根据题目id字符串获取题号，调用examPaperService的getById方法，将试卷和题目信息一起返回 |
| 方法 | addPaper | @PostMapping | 调用examPaperService的save方法，传入试卷信息，返回方法结果的布尔值 |
| 方法 | updatePaper | @PostMapping | 调用examPaperService的updateById方法，传入试卷信息，返回方法结果的布尔值 |
| 方法 | deletePaper | @DeleteMapping | 调用examPaperService的removeById方法，传入试卷信息，返回方法结果的布尔值 |

**ExamQuestionController题目管理控制类**

url路径：/train/question

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | examQuesion-Service | @Autowired | ExamQuestionService类 |
| 方法 | pageQuestion-Condition | @PostMapping | examQuestionService调用examQuestion-Service中的page方法，传入查询条件，返回分页后的数据和总页数 |
| 方法 | getQuestion | @GetMapping | 调用examQuestionService的getById方法，传入id值，获取题目信息并返回 |
| 方法 | addQuestion | @PostMapping | 调用examQuestionService的save方法，传入题目信息，返回方法结果的布尔值 |
| 方法 | updateQuestion | @PostMapping | 调用examQuestionService的updateById方法，传入题目信息，返回方法结果的布尔值 |
| 方法 | deleteQuestion | @DeleteMapping | 调用examQuestionService的removeById方法，传入题目信息，返回方法结果的布尔值 |

**TrainPaperService接口**

继承IService方法，定义getPaperList方法，传入Page<TrainPaper>对象，返回Map<String, Object>对象

**TrainQuestionService接口**

继承IService方法，定义getQuestionList方法，传入Page<TrainQuestion>对象，返回Map<String, Object>对象

**TrainPaperServiceImpl类**

继承ServiceImpl方法，实现TrainPaperService接口

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | trainPaper-Mapper | @Autowired | TrainPaperMapper类 |
| 方法 | getPaperList | @Override | 调用baseMapper的selectPage方法，传入筛选条件，获取分页信息（数据、当前页码、每页记录数、上一页、下一页、总页数等）并返回 |

**TrainQuestionServiceImpl类**

继承ServiceImpl方法，实现TrainQuestionService接口

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | trainQuestion-Mapper | @Autowired | TrainQuestionMapper类 |
| 方法 | getQuestionList | @Override | 调用baseMapper的selectPage方法，传入筛选条件，获取分页信息（数据、当前页码、每页记录数、上一页、下一页、总页数等）并返回 |

### 2.5、任务功能

**（1）功能汇总**

任务功能：

|  |  |
| --- | --- |
| 功能 | 描述 |
| 根据条件查询并分页 | 根据传入的查询条件（年级），获取任务记录，并根据当前页和每页记录数进行分页并返回 |
| 添加任务 | 根据传入的信息，新增任务记录 |
| 删除任务 | 根据传入的任务id，删除对应的任务记录 |

**（2）类详细描述**

**TaskController试卷管理控制类**

url路径：/api/admin/task

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | TaskController | @Autowired | TaskController类 |
| 方法 | pageList | @RequestMapping | pageList调用taskExamService中的page方法，传入查询条件，返回分页后的数据和总页数 |
| 方法 | edit | @RequestMapping | 调用taskExamService的edit方法，传入任务信息，返回方法结果的布尔值 |
| 方法 | select | @RequestMapping | 调用taskExamService的taskExamToVM方法，传入任务信息，返回方法结果的布尔值 |
| 方法 | delete | @DeleteMapping | 调用taskExamService的selectById方法，传入任务信息，返回方法结果的布尔值 |

**TaskExamCustomerAnswerService接口**

继承BaseService方法，定义edit方法，传入PageInfo<TaskExam>对象，返回List<TaskExam>对象

**TaskExamServiceImpl类**

继承BaseServiceImpl方法，实现TaskExamService接口

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | TaskExamServiceImpl | @Autowired | TaskExamService类 |
| 方法 | page | @Override | 调用baseMapper的selectPage方法，传入筛选条件，获取分页信息（数据、当前页码、每页记录数、上一页、下一页、总页数等）并返回 |
| 方法 | taskExamToVM | @Override | 调用taskExamMapper的selectByPrimaryKey方法，传入筛选条件，获取任务信息并返回 |
| 方法 | getByGradeLevel | @Override | 调用taskExamMapper的getByGradeLevel方法，获得返回信息 |

# 二、前台功能

## 1、接口设计

### 1.1、ExamPaperScoreController

|  |
| --- |
| ExamPaperScoreController |
| * getExamPaperScore() |
| ExamPaperScoreController考试成绩接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R getExamPaperScore(@PathVariable Integer id);  查询考试成绩 |

### 1.2、ExamAnswerPaperController

|  |
| --- |
| **ExamAnswerPaperController** |
| * getAnwerPaper() * addAnwerPaper() * updateAnwerPaper() |
| ExamAnswerPaperController学生答卷管理接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上。 |
| public R getAnswerPaper(@PathVariable Integer id);  查询答卷记录 |
| public R addAnswerPaper(@RequestBody ExamPaper examPaper);  添加答卷记录 |
| public R updateAnswerPaper(@RequestBody ExamPaper examPaper);  更新答卷记录 |

### 1.3、StuExamController

|  |
| --- |
| StuExamController |
| * selectPaper() |
| StuExamController  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上 |
| public R selectPaper(@PathVariable Integer id);  获取试卷 |

