|  |
| --- |
|  |
| **《在线考试&培训系统》**  **详细分析与设计**  **V2.0** |
|  |

**目 录**

[一、编写目的 1](#_Toc48594832)

[二、项目背景 2](#_Toc48594833)

[三、定义 2](#_Toc48594834)

[1、Web Service 2](#_Toc48594835)

[2、database（db） 3](#_Toc48594836)

[3、 spring boot 3](#_Toc48594837)

[4、 vue 3](#_Toc48594838)

[5、 Json 4](#_Toc48594839)

[6、 MVC 框架 4](#_Toc48594840)

[7、MyBatis 4](#_Toc48594841)

[四、参考资料 5](#_Toc48594842)

[一、技术架构设计 7](#_Toc48594843)

[二、核心控制流程 8](#_Toc48594844)

[1、核心控制流程图 8](#_Toc48594845)

[2、核心控制流程说明 9](#_Toc48594846)

[三、Spring Web环境下的集成配置 10](#_Toc48594847)

[1、pom.xml 10](#_Toc48594848)

[2、application.yml 14](#_Toc48594849)

[3、logback.xml 16](#_Toc48594850)

[一、界面设计 17](#_Toc48594851)

[1、界面框架设计 17](#_Toc48594852)

[2、PC界面设计 17](#_Toc48594853)

[2.1、登录界面设计 17](#_Toc48594854)

[2.2、学生主界面设计 18](#_Toc48594856)

[2.3、我的课程页面设计 18](#_Toc48594857)

[2.4、我的成绩页面设计 19](#_Toc48594858)

[2.5、 具体课程页面设计 20](#_Toc48594859)

[2.6、课程视频界面设计 21](#_Toc48594860)

[3、教师界面设计 23](#_Toc48594861)

[3.1、教师主界面设计 23](#_Toc48594862)

[3.2、查改具体课程界面设计 23](#_Toc48594863)

[3.3、发布试卷界面 24](#_Toc48594864)

[3.4、发布公告界面设计 25](#_Toc48594865)

[3.5、查看学生成绩 25](#_Toc48594866)

[3.6、查看学生信息界面设计 26](#_Toc48594867)

[3.7查看课程统计信息界面 26](#_Toc48594868)

[4.1管理员全部审批界面 27](#_Toc48594869)

[4.2 管理员待辦审批界面 28](#_Toc48594871)

[4.3 管理员已辦审批界面 28](#_Toc48594872)

[4.4 管理员教師管理界面 28](#_Toc48594874)

[4.5 管理员學生管理界面 29](#_Toc48594875)

[4.6 管理员課程管理界面 29](#_Toc48594876)

[4.7 管理员學生信息界面 30](#_Toc48594877)

[4.8 管理员教師信息界面 30](#_Toc48594878)

[4.9管理员审批信息界面 31](#_Toc48594879)

[4.10管理员課程信息界面 31](#_Toc48594880)

[4.11管理员登錄界面 32](#_Toc48594881)

[一、数据访问层设计 33](#_Toc48594882)

[1、类图设计 33](#_Toc48594883)

[2、类的详细设计描述 33](#_Toc48594884)

[2.1 BaseMapper<T>接口设计 33](#_Toc48594885)

[2.2 MAPPER接口实现类汇总 34](#_Toc48594886)

[二、业务逻辑层设计 35](#_Toc48594887)

[1、类图设计 35](#_Toc48594888)

[2、类的详细设计描述 35](#_Toc48594889)

[2.1 ITeacherService接口设计 35](#_Toc48594890)

[2.2 IStudentService接口设计 36](#_Toc48594891)

[2.3 ICourseService接口设计 37](#_Toc48594893)

[2.4 ITestPaperService接口设计 38](#_Toc48594894)

[2.5 IStudentGradeService接口设计 39](#_Toc48594895)

[2.6 INoticeService接口设计 39](#_Toc48594896)

[2.7 IStudentClassService接口设计 40](#_Toc48594897)

[2.8 IAdministratorService接口设计 41](#_Toc48594898)

[2.9 ICourseDataService接口设计 41](#_Toc48594899)

[2.10 ICourseApplicationService接口设计 42](#_Toc48594900)

[2.11 InfoService接口设计 43](#_Toc48594901)

[2.12 业务接口实现类汇总 43](#_Toc48594902)

[三、学生功能 44](#_Toc48594903)

[1、课程选择 44](#_Toc48594904)

[2、我的课程 48](#_Toc48594906)

[3. 成绩查询 51](#_Toc48594907)

[4. 个人信息更改 53](#_Toc48594908)

[四、 教师模块设计 56](#_Toc48594909)

[4.1.查找课程 56](#_Toc48594910)

[4.2.发布课程 58](#_Toc48594911)

[4.3.查找学生信息课程 59](#_Toc48594913)

[4.4.教师课程业务 60](#_Toc48594914)

[4.5教师登录及主界面UI 62](#_Toc48594915)

[五、 管理员模块设计 65](#_Toc48594918)

[5.1、管理員登錄 65](#_Toc48594919)

[5.2、課程申請审核 67](#_Toc48594920)

[5.3、教師管理 80](#_Toc48594921)

[5.4、學生管理 87](#_Toc48594922)

[5.5、課程管理 95](#_Toc48594923)

[六、 课程模块设计 102](#_Toc48594924)

[6.1、课程内容 102](#_Toc48594925)

[6.2、修改课程信息 109](#_Toc48594926)

[6.3、发布通知 111](#_Toc48594927)

[6.4、发布考试 113](#_Toc48594928)

[6.5、批改与读取学生成绩 116](#_Toc48594929)

[一、数据库整体结构图 121](#_Toc48594930)

[二、总模块 121](#_Toc48594931)

[1、Student表结构 121](#_Toc48594932)

[2、Teacher表结构 121](#_Toc48594933)

[3、Course表结构 121](#_Toc48594934)

[4、TestPaper表结构 122](#_Toc48594935)

[5、StudentGrade表结构 122](#_Toc48594936)

[6、Notice表结构 122](#_Toc48594937)

[7、StudentClass表结构 123](#_Toc48594938)

[8、Administrator表结构 123](#_Toc48594939)

[9、CourseData表结构 123](#_Toc48594940)

[10、CourseApplication表结构 124](#_Toc48594941)

[一、包路径与WEB目录结构设计 125](#_Toc48594942)

[1、包路径设计 125](#_Toc48594943)

[2、WEB目录结构设计 125](#_Toc48594944)

**第一部分 引言**

一、编写目的

编写本设计的目的是为了准确阐述基于在线考试&培训系统的具体实现思路和方法，即系统的详细架构和实现逻辑，主要包括程序系统的结构以及各层次中每个程序的设计考虑。

本文档供全体項目开发人员阅读，包括运营人员，维护人员等。

二、项目背景

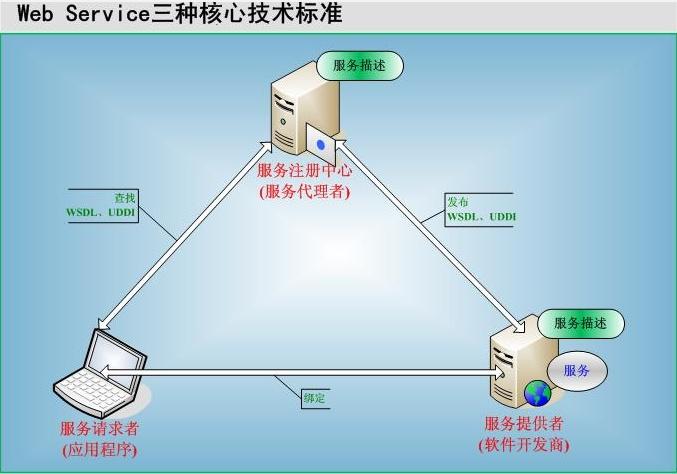
传统的线下教学与考试模式，由于物理空间上的限制会有许多不便，比如需要对教学与考试场所进行协调，考试时间可能遇到冲突，具有一定的局限性。

目前，随着互联网的进一步发展，网络对于大多数人来说都触手可得。通过将教学活动搬入线上，学员能够随时随地进行学习，突破传统教学模式的限制，大大提高学习效率。同时，新冠疫情以来，许多高校的教学都在网络上进行，使用了大量的网络教学软件，这也显示出一款在线考试&培训系统所具有的广阔前景。

三、定义

1、Web Service

Web Service 是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得 Web Service 能与其他兼容的组件进行互操作。Internet Inter-Orb Protocol(IIOP)都已经发布了很长时间了，但是这些模型都依赖于特殊对象模型协议，而 Web Services 利用 SOAP 和 XML 对这些模型在通讯方面作了进一步的扩展以消除特殊对象模型的障碍。Web Services 主要利用 HTTP 和 SOAP 协议使商业数据在 Web 上传输，SOAP 通过 HTTP 调用商业对象执行远程功能调用，Web 用户能够使用 SOAP 和 HTTP 通过 Web 调用的方法来调用远程对象.



2、database（db）

数据库(Database)是按照数据结构来组织、存储和管理数据的仓库，它产生于距今六十多年前，随着信息技术和市场的发展，特别是二十世纪九十年代以后，数据管理不再仅仅是存储和管理数据，而转变成用户所需要的各种数据管理的方式。数据库有很多种类型，从最简单的存储有各种数据的表格到能够进行海量数据存储的大型数据库系统都在各个方面得到了广泛的应用。

在信息化社会，充分有效地管理和利用各类信息资源，是进行科学研究和决策管理的前提条件。数据库技术是管理信息系统、办公自动化系统、决策支持系统等各类信息系统的核心部分，是进行科学研究和决策管理的重要技术手段。

3、 spring boot

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 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来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot 致力于在蓬勃发展的快速应用开发领域(rapid application development)成为领导者

4、 vue

Vue 是一套用于构建用户界面的渐进式 JavaScript 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，方便与第三方库或既有项目整合。

5、 Json

JSON(JavaScript Object Notation, JS 对象简谱) 是一种轻量级的数据交换格式。它基于 ECMAScript (欧洲计算机协会制定的 js 规范)的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。

6、 MVC 框架

MVC 全名是 Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC 被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

7、MyBatis

MyBatis 是一款优秀的持久层框架，它支持自定义 SQL、存储过程以及高级映射。MyBatis 免除了几乎所有的 JDBC 代码以及设置参数和获取结果集的工作。MyBatis 可以通过简单的 XML 或注解来配置和映射原始类型、接口和 Java POJO（Plain Old Java Objects，普通老式 Java 对象）为数据库中的记录。

四、参考资料

《系统详细设计》

百度百科. <https://baike.baidu.com/item/Database/1264765>

百度百科. <https://baike.baidu.com/item/Spring%20Boot>

百度百科. <https://baike.baidu.com/item/Vue.js/19884851?fr=aladdin>

百度百科. <https://baike.baidu.com/item/JSON/2462549?fr=aladdin>

百度百科. <https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/9241230?fromtitle=mvc&fromid=85990&fr=aladdin>

Mybatis. <https://mybatis.org/mybatis-3/zh/index.html>

**第二部分 项目概述**

传统的线下教学与考试模式，由于物理空间上的限制会有许多不便，比如需要对教学与考试场所进行协调，考试时间可能遇到冲突，具有一定的局限性。

目前，随着互联网的进一步发展，网络对于大多数人来说都触手可得。通过将教学活动搬入线上，学员能够随时随地进行学习，突破传统教学模式的限制，大大提高学习效率。同时，新冠疫情以来，许多高校的教学都在网络上进行，使用了大量的网络教学软件，这也显示出一款在线考试&培训系统所具有的广阔前景。

對于學生來說,网上教學平台能給學生拋開地点和時間的限制,可以學習到更多不同类型的課程知識。

對于教師來說,网上教學平台能夠減輕教師的教學負擔,因為网上教學平台只一次授課,不同的群体學生都可以學習。

用户业务主要包含学生业务、教师业务和管理员业务。

学生业务包含八个主要业务审核：添加课程、删除课程、观看视频、做练习题、在线考试、成绩查询、更改身份信息和更改密码。

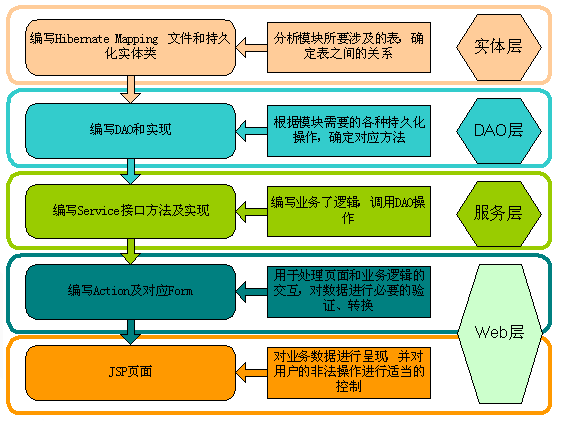
教师业务包含八个主要业务查询：发布课程、查看具体课程信息、发布考试、批改试卷、查看成绩、发布公告、查看课程统计信息、查询学生信息。

管理员业务包含八个主要业务查询：课程申请审核、学生管理、教师管理、课程信息查询、课程信息修改。

**第三部分 总体设计**

一、技术架构设计

SSH 为 Struts+Spring+（JPA）Hibernate 的组成方式，Struts 实现 MVC，Spring 负责架构的结合，（JPA）Hibernate 进行数据的持久化。



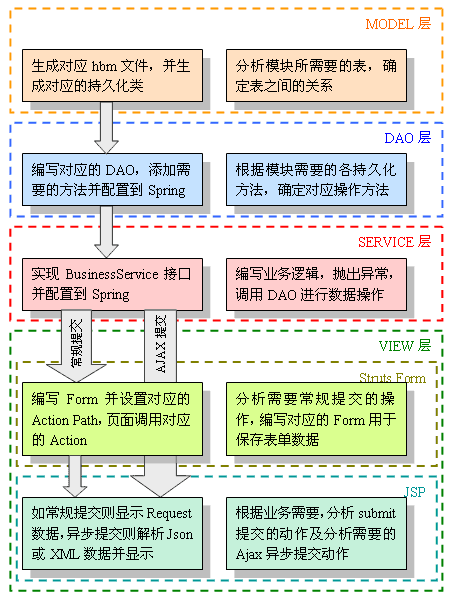
这样的结构，满足了一般的业务需要，但是对于当前日益复杂化的 WEB2.0 的开发，却存在不少问题，归纳起来主要有以下几点的不足：

DAO 和服务层容易出现职责不明，由于按照 MVC 逻辑，业务代码应该写在 Struts Action 里，但是其事务的提供，却是配置在 Service 层。为了一组在逻辑上完整的数据操作业务逻辑，需要涉及两个层（Serveice、Action）来进行编写，遇到判断的情况下，为了保证完整的事务操作，则需要将业务代码移到 Service 层完成，而通常习惯了在 Struts Action 里调用多次 Service 而产生多个事务而在出现 Exception 时导致出错时操作之前调用的 Service 事务的业务数据没有回滚。

当需要返回的数据供 AJAX 使用，操作 JSON 或 XML 的的大量使用时。开发起来会很费力，一段同样的业务代码，为了使用 AJAX 和 XML 可能需要重新编写一次，或者在同一个 ACTION 里通过标志来判断，对分层结构造成了比较糟糕的破坏。如果设计得不好，为了使用 JSON 和 XML 还得额外增加大量的配置，严重降低了开发效率。

因此，为了克服这些缺点，本系统对于 SSH 架构，进行了重新的分层，共享了业务代码。简化了开发、增强了与 AJAX 技术、XML 技术的结合。提供了一种更高效的开发模式。

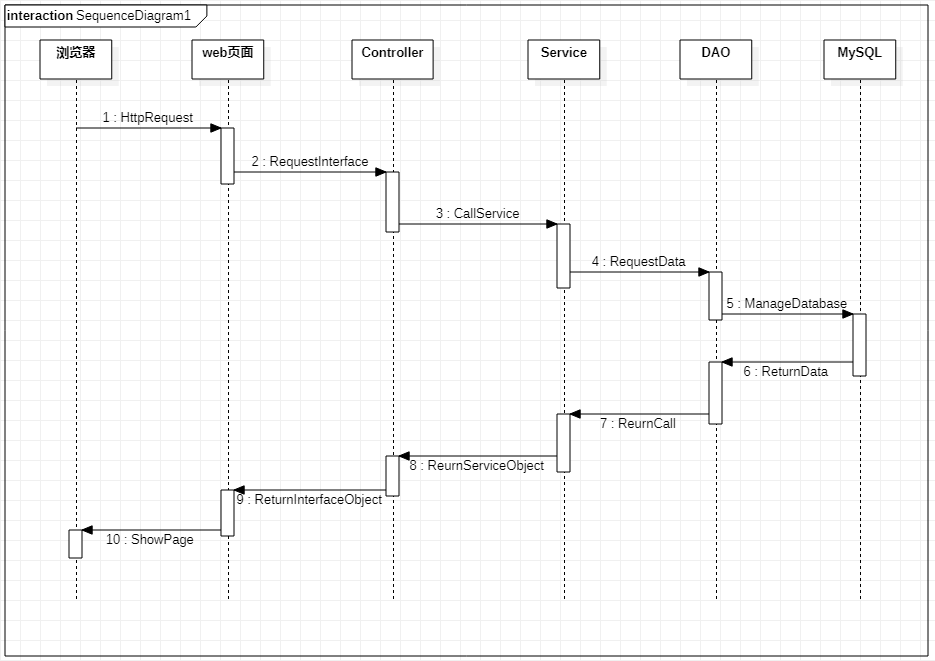
其开发的结构图如下：



由于业务代码统一实现 IbusinessService 接口，使得只需要相对固定的几个 Struts Action 类调用 Service 层的方法，便可以完成工作。包括 JSON 格式输出，XML 输出及 WebService 输出均调用 Service 层方法来完成功能。这样便实现了业务代码的分离，以及与前端框架的极大解耦。

二、核心控制流程

1、核心控制流程图



2、核心控制流程说明

结合上图，针对每一个消息请求，按照消息请求/回复顺序，作如下说明。

|  |  |  |
| --- | --- | --- |
| **编号** | **消息名称** | **消息说明** |
| 1 | HttpRequest | 浏览器发来的HttpRequest 请求 |
| 2 | RequestInterface | web页面向controller层请求接口数据 |
| 3 | CallService | 调用服务方法 |
| 4 | RequestData | 使用dao方法实现数据访问 |
| 5 | ManageDatabase | 操作数据库 |
| 6 | ReturnData | 返回数据 |
| 7 | ReturnCall | 返回数据访问 |
| 8 | ReturnServiceObject | 返回服务对象 |
| 9 | ReturnInterfaceData | 返回接口数据 |
| 10 | ShowPage | 展示页面 |

三、Spring Web环境下的集成配置

1、pom.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 <https://maven.apache.org/xsd/maven-4.0.0.xsd>">      <modelVersion>4.0.0</modelVersion>      <parent>          <groupId>org.springframework.boot</groupId>          <artifactId>spring-boot-starter-parent</artifactId>          <version>2.3.1.RELEASE</version>          <relativePath/> <!-- lookup parent from repository -->      </parent>      <groupId>cn.com.chinahitech</groupId>      <artifactId>bjmymarket</artifactId>      <version>0.0.1-SNAPSHOT</version>      <name>bjmymarket</name>      <description>Demo project for Spring Boot</description>        <properties>          <java.version>1.8</java.version>      </properties>        <!-- 配置国内maven仓库源 -->      <repositories>          <repository>              <id>ali-maven</id>              <url>http://maven.aliyun.com/nexus/content/groups/public</url>              <releases>                  <enabled>true</enabled>              </releases>              <snapshots>                  <enabled>true</enabled>                  <updatePolicy>always</updatePolicy>                  <checksumPolicy>fail</checksumPolicy>              </snapshots>          </repository>            <repository>              <id>central</id>              <name>Maven Repository Switchboard</name>              <layout>default</layout>              <url>http://repo1.maven.org/maven2</url>              <snapshots>                  <enabled>false</enabled>              </snapshots>          </repository>      </repositories>        <dependencies>          <!-- spring-boot-starter -->          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-starter</artifactId>          </dependency>            <!-- spring boot web -->          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-starter-web</artifactId>          </dependency>            <!--thymeleaf模板-->          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-starter-thymeleaf</artifactId>          </dependency>            <!-- spring-boot-devtools -->          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-devtools</artifactId>              <scope>runtime</scope>              <optional>true</optional>          </dependency>            <!-- lombok 实现实体类运行时添加构造器/setter/getter -->          <dependency>              <groupId>org.projectlombok</groupId>              <artifactId>lombok</artifactId>              <optional>true</optional>          </dependency>            <!--    json object对象支持    -->          <dependency>              <groupId>com.alibaba</groupId>              <artifactId>fastjson</artifactId>              <version>1.2.71</version>          </dependency>            <!--   mybatis plus     -->          <dependency>              <groupId>com.baomidou</groupId>              <artifactId>mybatis-plus-boot-starter</artifactId>              <version>3.3.0</version>          </dependency>            <!--    MySQL驱动    -->          <!-- <https://mvnrepository.com/artifact/mysql/mysql-connector-java> -->          <dependency>              <groupId>mysql</groupId>              <artifactId>mysql-connector-java</artifactId>              <version>8.0.18</version>          </dependency>            <!--    数据连接池 druid    -->          <dependency>              <groupId>com.alibaba</groupId>              <artifactId>druid-spring-boot-starter</artifactId>              <version>1.1.20</version>          </dependency>            <!--    代码生成器中使用freemarker    -->          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-starter-freemarker</artifactId>              <version>2.2.2.RELEASE</version>          </dependency>            <!--    代码自动生成工具 mybatis-plus    -->          <dependency>              <groupId>com.baomidou</groupId>              <artifactId>mybatis-plus-generator</artifactId>              <version>3.3.0</version>          </dependency>            <!-- springfox-swagger2-->          <dependency>              <groupId>io.springfox</groupId>              <artifactId>springfox-swagger2</artifactId>              <version>2.9.2</version>          </dependency>            <!--springfox-swagger2 -->          <dependency>              <groupId>io.springfox</groupId>              <artifactId>springfox-swagger-ui</artifactId>              <version>2.9.2</version>          </dependency>            <!-- junit单元测试 -->          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-starter-test</artifactId>              <scope>test</scope>              <exclusions>                  <exclusion>                      <groupId>org.junit.vintage</groupId>                      <artifactId>junit-vintage-engine</artifactId>                  </exclusion>              </exclusions>          </dependency>      </dependencies>          <build>          <plugins>              <plugin>                  <groupId>org.springframework.boot</groupId>                  <artifactId>spring-boot-maven-plugin</artifactId>              </plugin>          </plugins>      </build>    </project> |

2、application.yml

|  |
| --- |
| server:  port: 8081  spring:  application:  name: market //需要修改  aop:  proxy-target-class: true  datasource:  type: com.alibaba.druid.pool.DruidDataSource  driver-class-name: com.mysql.cj.jdbc.Driver  url: jdbc:mysql://192.168.1.181:3306/bubusheng? //项目名  allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=utf8&serverTimezo  ne=Asia/Shanghai&useSSL=false  username: root  password: ROOTroot\_1  druid:  validation-query: SELECT 1 FROM DUAL  initial-size: 10  min-idle: 10  max-active: 200  min-evictable-idle-time-millis: 300000  test-on-borrow: false  test-while-idle: true  time-between-eviction-runs-millis: 30000  pool-prepared-statements: true  max-open-prepared-statements: 100  mybatis-plus:  type-aliases-package: cn.com.chinahitech.bjmymarket.\*.entity //需要修改  global-config:  db-config:  id-type: auto  logic-delete-field: flag  logic-delete-value: 1  logic-not-delete-value: 0  configuration:  map-underscore-to-camel-case: true  cache-enabled: false  jdbc-type-for-null: 'null'  log-impl: org.apache.ibatis.logging.stdout.StdOutImpl  logging:  config:  classpath: logback.xml |

3、logback.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <configuration>  <property name="LOG\_PATTERN"  value="%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -%msg%n" />  <property name="LOG\_LEVEL" value="INFO"/>  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  <encoder>  <pattern>${LOG\_PATTERN}</pattern>  </encoder>  <filter class="ch.qos.logback.classic.filter.ThresholdFilter">  <level>${LOG\_LEVEL}</level>  </filter>  </appender>  <root level="${LOG\_LEVEL}">  <appender-ref ref="STDOUT" />  </root>  </configuration> |

