软件工程课程项目

——实验教学管理系统

软件设计规格说明书

SOFTWARE DESIGNS SPECIFICATION

小组成员：2051196 刘一飞

2052348 王杨乐

2050747 赵帅涛

指导教师：黄杰

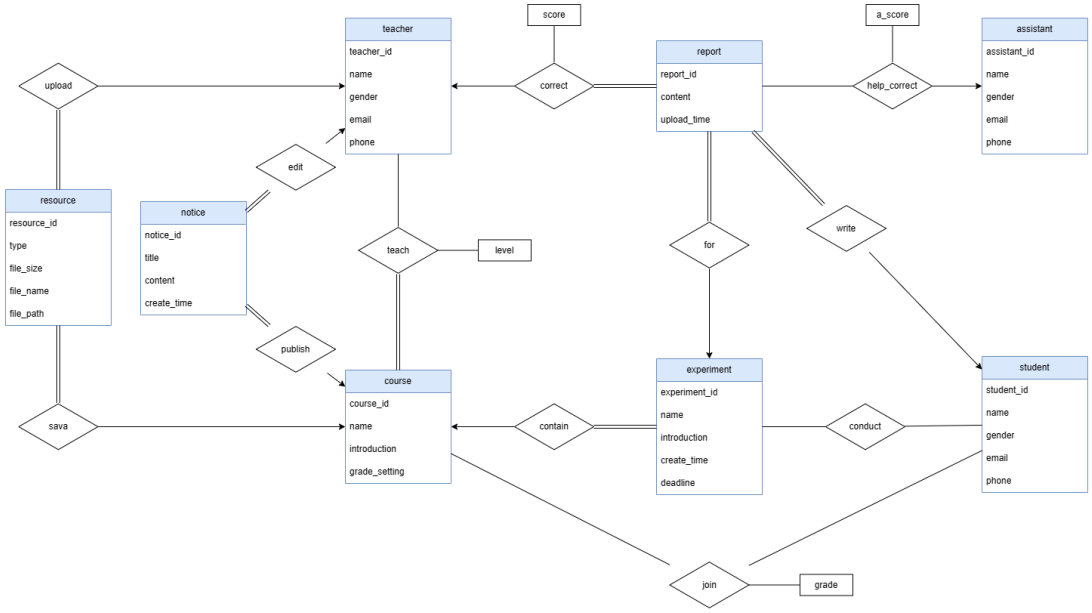
目 录

1 数据设计

1.1 数据库设计

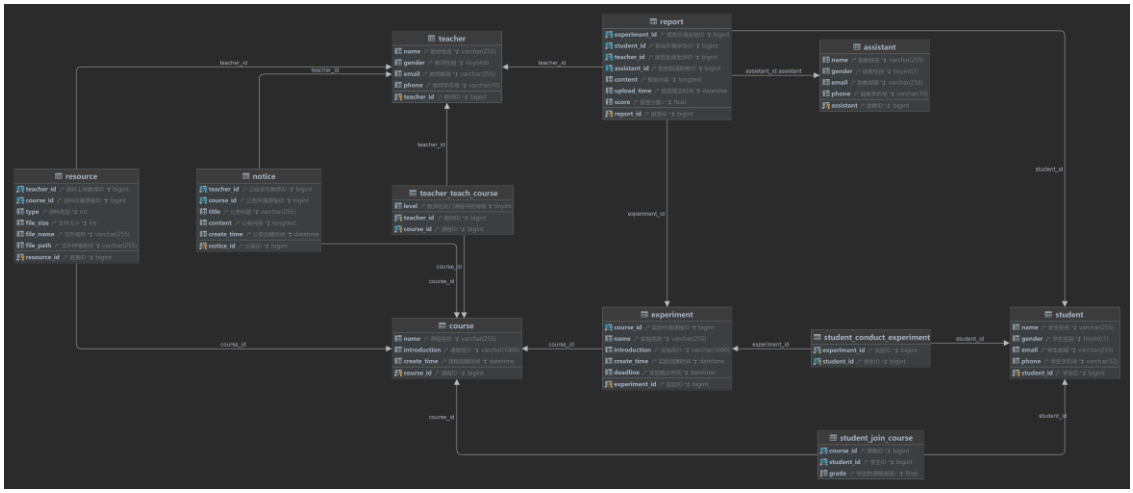
1.1.1 逻辑设计

针对支撑实验教学管理系统所需的数据模型，我们利用语法解析识别系统所涉及的实体，并根据其是否具有 Retained information，Needed Services，Multiple Attribute 等特征进行过滤，识别出系统所需数据实体，如教师，学生，实验报告，通知公告等，并根据具体的业务逻辑需求，设计出如下的 ER 图：



1.1.2 物理设计

在完成 ER 图的设计之后，我们根据 ER 图与数据库的关系，根据实体间一对一，多对一等关系，合理设计数据库表，通过增加数据库表或在表中增添字段，将各表之间的关系联系起来，为满足较高的性能需求，我们力求用最小的存储代价描述数据模型，为此数据库的设计严格遵循了第三范式，消除非主属性对于码的传递函数依赖，具体数据库表如下：