### 1.4、TrainSelectCourseController

|  |
| --- |
| TrainSelectCourseController |
| * pageQuestionCondition() |
| TrainSelectCourseController学生选课管理接口  @RestController和@RequestMapping("/train/exam-questions")声明在接口名上方，@RestController = @Controller + @ResponseBody，@RequestMapping注解是用来映射请求的，即指明处理器可以处理哪些URL请求，该注解既可以用在类上，也可以用在方法上 |
| public R pageQuestionCondition(@PathVariable Integer current, @PathVariable Integer limit, @RequestBody(required = false) CourseQuery CourseQuery);  课程条件分页 |

## 2、功能描述

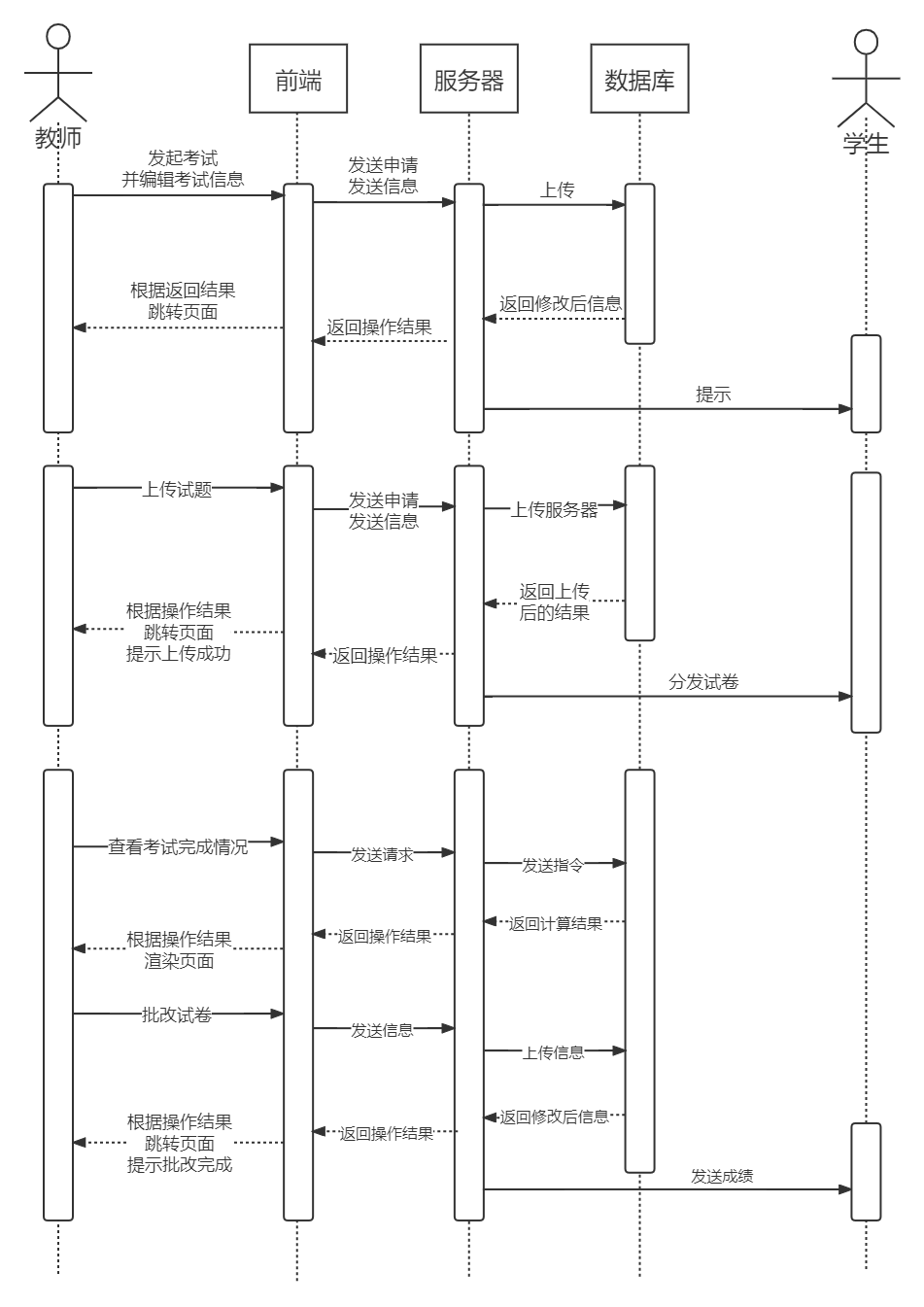
### 2.1、答题功能

**（1）功能汇总**

答卷功能：

|  |  |
| --- | --- |
| 功能 | 描述 |
| 根据id查询答卷信息 | 根据传入的id获取答卷信息，根据答卷题号和学生题号，获取全部答题信息并返回 |
| 根据条件查询并分页 | 根据传入的查询条件（无条件、所属课程、试卷类型、试卷名称），获取试卷记录，并根据当前页和每页记录数进行分页并返回 |
| 添加答卷记录 | 根据传入的信息，新增答卷记录和对应的答题记录 |
| 修改答卷记录 | 根据传入的信息，修改答卷记录和对应的答题记录 |