**第四部分 界面和业务单设计**

一、界面设计

1、界面框架设计

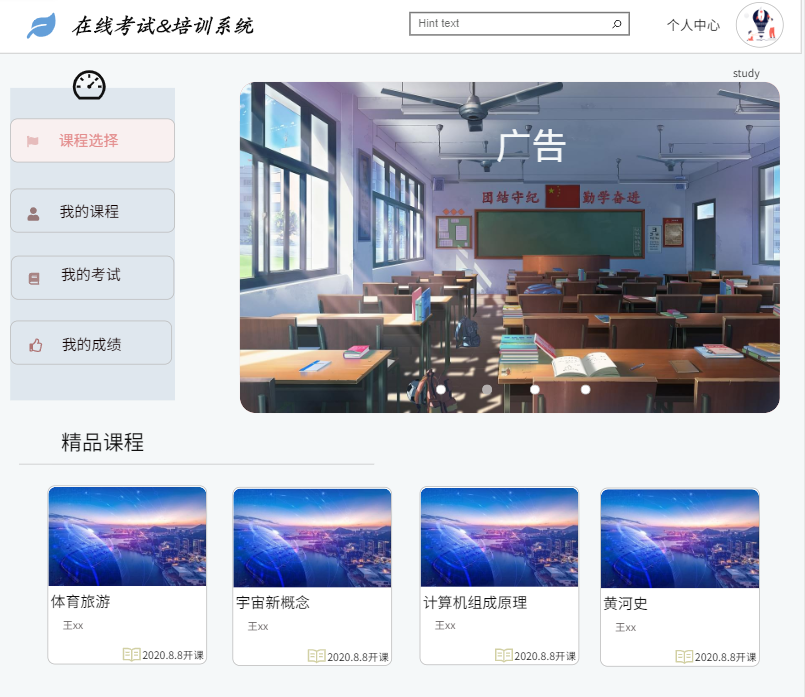
2、PC界面设计

2.1、登录界面设计

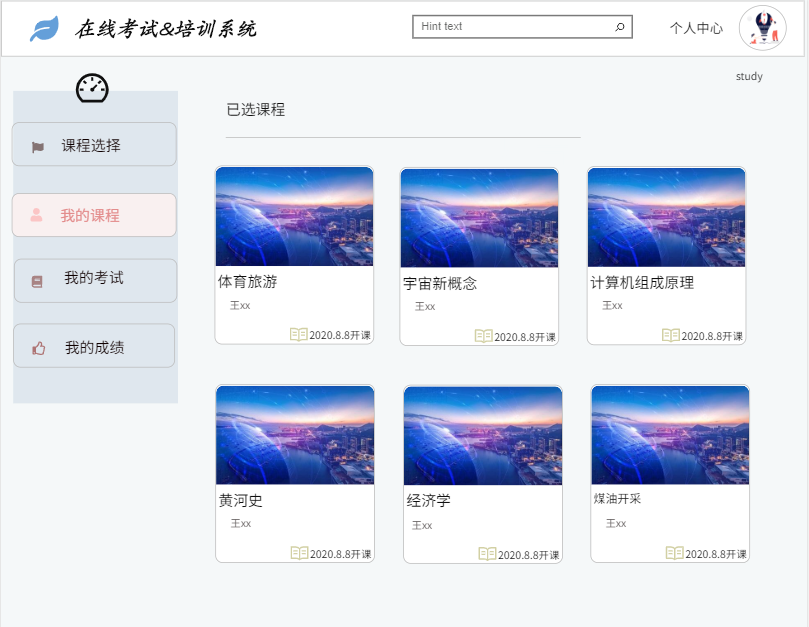




2.2、学生主界面设计



2.3、我的课程页面设计



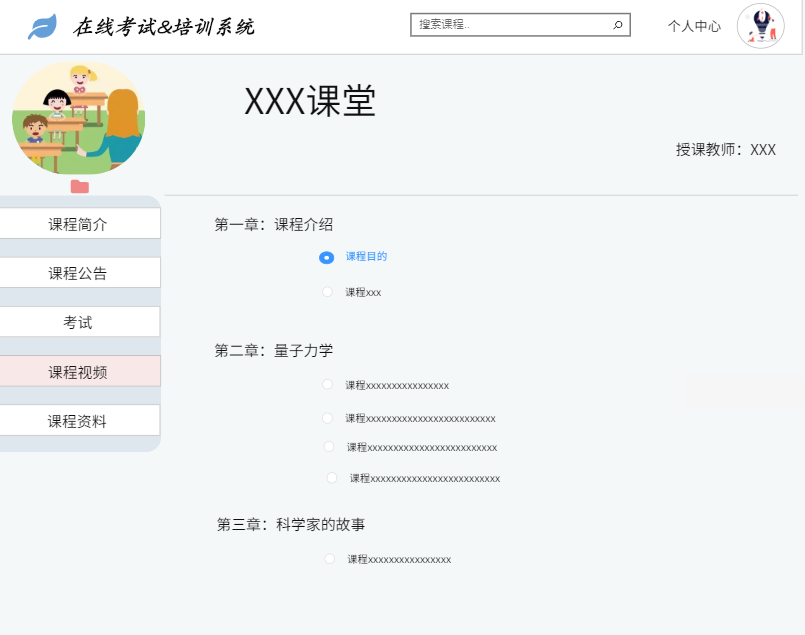
2.4、我的成绩页面设计



2.5、 具体课程页面设计



2.6、课程视频界面设计



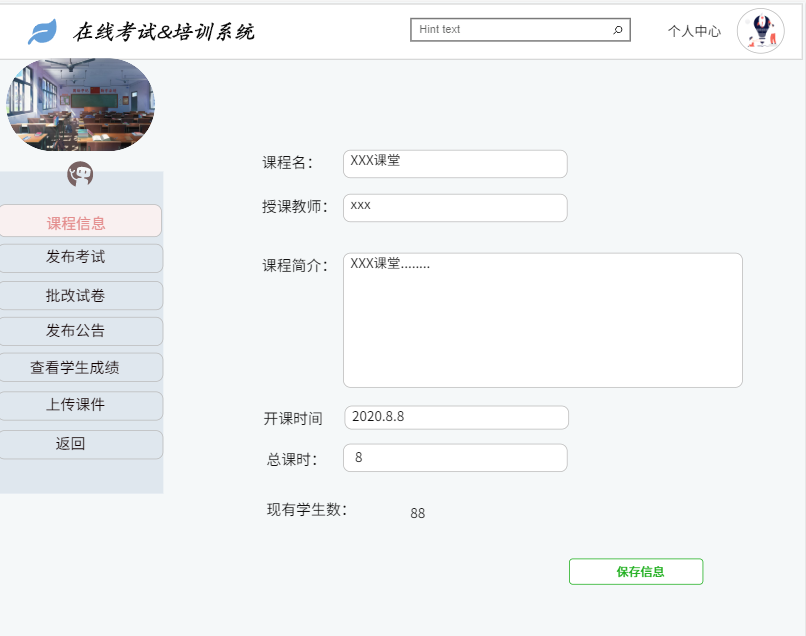


3、教师界面设计

3.1、教师主界面设计



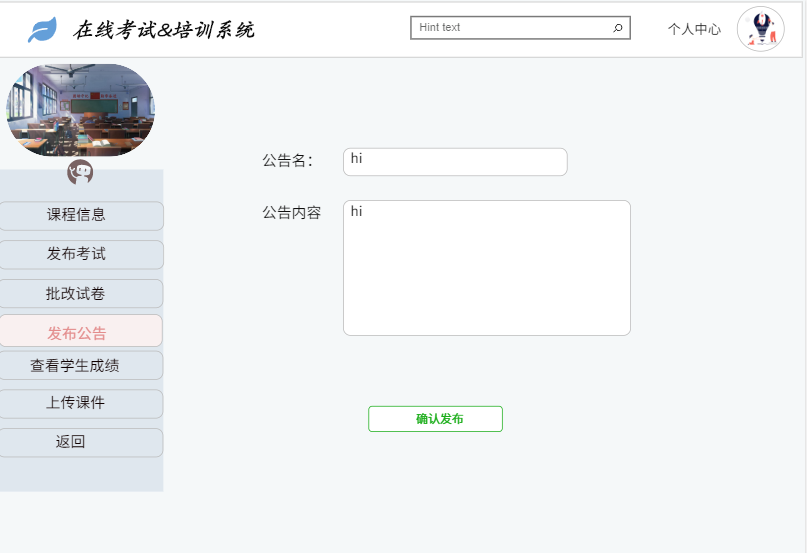
3.2、查改具体课程界面设计



3.3、发布试卷界面



3.4、发布公告界面设计



3.5、查看学生成绩



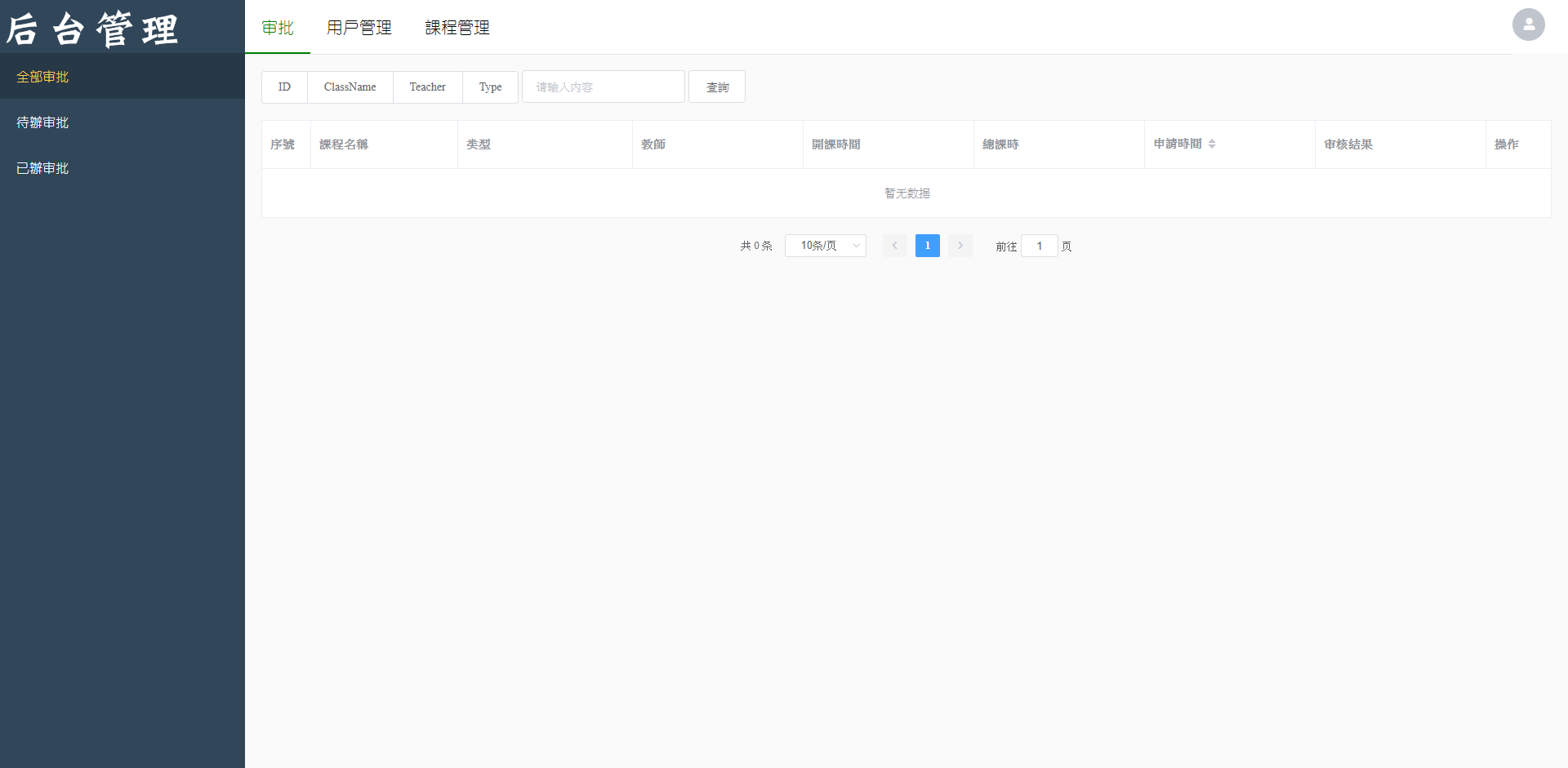
3.6、查看学生信息界面设计



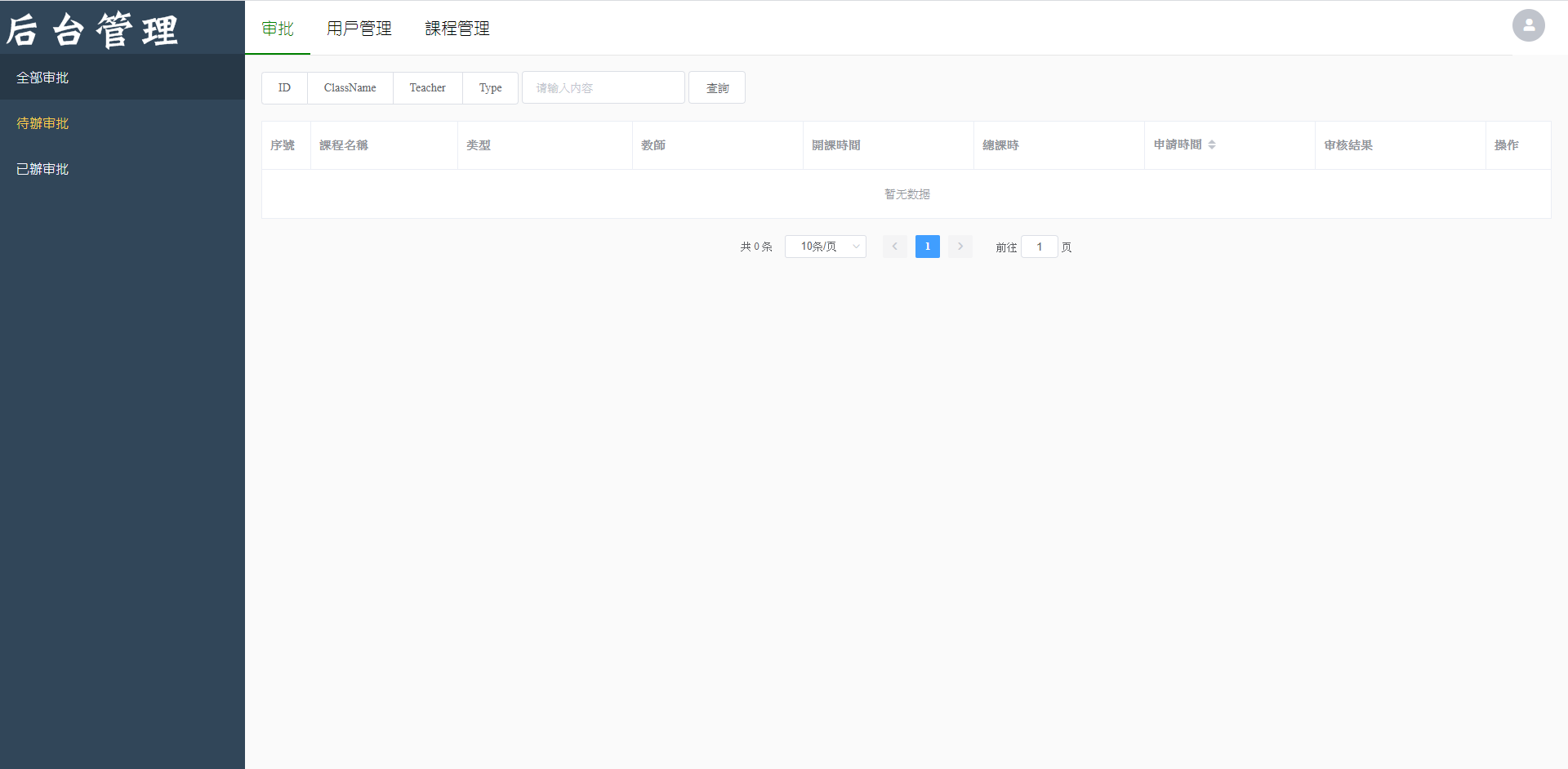
3.7查看课程统计信息界面



4.1管理员全部审批界面



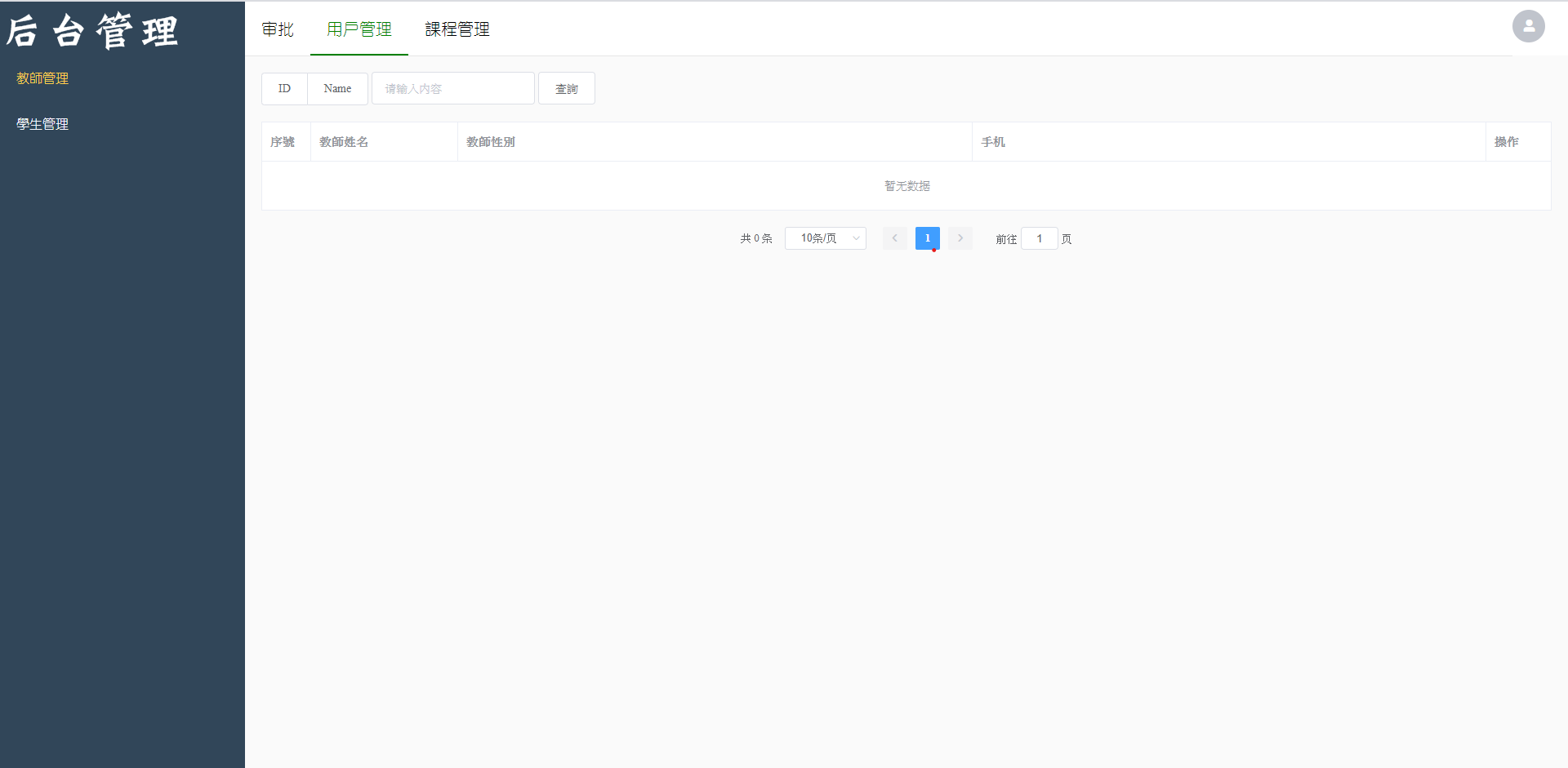
4.2 管理员待辦审批界面



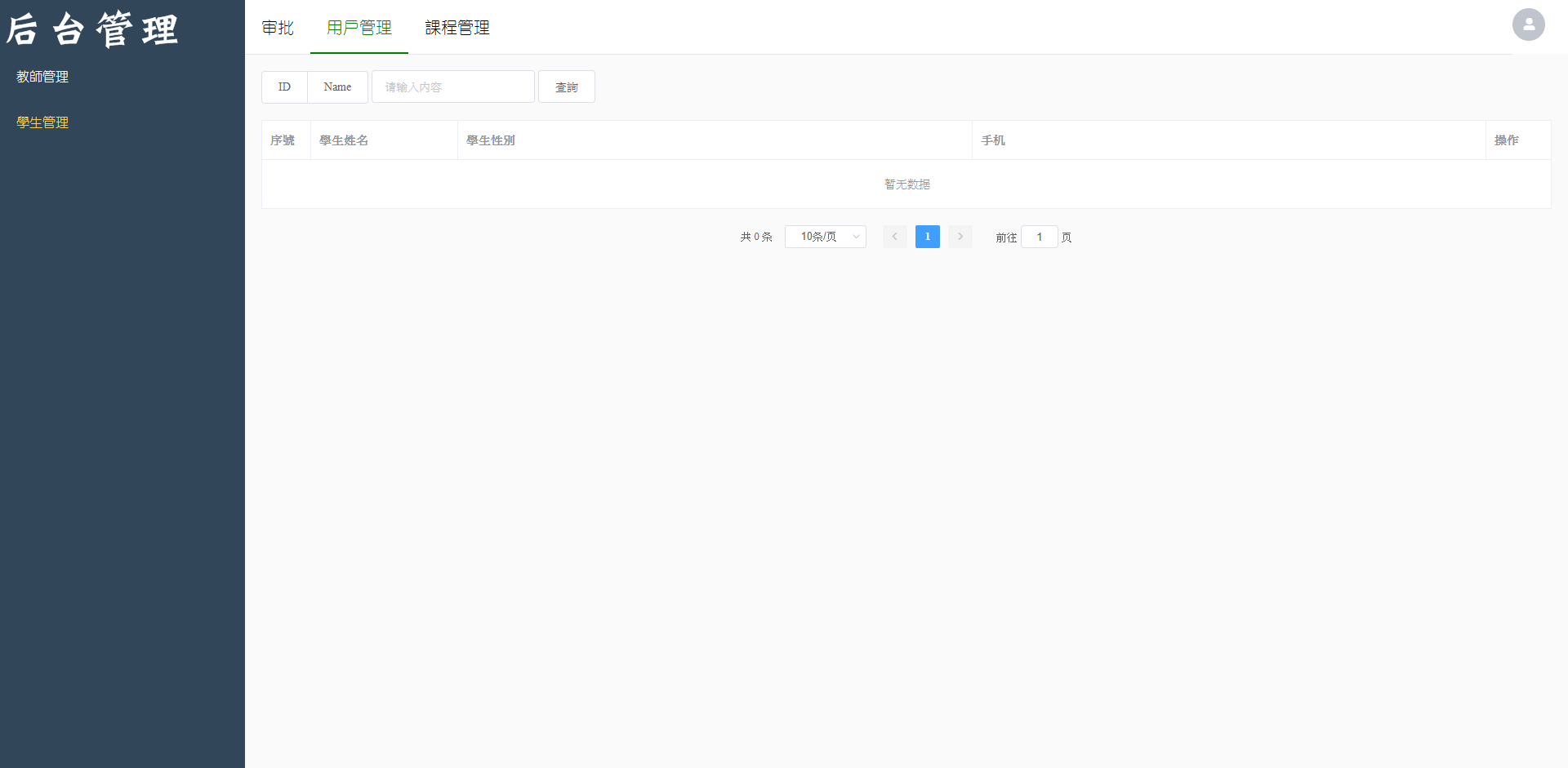
4.3 管理员已辦审批界面



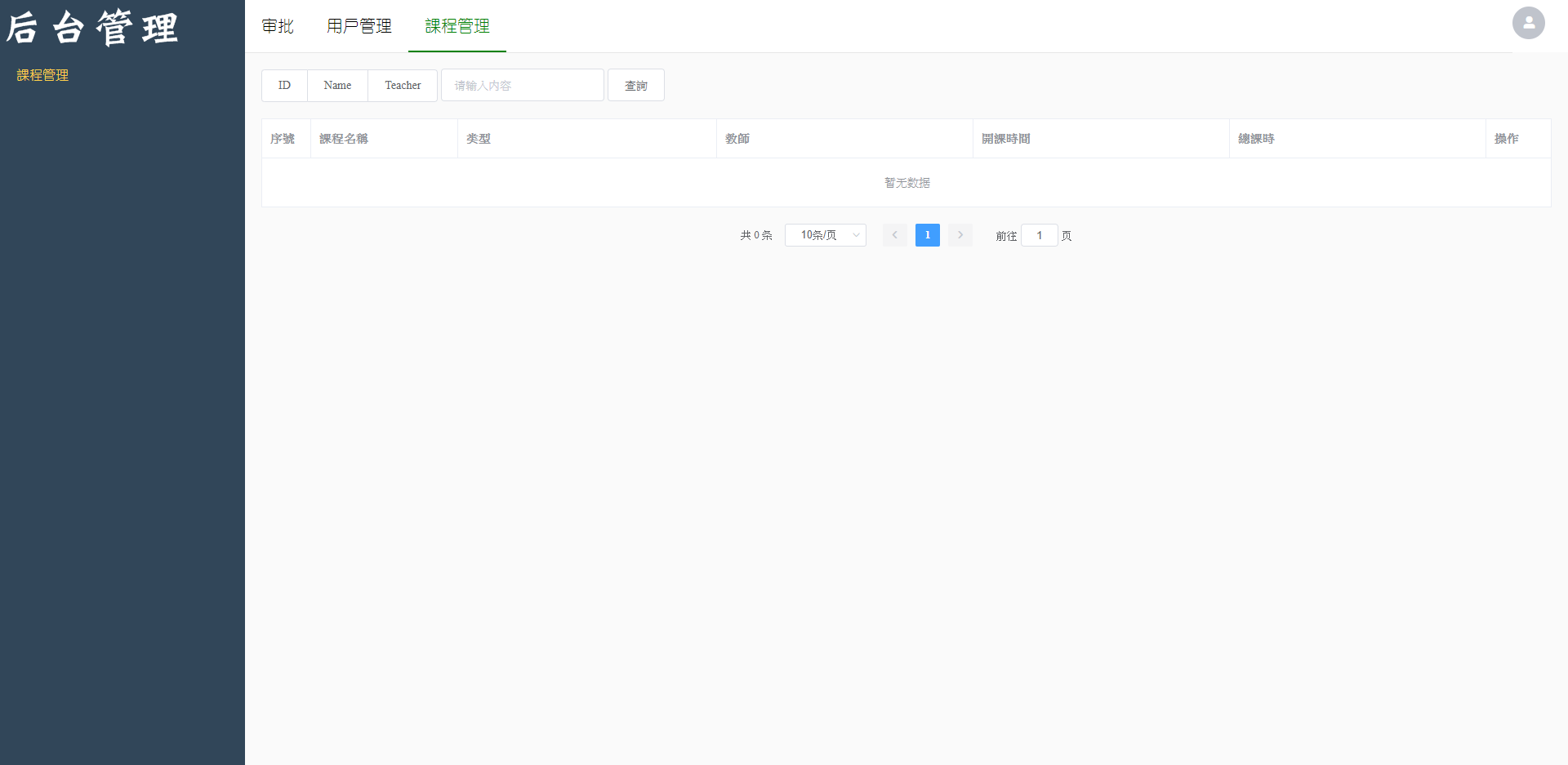
4.4 管理员教師管理界面



4.5 管理员學生管理界面



4.6 管理员課程管理界面



4.7 管理员學生信息界面



4.8 管理员教師信息界面



4.9管理员审批信息界面



4.10管理员課程信息界面



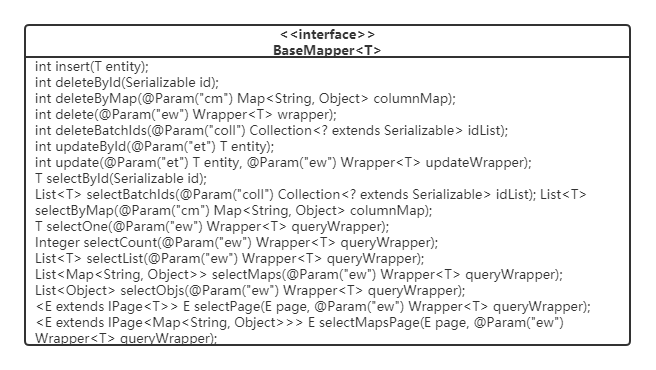
4.11管理员登錄界面



**第五部分 单元模块设计**

一、数据访问层设计

1、类图设计



2、类的详细设计描述

2.1 BaseMapper<T>接口设计

详细描述：

|  |
| --- |
| BaseMapper<T>通用数据库CRUD操作 |
| int insert(T entity);  通過实体类插入到數据庫 |
| int deleteById(Serializable id);  通過id刪除數据 |
| int deleteByMap(@Param("cm") Map<String, Object> columnMap);  通過多个条件刪除數据 |
| int delete(@Param("ew") Wrapper<T> wrapper);  通過封裝类刪除 |
| int deleteBatchIds(@Param("coll") Collection<? extends Serializable> idList);  通過多个ID进行刪除 |
| int updateById(@Param("et") T entity);  通過id進行更新 |
| int update(@Param("et") T entity, @Param("ew") Wrapper<T> updateWrapper);  通過封裝类更新 |
| T selectById(Serializable id);  通過id查詢某一數据 |
| List<T> selectBatchIds(@Param("coll") Collection<? extends Serializable> idList);  通過一系列id查詢一系列數据 |
| List<T> selectByMap(@Param("cm") Map<String, Object> columnMap);  通過多个条件查詢一系列數据 |
| T selectOne(@Param("ew") Wrapper<T> queryWrapper);  通過封裝类查詢某一數据 |
| Integer selectCount(@Param("ew") Wrapper<T> queryWrapper);  通過封裝类查詢數据數量 |
| List<T> selectList(@Param("ew") Wrapper<T> queryWrapper);  通過封裝类查詢數据 |
| List<Map<String, Object>> selectMaps(@Param("ew") Wrapper<T> queryWrapper);  返回指定列为key的map集合或对象 |
| List<Object> selectObjs(@Param("ew") Wrapper<T> queryWrapper);  返回指定列为key的对象 |
| <E extends IPage<T>> E selectPage(E page, @Param("ew") Wrapper<T> queryWrapper);  返回指定分頁 |
| <E extends IPage<Map<String, Object>>> E selectMapsPage(E page, @Param("ew") Wrapper<T> queryWrapper);  返回指定列为key的map集合或对象的分頁 |

2.2 MAPPER接口实现类汇总

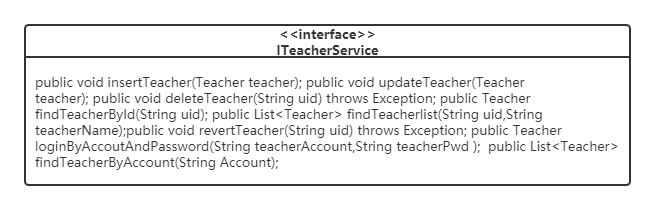
|  |  |
| --- | --- |
| 接口 | 实现类 |
| BaseMapper<Teacher> | TeacherMapper |
| BaseMapper<Student> | StudentMapper |
| BaseMapper<Course> | CourseMapper |
| BaseMapper<TestPaper> | TestPaperMapper |
| BaseMapper<StudentGrade> | StudnetGradeMapper |
| BaseMapper<Notice> | NoticeMapper |
| BaseMapper<StudentClass> | StudentMapper |
| BaseMapper<Administrator> | AdministratorMapper |
| BaseMapper<CourseData> | CourseDataMapper |
| BaseMapper<CourseApplication> | CourseApplicationMapper |

二、业务逻辑层设计

1、类图设计

2、类的详细设计描述

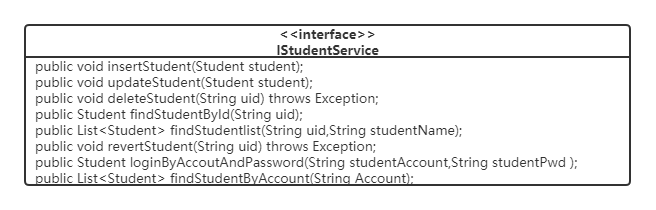
2.1 ITeacherService接口设计



详细描述：

|  |
| --- |
| ITeacherService教師接口 |
| public void insertTeacher(Teacher teacher);  新增教師 |
| public void updateTeacher(Teacher teacher);  更新教師信息 |
| public void deleteTeacher(String id) throws Exception;  删除教師，将教師状态置为【删除】状态 |
| public IPage<Teacher> getAllTeacher(int currentPage,int limit);  獲取所有教師信息 |
| public Teacher getTeacherById(String id);  根据教師ID查询教師詳細信息 |
| public IPage<Teacher> getTeacherByKey(int currentPage,int limit,String key,String keyword);  根据条件模糊条件查询教師列表 |
| public void revertTeacher(String uid) throws Exception;  恢复教師，将状态置为【活动】 |
| public Teacher loginByAccoutAndPassword(String teacherAccount,String teacherPwd );  教師登录，根据登录帐号和密码 |
| Teacher getAccount(int id)根据教师id，返回教师对象 |
| int signup(int teaid,String name,String phone,String password)教师注册 |