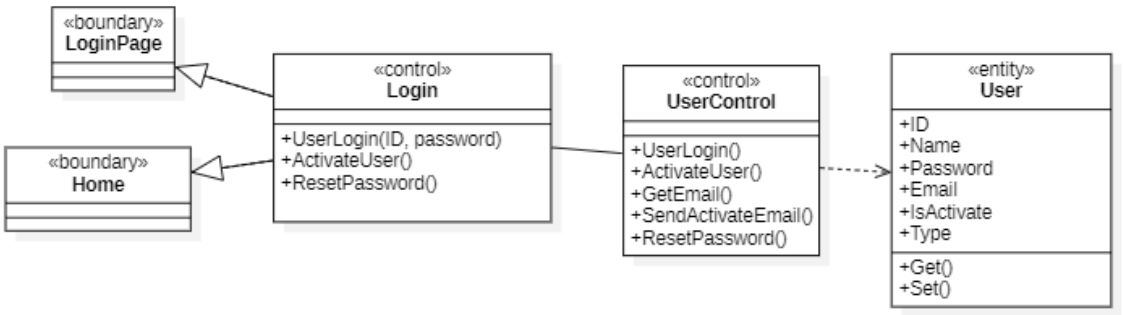
1.2 类设计

1.2.1 用户登录子系统

1.2.1.1 系统简述

该子系统主要承担用户在登录过程中的有关工作，包括了发送验证邮件、激活账户、用户登录、修改密码等操作。

1.2.1.2 设计类图

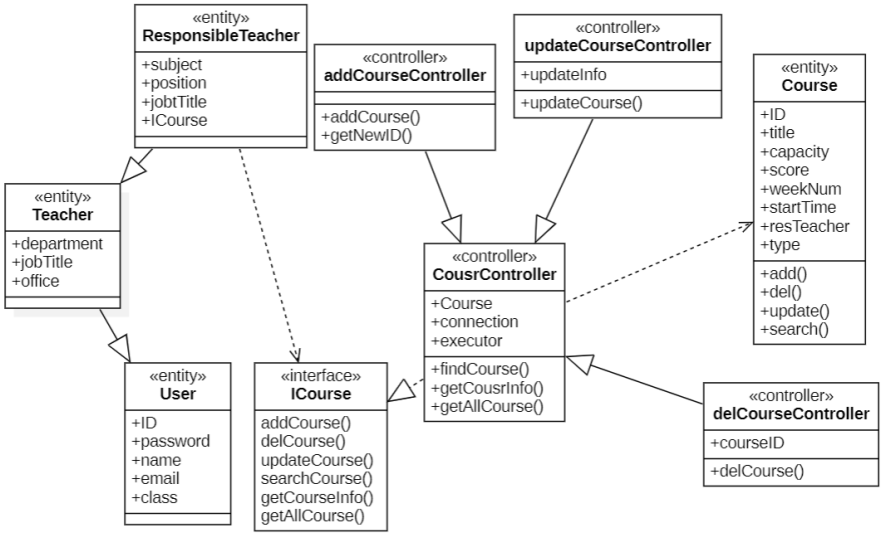


1.2.2 实验课程管理子系统

1.2.2.1 系统简述

该子系统主要承担与实验课程管理有关的所有工作，包括了教师添加、修改、删除课程等操作。

1.2.2.2 设计类图

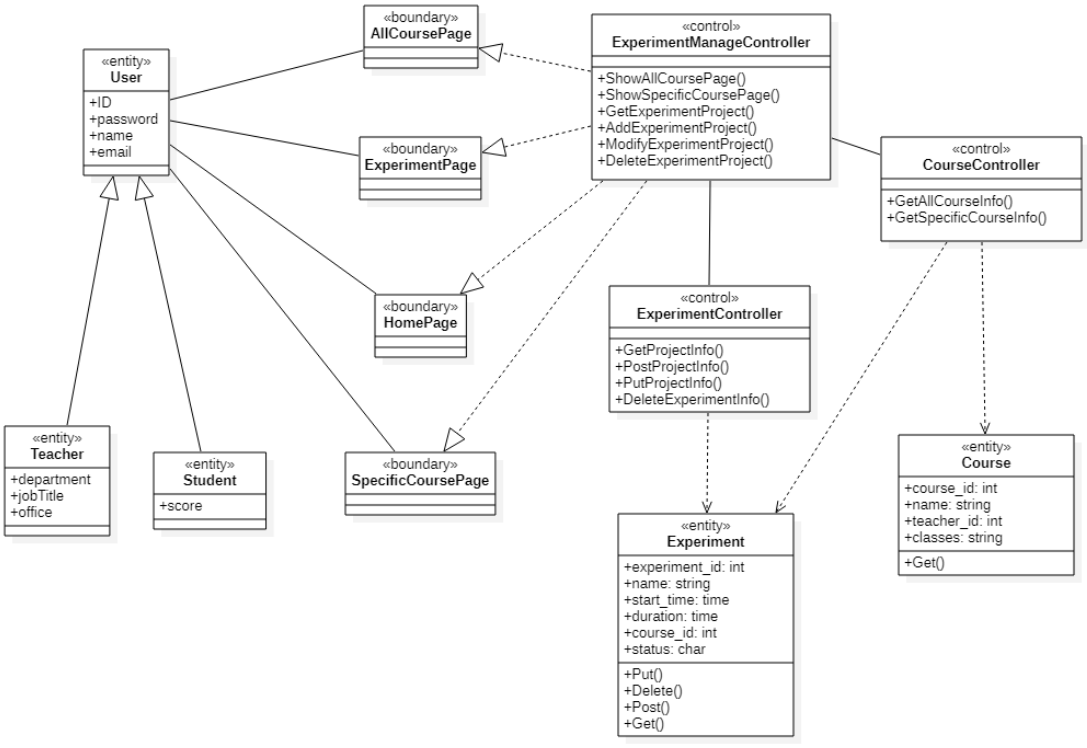