**（2）时序图**



**（3）类详细描述**

**ExamAnswerPaperController答卷管理控制类**

url路径：/train/exam-answer

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | examAnwer-PaperService | @Autowired | ExamAnswerPaperService类 |
| 属性 | examAnwerQuestionService | @Autowired | ExamAnswerQuestionService类 |
| 方法 | getAnwerPaper | @GetMapping | 调用examAnswerPaperService的getById方法，传入id值，获取答卷信息，根据paperId调用examAnswerQuestionService中查询答题记录方法，返回答卷和对应的答题数据 |
| 方法 | addAnwerPaper | @PostMapping | 调用examAnswerPaperService和examAnswerQuestionService的save方法，传入答卷信息和答题信息，返回方法结果的布尔值 |
| 方法 | updateAnwerPaper | @PostMapping | 调用examAnswerPaperService和examAnswerQuestionService的updateById方法，传入试卷信息和答题，返回方法结果的布尔值 |

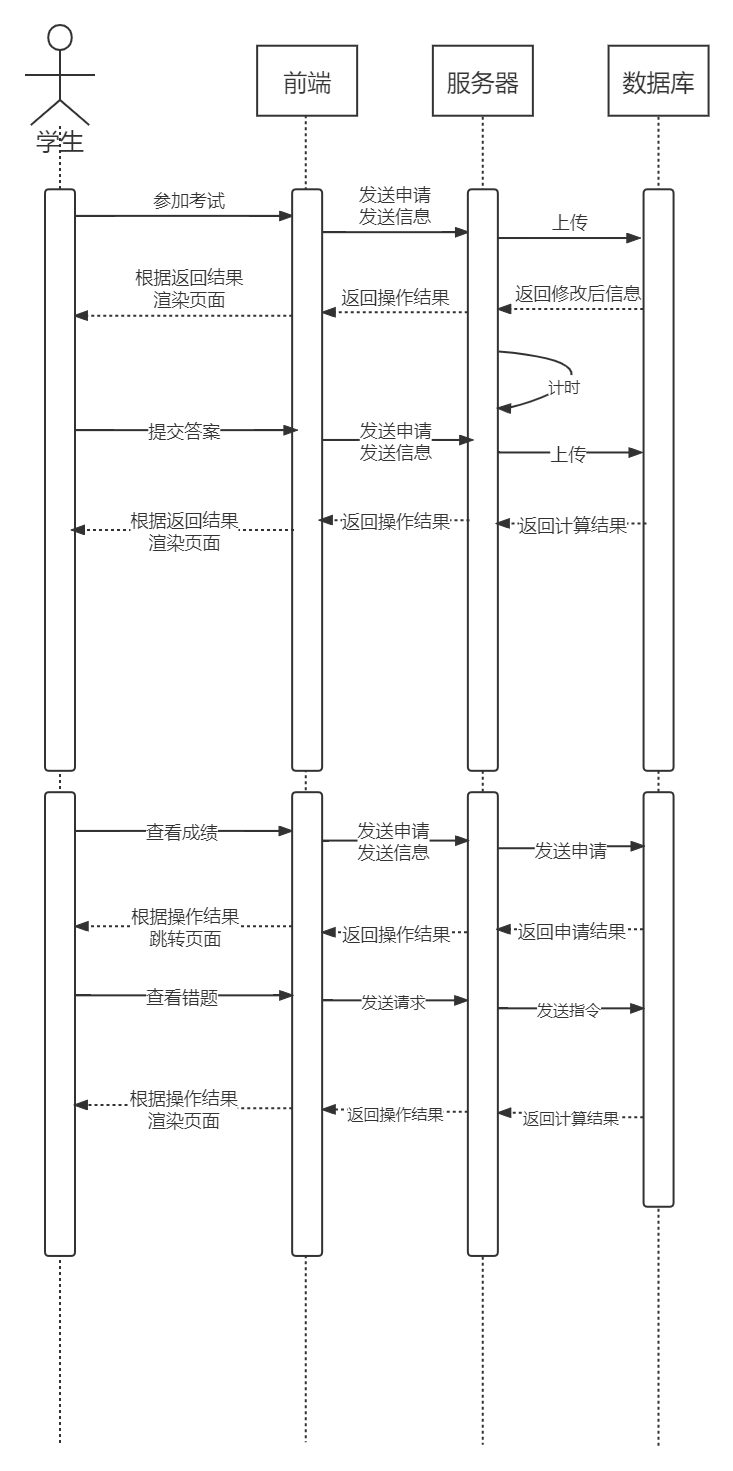
### 2.1、错题本功能

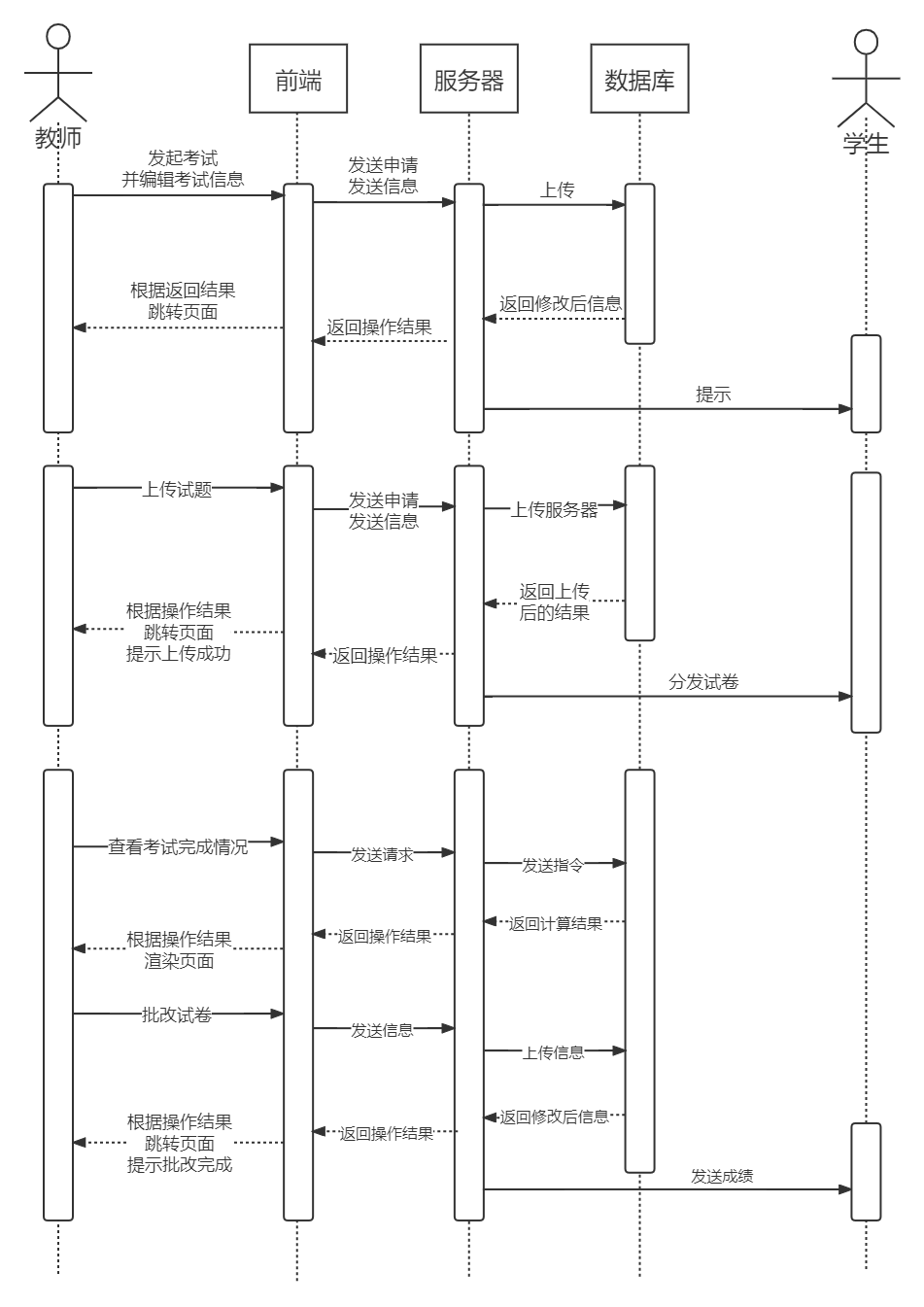
**（1）功能汇总**

错题本功能：

|  |  |
| --- | --- |
| 功能 | 描述 |
| 根据id查询答卷信息 | 根据传入的id获取答卷信息，根据答卷题号和学生题号，获取全部答题信息并返回 |
| 根据条件查询并分页 | 根据传入的查询条件（无条件、所属课程、试卷类型、试卷名称），获取试卷记录，并根据当前页和每页记录数进行分页并返回 |
| 添加答卷记录 | 根据传入的信息，新增答卷记录和对应的答题记录 |
| 修改答卷记录 | 根据传入的信息，修改答卷记录和对应的答题记录 |