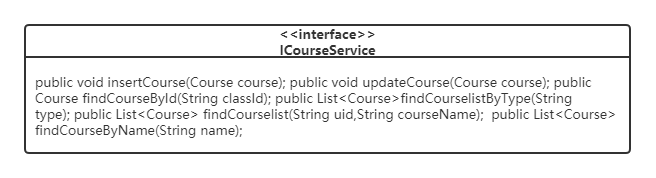
2.2 IStudentService接口设计



详细描述：

|  |
| --- |
| IStudentService學生接口 |
| public void insertStudent(Student student);  新增學生 |
| public void updateStudent(Student student);  更新學生信息 |
| public void deleteStudent(String uid) throws Exception;  删除學生，将學生状态置为【删除】状态 |
| public Student getStudentById(String id);  根据學生ID查询學生詳細信息 |
| public IPage<Student> getAllStudent(int currentPage,int limit)  獲取所有學生信息 |
| public IPage<Student> getStudentKey(int currentPage,int limit,String key,String keyword);  根据条件模糊条件查询學生列表 |
| public void revertStudent(String uid) throws Exception;  恢复學生，将状态置为【活动】 |
| public Student loginByAccoutAndPassword(String studentAccount,String studentPwd );  學生登录，根据登录帐号和密码 |

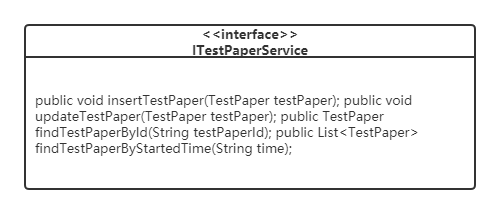
2.3 ICourseService接口设计



详细描述：

|  |
| --- |
| ICourseService課程接口 |
| public void insertCourse(Course course);  新增課程 |
| public void updateCourse(Course course);  更新課程信息 |
| public IPage<Course> getAllCourse(int currentPage, int limit);  獲取所有課程信息 |
| public Course getCourseById(String Id);  根据課程ID查询課程詳細信息 |
| public Course getCourseByTeacherId(int currentPage, int limit,String Id);  根据教師ID查询該教師的所有課程 |
| public IPage<Course> getCourseByKey(int currentPage, int limit, String key, String keyword);  根据条件模糊条件查询課程 |
| Course findCourse(int courseid)显示具体课程信息 |
| List<Course> showCourse(int courseid)显示教师对应课程 |

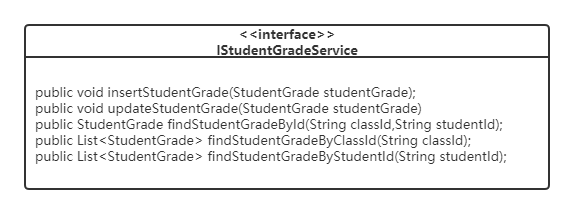
2.4 ITestPaperService接口设计



详细描述：

|  |
| --- |
| ITestPaperService考卷接口 |
| public void insertTestPaper(TestPaper testPaper);  新增考卷 |
| public void updateTestPaper(TestPaper testPaper);  更新考卷信息 |
| public TestPaper findTestPaperById(String testPaperId);  根据考卷ID查询考卷 |
| public List<TestPaper> findTestPaperByStartedTime(String time);  根据考卷開始時間查找考卷 |

2.5 IStudentGradeService接口设计



详细描述：

|  |
| --- |
| IStudentGradeService學生分數接口 |
| public void insertStudentGrade(StudentGrade studentGrade);  新增學生分數 |
| public void updateStudentGrade(StudentGrade studentGrade)  更新學生分數 |
| public StudentGrade findStudentGradeById(String classId,String studentId);  根据學生ID和課程ID查询學生分數 |
| public List<StudentGrade> findStudentGradeByClassId(String classId);  根据課程ID查找學生分數 |
| public IPage<AdminStudentgradeVo> getStudentGrade(int currentPage, int limit, String id);  根据學生ID查找學生分數 |

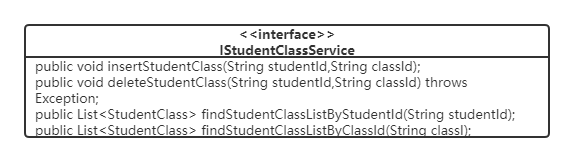
2.6 INoticeService接口设计



详细描述：

|  |
| --- |
| INoticeService通知接口 |
| public void insertNotice(Notice notice);  新增通知 |
| public void updateNotice(Notice notice);  更新通知信息 |
| public List<Notice> findNoticelistByClassId(String classId);  根据課程ID查询通知 |

2.7 IStudentClassService接口设计



详细描述：

|  |
| --- |
| IStudentClassService學生選課接口 |
| public void insertStudentClass(String studentId,String classId);  新增學生選課 |
| public void deleteStudentClass(String studentId,String classId) throws Exception;  删除學生選課 |
| public List<StudentClass> findStudentClassListByStudentId(String studentId);  根据學生ID查询學生選課 |
| public IPage<AdminStudentClassVo> getClassStudent(int currentPage, int limit, String id);  根据課程ID查询課程所有學生 |

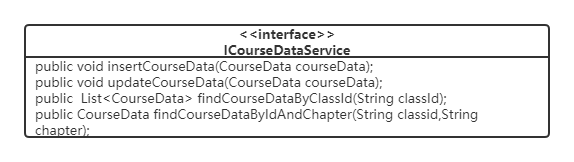
2.8 IAdministratorService接口设计



详细描述：

|  |
| --- |
| IAdminService管理員接口 |
| public boolean adminLogin(String uid);  管理員登录，根据uid |
| public Administrator adminLogin(String username,String password);  管理員登录，根据登录帐号和密码 |

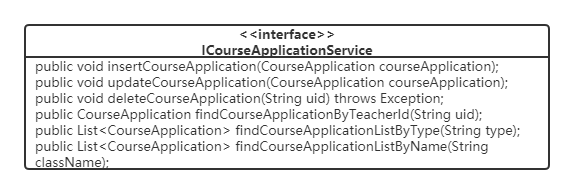
2.9 ICourseDataService接口设计



详细描述：

|  |
| --- |
| ICourseDataService課程资源接口 |
| public void insertCourseData(CourseData courseData);  新增課程资源 |
| public void updateCourseData(CourseData courseData);  更新課程资源 |
| public List<CourseData> findCourseDataByClassId(String classId);  根据課程ID查询課程资源 |
| public CourseData findCourseDataByIdAndChapter(String classid,String chapter);  根据課程ID和章節查询課程资源 |

2.10 ICourseApplicationService接口设计



详细描述：

|  |
| --- |
| ICourseApplicationService課程申請接口 |
| public void insertCourseApplication(CourseApplication courseApplication);  新增課程申請 |
| pic IPage<CourseApplication> getAllAduit(int currentPage, int limit);  獲取所有課程申請 |
| CourseApplication getAduitById(String id);  根据申請ID獲取申請詳細信息 |
| IPage<AdminCourseApplication> getAduitByKey(int currentPage, int limit, String key, String keyword);  搜索課程申請 |
| void sumbitAduit(String id);  通過审核 |
| void unsumbitAduit(String id);  不通過审核 |
| IPage<AdminCourseApplication> getFinishedAduit(int currentPage, int limit);  獲取已审核的申請 |
| IPage<AdminCourseApplication> getFinishedAduitByKey(int currentPage, int limit, String key, String keyword);  搜索已审核的課程申請 |
| IPage<AdminCourseApplication> getDealAduit(int currentPage, int limit);  獲取未审核的申請 |
| IPage<AdminCourseApplication> getDealAduitByKey(int currentPage, int limit, String key, String keyword);  搜索未审核的課程申請 |
| createCourse(String classname,String teachername,int teacherid,String intro,String beginning,String type)课程申请 |

2.11 InfoService接口设计

详细描述：

|  |
| --- |
| int createCourse(String classname,String teachername,int teacherid,String intro,String beginning,String type)实现查找老师对应学生的信息 |

2.12 业务接口实现类汇总

|  |  |
| --- | --- |
| 接口 | 实现类 |
| ITeacherService | TeacherServiceImpl |
| IStudentService | StudentServiceImpl |
| ICourseService | CourseServiceImpl |
| ITestPaperService | TestPaperServiceImpl |
| IStudentGradeService | StudentGradeImpl |
| INoticeService | NoticeServiceImpl |
| IStudentClassService | StudentClassServiceImpl |
| IAdminService | AdminServiceImpl |
| ICourseDataService | CourseDataServiceImpl |
| InfoService | InfoServiceImpl |
| ICourseApplicationService | CourseApplicationImpl |

三、学生功能

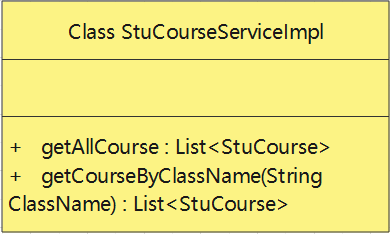
1、课程选择

**1.1、接口**

|  |
| --- |
| 接口 |
| StuICourseService |
| StuIStudentclassService |
| StuIStudentgradeService |

**1.2、实现类**

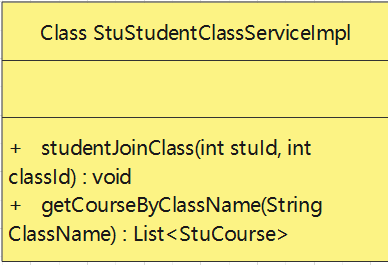
1.2.1 StuCourseServiceImpl描述



详细描述：

|  |
| --- |
| public List<StuCourse> getAllCourse()  获得所有课程 |
| 成员变量：无 |
| public List<StuCourse> getCourseByClassName(String ClassName)  根据输入的课程关键字查找相关课程 |

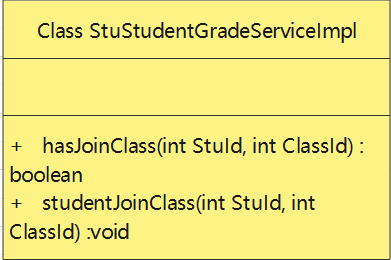
1.2.2、StuStudentClassServiceImpl**描述**



详细描述：

|  |
| --- |
| public void studentJoinClass(int stuId, int classId)  根据学生id和课程id将学生加入课程 |

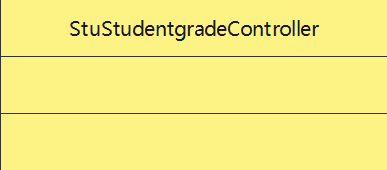
1.2.3、StuStudentGradeServiceImpl**描述**



详细描述：

|  |
| --- |
| public boolean hasJoinClass(int StuId, int ClassId)  根据学生id和课程id判断学生是否已经加入课程 |
| public void studentJoinClass(int StuId, int ClassId)  将学生加入课程 |

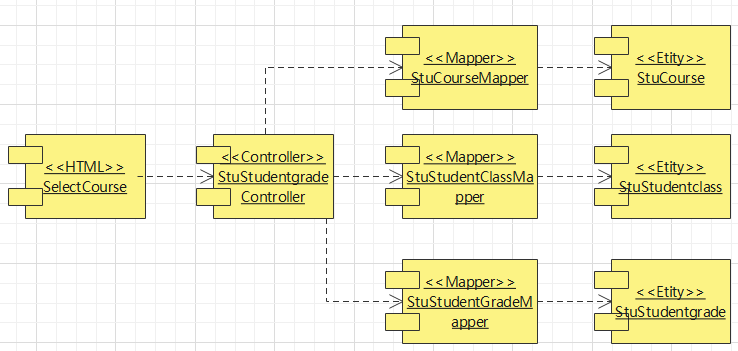
**1.3、控制器类图**



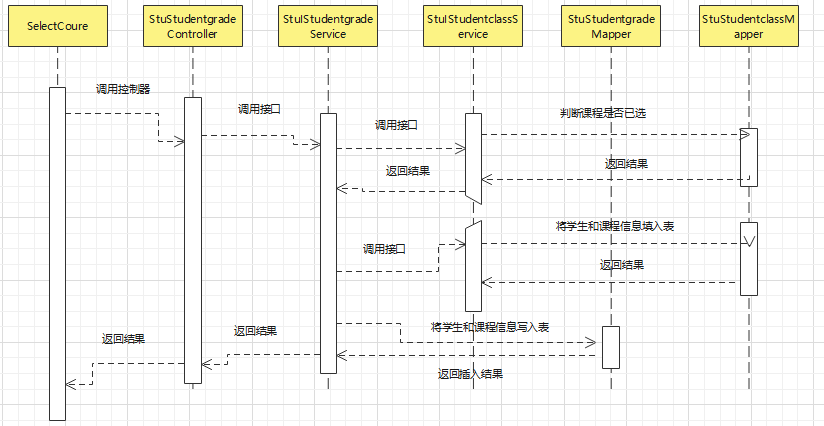
**1.3.1、控制方法**

|  |
| --- |
| public String studentJoinClass(int stuId, int classId)  根据学生id和课程id将学生加入课程 |

**1.4、类关系结构图**



**1.5、时序图**



**1.6、UI界面**



**1.7、接口说明文档**

1. 请求方式：Get｜HTTP

2. 请求地址：student/studentgrade/studentJoinClass

3. 接口功能说明：学生加入课程

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| stuId | 是 | 学号 |
| classId | 是 | 课程号 |

5. 结果数据：Json

{

status:value

}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 是否必须 | 参数说明 |
| status | 是 | 加入状态（200为加入成功，201为已加入课程，400为检索出错） |

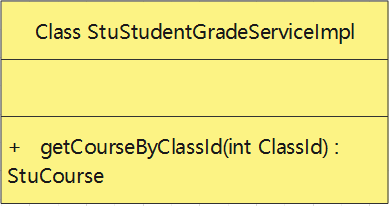
2、我的课程

**1.1、接口**

|  |
| --- |
| 接口 |
| StuICourseService |
| StuIStudentclassService |

**1.2、实现类**

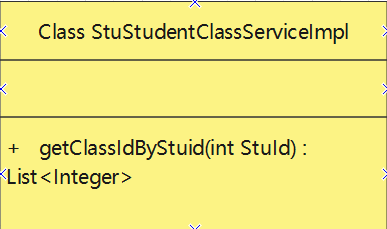
1.2.1 StuCourseServiceImpl**描述**



详细描述：

|  |
| --- |
| public StuCourse getCourseByClassId(int ClassId)  根据课程id查询课程信息 |

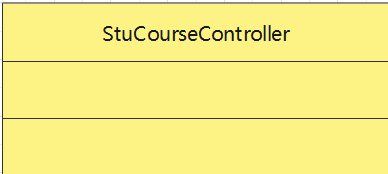
1.2.2、StuStudentClassServiceImpl**描述**



详细描述：

|  |
| --- |
| List<Integer> getClassIdByStuid(int StuId);  根据学生id查询课程id |

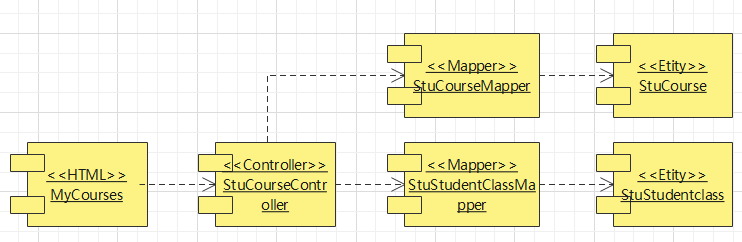
**1.3、控制器类图**



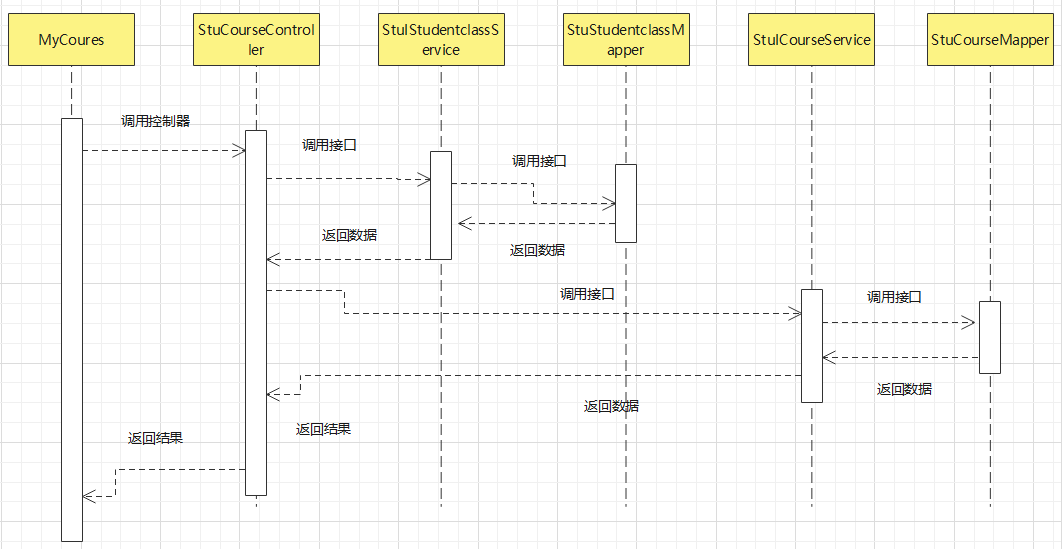
**1.3.1、控制方法**

|  |
| --- |
| public String getCourseInfoByStuId(int StuId)  根据学生id查询课程信息 |

**1.4、类关系结构图**



**1.5、时序图**



**1.6、UI界面**



**1.7、接口说明文档**

1. 请求方式：Get｜HTTP

2. 请求地址：student/course/getCourseInfoByStuId

3. 接口功能说明：查询学生已选的课程

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| StuId | 是 | 学号 |

5. 结果数据：Json

{

status:value，courses：value

}

6. 结果数据说明：

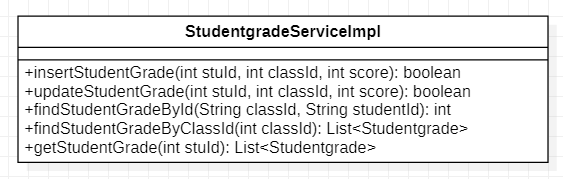
|  |  |  |
| --- | --- | --- |
| 参数 | 是否必须 | 参数说明 |
| status | 是 | 加入状态（200为查询成功，，400为检索出错） |
| courses | 是 | 返回的已选的课程信息 |

3. 成绩查询

**1.1 接口**

|  |
| --- |
| 接口 |
| IStudentgradeService |

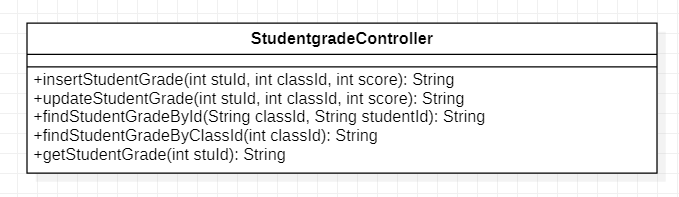
**1.2 实现类**



详细描述：

|  |
| --- |
| public boolean insertStudentGrade(int stuId,int classId,int score);  根据课程号和学号添加学生的分数 |
| public boolean updateStudentGrade(int stuId,int classId,int score);  更改学生的成绩 |
| public int findStudentGradeById(String classId,String studentId);  根据id和课程查询某位同学的分数 |
| public List<Studentgrade> findStudentGradeByClassId(int classId);  查询某个班级的所有学生的成绩 |
| public List<Studentgrade> getStudentGrade(int stuId);  查询某个学生的所有课程的成绩 |

**1.3 控制器类图**



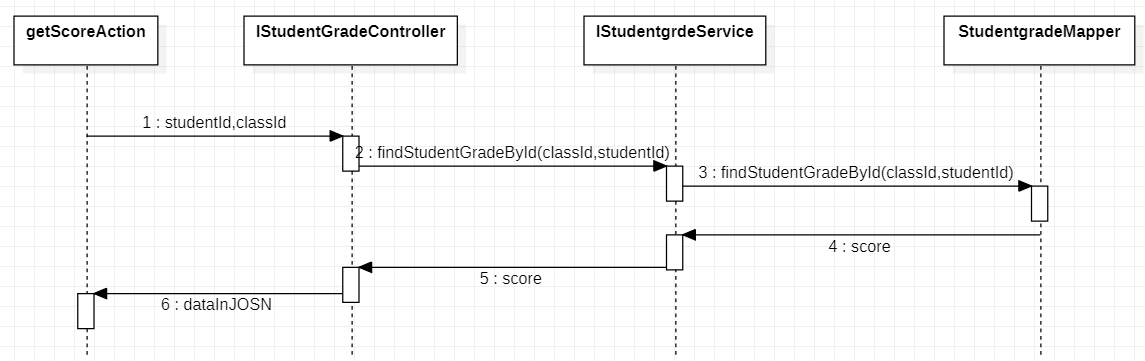
详细描述：

|  |
| --- |
| public String insertStudentGrade(int stuId,int classId,int score);  根据课程号和学号添加学生的分数 |
| public String updateStudentGrade(int stuId,int classId,int score);  更改学生的成绩 |
| public String findStudentGradeById(String classId,String studentId);  根据id和课程查询某位同学的分数 |
| public String findStudentGradeByClassId(int classId);  查询某个班级的所有学生的成绩 |
| public String getStudentGrade(int stuId);  查询某个学生的所有课程的成绩 |

**1.4 类关系结构图**



**1.5 时序图**



**1.6 UI设计**



**1.7 接口说明文档**

1. 请求方式：Post｜HTTP

2. 请求地址：student/student/studentgrade/ findStudentGradeById

3. 接口功能说明：学生查询成绩

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| stuId | 是 | 学号 |
| classId | 是 | 课程号 |

5. 结果数据：Json

{

status:value

data:score

}

6. 结果数据说明：

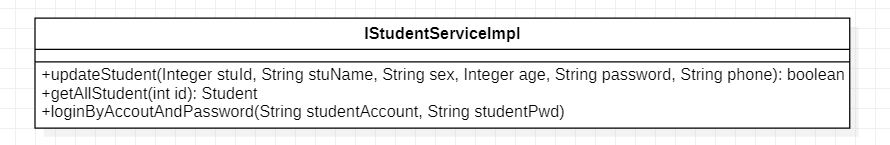
|  |  |  |
| --- | --- | --- |
| 参数 | 是否必须 | 参数说明 |
| status | 是 | 加入状态（200为查询成功，400为检索出错） |

4. 个人信息更改

**1.1 接口**

|  |
| --- |
| 接口 |
| IStudentService |

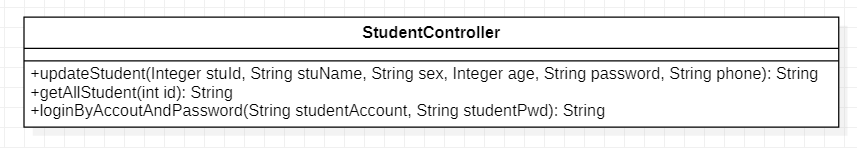
**1.2 实现类**



详细描述：

|  |
| --- |
| public boolean updateStudent(Integer stuId,String stuName,String sex,Integer age,String password,String phone)  更改学生对的个人信息和密码 |
| public Student getStudentById(int id)  获取学生的所有信息 |
| public boolean loginByAccoutAndPassword(String studentAccount,String studentPwd )  使用账号和密码登录 |

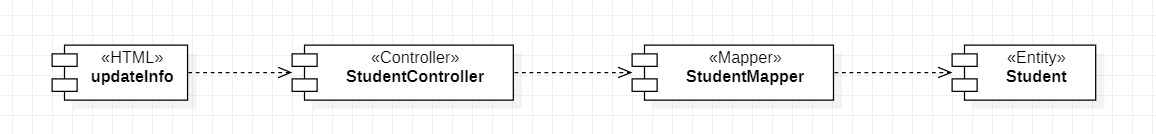
**1.3 控制器类图**



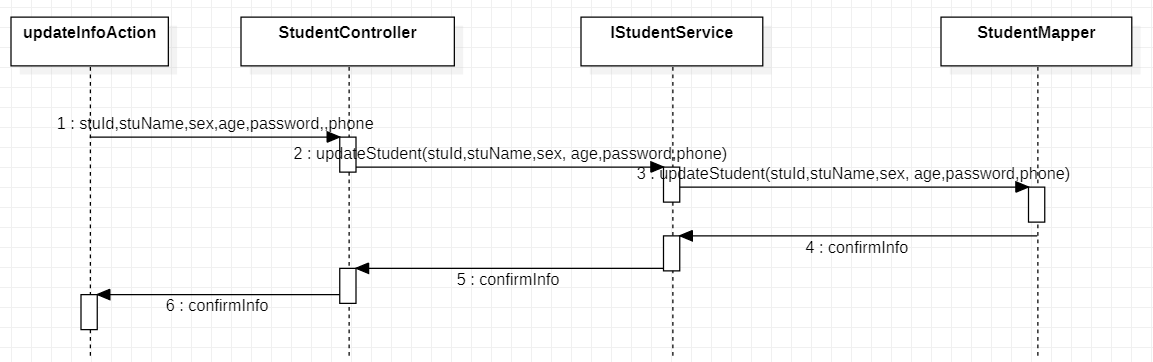
详细描述:

|  |
| --- |
| public String updateStudent(Integer stuId,String stuName,String sex,Integer age,String password,String phone)  更改学生对的个人信息和密码 |
| public String getStudentById(int id)  获取学生的所有信息 |
| public String loginByAccoutAndPassword(String studentAccount,String studentPwd )  使用账号和密码登录 |

**1.4 类关系结构图**



**1.5 时序图**



**1.6 UI设计**



**1.7 接口说明文档**

1. 请求方式：POST｜HTTP

2. 请求地址：student/student/updateStudent

3. 接口功能说明：学生更改个人信息

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| stuId | 是 | 学号 |
| stuName | 是 | 用户名 |
| sex | 是 | 性别 |
| age | 是 | 年龄 |
| password | 是 | 密码 |
| phone | 是 | 手机号 |

5. 结果数据：Json

{

status:value

data:confirmInfo

}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 是否必须 | 参数说明 |
| status | 是 | 加入状态（200为更改成功，400为更改失败） |

四、 教师模块设计

4.1.查找课程

1.1 接口

|  |
| --- |
| 接口 |
| ICourseService |

1.2 实现类

createCourse

详细描述：

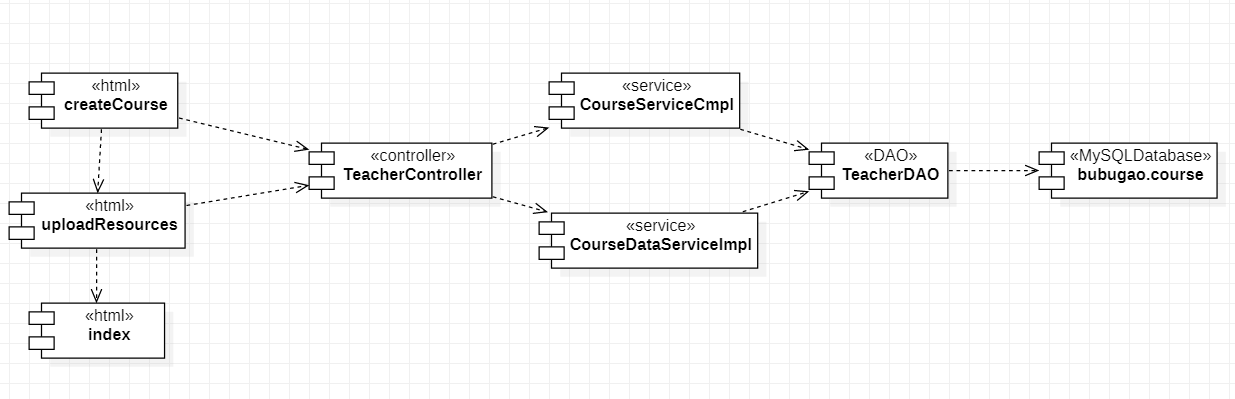
|  |
| --- |
| findCourse(int courseid)根据课程id查找课程信息 |
| List<Course>(String name,Int teaid)根据模糊查询课程名找到课程 |
| List<Coures> showCourse(int teacherid)根据教师id返回其所有课程 |
| public void insertCourse(Course course);  新增課程 |
| public void insertCourseData(CourseData ourseData);  新增課程资源 |
| public void updateCourseData(CourseData ourseData);  更新課程资源 |
| public Course findCourseById(String uid);  根据課程ID查询課程 |
| public List<Course>findCourselistByType(String type);  根据課程类型查询課程 |
| public List<Course> findCourselist(String uid,String courseName);  根据条件模糊条件查询課程列表 |
| public List<Course> findCourseByName(String name);  根据課程名查找課程 |
| public List<CourseData> findCourseDataByClassId(String classId);  根据課程ID查询課程资源 |
| public CourseData findCourseDataByIdAndChapter(String classid,String chapter);  根据課程ID和章節查询課程资源 |