1.2.3 实验项目管理子系统

1.2.3.1 系统简述

该子系统主要承担与实验项目管理有关的所有工作，包括了教师添加、修改、删实验，学生参加实验等操作。

1.2.3.2 设计类图

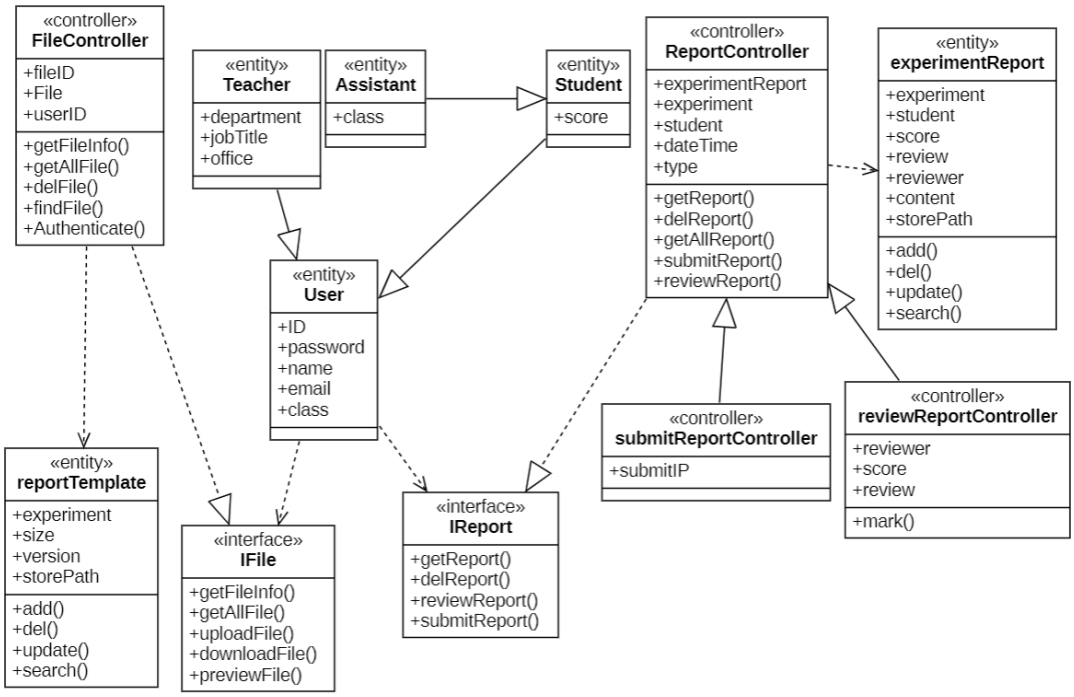


1.2.4 实验报告管理子系统

1.2.4.1 系统简述

该子系统主要承担与实验报告管理有关的所有工作，包括了学生完成实验报告，教师和助教批改实验报告等操作。

1.2.4.2 设计类图

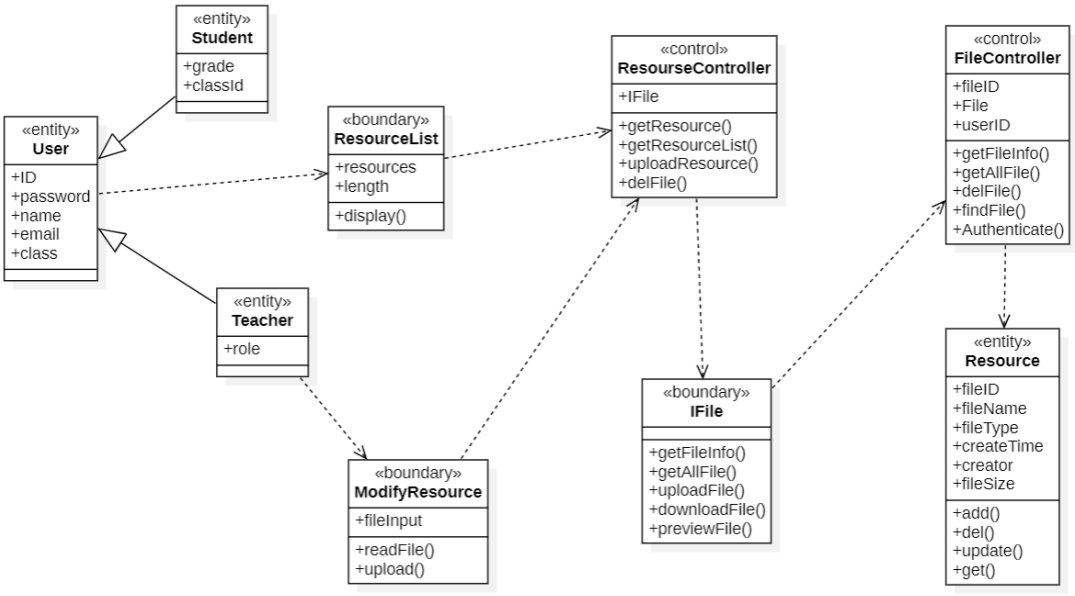


1.2.5 课程资源管理子系统

1.2.5.1 系统简述