**（2）时序图**





**（3）类详细描述**

**ExamAnswerPaperController答卷管理控制类**

url路径：/train/exam-answer

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 注解 | 描述 |
| 属性 | examAnwer-PaperService | @Autowired | ExamAnswerPaperService类 |
| 属性 | examAnwerQuestionService | @Autowired | ExamAnswerQuestionService类 |
| 方法 | getAnwerPaper | @GetMapping | 调用examAnswerPaperService的getById方法，传入id值，获取答卷信息，根据paperId调用examAnswerQuestionService中查询答题记录方法，返回答卷和对应的答题数据 |
| 方法 | addAnwerPaper | @PostMapping | 调用examAnswerPaperService和examAnswerQuestionService的save方法，传入答卷信息和答题信息，返回方法结果的布尔值 |
| 方法 | updateAnwerPaper | @PostMapping | 调用examAnswerPaperService和examAnswerQuestionService的updateById方法，传入试卷信息和答题，返回方法结果的布尔值 |

# 三、配置文件与工具类

### 1、application.properties配置文件

配置应用名称与访问端口：

spring.application.name=train  
server.port=8001

配置MySQL数据库的连接：

spring.datasource.username=root  
spring.datasource.password=123456  
spring.datasource.url=jdbc:mysql://localhost:3306/wdndb?useSSL=false&useUnicode=true&characterEncoding=utf-8&serverTimezone=GMT%2B8  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