**1.2、控制器类图**

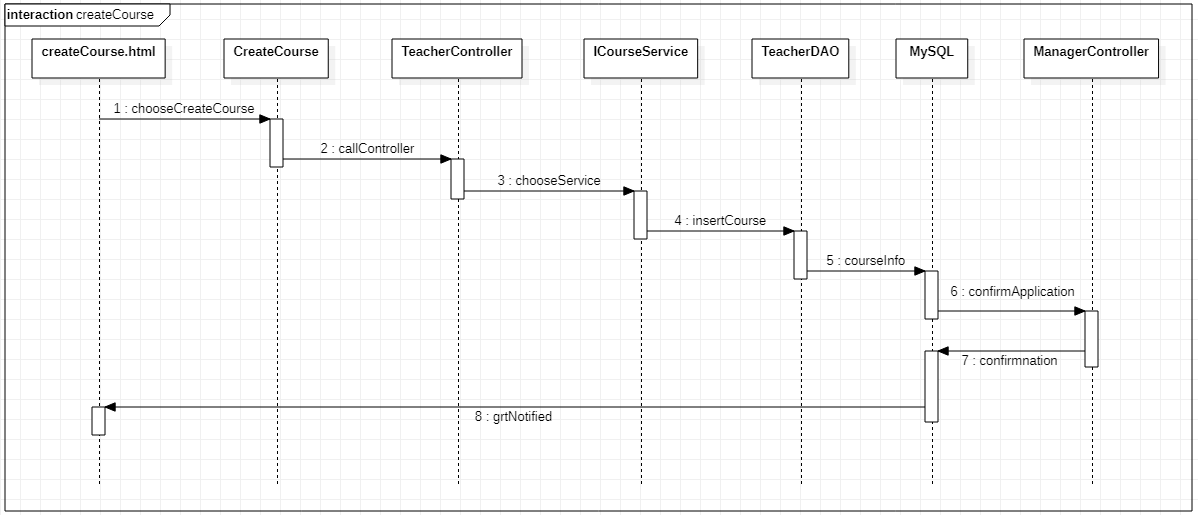
控制方法：CourseController

|  |
| --- |
| public Sting getCourse(int id)  public String findCourse(int id) |

1.3 类关系结构图



1.4 时序图



4.2.发布课程

1.1 接口

|  |
| --- |
| 接口 |
| ICourseapplicationService |

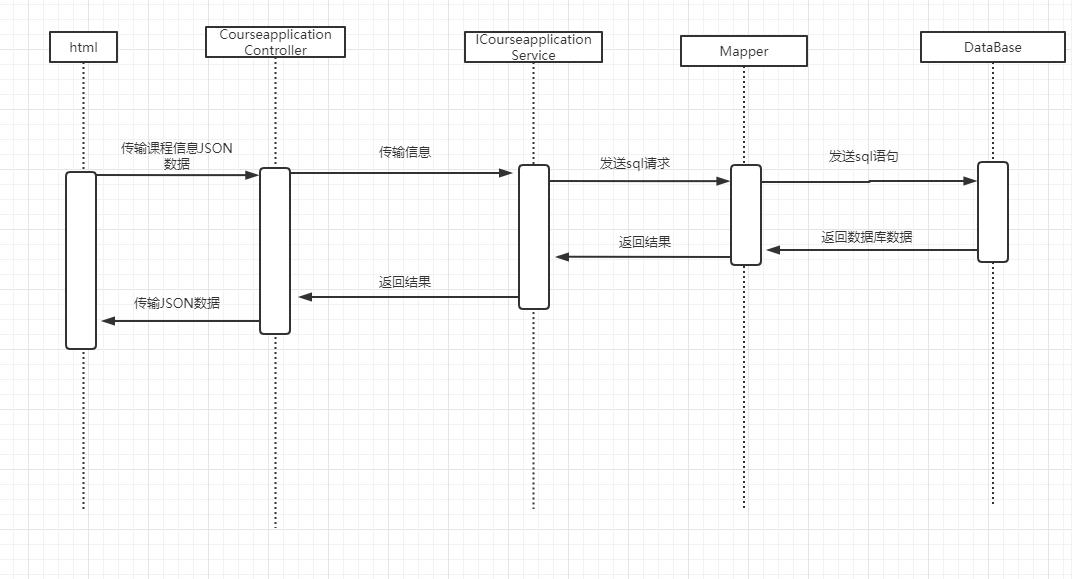
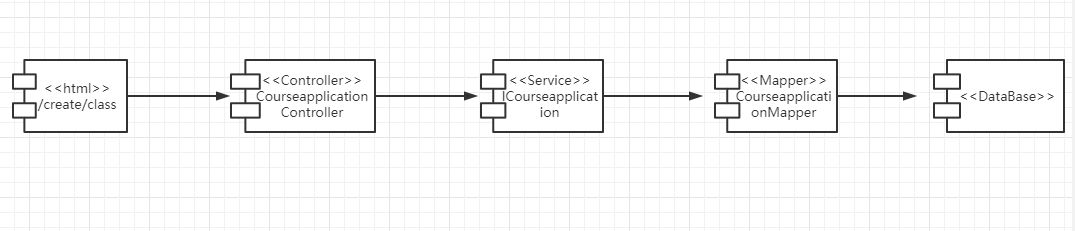
1.2 实现类

详细描述：CourseapplicationServiceImpl

|  |
| --- |
| findCourse(int courseid)根据课程id查找课程信息 |

控制方法：CourseapplicationController

|  |
| --- |
| public Sting createCourse(String classname,String teachername,int teacherid,String intro,String beginning,String type) |



4.3.查找学生信息课程

1.1 接口

|  |
| --- |
| 接口 |
| InfoService |

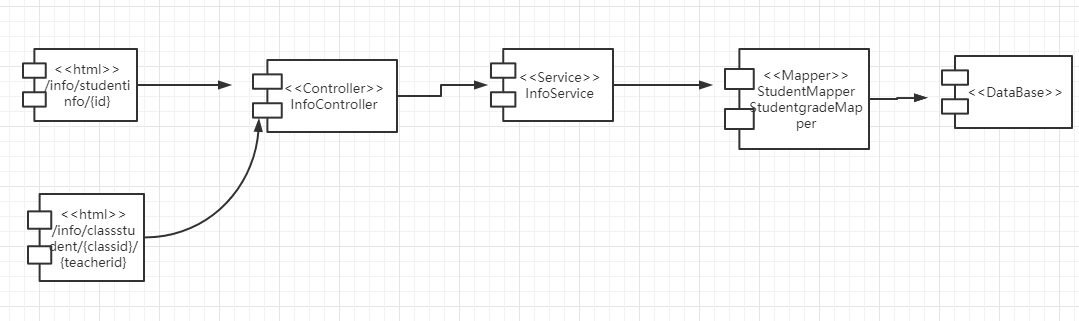
1.2 实现类InfoServiceImplement

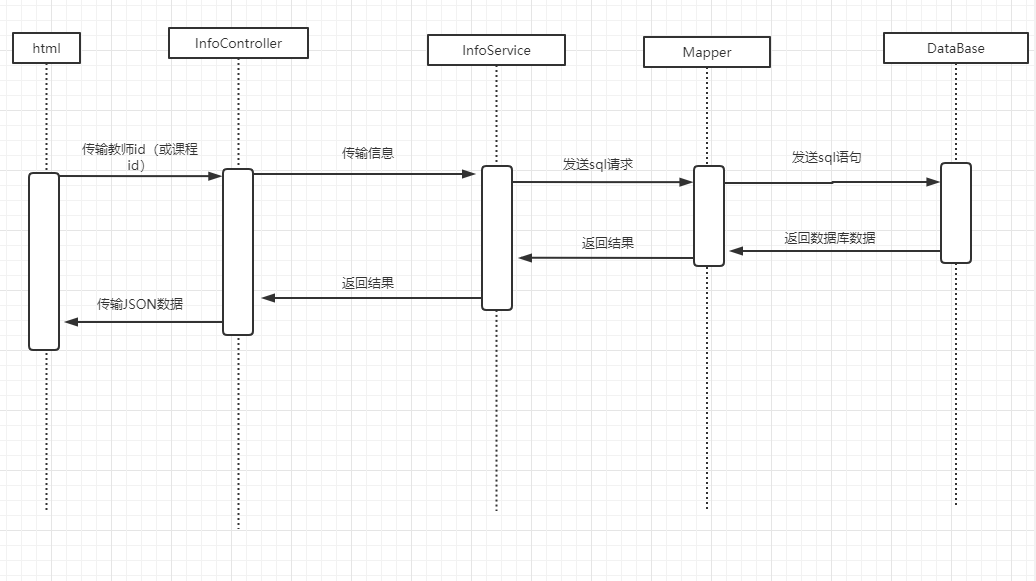
详细描述：

|  |
| --- |
| List<Mystudentinfo> statisticStuInfo(int teacherid)  List<Mystudentinfo> findStuInfoByClassid(int classid,int teacherid) |

控制方法

|  |
| --- |
| public Sting getStudent(int id)  pubulic String getStudentByCkass(int classid,int teacherid) |





4.4.教师课程业务

1.1 接口

|  |
| --- |
| 接口 |
| ITeacherService |

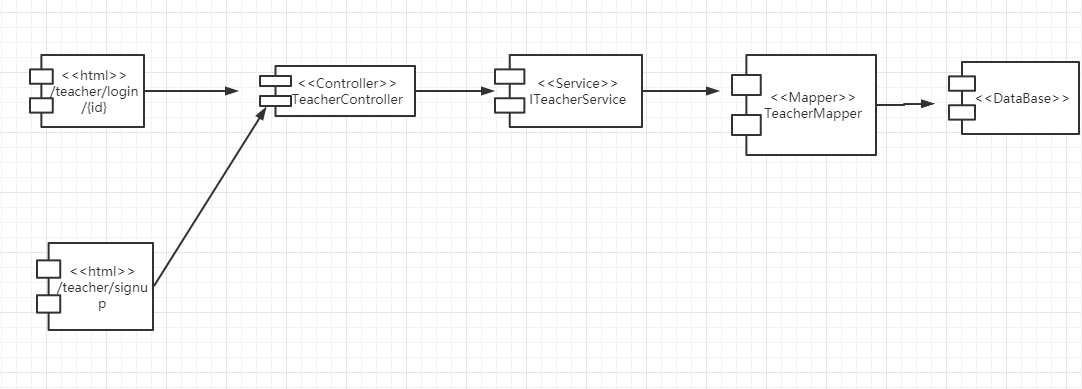
1.2 实现类

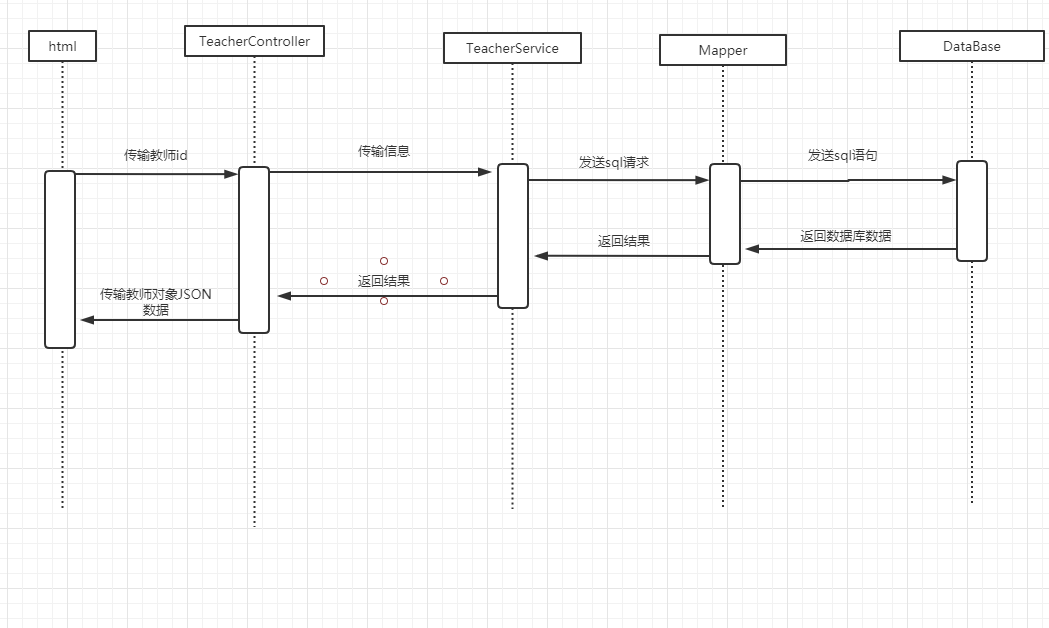
详细描述：TeacherServiceImpl

|  |
| --- |
| Teacher geyAccount（int id)  int signup(int teaid, String name,String phone,String password) |

控制方法：TeacherController

|  |
| --- |
| public Sting logint(int id)  pubulic Integer signup(int teaid, String name,String phone,String password) |





4.5教师登录及主界面UI













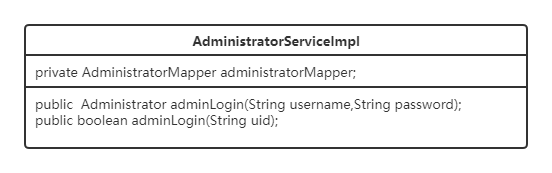
五、 管理员模块设计

5.1、管理員登錄

5.1.1 接口

|  |
| --- |
| 接口 |
| IAdministratorService |

5.1.2 实现类



详细描述：

|  |
| --- |
| AdministratorServiceImpl |
| private AdministratorMapper administratorMapper; |
| public Administrator adminLogin(String username,String password);  利用用戶名和密碼登錄 |
| public boolean adminLogin(String uid);  利用管理員uid進行登錄 |

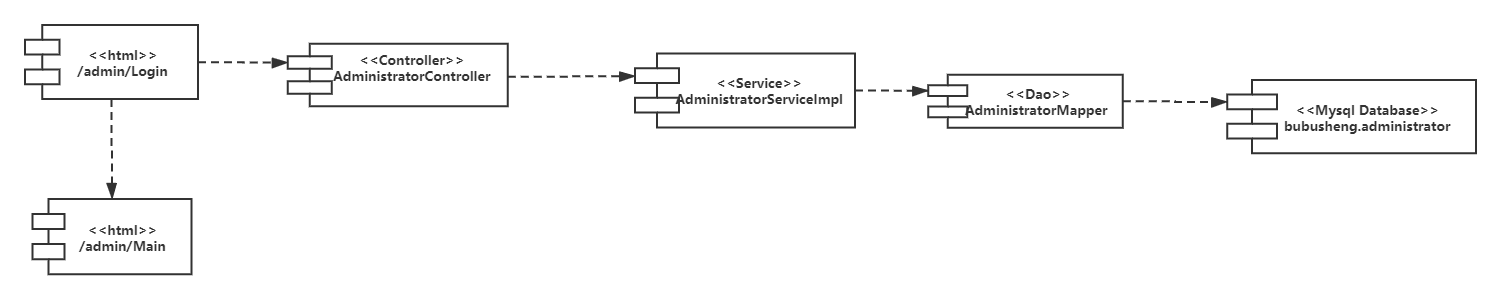
5.1.3控制器类图



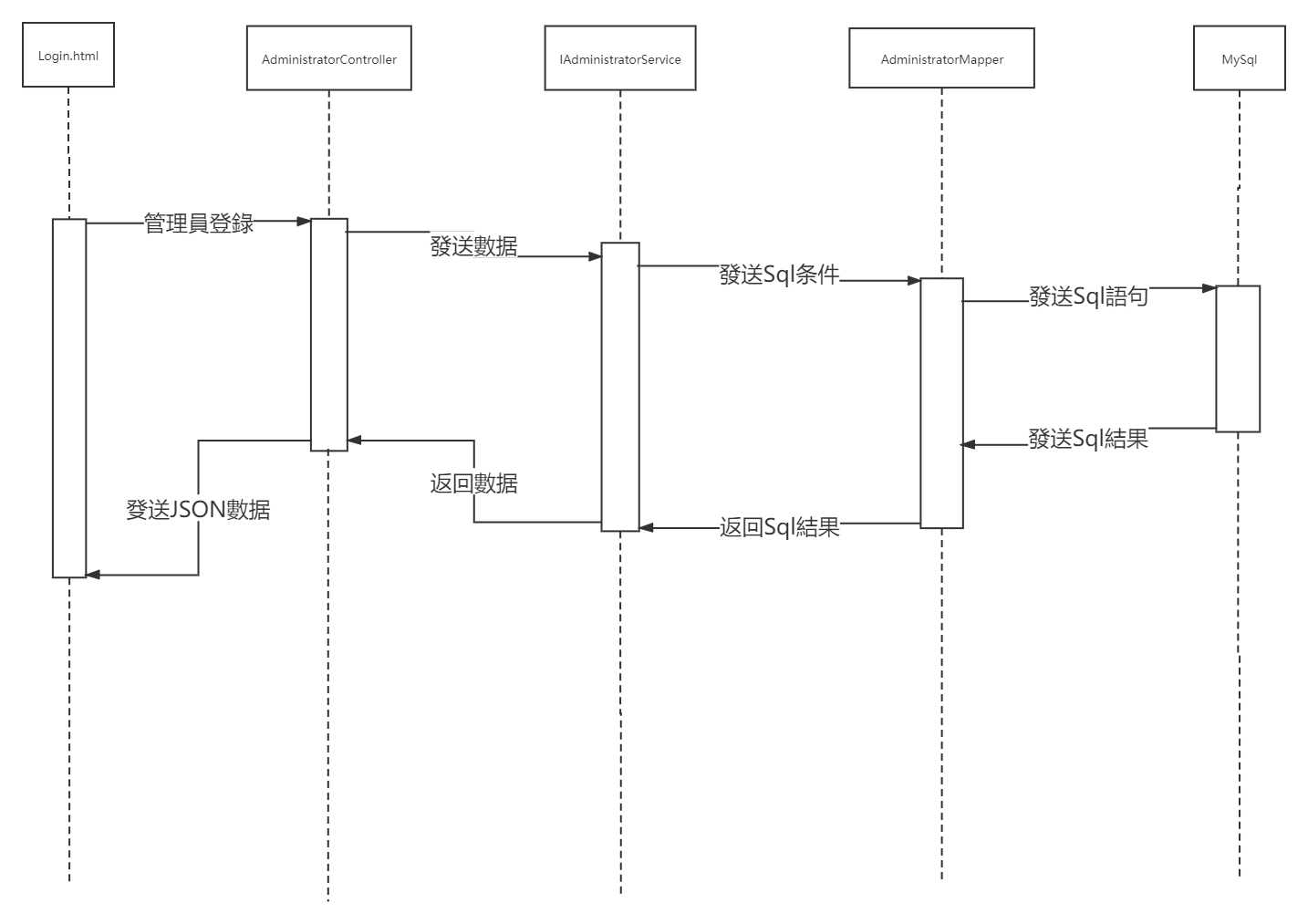
控制方法：

|  |
| --- |
| AdministratorController |
| private IAdministratorService administratorService; |
| public String login(String account, String password, String uid)  管理員登錄 |

5.1.4 类关系结构图



5.1.5 时序图



5.1.6**、UI界面**



5.1.7**、接口说明文档**

5.1.7.1登錄接口

1. 请求方式：Post｜HTTP

2. 请求地址：/admin/Login

3. 接口功能说明：管理員登錄

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| account | 是(使用帳號密碼登錄必須) | 帳號名稱 |
| password | 是(使用帳號密碼登錄必須) | 密碼 |
| uid | 否 | 管理員ID(可用作免登錄) |

5. 结果数据：Json

帳密登錄:{status:value,uid:value,isExist:value}

uid登錄:{status:value,isExist:value}

6. 结果数据说明：

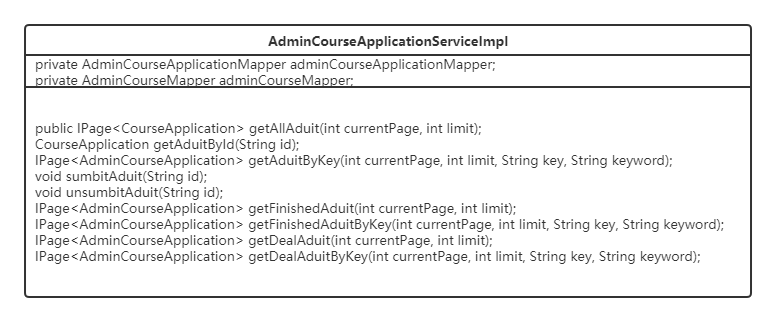
|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| uid | 否 | 管理員ID |
| isExist | 是 | true(登錄成功),false(登錄失敗) |

5.2、課程申請审核

5.2.1 接口

|  |
| --- |
| 接口 |
| IAdminCourseApplicationService |

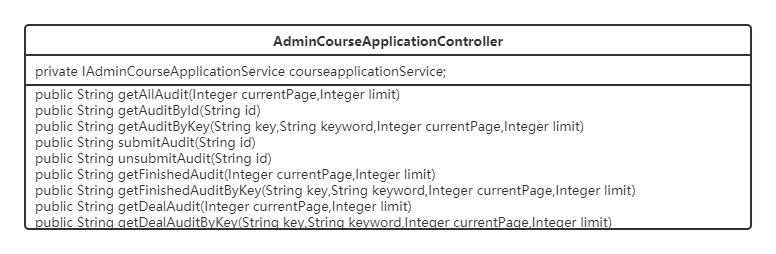
5.2.2 实现类



详细描述：

|  |
| --- |
| AdminCourseApplicationServiceImpl |
| private AdminCourseApplicationMapper adminCourseApplicationMapper; |
| private AdminCourseMapper adminCourseMapper; |
| public IPage<CourseApplication> getAllAduit(int currentPage, int limit);  獲取所有課程申請 |
| CourseApplication getAduitById(String id);  根据申請ID獲取申請詳細信息 |
| IPage<AdminCourseApplication> getAduitByKey(int currentPage, int limit, String key, String keyword);  搜索課程申請 |
| void sumbitAduit(String id);  通過审核 |
| void unsumbitAduit(String id);  不通過审核 |
| IPage<AdminCourseApplication> getFinishedAduit(int currentPage, int limit);  獲取已审核的申請 |
| IPage<AdminCourseApplication> getFinishedAduitByKey(int currentPage, int limit, String key, String keyword);  搜索已审核的課程申請 |
| IPage<AdminCourseApplication> getDealAduit(int currentPage, int limit);  獲取未审核的申請 |
| IPage<AdminCourseApplication> getDealAduitByKey(int currentPage, int limit, String key, String keyword);  搜索未审核的課程申請 |

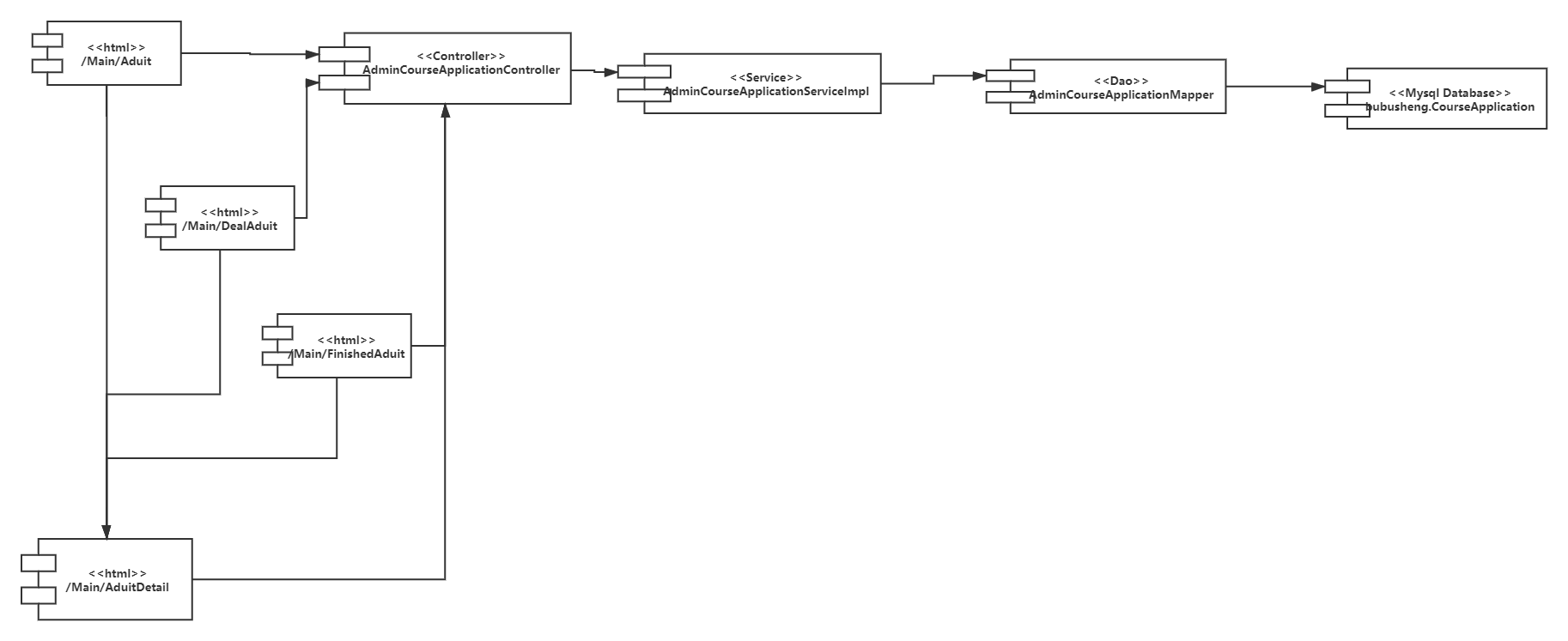
5.2.3控制器类图



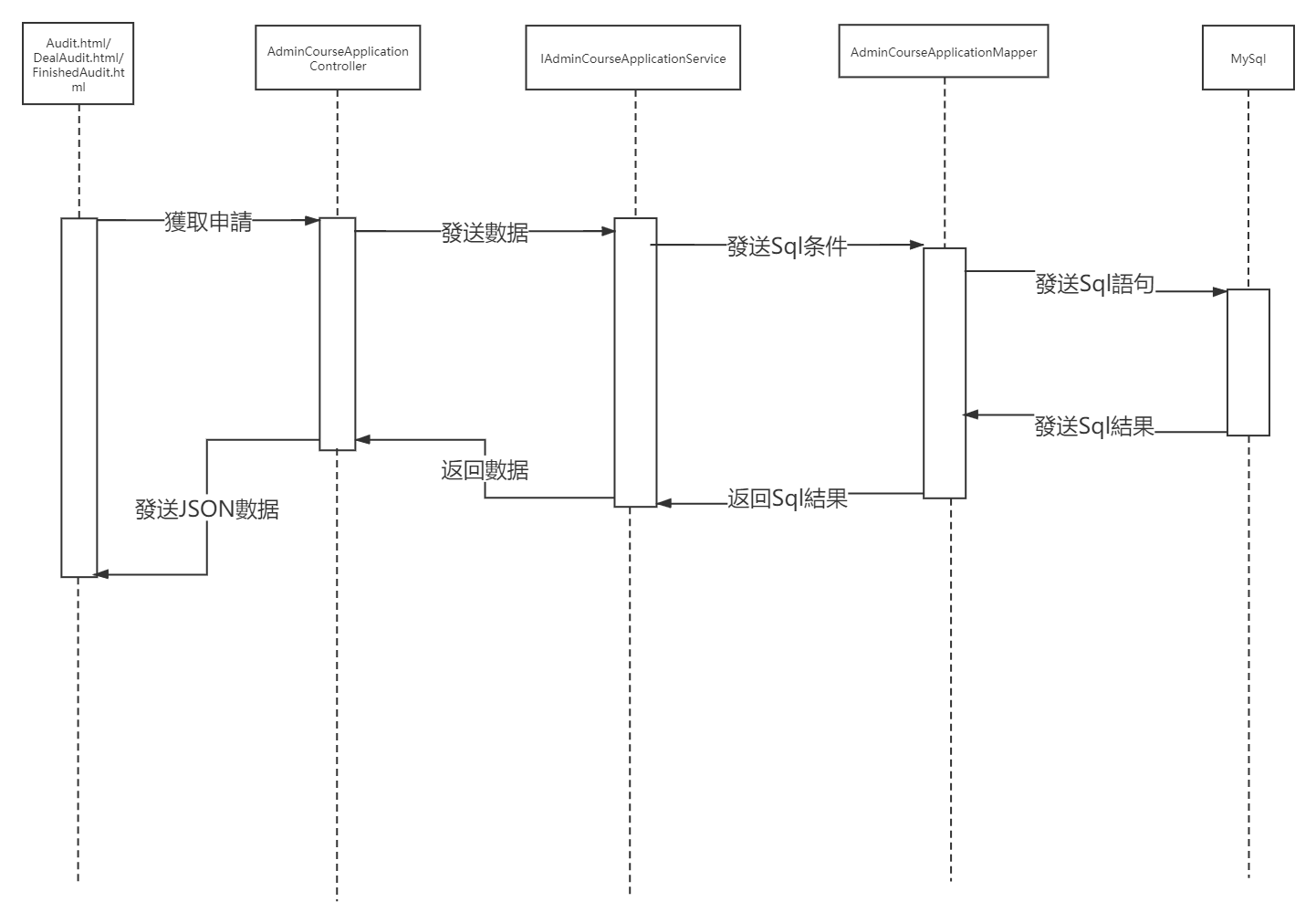
控制方法：

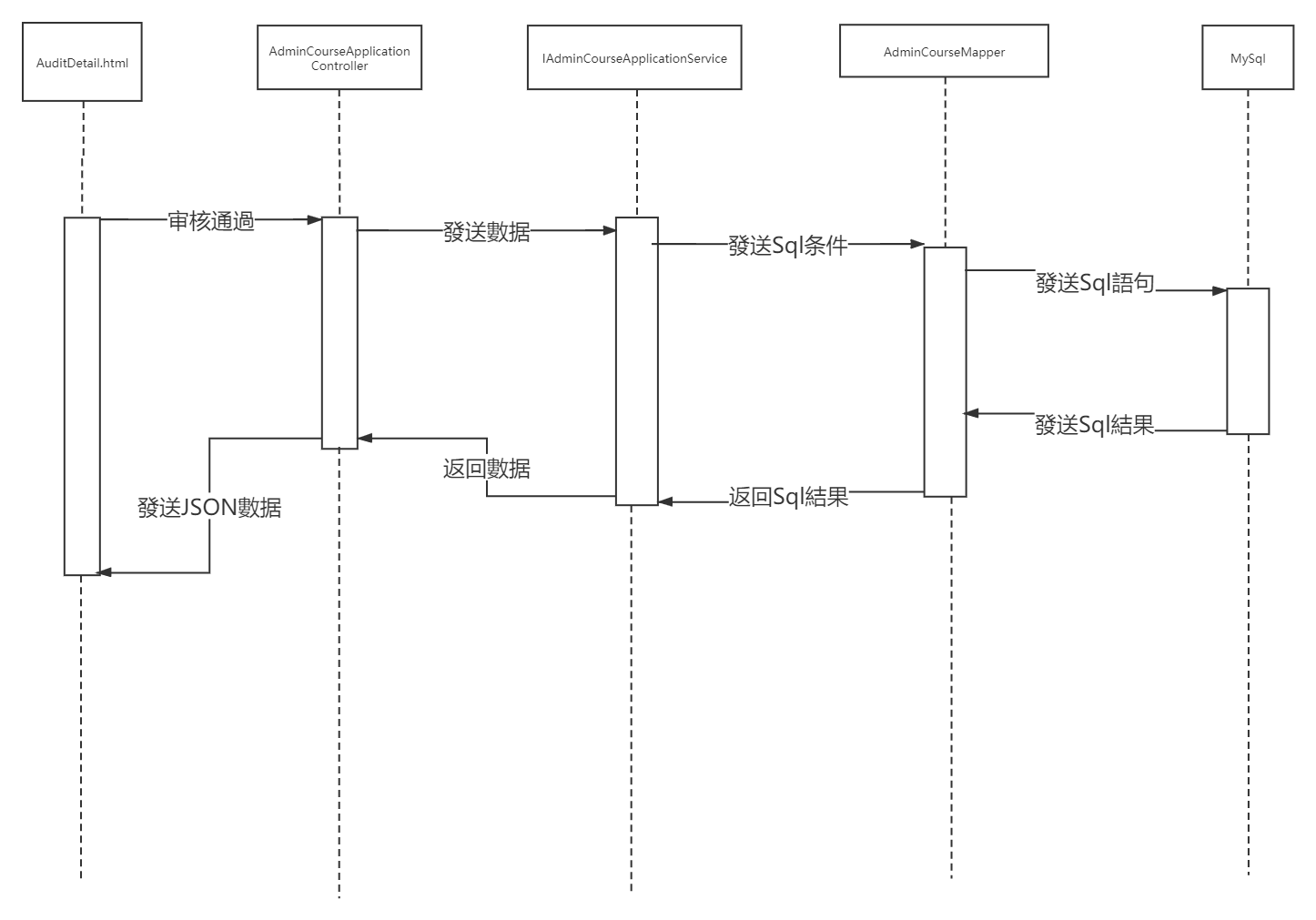
|  |
| --- |
| AdminCourseApplicationController |
| private IAdminCourseApplicationService courseapplicationService; |
| public String getAllAudit(Integer currentPage,Integer limit)  獲取所有課程申請 |
| public String getAuditById(String id)  根据申請ID獲取申請詳細信息 |
| public String getAuditByKey(String key,String keyword,Integer currentPage,Integer limit)  搜索課程申請 |
| public String submitAudit(String id)  通過审核 |
| public String unsubmitAudit(String id)  不通過审核 |
| public String getFinishedAudit(Integer currentPage,Integer limit)  獲取已审核的申請 |
| public String getFinishedAuditByKey(String key,String keyword,Integer currentPage,Integer limit)  搜索已审核的課程申請 |
| public String getDealAudit(Integer currentPage,Integer limit)  獲取未审核的申請 |
| public String getDealAuditByKey(String key,String keyword,Integer currentPage,Integer limit)  搜索未审核的課程申請 |