该子系统主要承担与实验课程资源管理有关的所有工作，包括了教师上传实验资源、学生和教师下载实验资源等操作。

1.2.5.2 设计类图

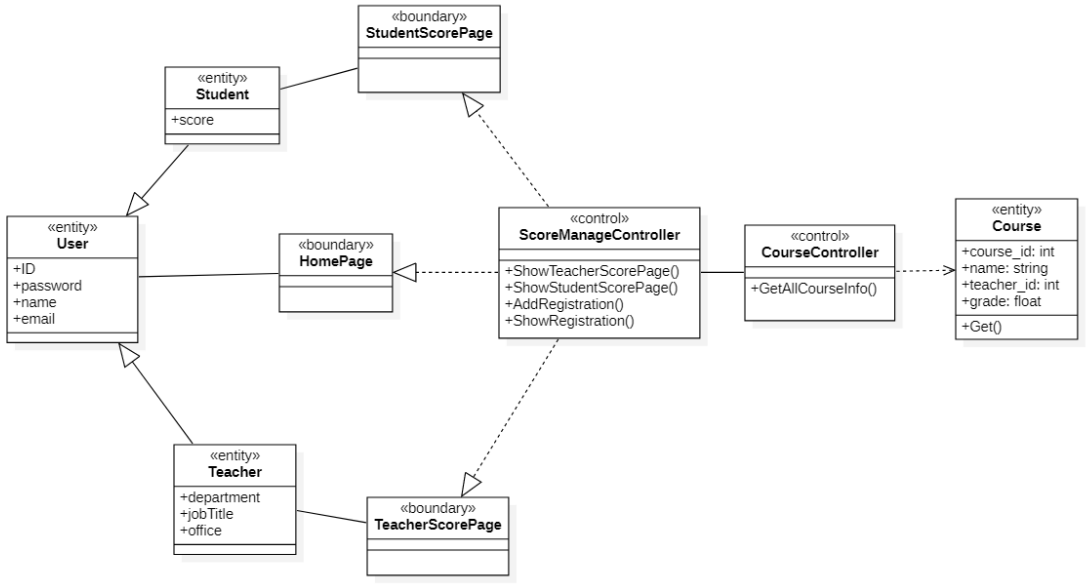


1.2.6 成绩管理子系统

1.2.6.1 系统简述

该子系统主要承担与实验成绩管理有关的所有工作，包括了教师发布签到、学生完成签到、教师和学生查看成绩等操作。

1.2.6.2 设计类图

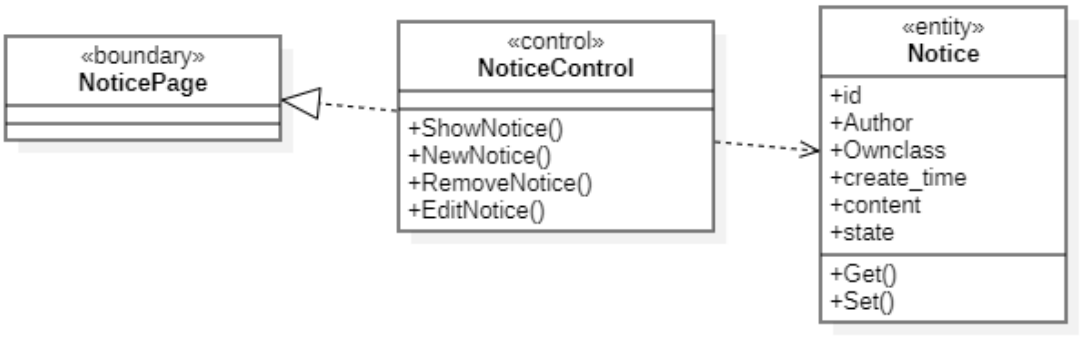


1.2.7 通知公告子系统

1.2.7.1 系统简述

该子系统主要承担与实验课程公告管理有关的所有工作，包括了教师发布、修改、删除公告等操作。