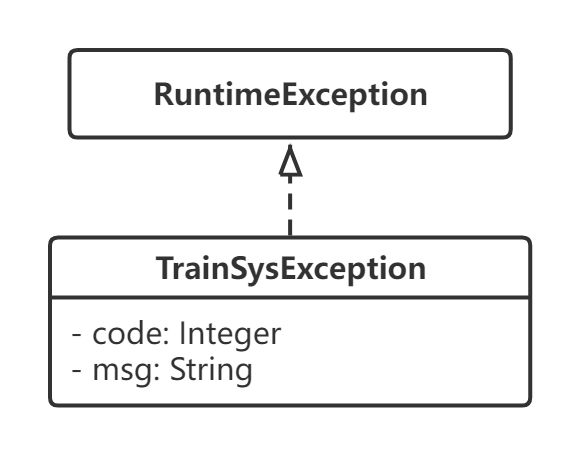
配置json日期格式：

spring.jackson.date-format=yyyy-MM-dd HH:mm:ss  
spring.jackson.time-zone=GMT+8

配置mapper路径：

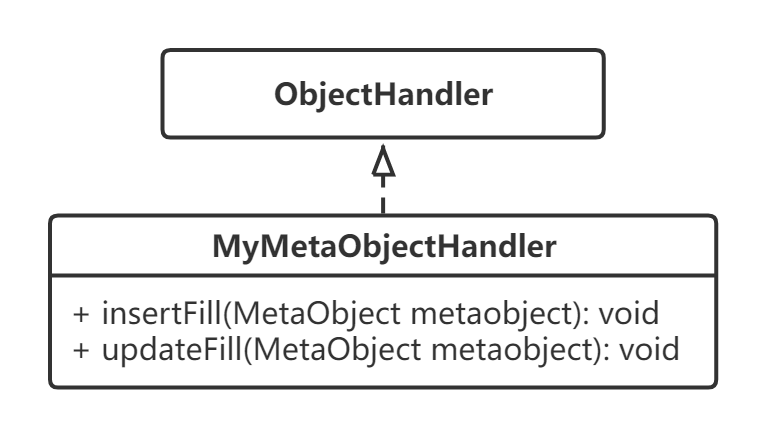
mybatis-plus.mapper-locations = classpath:com/education/train/mapper/xml/\*.xml

### 2、TrainSysException异常处理类



|  |
| --- |
| TrainSysException异常处理类  @Data自动生成set和get方法  @AllArgsConstructor生成变量的有参构造  @NoArgsConstructor生成变量  的无参构造  自定义异常处理类。实现RuntimeException接口，定义状态码和消息，供全局异常抛出。 |

### 3、MyMetaObjectHandler日期类型自动填充配置类



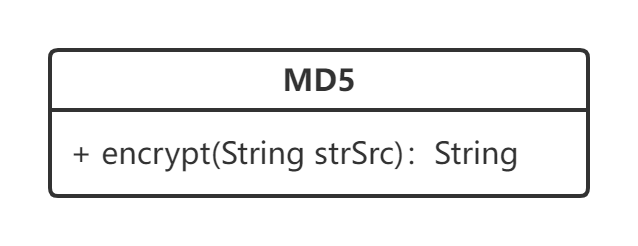
|  |
| --- |
| MyBatisPlus日期类型自动填充配置  @Component实现Bean的注入  实现ObjectHandler接口，用于时间属性gmtCreate和gmtModified的自动填充 |
| public void insertFill(MetaObject metaObject)  对象创建时，自动填充gmtCreate和gmtModified时间 |
| public void updateFill(MetaObject metaObject)  对象更新时，自动填充gmtModified时间 |

### 4、JwtUtils工具类



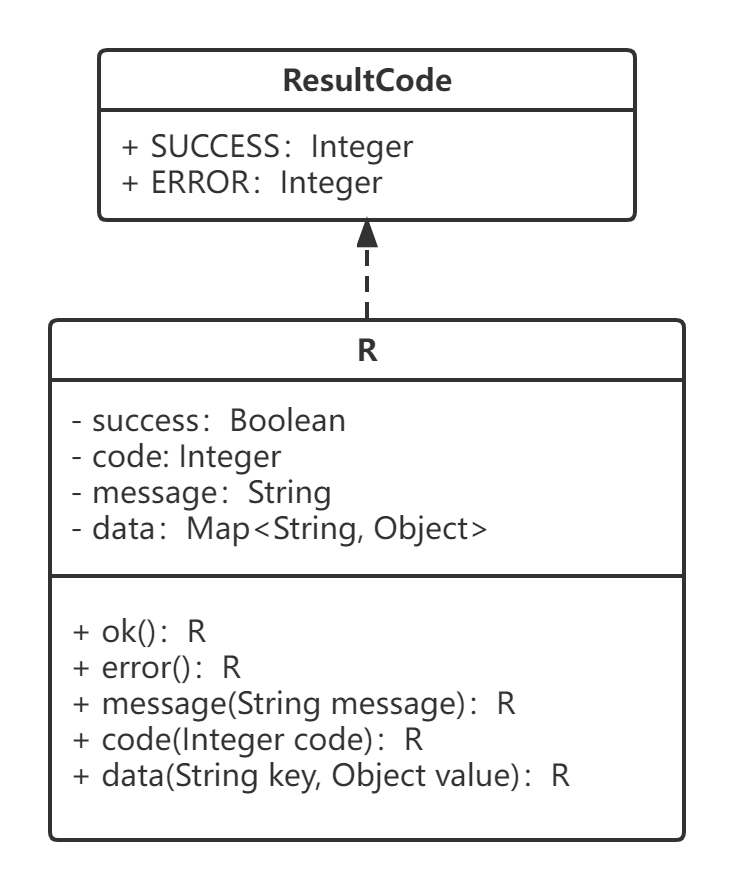
|  |
| --- |
| JwtUtils工具类  实现基于Jwt的令牌管理 |
| public static String getJwtToken(String id, String nickname)  根据id和nickname生成令牌token并返回 |
| public static boolean checkToken(String jwtToken)  判断传入的token是否存在且有效 |
| public static String getUserIdByJwtToken(String jwtToken)  根据传入的token获取用户id并返回 |