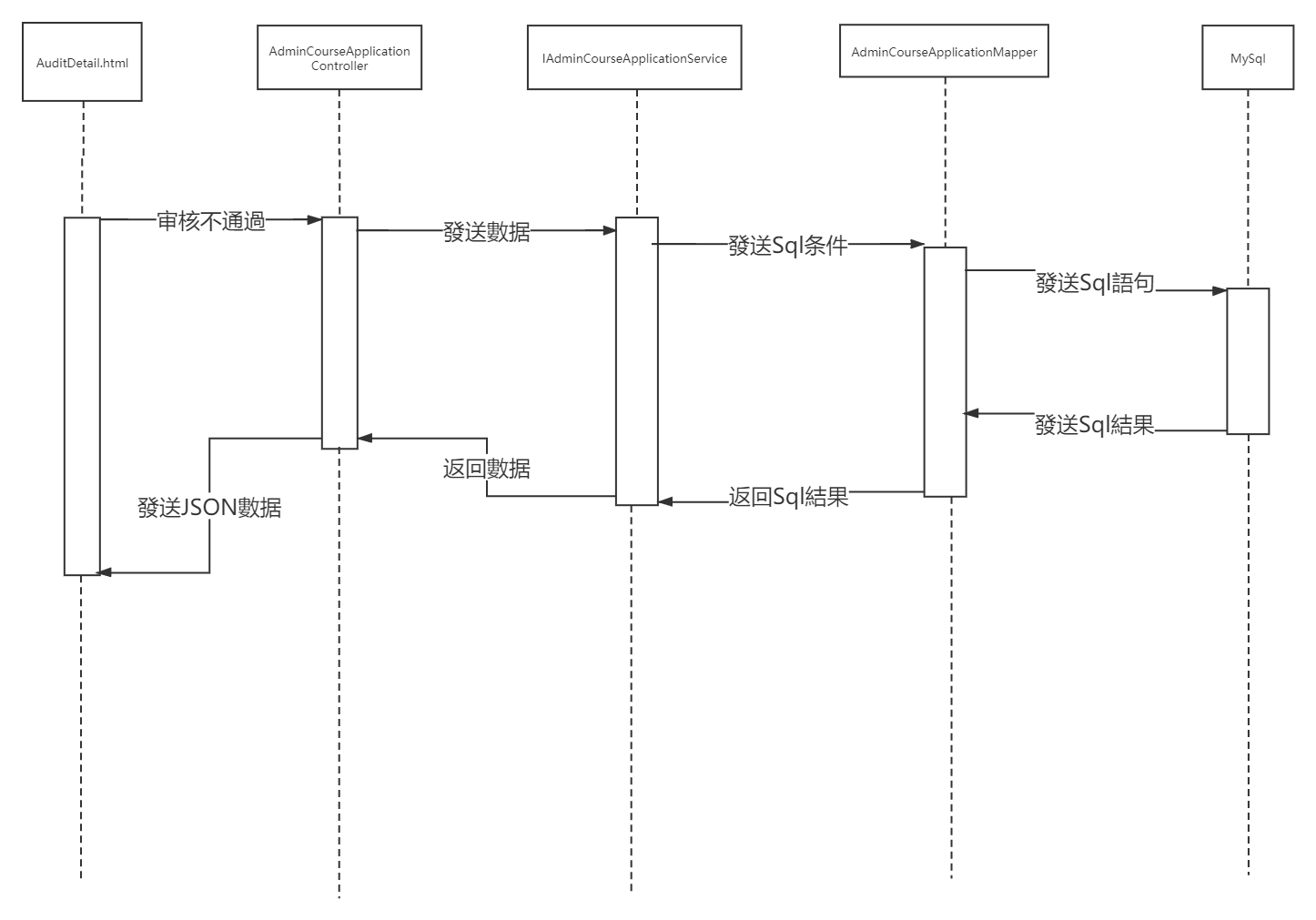
5.2.4 类关系结构图



5.2.5 时序图

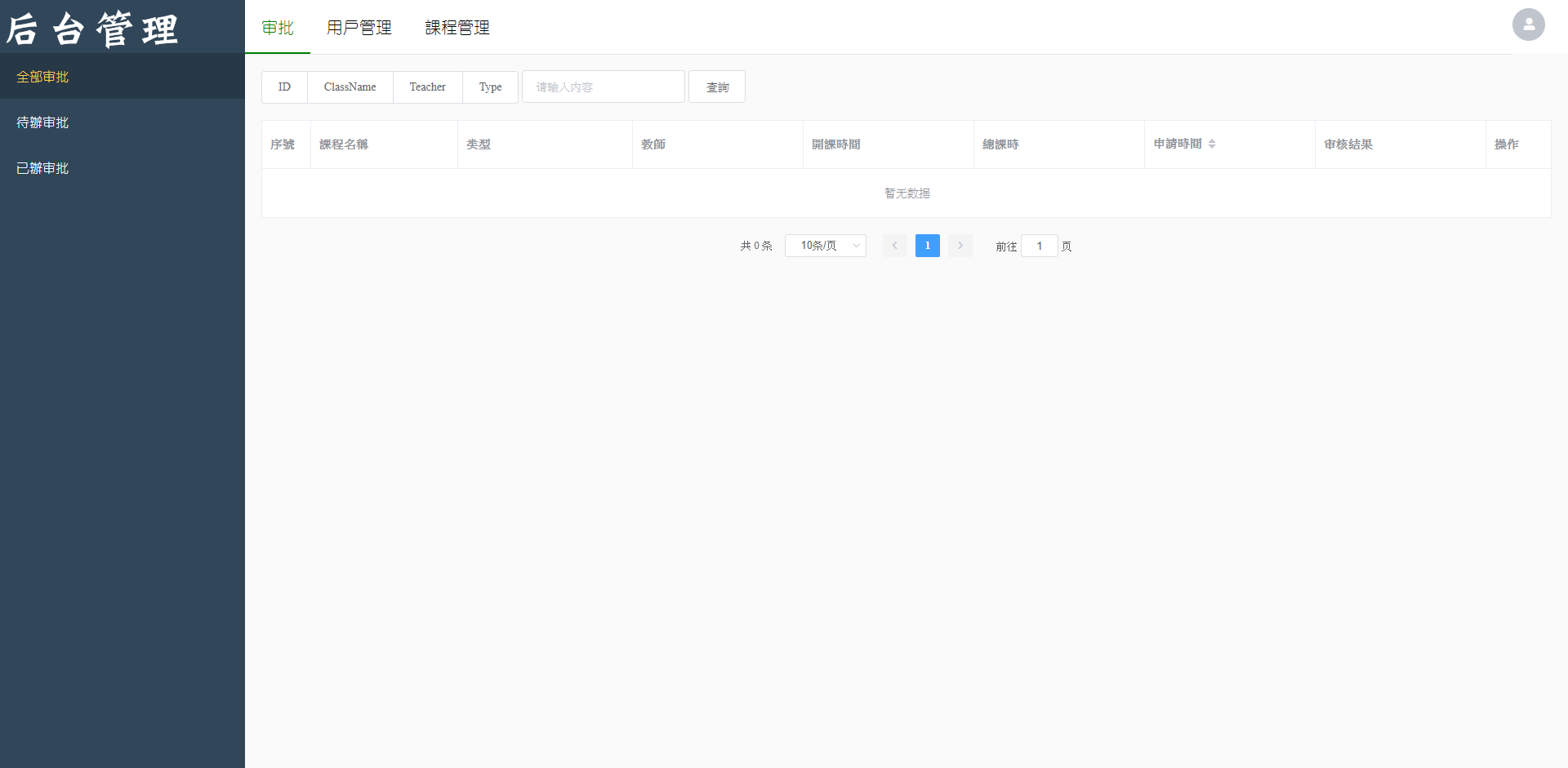




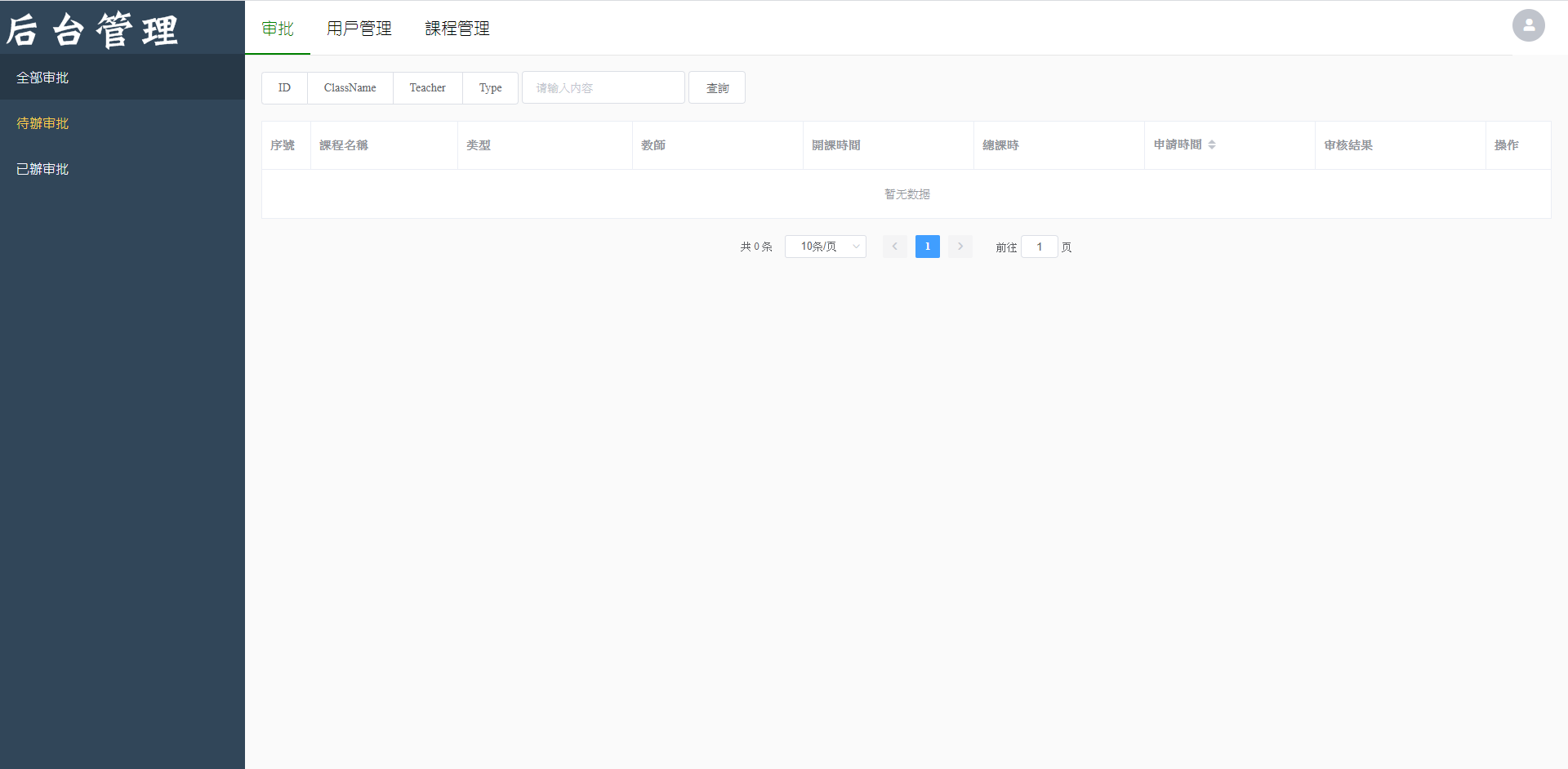


5.2.6**、UI界面**

5.2.6.1管理员全部审批界面



5.2.6.2 **管理员待辦审批界面**



5.2.6.3 **管理员已辦审批界面**



5.2.6.4**管理员审批信息界面**



5.2.7**、接口说明文档**

5.2.7.1全部审核數据獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAllAudit

3. 接口功能说明：獲取全部审核數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |
| total | 是 | 所有數据总數 |

5.2.7.2指定条件搜索审核數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAuditByKey

3. 接口功能说明：搜索审核

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
|  | 是 | 条件 |
|  | 是 | 搜索字 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |
| total | 是 | 所有數据总數 |

5.2.7.3獲取指定审核詳細數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAuditById

3. 接口功能说明：獲取指定审核詳細數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 审核ID |

5. 结果数据：Json

{status:value,data:{value}}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |

5.2.7.4待辨审核數据獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getDealAudit

3. 接口功能说明：獲取待辨审核數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |
| total | 是 | 所有數据总數 |

5.2.7.5指定条件搜索待辨审核數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getDealAuditByKey

3. 接口功能说明：搜索审核

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
| key | 是 | 条件 |
| keyword | 是 | 搜索字 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |
| total | 是 | 所有數据总數 |

5.2.7.6獲取指定审核詳細數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAuditById

3. 接口功能说明：獲取指定审核詳細數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 审核ID |

5. 结果数据：Json

{status:value,data:{value}}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |

5.2.7.7已辨审核數据獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getFinishedAudit

3. 接口功能说明：獲取已辨审核數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |
| total | 是 | 所有數据总數 |

5.2.7.8指定条件搜索已辨审核數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getFinishedAuditByKey

3. 接口功能说明：搜索审核

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
| key | 是 | 条件 |
| keyword | 是 | 搜索字 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |
| total | 是 | 所有數据总數 |

5.2.7.9獲取指定审核詳細數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAuditById

3. 接口功能说明：獲取指定审核詳細數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 审核ID |

5. 结果数据：Json

{status:value,data:{value}}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 审核數据 |

5.2.7.10审核通過接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/SubmitAudit

3. 接口功能说明：审核通過

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 課程ID |

5. 结果数据：Json

{status:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |

5.2.7.11审核不通過接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/unSubmitAudit

3. 接口功能说明：审核不通過

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 課程ID |

5. 结果数据：Json

{status:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |

5.3、教師管理

5.3.1 接口

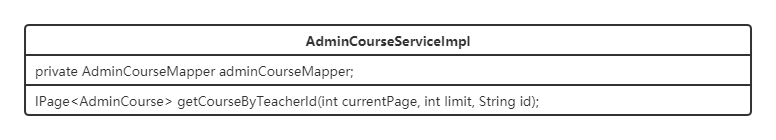
|  |
| --- |
| 接口 |
| IAdminTeacherService |
| IAdminCourseService |

5.3.2 实现类



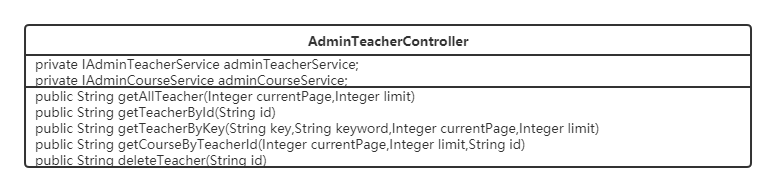
详细描述：

|  |
| --- |
| AdminTeacherServiceImpl |
| private AdminTeacherMapper adminTeacherMapper; |
| public void deleteTeacher(String id) throws Exception;  删除教師，将教師状态置为【删除】状态 |
| public IPage<Teacher> getAllTeacher(int currentPage,int limit);  獲取所有教師信息 |
| public Teacher getTeacherById(String id);  根据教師ID查询教師詳細信息 |
| public IPage<Teacher> getTeacherByKey(int currentPage,int limit,String key,String keyword);  根据条件模糊条件查询教師列表 |



|  |
| --- |
| AdminCourseServiceImpl |
| private AdminCourseMapper adminCourseMapper; |
| IPage<AdminCourse> getCourseByTeacherId(int currentPage, int limit, String id);  根据教師ID查询教授課程 |

5.3.3控制器类图



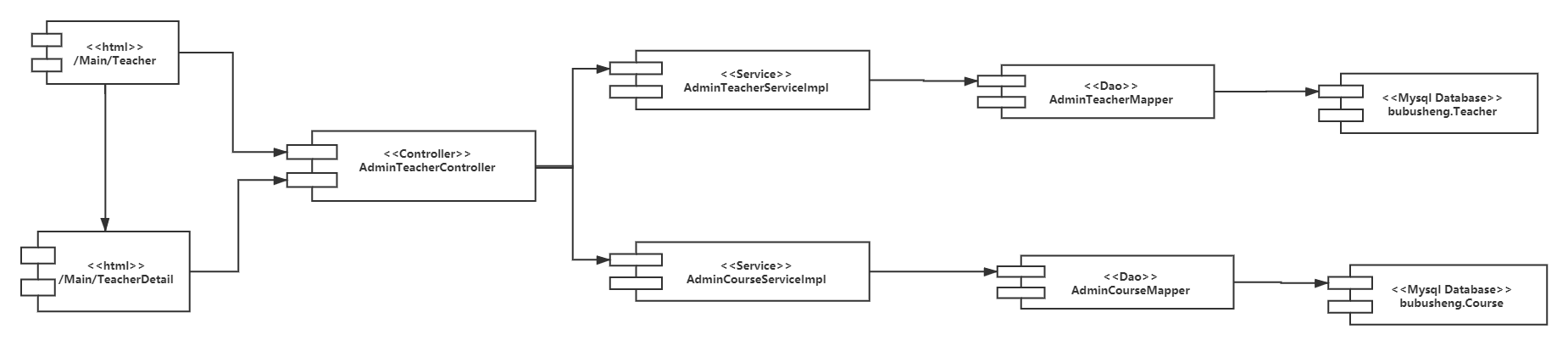
控制方法：

|  |
| --- |
| AdminTeacherController |
| private IAdminTeacherService adminTeacherService; |
| private IAdminCourseService adminCourseService; |
| public String getAllTeacher(Integer currentPage,Integer limit)  獲取所有教師信息 |
| public String getTeacherById(String id)  根据教師ID查询教師詳細信息 |
| public String getTeacherByKey(String key,String keyword,Integer currentPage,Integer limit)  根据条件模糊条件查询教師列表 |
| public String getCourseByTeacherId(Integer currentPage,Integer limit,String id)  根据教師ID查询教師詳細信息 |
| public String deleteTeacher(String id)  删除教師，将教師状态置为【删除】状态 |