1.2.7.2 设计类图

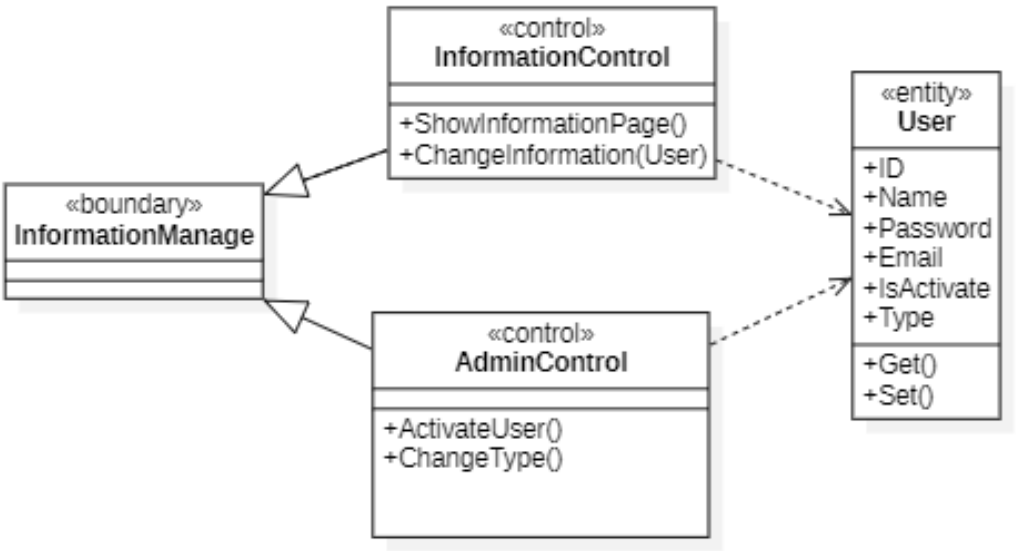


1.2.8 个人信息子系统

1.2.8.1 系统简述

该子系统主要承担与实验课程公告管理有关的所有工作，包括了用户修改个人信息等操作。

1.2.8.2 设计类图

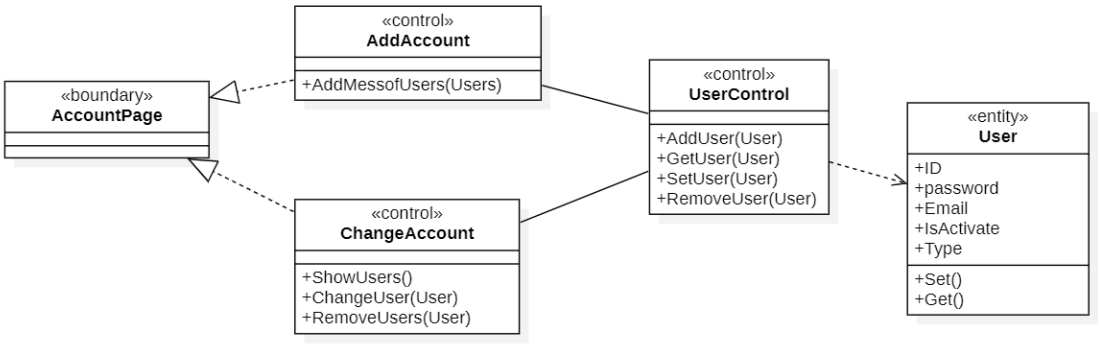


1.2.9 系统管理子系统

1.2.9.1 系统简述

该子系统主要承担与整个系统管理有关的所有工作，包括了用管理员批量添加用户、删除用户等操作。

1.2.9.2 设计类图



2 架构设计

2.1 总体架构

2.1.1 前端架构设计

为满足系统的跨平台要求和提高易用性，前端借助浏览器以 web 页面的形式呈现。以 vue.js 作为主体开发框架；为达到系统界面的美观性，使用了 Element UI 组件库，借助 vue-router 进行路由导航；vuex 做前端存储管理；利用 Ajax 动态请求数据，更新页面，提升用户友好性，使用第三方库 Axios 对 Ajax 进行封装，降低开发难度。

vue 是一款用于构建用户界面的 JavaScript 框架，其基于标准 HTML，CSS 和 JavaScript，并提供了声明式，组件化的编程模型。借助其响应性特性，动态更新页面 DOM，做到页面渲染和内部数据的双向同步。利用 vue 提供的单页面文件支持，模块化前端页面元素，降低系统耦合性。