### 5、MD5工具类



|  |
| --- |
| MD5工具类  用于MD5加密 |
| public static String encrypt(String strSrc)  将传入的字符串进行MD5加密，并返回加密后的字符串 |

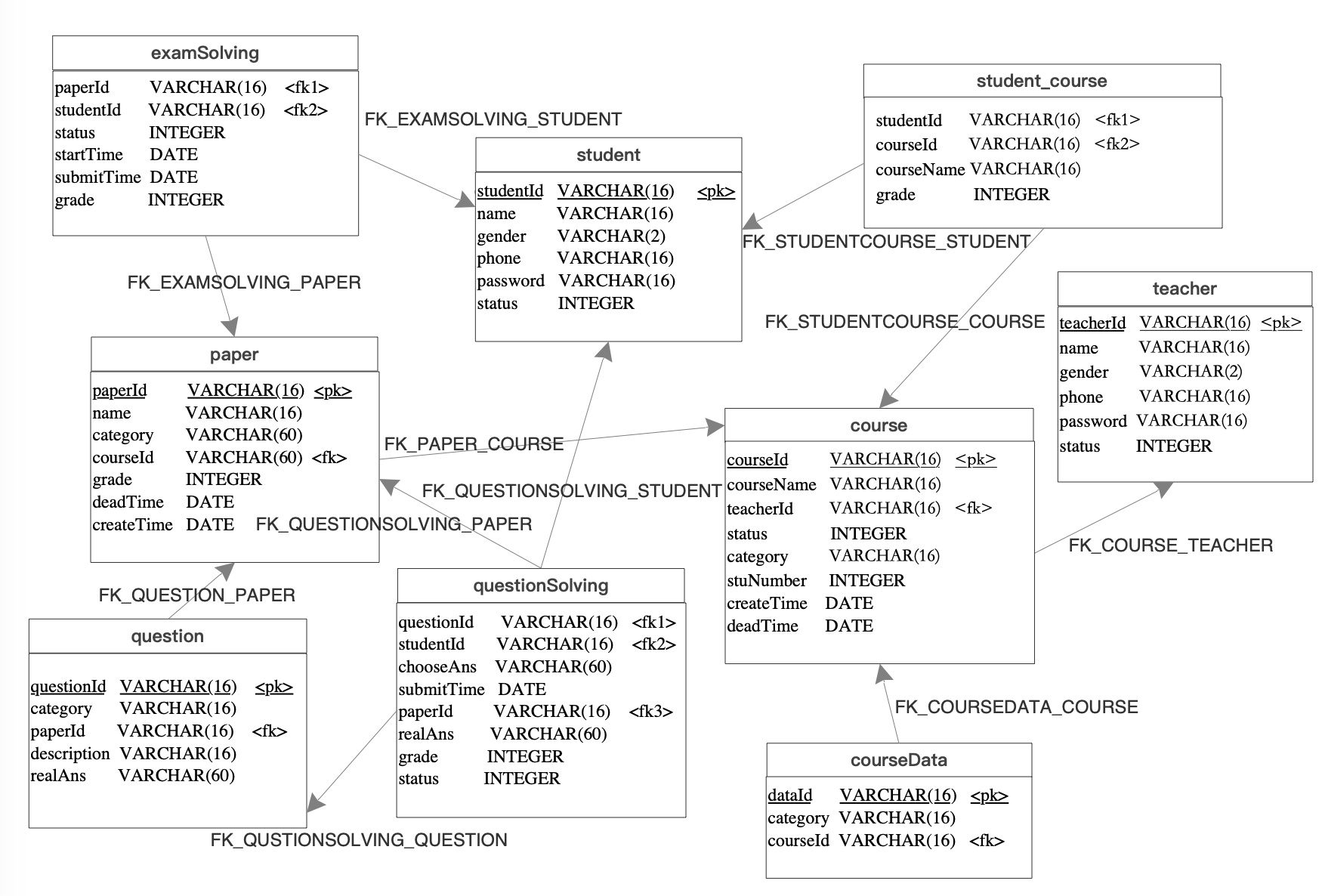
### 6、R和ResultCode统一返回类型



|  |
| --- |
| ResultCode接口  定义状态码SUCCESS=20000, ERROR=20001用于后端处理结果的状态返回 |
| R统一返回结果类  @Data自动生成set和get方法  用于返回是否成功、状态码、消息和数据等内容，是Controller接口的统一返回对象 |
| public static R ok()  返回成功、成功状态码和成功消息 |
| public R message(String message)  设置返回的消息 |
| public R code(Integer code)  设置返回的状态码 |
| public R data(String key, Object value)  设置返回的数据 |

# 第六部分 数据库设计

## 一、数据库整体结构图



## 二、数据库表设计

### 1、数据库表清单

|  |  |  |
| --- | --- | --- |
| 序号 | 表名 | 注释 |
| 1 | student | 学生表 |
| 2 | student\_course | 选课表，学生和课程的中间表 |
| 3 | teacher | 教师表 |
| 4 | questionSolving | 解题表，记录学生所做的练习题的记录，包括所选答案，提交时间等 |
| 5 | examSolving | 做卷表，记录学生所做试卷的记录 |
| 6 | paper | 试卷表 |
| 7 | question | 题目表 |

### （1）student表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | studentId | VARCHAR(16) | 学号 |
| 2 | name | VARCHAR(16) | 姓名 |
| 3 | gender | VARCHAR(2) | 性别 |
| 4 | phone | VARCHAR(16) | 手机号码 |
| 5 | password | VARCHAR(16) | 密码 |
| 6 | status | INTEGER | 账号状态 |