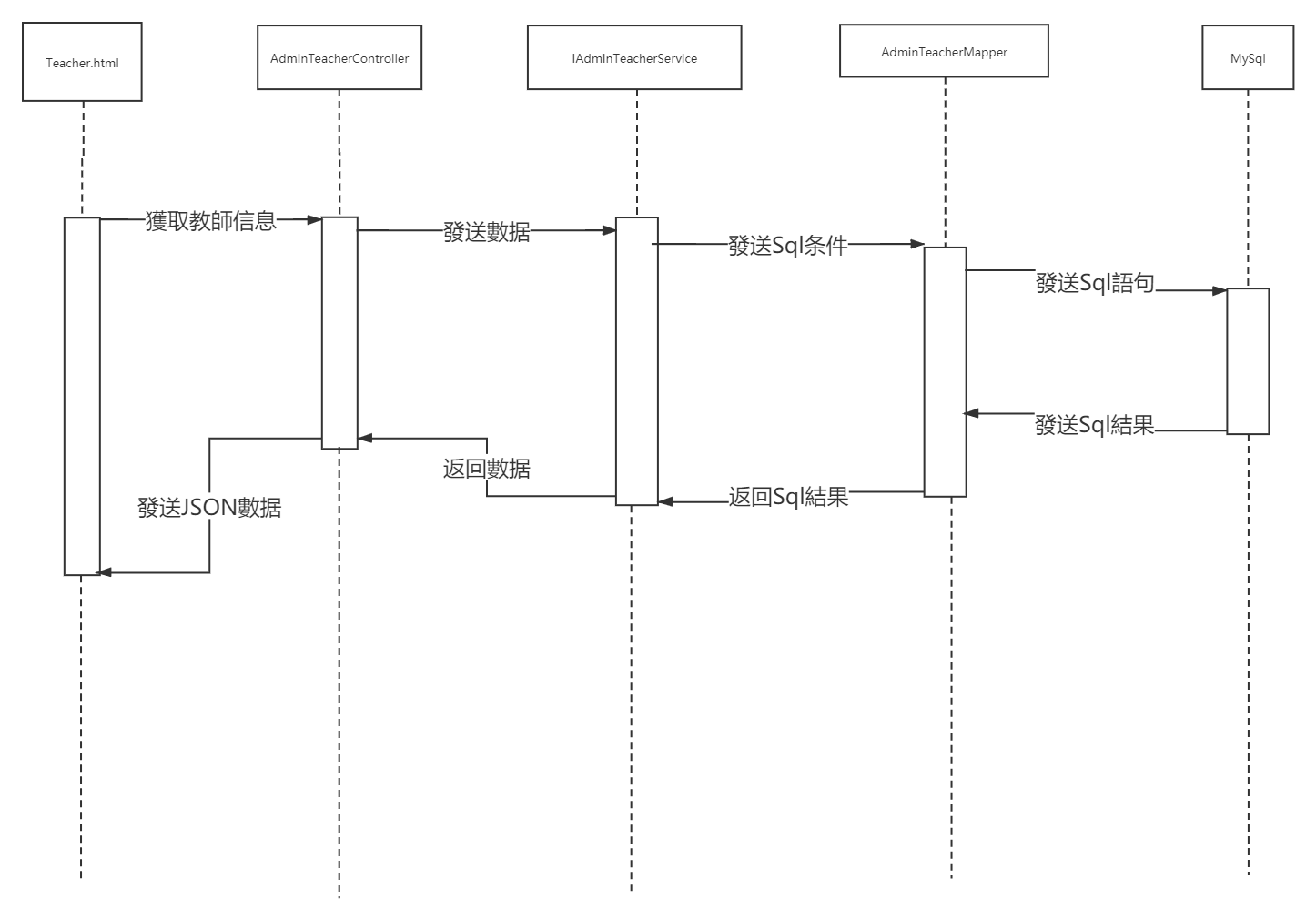
|  |
| --- |
|  |

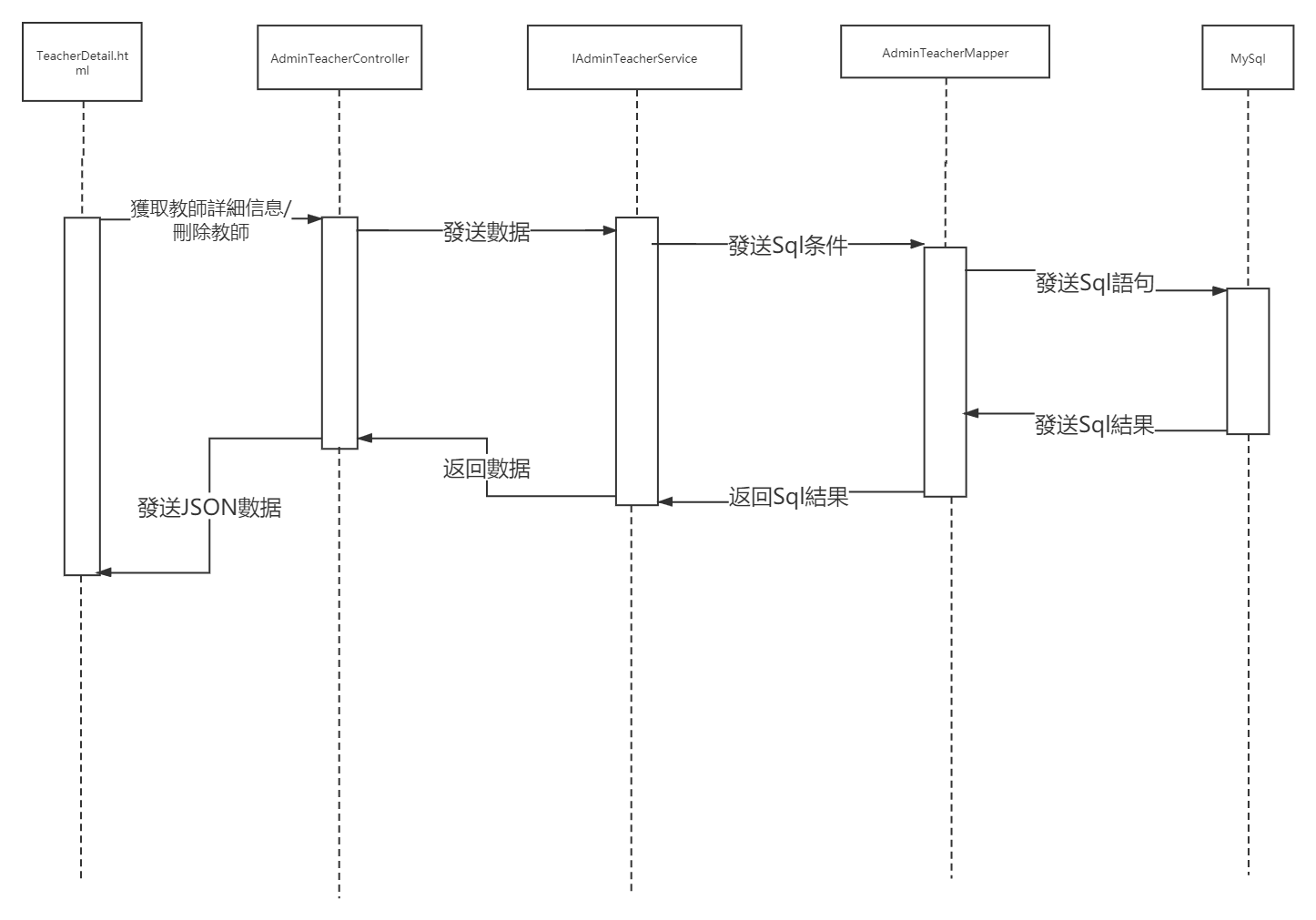
|  |
| --- |
|  |

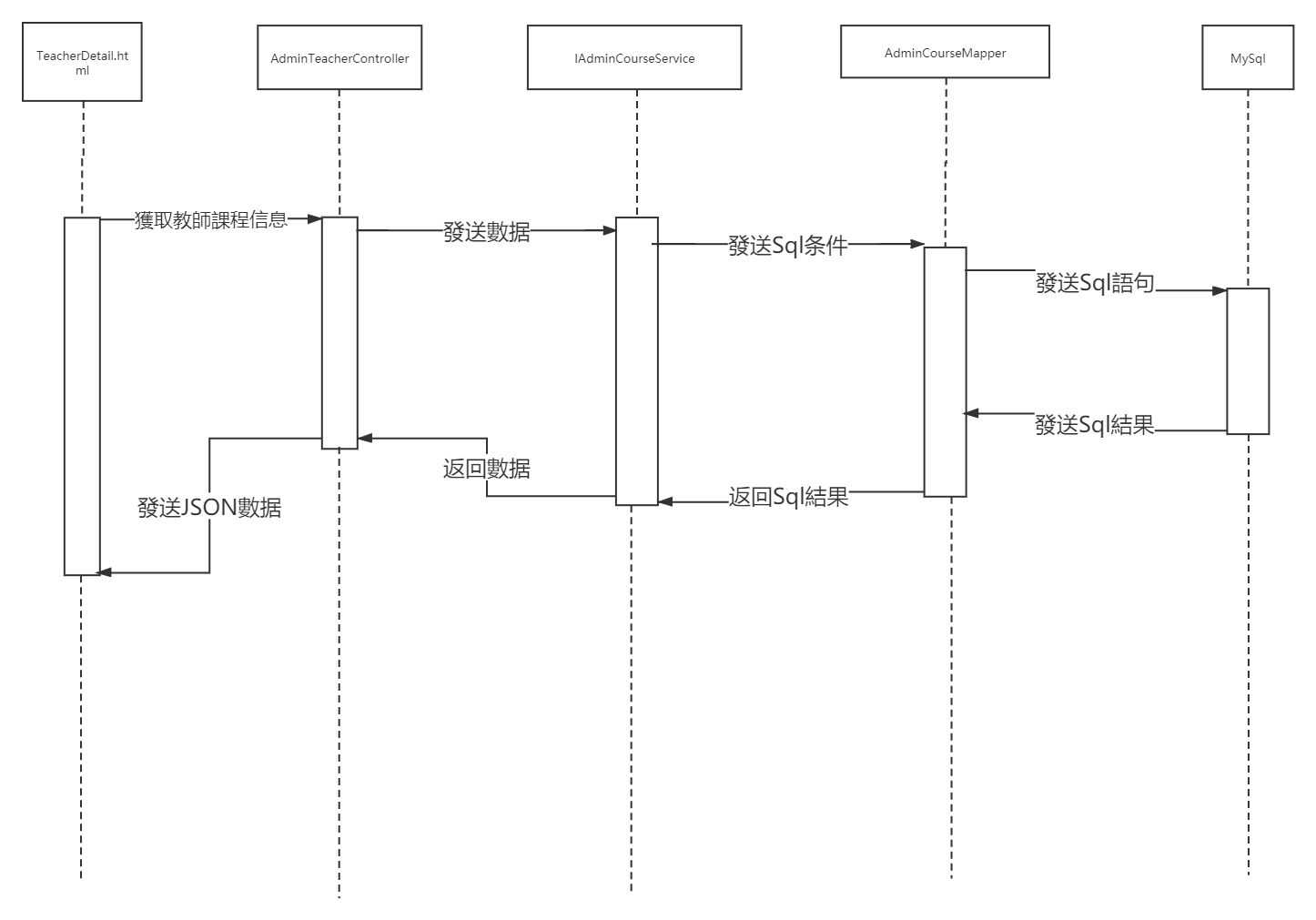
5.3.4 类关系结构图



5.3.5 时序图

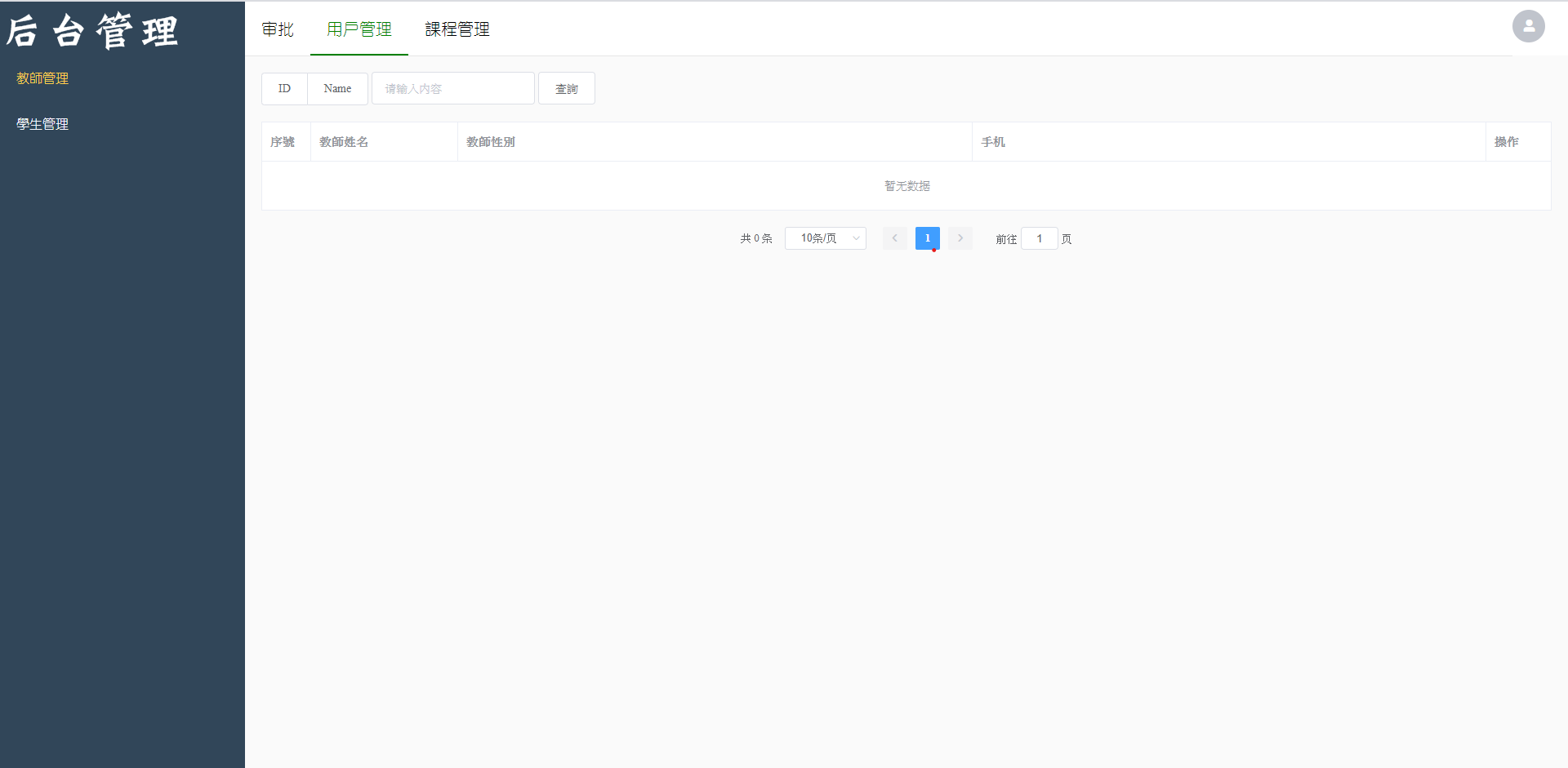






**5.3.6、UI界面**

5.3.6.1 管理员教師管理界面



5.3.6.2 管理员教師信息界面



**5.3.7、接口说明文档**

5.3.7.1教師課程獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getCourseByTeacherId

3. 接口功能说明：獲取教師課程

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
| id | 是 | 教師ID |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 教師課程數据 |
| total | 是 | 所有數据总數 |

5.3.7.2刪除教師接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/deleteTeacher

3. 接口功能说明：刪除教師

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 學生ID |

5. 结果数据：Json

{status:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |

5.3.7.3教師數据獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAllTeacher

3. 接口功能说明：獲取教師數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 教師數据 |
| total | 是 | 所有數据总數 |

5.3.7.4指定条件搜索教師數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getTeacherByKey

3. 接口功能说明：搜索教師

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
| key | 是 | 条件 |
| keyword | 是 | 搜索字 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 教師數据 |
| total | 是 | 所有數据总數 |

5.3.7.5獲取指定教師詳細數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAuditById

3. 接口功能说明：獲取指定教師詳細數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 审核ID |

5. 结果数据：Json

{status:value,data:{value}}

6. 结果数据说明：

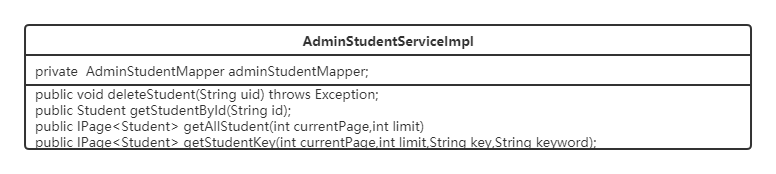
|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 教師數据 |

5.4、學生管理

5.4.1 接口

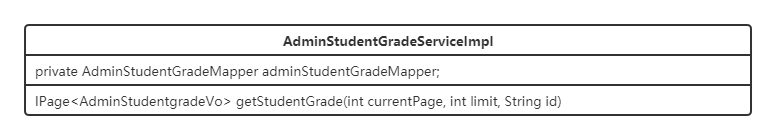
|  |
| --- |
| 接口 |
| ICourseService |
| CourseDataServiceImplICourseDataService |

5.4.2 实现类



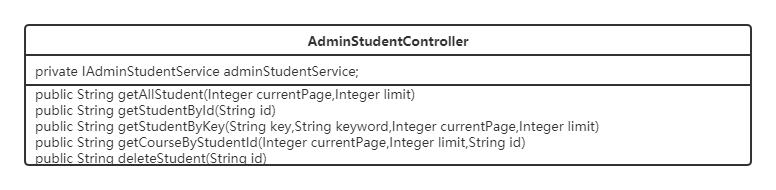
详细描述：

|  |
| --- |
| AdminStudentServiceImpl |
| private AdminStudentMapper adminStudentMapper; |
| public void deleteStudent(String uid) throws Exception;  删除學生，将學生状态置为【删除】状态 |
| public Student getStudentById(String id);  根据學生ID查询學生詳細信息 |
| public IPage<Student> getAllStudent(int currentPage,int limit)  獲取所有學生信息 |
| public IPage<Student> getStudentKey(int currentPage,int limit,String key,String keyword);  根据条件模糊条件查询學生列表 |



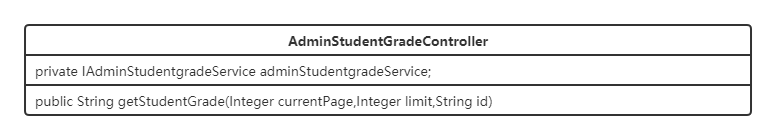
|  |
| --- |
| AdminStudentGradeServiceImpl |
| private AdminStudentGradeMapper adminStudentGradeMapper; |
| IPage<AdminStudentgradeVo> getStudentGrade(int currentPage, int limit, String id)  根据學生ID獲取學生分數 |

5.4.3控制器类图



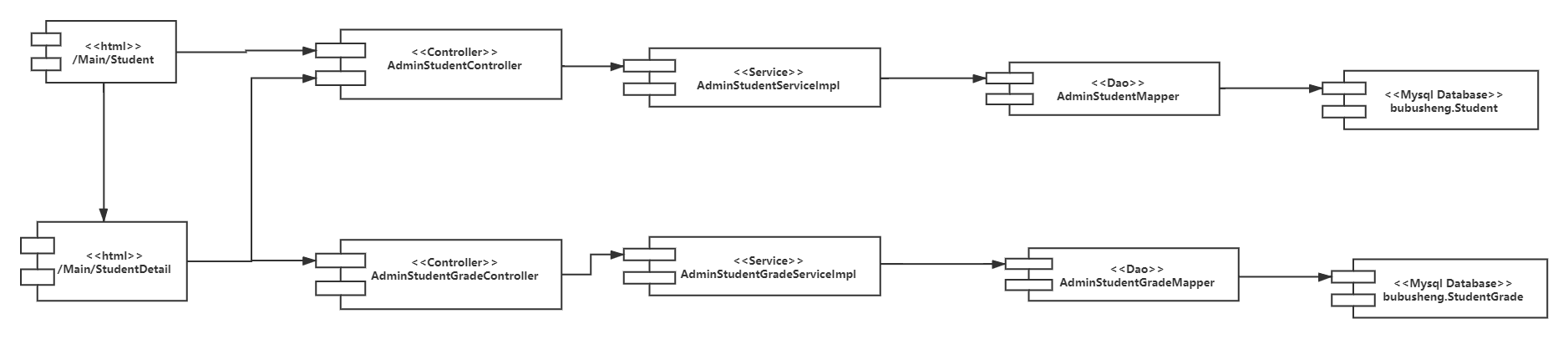
控制方法：

|  |
| --- |
| AdminStudentController |
| private IAdminStudentService adminStudentService; |
| public String getAllStudent(Integer currentPage,Integer limit)  獲取所有學生信息 |
| public String getStudentById(String id)  根据學生ID查询學生詳細信息 |
| public String getStudentByKey(String key,String keyword,Integer currentPage,Integer limit)  根据条件模糊条件查询學生列表 |
| public String getCourseByStudentId(Integer currentPage,Integer limit,String id)  根据學生ID查询學生詳細信息 |
| public String deleteStudent(String id)  删除學生，将學生状态置为【删除】状态 |

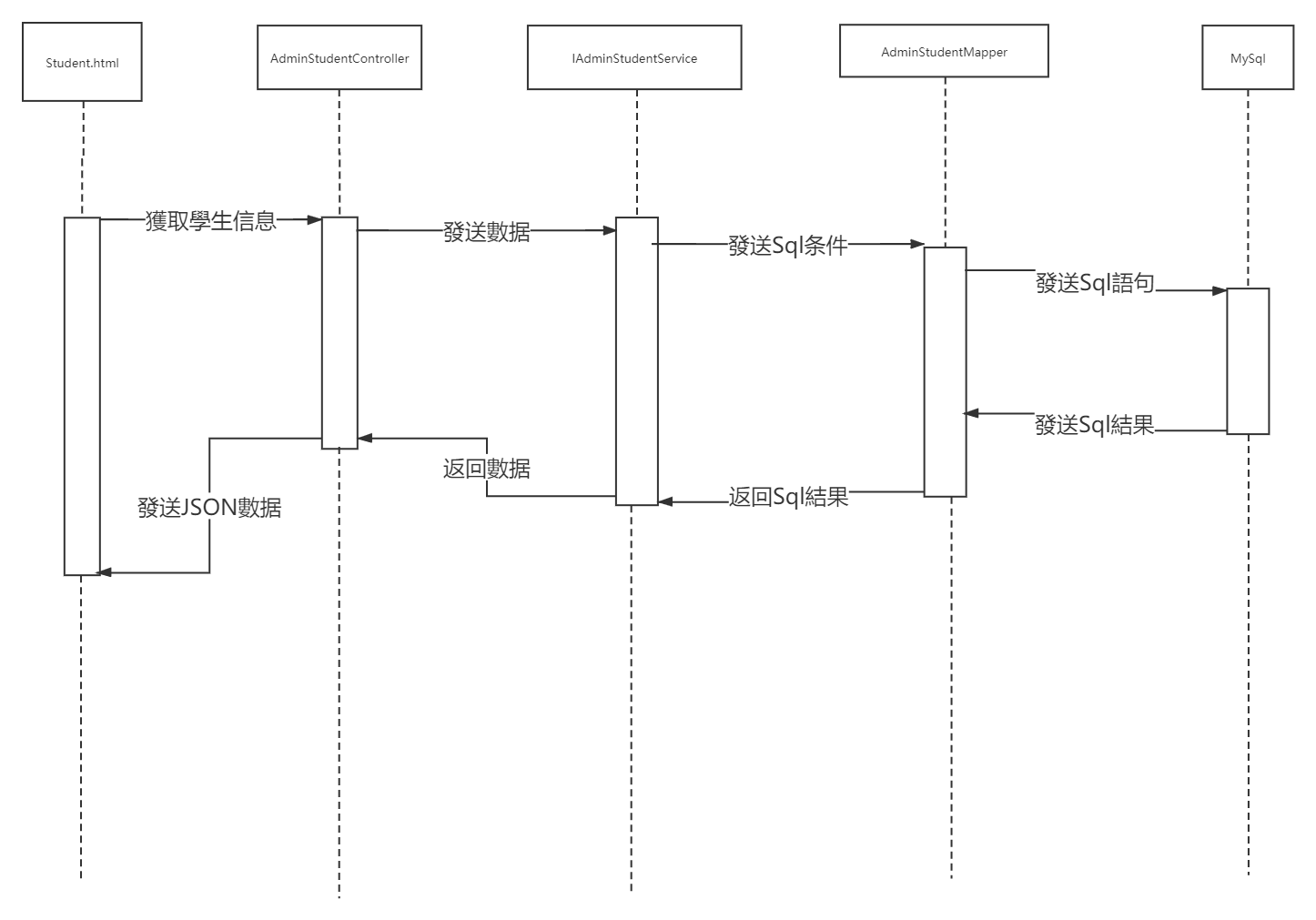


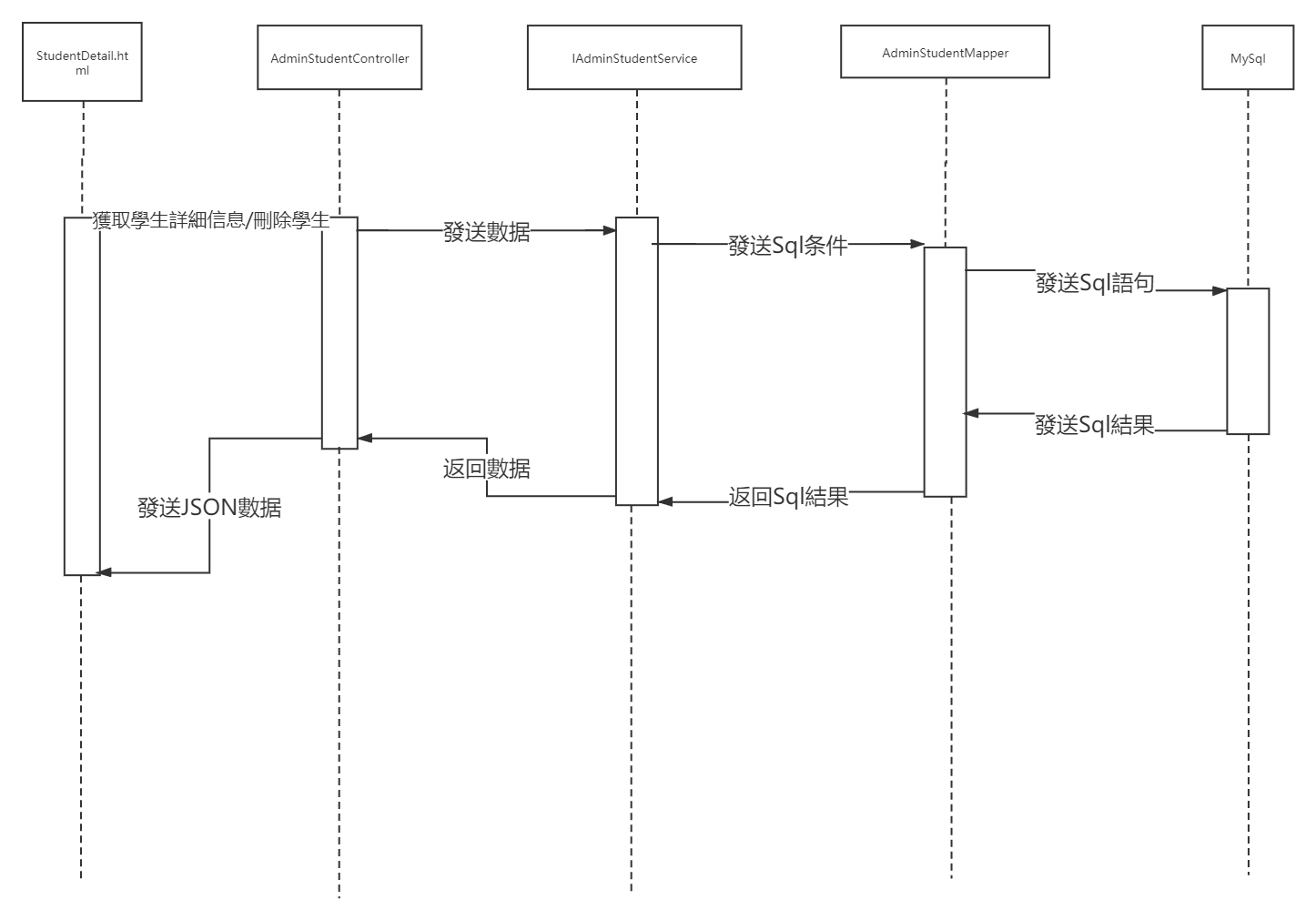
|  |
| --- |
| AdminStudentGradeController |
| private IAdminStudentgradeService adminStudentgradeService; |
| public String getStudentGrade(Integer currentPage,Integer limit,String id)  根据學生ID獲取學生分數 |

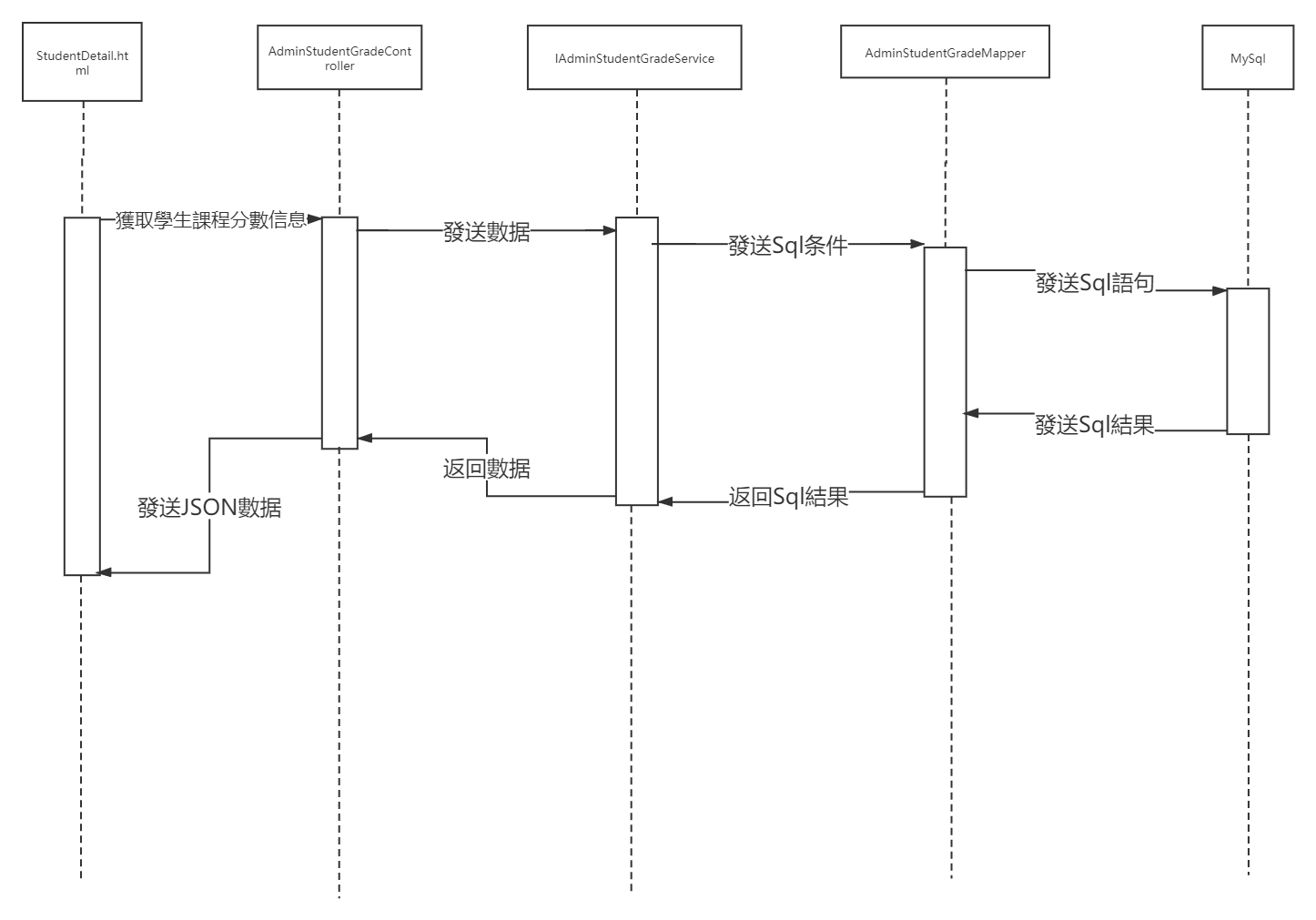
5.4.4 类关系结构图



5.4.5 时序图

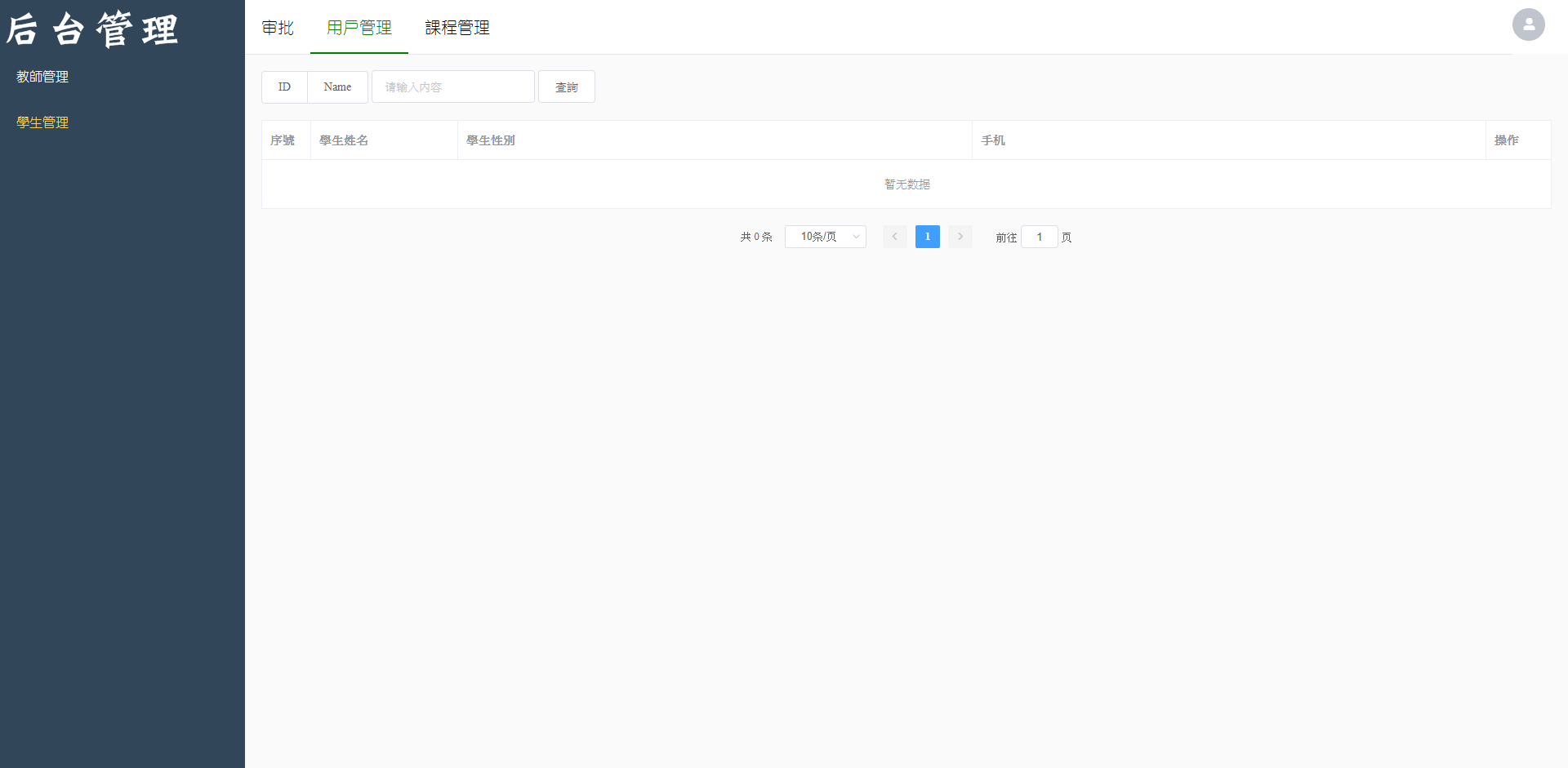






5.4.6 **UI界面**

5.4.6.1 **管理员學生管理界面**



5.4.6.2 **管理员學生信息界面**



**5.4.7 接口说明文档**

5.4.7.1學生數据獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAllStudent

3. 接口功能说明：獲取已辨审核數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 學生數据 |
| total | 是 | 所有數据总數 |

5.4.7.2指定条件搜索學生數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getStudentByKey

3. 接口功能说明：搜索學生

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
| key | 是 | 条件 |
| keyword | 是 | 搜索字 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 學生數据 |
| total | 是 | 所有數据总數 |

5.4.7.3獲取指定學生詳細數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getStudentById

3. 接口功能说明：獲取指定學生詳細數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 學生ID |

5. 结果数据：Json

{status:value,data:{value}}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 學生數据 |

**5.4.7.4**學生分數獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getStudentGrade

3. 接口功能说明：獲取學生分數

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
| id | 是 | 學生ID |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 學生分數數据 |
| total | 是 | 所有數据总數 |

5.4.7.5刪除學生接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/deleteStudent

3. 接口功能说明：刪除學生

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 學生ID |

5. 结果数据：Json

{status:value}

6. 结果数据说明：

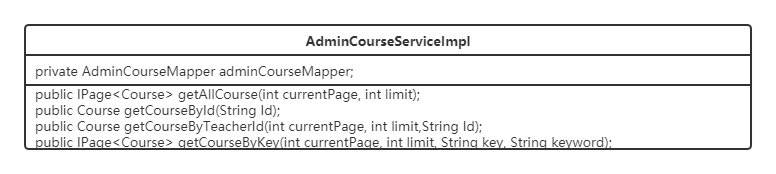
|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |

5.5、課程管理

5.5.1 接口

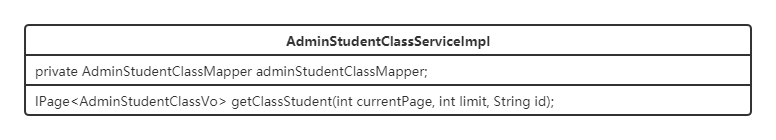
|  |
| --- |
| 接口 |
| IAdminCourseService |
| IAdminStudentClassService |

5.5.2 实现类



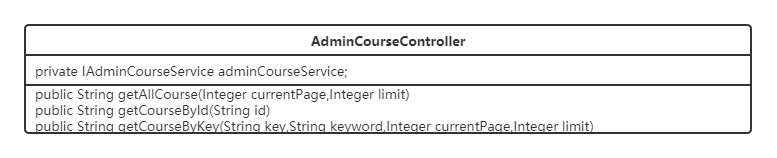
详细描述：

|  |
| --- |
| AdminCourseServiceImpl |
| private AdminCourseMapper adminCourseMapper; |
| public IPage<Course> getAllCourse(int currentPage, int limit);  獲取所有課程信息 |
| public Course getCourseById(String Id);  根据課程ID查询課程詳細信息 |
| public Course getCourseByTeacherId(int currentPage, int limit,String Id);  根据教師ID查询該教師的所有課程 |
| public IPage<Course> getCourseByKey(int currentPage, int limit, String key, String keyword);  根据条件模糊条件查询課程 |



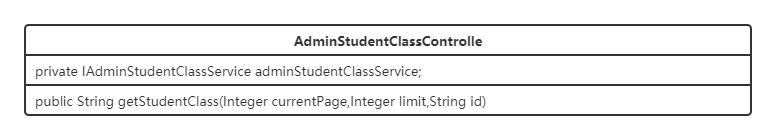
|  |
| --- |
| AdminStudentClassServiceImpl |
| private AdminStudentClassMapper adminStudentClassMapper; |
| IPage<AdminStudentClassVo> getClassStudent(int currentPage, int limit, String id);  根据課程ID獲取課程學生 |

5.5.3控制器类图



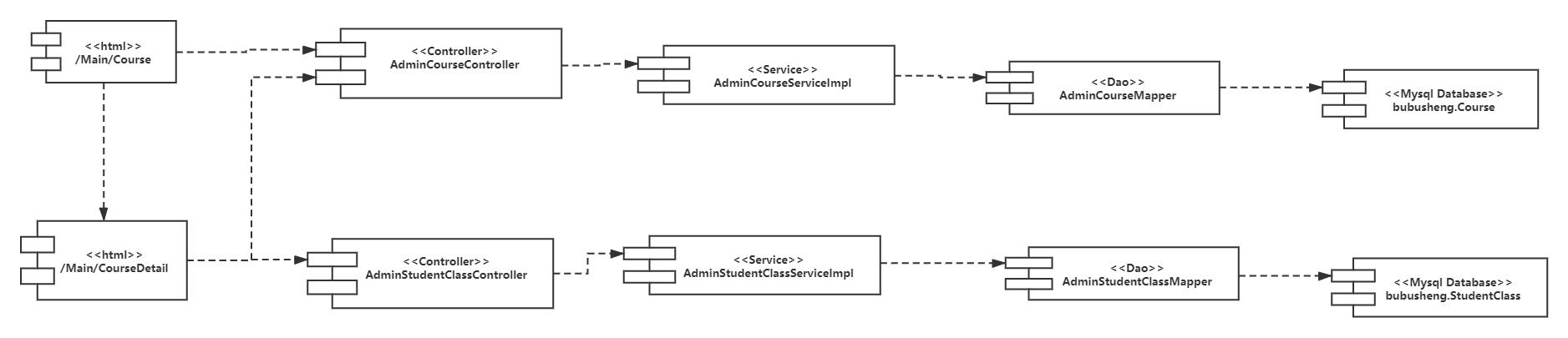
控制方法：

|  |
| --- |
| AdminCourseController |
| private IAdminCourseService adminCourseService; |
| public String getAllCourse(Integer currentPage,Integer limit)  獲取所有課程信息 |
| public String getCourseById(String id)  根据課程ID查询課程詳細信息 |
| public String getCourseByKey(String key,String keyword,Integer currentPage,Integer limit)  根据条件模糊条件查询課程列表 |

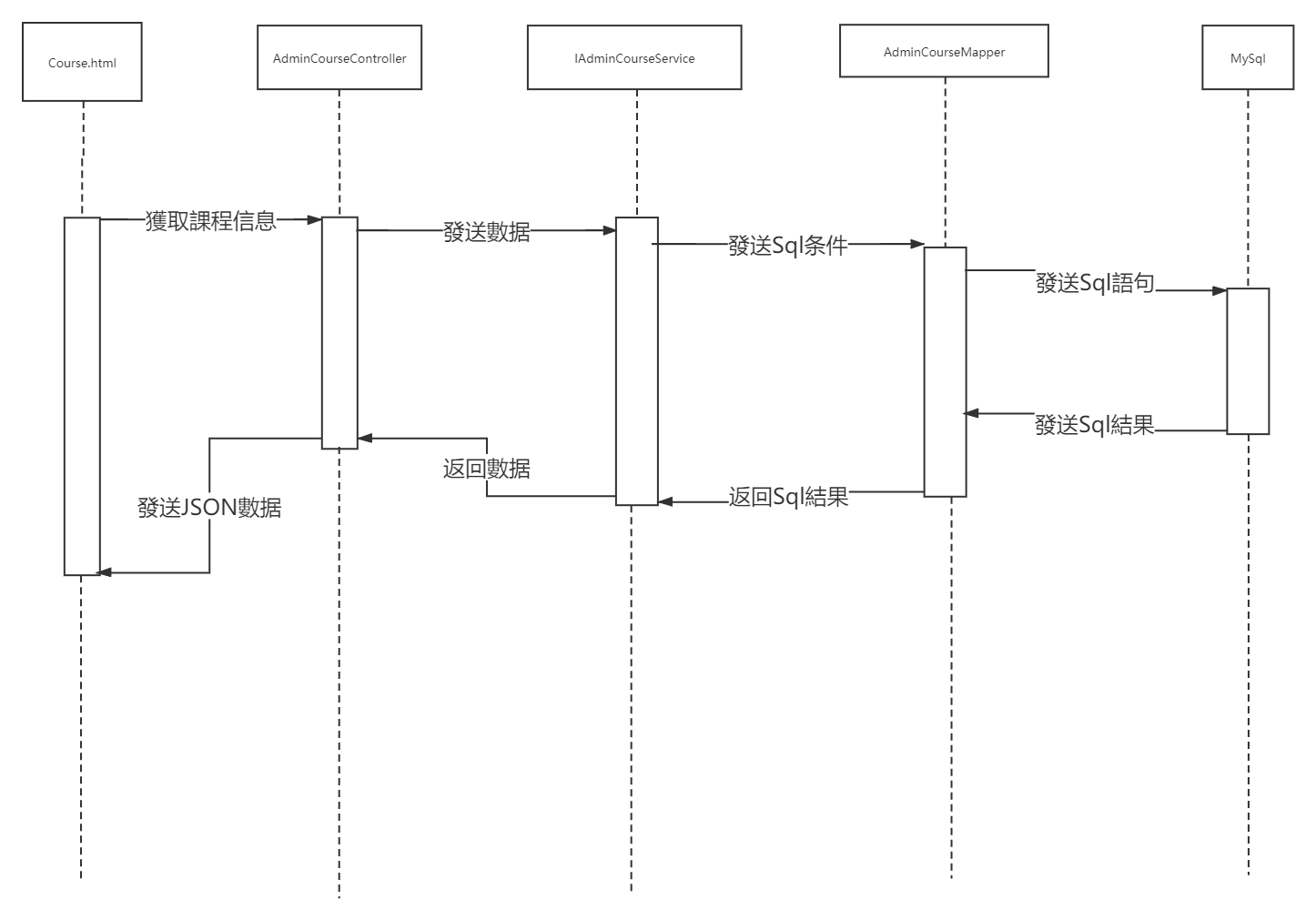


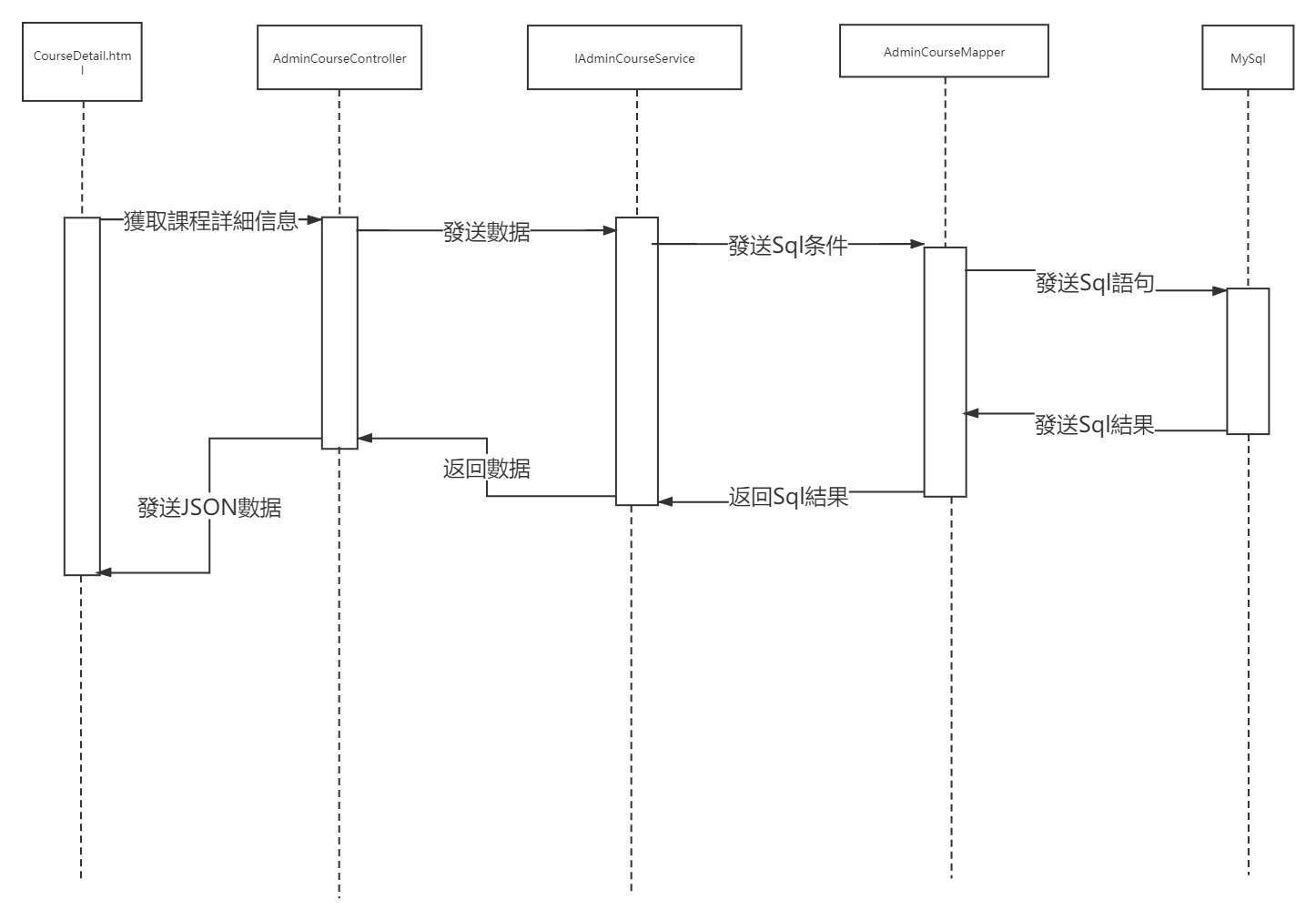
|  |
| --- |
| AdminStudentClassController |
| private IAdminStudentClassService adminStudentClassService; |
| public String getStudentClass(Integer currentPage,Integer limit,String id)  根据課程ID查询課程學生 |

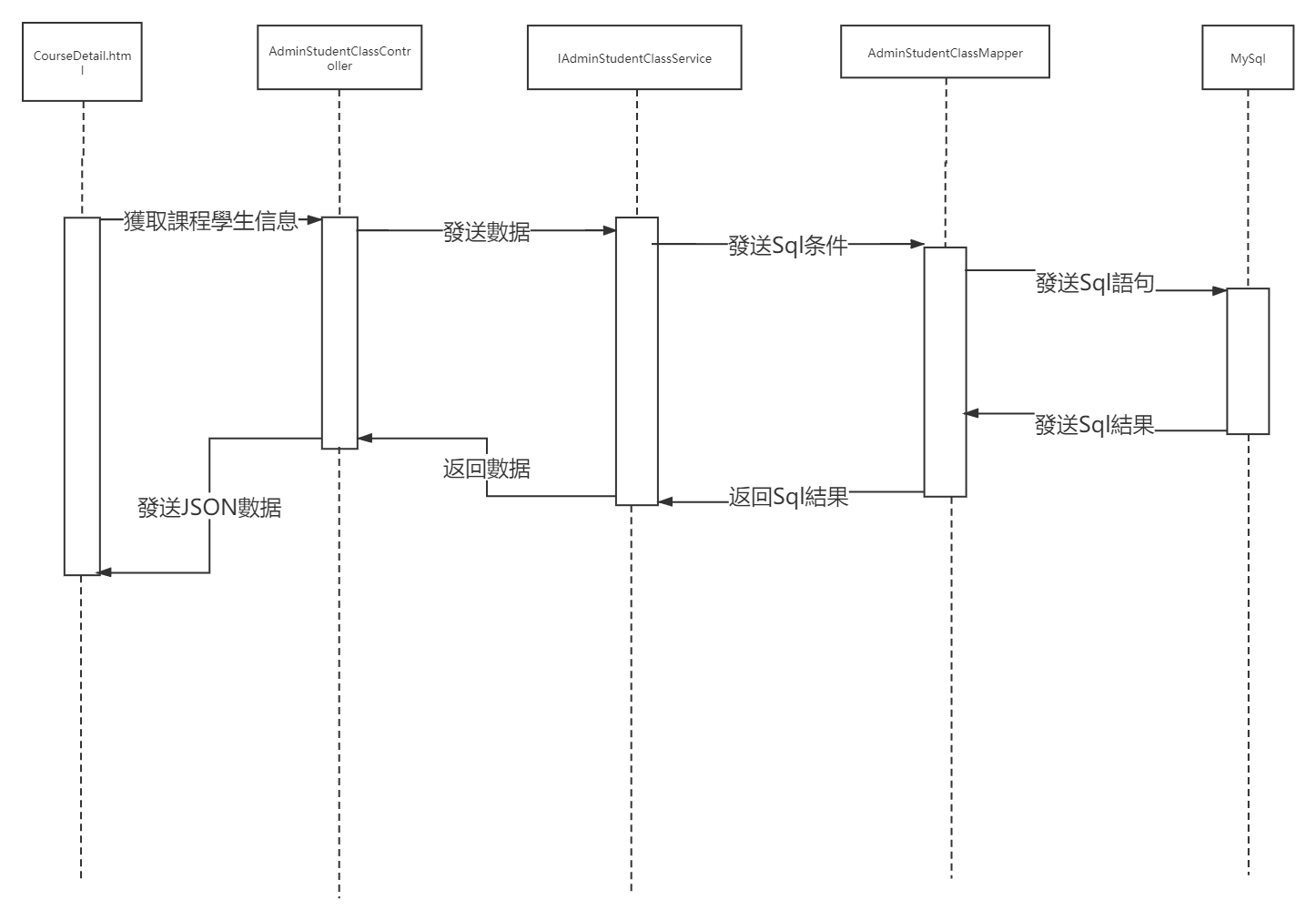
5.5.4 类关系结构图



5.5.5 时序图

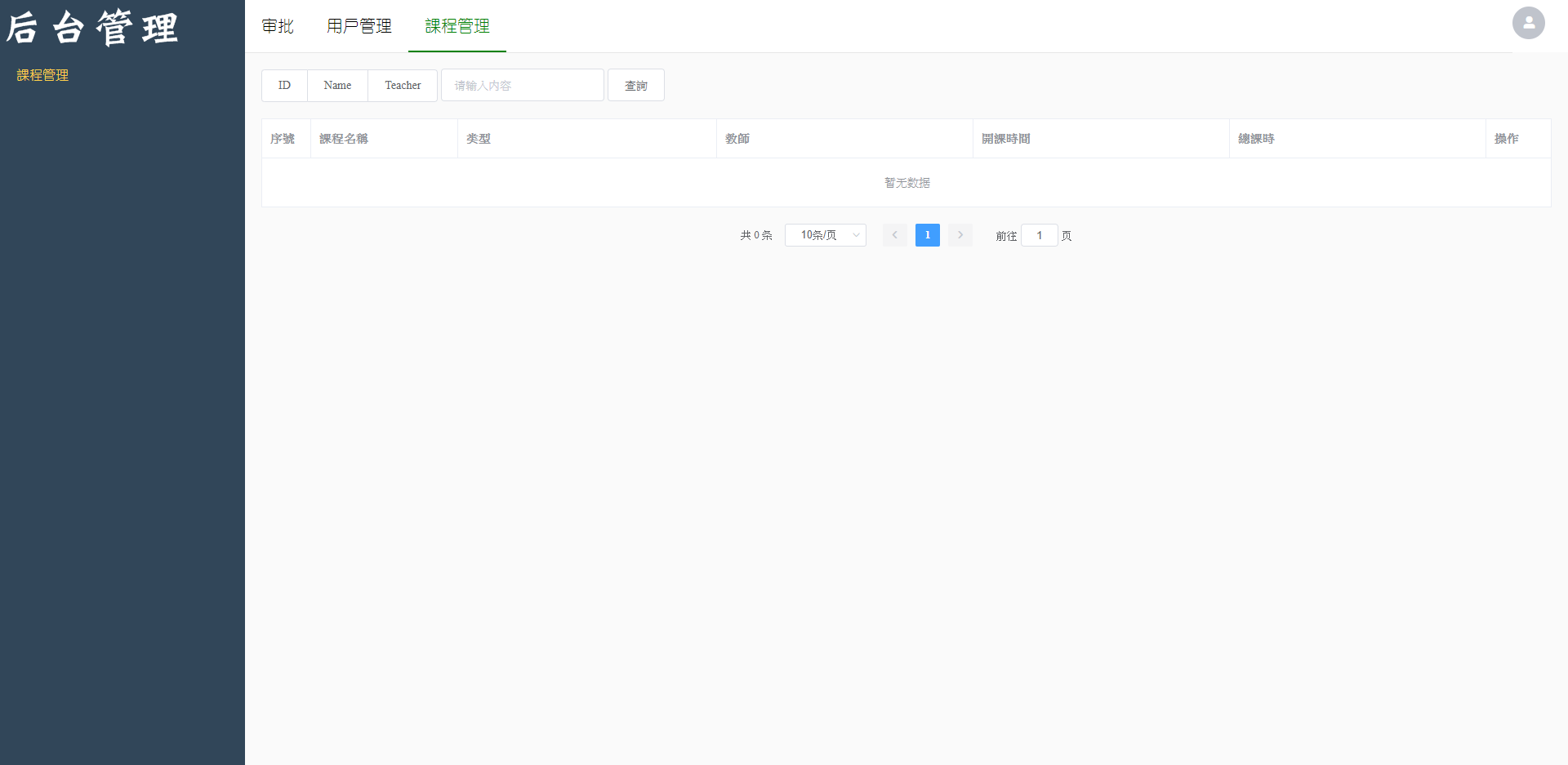






**5.5.6 UI界面**

**5.5.6.1** **管理员課程管理界面**



**5.5.6.2管理员課程信息界面**



**5.5.7 接口说明文档**

**5.5.7**.1課程數据獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getAllCourse

3. 接口功能说明：獲取課程數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 課程數据 |
| total | 是 | 所有數据总數 |

**5.5.7**.2指定条件搜索課程數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getCourseByKey

3. 接口功能说明：搜索課程

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
| key | 是 | 条件 |
| keyword | 是 | 搜索字 |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 課程數据 |
| total | 是 | 所有數据总數 |

**5.5.7**.3獲取指定課程詳細數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getCourseById

3. 接口功能说明：獲取指定課程詳細數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 課程ID |

5. 结果数据：Json

{status:value,data:{value}}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 課程數据 |

**5.5.7**.4課程學生數据獲取接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getStudentByClass

3. 接口功能说明：獲取課程學生數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| currentPage | 是 | 分頁頁數,初始值為1 |
| limit | 是 | 每頁多少条數据,,初始值為10 |
| id | 是 | 課程ID |

5. 结果数据：Json

{status:value,data:[{value},{value}],total:value}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 課程學生數据 |
| total | 是 | 所有數据总數 |

**5.5.7.5**獲取指定學生詳細數据接口

1. 请求方式：get｜HTTP

2. 请求地址：/admin/getStudentById

3. 接口功能说明：獲取指定學生詳細數据

4. 请求参数：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| id | 是 | 學生ID |

5. 结果数据：Json

{status:value,data:{value}}

6. 结果数据说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),500(服務器出錯) |
| data | 是 | 學生數据 |

六、 课程模块设计

6.1、课程内容

6.1.1 接口

|  |
| --- |
| 接口 |
| ICoursedataService |

6.1.2 实现类

详细描述：CoursedataServiceImpl

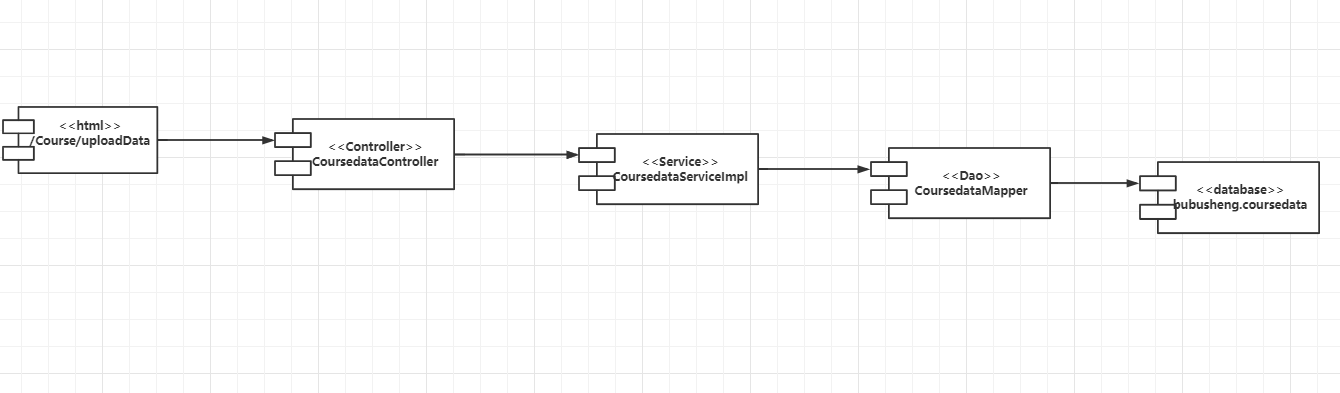
|  |
| --- |
| CoursedataServiceImpl |
| private CoursedataMapper coursedataMapper; |
| int insertCourseData(int classId,int chapter,MultipartFile ppt,MultipartFile document)  上传课件与文档 |
| int insertVideo(int classId , int chapter , MultipartFile video)  上传视频 |
| int insertExerciseAndAnswer(int classId , int chapter , String exercise,String answer)  上传习题 |
| List<Coursedata> getCourseDataListByClassId(int classId);  根据课程Id获取课程信息列表 |
| Coursedata getCourseDataByIdAndChapter(int classId,int chapter);  根据课程Id与章节获取课程信息 |

6.1.3控制器类

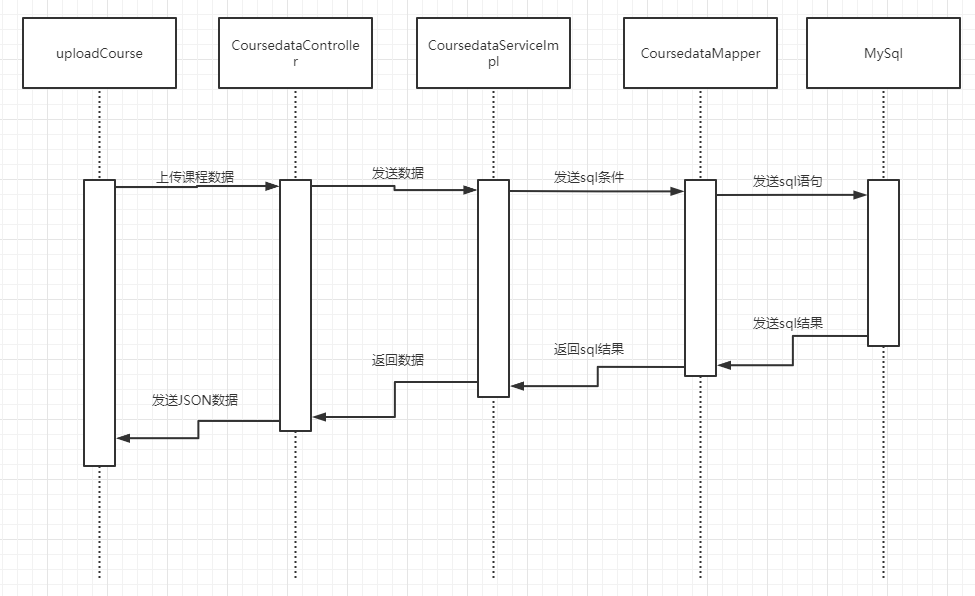
控制方法：CoursedataController

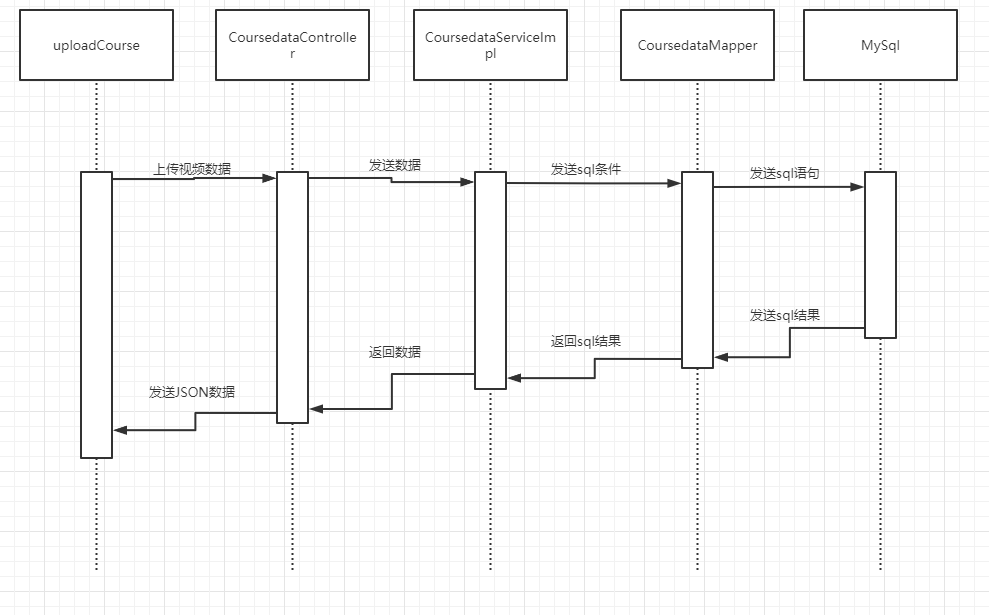
|  |
| --- |
| CoursedataController |
| private ICoursedataService courseDataService; |
| public String uploadData(int classId, int chapter, MultipartFile ppt, MultipartFile document)  上传课件与文档 |
| public String insertExerciseAndAnswer(int classId , int chapter , String exercise,String answer)  上传习题 |
| public String uploadVideo(int classId , int chapter , MultipartFile video)  上传视频 |
| public String getCourseDataListByClassId(int classId)  根据课程Id获取课程信息列表 |
| public String getCourseDataByIdAndChapter(int classId ,int chapter)  根据课程Id与章节获取课程信息 |