### （2）student\_course表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | studentId | VARCHAR(16) | 学号，外键 |
| 2 | courseId | VARCHAR(16) | 课程号，外键 |
| 3 | courseName | VARCHAR(16) | 课程名 |
| 4 | grade | INTEGER | 成绩 |

### （3）teacher表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | teacherId | VARCHAR(16) | 教师工号 |
| 2 | name | VARCHAR(16) | 姓名 |
| 3 | gender | VARCHAR(2) | 性别 |
| 4 | phone | VARCHAR(16) | 手机号码 |
| 5 | password | VARCHAR(16) | 密码 |
| 6 | status | INTEGER | 账号状态 |
| 6 | realAns | VARCHAR(60) | 正确答案 |
| 7 | grade | INTEGER | 得分 |
| 8 | status | INTEGER | 批改状态 |

### （4）examSolving表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | paperId | VARCHAR(16) | 试卷号 |
| 2 | studentId | VARCHAR(16) | 答卷学生学号 |
| 3 | status | INTEGER | 试卷完成状态 |
| 4 | startTime | DATE | 开始时间 |
| 5 | submitTime | DATE | 提交时间 |
| 6 | grade | INTEGER | 分值 |

### （5）paper表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | paperId | VARCHAR(16) | 试卷号 |
| 2 | name | VARCHAR(16) | 试卷名称 |
| 3 | category | VARCHAR(60) | 试卷类型 |
| 4 | courseId | VARCHAR(60) | 所属课程 |
| 5 | grade | INTEGER | 分数 |
| 6 | deadTime | DATE | 截止时间 |
| 7 | createTime | DATE | 发布时间 |

### （6）questionSolving表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | questionId | VARCHAR(16) | 题号 |
| 2 | studentId | VARCHAR(16) | 解题学生学号 |
| 3 | chooseAns | VARCHAR(60) | 所选答案 |
| 4 | submitTime | DATE | 提交时间 |
| 5 | paperId | VARCHAR(16) | 所属试卷号 |
| 6 | realAns | VARCHAR(60) | 正确答案 |
| 7 | grade | INTEGER | 得分 |
| 8 | status | INTEGER | 批改状态 |

### （6）question表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | questionId | VARCHAR(16) | 题号 |
| 2 | category | VARCHAR(16) | 题型 |
| 3 | paperId | VARCHAR(16) | 试卷号 |
| 4 | description | VARCHAR(16) | 题目描述 |
| 5 | realAns | VARCHAR(60) | 正确答案 |

### 2、外键清单

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 外键名称 | 父表 | 父键列 | 子表 | 外键列 | 关系 | 说明 |
| FK\_TEACHER\_COURSE | teacher | teacherId | course | teacherId | 1..\* | 一个教师可教授多门课程 |
| FK\_PAPER\_COURSE | course | courseId | paper | courseId | 1..\* | 一个课程可以有多张试卷 |
| FK\_PAPER\_QUESTION | paper | paperId | question | paperId | 1..\* | 一张试卷由很多题目组成 |
| FK\_COURSEDATA\_COURSE | course | courseId | coursedata | courseId | 1..\* | 一门课可以有多个教学资源 |
| FK\_EXAMSOLVING\_PAPER | paper | paperId | examSolving | paperId | 1..1 | 一个答卷记录对应一张试卷 |

# 第七部分 补充设计和说明

### 一、编译运行环境设计

### 1、服务器

1）OS：WindowsServer 2003

2）2G以上内存，建议使用4GMB内存；

3）500GB以上的硬盘空间，不包括数据存储空间

4）JDK：JavaSE 6

5）DB：Database10g

6）ApplicationServer：WebLogic Server 10.3或者Tomcat5.5以上版本

### 2、客户机器环境

1）PentiumIII或以上微处理器（CPU）；

2）MicrosoftWindows XP、 Windows Vista、Windows 7操作系统；

3）512MB以上内存，建议使用1024MB内存；

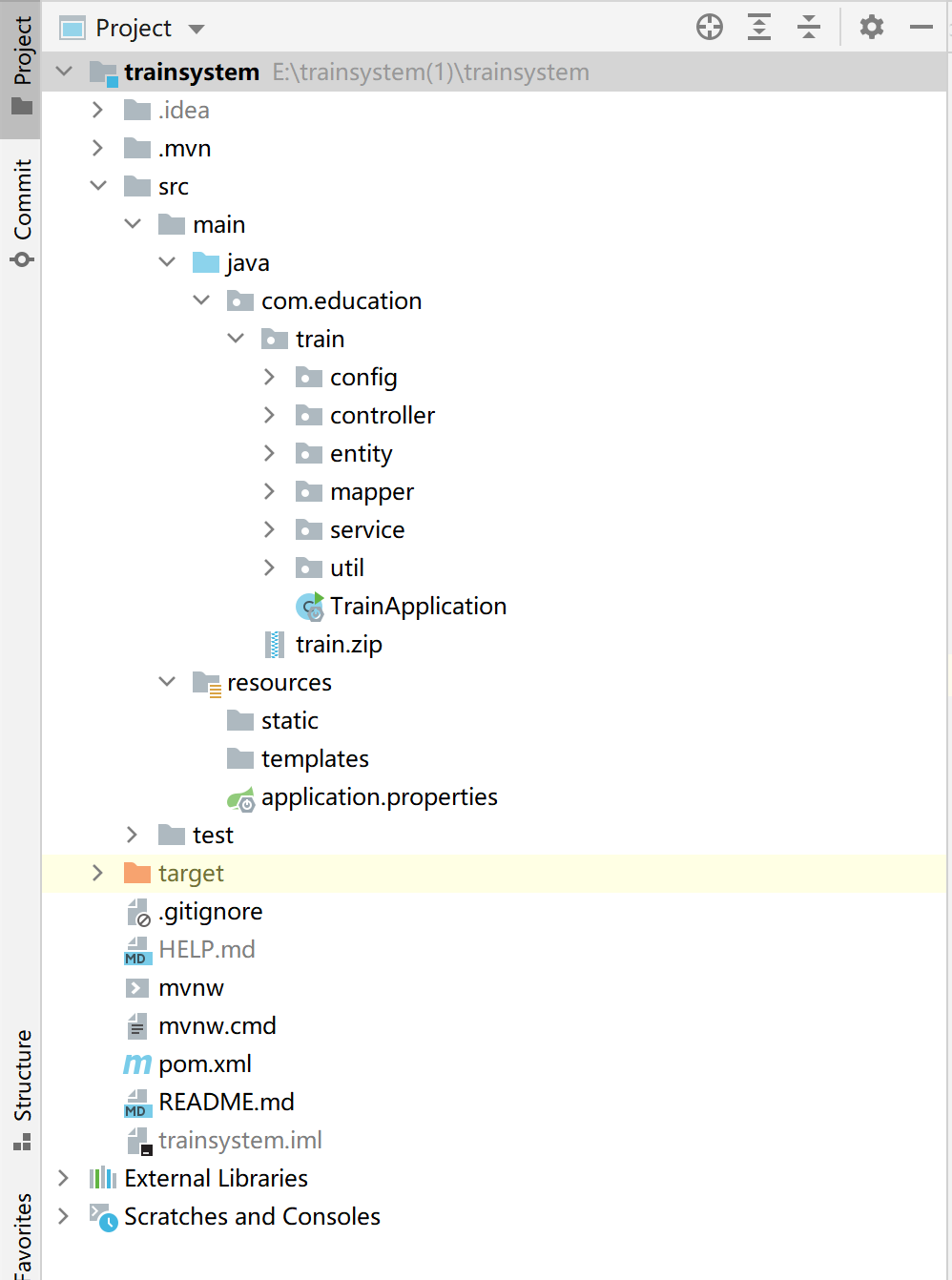
4）WEB：要求IE7以上版本，最好IE8以上版本。

### 3、开发环境

|  |  |  |
| --- | --- | --- |
| **项目** | **名称** | **版本** |
| 开发平台 | Mac OS | 10.7 |
|  | Windows OS | 10 |
| 开发工具 | IDEA | 2021.1 |
|  | Visual Studio Code | 1.64.2 |
| 代码管理工具 | Git | 2.31.1 |
| 开发环境 | Java JDK | 1.8 |

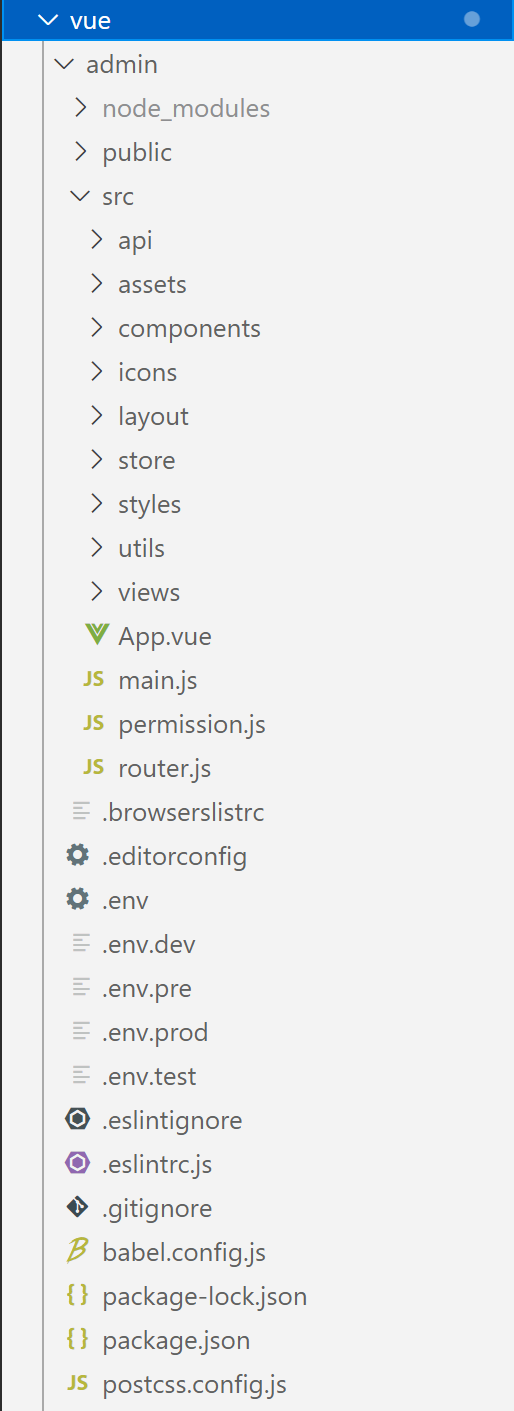
### 二、包路径与WEB目录结构设计

1、后端包路径设计



2、WEB目录结构设计

后台前端目录（基于vue-admin-template）：



前台前端目录（基于Nuxt）：