6.1.4 类关系结构图



6.4 时序图





6.1.6接口说明文档

6.1.6.1教师上传课件

1.请求方式：Post | HTTP

2.请求地址：/course/uploadData

3.接口功能说明：教师上传课件

4.请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| classId | 是 | 课程编号 |
| chapter | 是 | 章节 |
| ppt | 否 | ppt |
| document | 否 | 文档 |

5.结果数据：Json

{state:value}

6.结果数据说明

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),1(失败  ),500(服務器出錯) |

6.1.6.2教师上传视频

1.请求方式：Post | HTTP

2.请求地址：/course/uploadVideo

3.接口功能说明：教师上传课程视频

4.请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| classId | 是 | 课程编号 |
| chapter | 是 | 章节 |
| video | 是 | 课程视频 |

5.结果数据：Json

{state:value}

6.结果数据说明

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),1(失败  ),500(服務器出錯) |

6.1.6.3教师上传习题

1.请求方式：Post | HTTP

2.请求地址：/course/insertExercise

3.接口功能说明：教师上传习题

4.请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| classId | 是 | 课程编号 |
| chapter | 是 | 章节 |
| exercise | 是 | 练习题 |
| answer | 是 | 习题答案 |

5.结果数据：Json

{state:value}

6.结果数据说明

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),1(失败  ),500(服務器出錯) |

6.2、修改课程信息

6.2.1 接口

|  |
| --- |
| 接口 |
| ICourseService |

6.2.2 实现类

详细描述：CourseServiceImpl

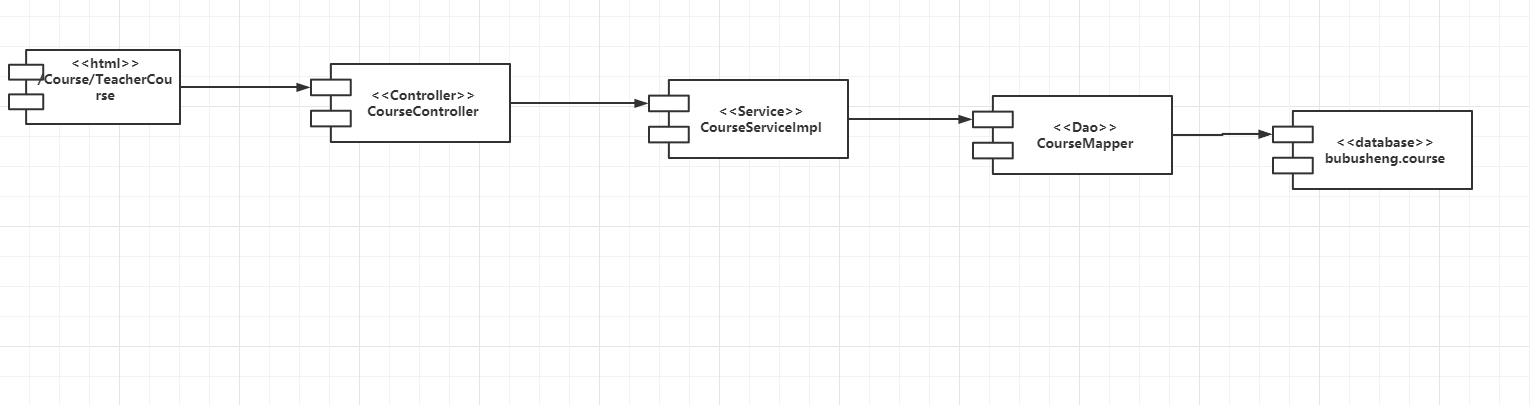
|  |
| --- |
| CourseServiceImpl |
| private CourseMapper courseMapper; |
| public int updateCourse(int classId, String teacherName, String className, String intro, String beginning, String type)  修改课程信息 |

6.2.3控制器类

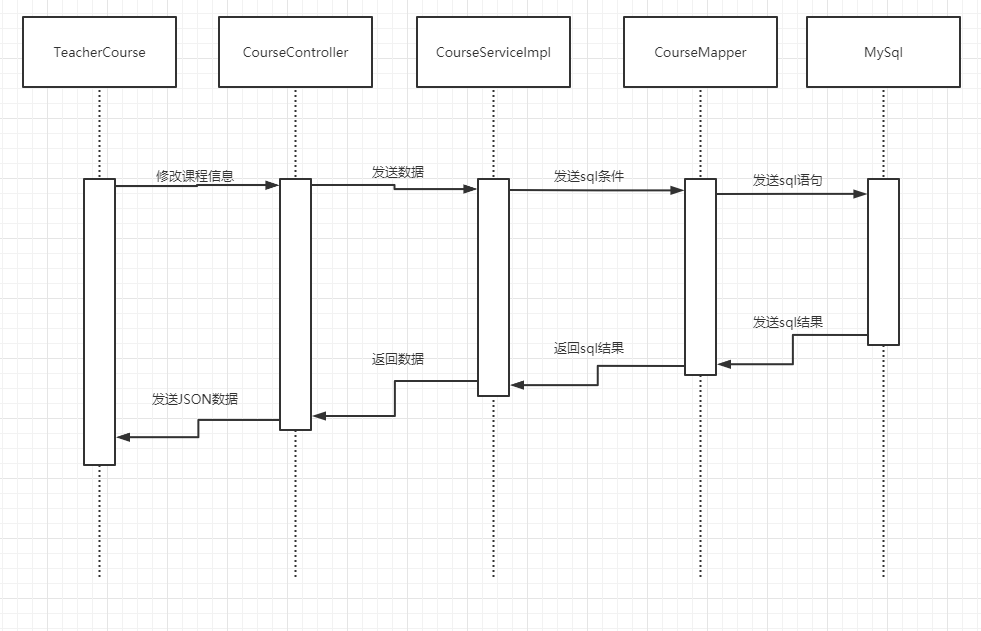
控制方法：CourseController

|  |
| --- |
| CourseController |
| private ICourseService courseService; |
| public String updateCourse(int classId, String teacherName, String className, String intro, String beginning, String type)  修改课程信息 |

6.2.4 类关系结构图



6.2.5 时序图



6.2.6接口说明文档

6.2.6.1教师修改课程信息

1.请求方式：Post | HTTP

2.请求地址：/course/updateCourse

3.接口功能说明：教师修改课程信息

4.请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| classId | 是 | 课程编号 |
| teacherName | 否 | 教师姓名 |
| className | 否 | 课程名 |
| intro | 否 | 课程介绍 |
| beginning | 否 | 开课时间 |
| type | 否 | 课程类型 |

5.结果数据：Json

{state:value}

6.结果数据说明

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),1(失败  ),500(服務器出錯) |

6.3、发布通知

6.3.1 接口

|  |
| --- |
| 接口 |
| INoticeService |

6.3.2 实现类

详细描述：NoticeServiceImpl

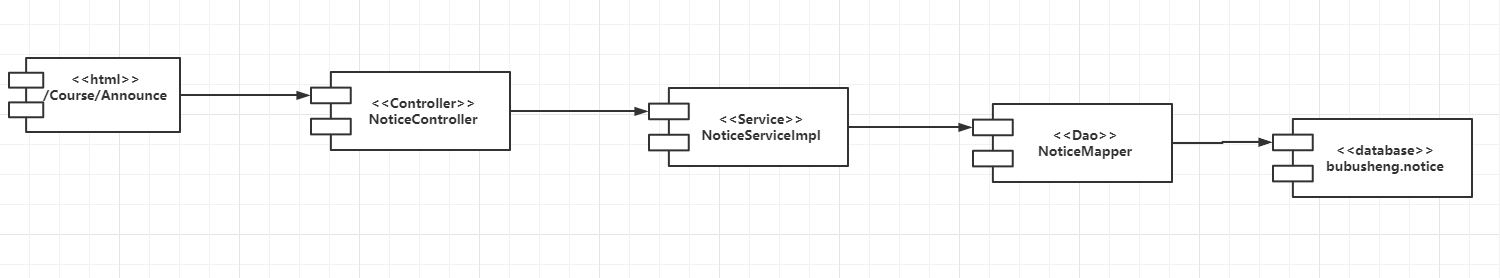
|  |
| --- |
| NoticeServiceImpl |
| private NoticeMapper noticeMapper; |
| public int insertNotice(int classId, String notice, String noticeName, String time)  发布通知 |

6.3.3控制器类

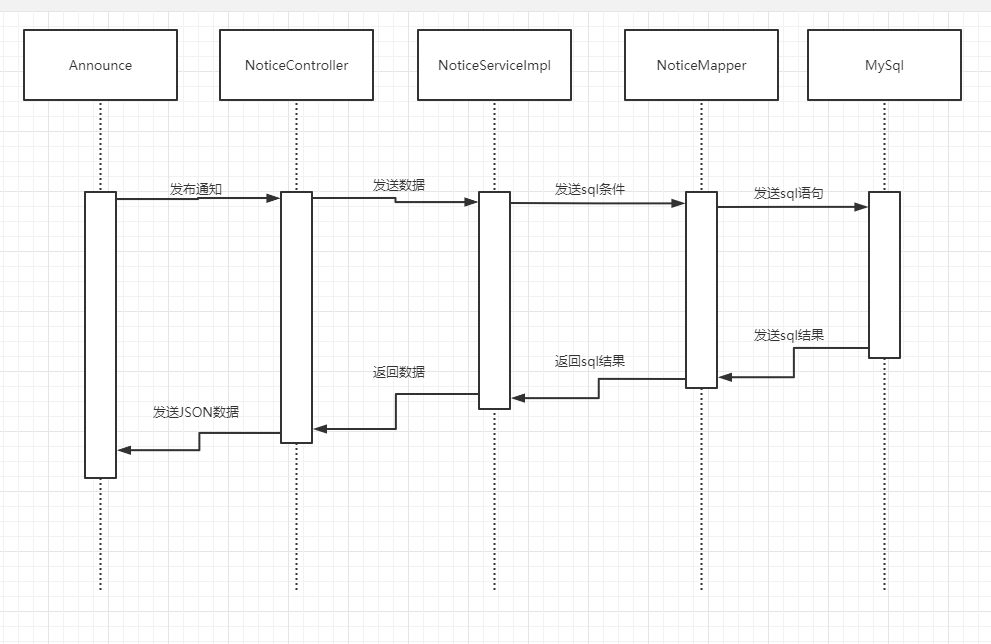
控制方法NoticeController

|  |
| --- |
| NoticeController |
| private INoticeService noticeService; |
| public String insertNotice(int classId, String notice, String noticeName, String time)  发布通知 |

6.3.4 类关系结构图



6.3.5 时序图



6.3.6接口说明文档

6.3.6.1教师发布通知

1.请求方式：Post | HTTP

2.请求地址：/course/insertNotice

3.接口功能说明：教师发布通知

4.请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| classId | 是 | 课程编号 |
| notice | 是 | 通知内容 |
| noticeName | 是 | 通知标题 |
| time | 是 | 通知时间 |

5.结果数据：Json

{state:value}

6.结果数据说明

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),1(失败  ),500(服務器出錯) |

6.4、发布考试

6.4.1 接口

|  |
| --- |
| 接口 |
| ITestpaperService |

6.4.2 实现类

详细描述：TestpaperServiceImpl

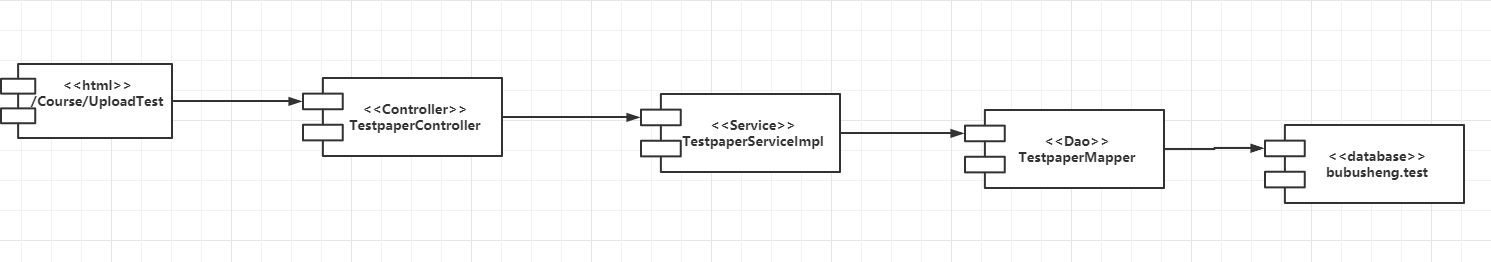
|  |
| --- |
| TestpaperServiceImpl |
| private TestpaperMapper TestpaperMapper; |
| public int insertTestpaper(int classId, MultipartFile test, String testTime, String deadline)  发布考试 |

6.4.3控制器类

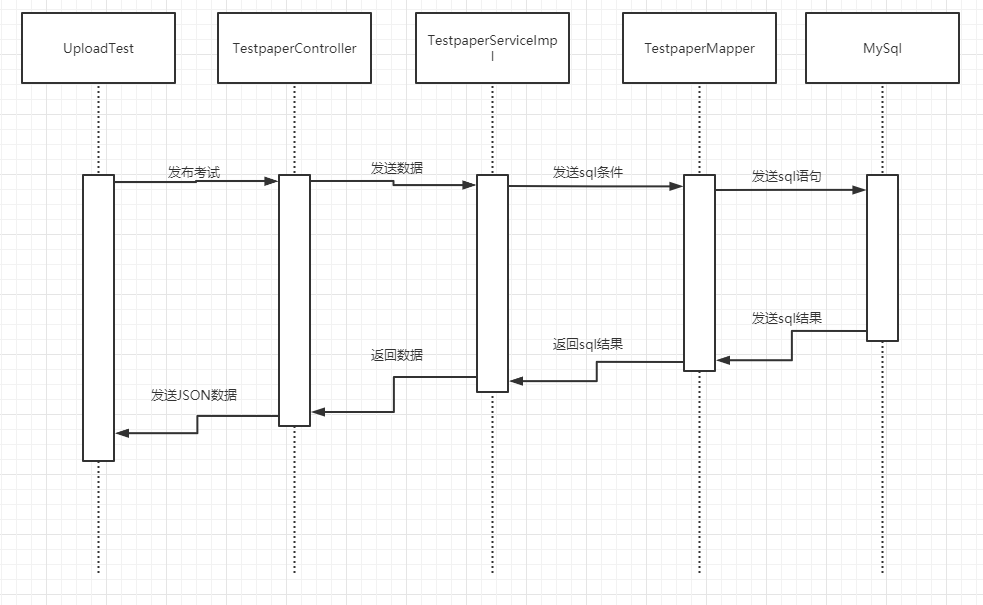
控制方法NoticeController

|  |
| --- |
| TestpaperController |
| private ITestpaperService testpaperService; |
| public String insertTestpaper(int classId, MultipartFile test, String testTime, String deadline)  发布考试 |

6.4.4 类关系结构图



6.4.5 时序图



6.4.6接口说明文档

6.4.6.1教师上传试卷

1.请求方式：Post | HTTP

2.请求地址：/course/insertTestpaper

3.接口功能说明：教师上传试卷

4.请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| classId | 是 | 课程编号 |
| test | 是 | 考试试卷 |
| testTime | 是 | 开始时间 |
| deadline | 是 | 截止时间 |

5.结果数据：Json

{state:value}

6.结果数据说明

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),1(失败  ),500(服務器出錯) |

6.5、批改与读取学生成绩

6.5.1 接口

|  |
| --- |
| 接口 |
| IStudentGradeService |

6.5.2 实现类

详细描述：StudentGradeServiceImpl

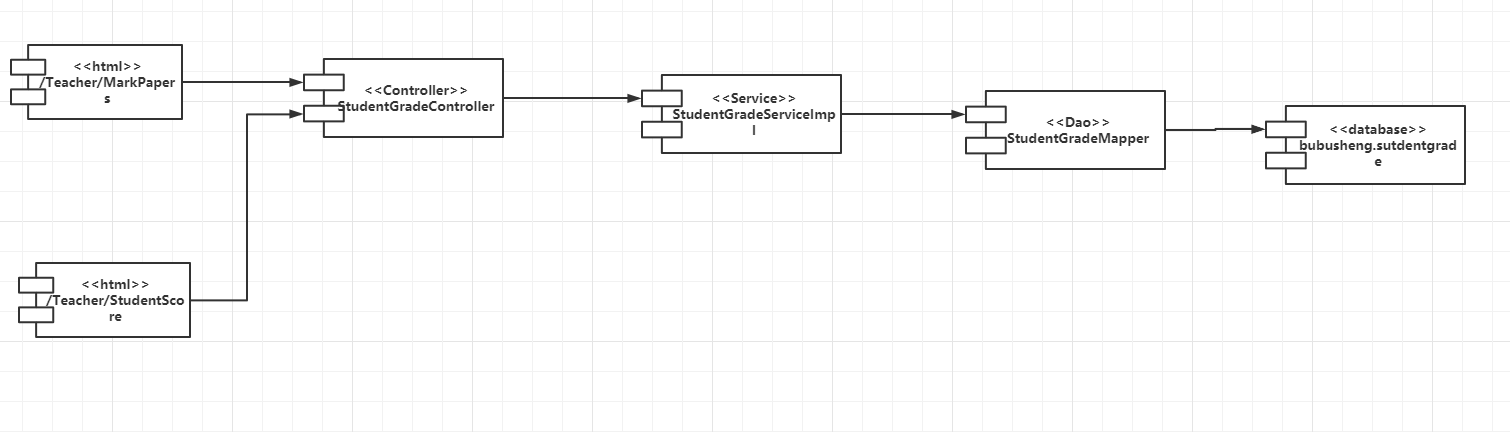
|  |
| --- |
| StudentGradeServiceImpl |
| private StudentGradeMapper StudentGradeMapper; |
| public IPage<Studentgrade> getStudentGrade(int classId,int currentPage,int limit)  读取学生成绩 |
| public int setStudentGrade(int studentId, int classId, int score, String annotation,String answer)  批改学生成绩 |

6.5.3控制器类

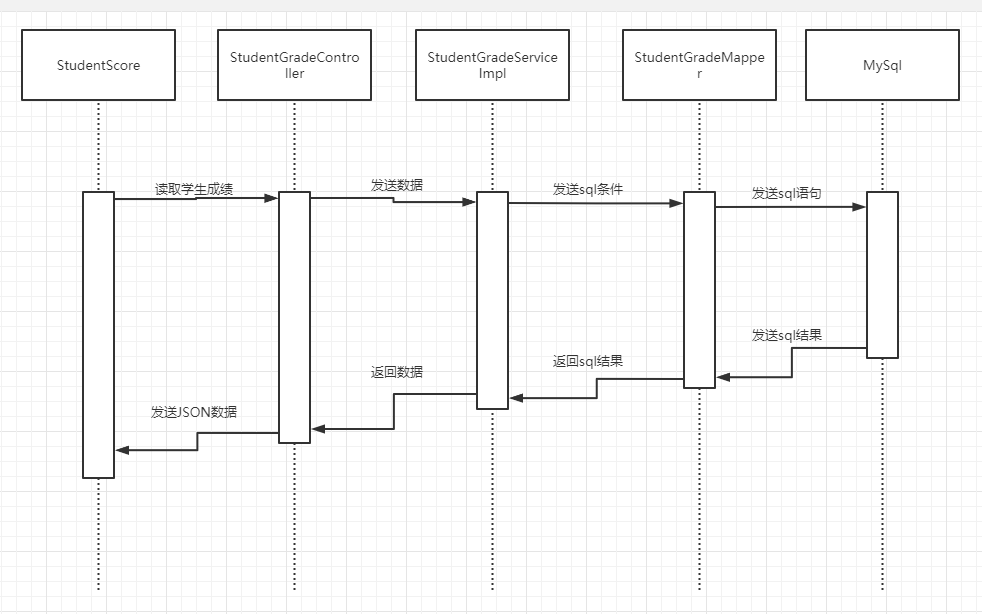
控制方法StudentGradeController

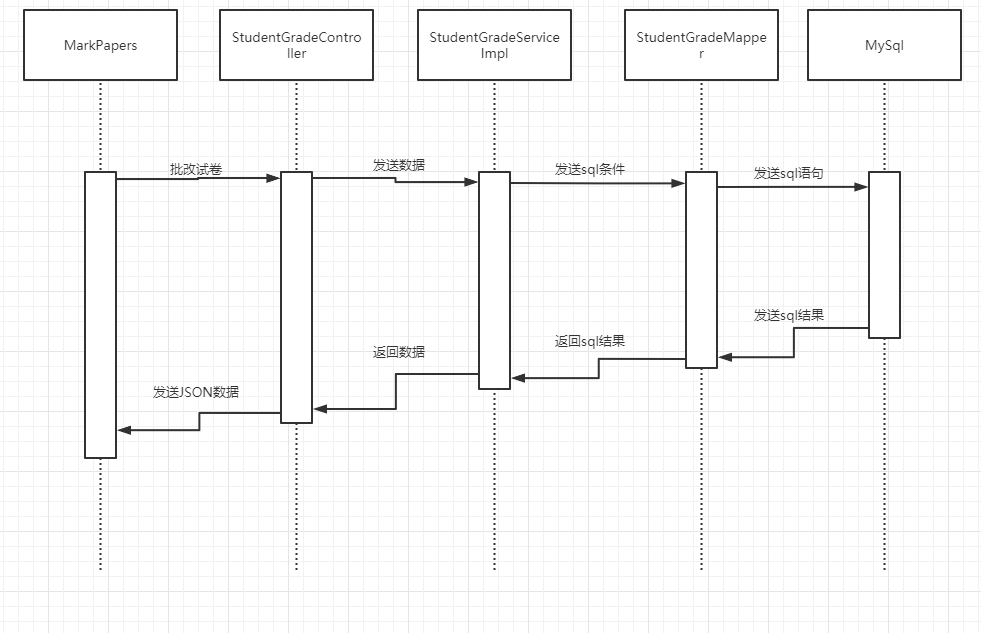
|  |
| --- |
| StudentGradeController |
| private IStudentGradeService studentgradeService; |
| public String getStudentGrade(int classId ,int currentPage ,int limit)  读取学生成绩 |
| public String setStudentGrade(int studentId,int classId,int score,String annotation,String answer)  批改学生成绩 |

6.5.4 类关系结构图



6.5.5 时序图





6.5.6接口说明文档

6.5.6.1教师批改学生成绩

1.请求方式：Post | HTTP

2.请求地址：/course/setStudentGrade

3.接口功能说明：教师批改学生成绩

4.请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| studentId | 是 | 学生编号 |
| classId | 是 | 课程编号 |
| score | 是 | 成绩 |
| annotation | 是 | 教师评注 |
| answer | 是 | 试卷答案 |

5.结果数据：Json

{state:value}

6.结果数据说明

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),1(失败  ),500(服務器出錯) |

6.5.6.2教师读取学生成绩

1.请求方式：Post | HTTP

2.请求地址：/course/getStudentGrade

3.接口功能说明：教师读取学生成绩

4.请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| classId | 是 | 课程编号 |
| currentPage | 是 | 分页数据，初始值为1 |
| limit | 是 | 每页多少条数据，初始值为10 |

5.结果数据：Json

{state:value,data:[{value},{value}],total:value}

6.结果数据说明

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否必须 | 参数说明 |
| status | 是 | 200(成功),1(失败  ),500(服務器出錯) |
| data | 是 | 学生成绩列表 |
| total | 是 | 所有數据总數 |

**第六部分 数据库设计**

一、数据库整体结构图

二、总模块

1、Student表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | studentId | INT | 学生id |
| 2 | studentName | VARCHAR(10) | 学生姓名 |
| 3 | sex | VARCHAR(4) | 学生性别 |
| 4 | age | INT | 学生年龄 |
| 5 | password | VARCHAR(20) | 密码 |
| 6 | phone | VARCHAR(11) | 手机号 |
| 7 | flag | int | 逻輯刪除值 |

2、Teacher表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | teacherId | INT | 老师id |
| 2 | teacherName | VARCHAR(10) | 老师名 |
| 3 | phone | VARCHAR(11) | 手机号 |
| 4 | password | VARCHAER(20) | 密码 |
| 5 | flag | INT | 逻輯刪除值 |

3、Course表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | classId | INT | 课程编号 |
| 2 | className | VARCHAR(56) | 课程名 |
| 3 | teacherId | INT | 教师id |
| 4 | teacherName | VARCHAR(10) | 授课教师名称 |
| 5 | intro | VARCHAR(56) | 课程简介 |
| 6 | beginning | VARCHAR(56) | 开课时间 |
| 7 | type | VARCHAR(56) | 课程类型 |

4、TestPaper表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | classId | INT | 试卷对应的编号 |
| 2 | test | VARCHAR(56) | 考试试卷存储地址 |
| 3 | testtime | VARCHAR(56) | 考试时间 |
| 4 | deadline | VARCHAR(56) | 考试截止时间 |

5、StudentGrade表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | studentId | INT | 学生id（主键） |
| 2 | classId | INT | 课程id（主键） |
| 3 | score | INT | 考试分数 |
| 4 | annotation | VARCHAR(56) | 批注 |
| 5 | answer | VARCHAR(56) | 试卷答案 |
| 6 | state | VARCHAR(4) | 是否批改 |
| 7 | studentName | VARCHAR(10） | 学生姓名 |

6、Notice表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | classId | int | 课程id |
| 2 | notice | VARCHAR(56) | 公告内容 |
| 3 | time | VARCHAR(56) | 发布时间 |
| 4 | noticeName | VARCHAR(56) | 公告名 |

7、StudentClass表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | studentid | int | 学生id |
| 2 | classid | int | 课程id |

8、Administrator表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | id | int | 主鍵 |
| 2 | account | VARCHAR(56) | 管理员用戶名 |
| 3 | password | VARCHAR(56) | 管理员密码 |
| 4 | uid | VARCHAR(56) | 管理员id |

9、CourseData表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | classid | int | 课程id |
| 2 | chapters | int | 章节 |
| 3 | name | VARCHAR(56) | 章节名 |
| 4 | video | VARCHAR(56) | 教学视频 |
| 5 | document | VARCHAR(56) | 文档 |
| 6 | ppt | VARCHAR(56) | 教学ppt |
| 7 | exercise | VARCHAR(56) | 习题 |
| 8 | answer | VARCHAR(56) | 习题答案 |

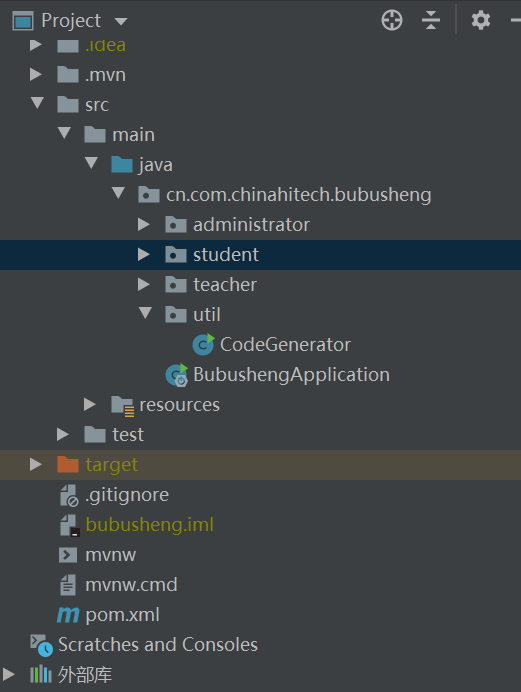
10、CourseApplication表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | id | int | id |
| 2 | classId | int | 课程id |
| 3 | className | VARCHAR(56) | 课程名 |
| 4 | teacherId | INT | 授课教师id |
| 5 | teacherName | VARCHAR(56) | 授课教师名稱 |
| 6 | intro | VARCHAR(56) | 课程简介 |
| 7 | classType | VARCHAR(56) | 課程类型 |
| 8 | beginning | VARCHAR(56) | 开课时间 |
| 9 | result | VARCHAR(56) | 审核結果 |

**第七部分 补充设计和说明**

一、包路径与WEB目录结构设计

1、包路径设计



2、WEB目录结构设计