Element UI 将前端页面常用组件进行封装，提供了大量美观、易用的 UI 元素，借助该组件库，有效降低系统开发难度，提升系统观感。

vue-router 为单页面应用而生，可方便的做到嵌套路由映射，动态路由导航，通过与 vuex 配合使用，动态生成路由表，做到细致的权限管理。

2.1.2 后端架构设计

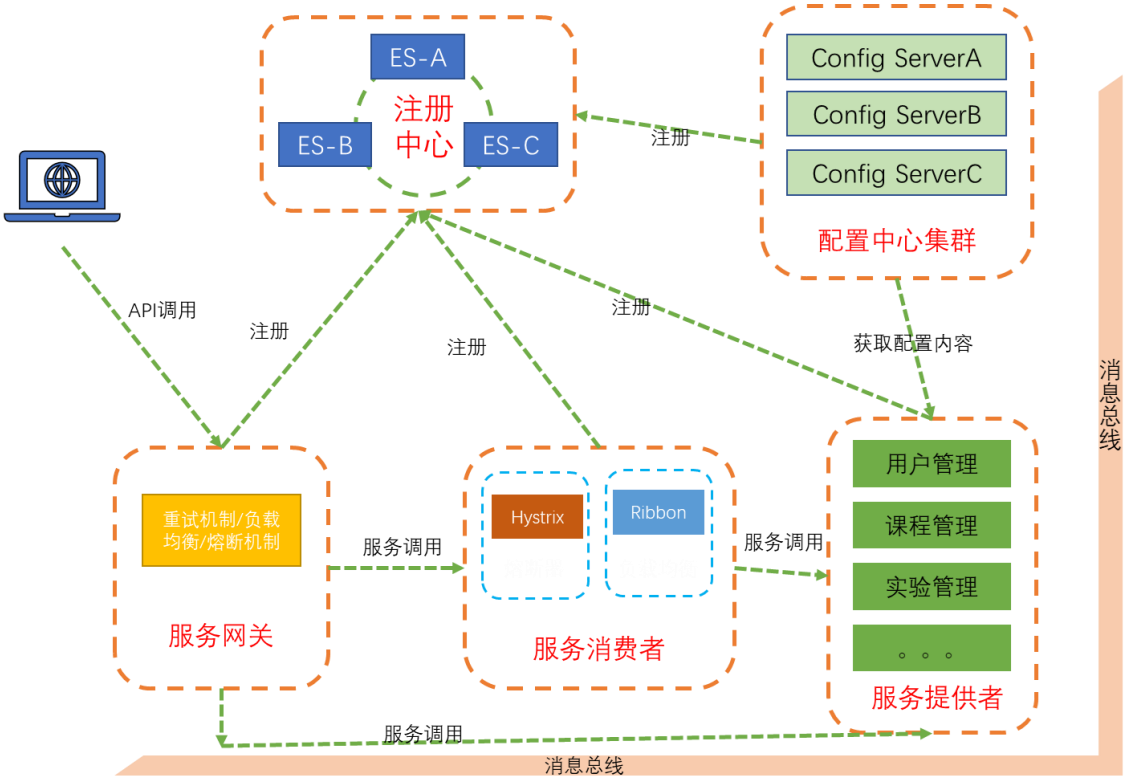
后端以 Java 作为开发语言，利用 Spring 生态，搭建微服务体系。微服务体系结构相比于单体服务，将业务模块进行了充分的解耦，提升了系统的健壮性，各模块也可按需扩容，业务切分同时降低了维护难度，便于融合其它技术栈，便于组织管理。

根据本实验教学管理平台的业务需求，将其分割为相对独立的 7 大模块，借助 SpringCloud 框架进行微服务体系的快速开发。SpringCloud 借助 Spring Boot 对微服务架构中的技术框架进行了有序集成，简化了分布式系统中基础设施的开发，如服务注册发现，配置中心，负载均衡，熔断，数据监控等。

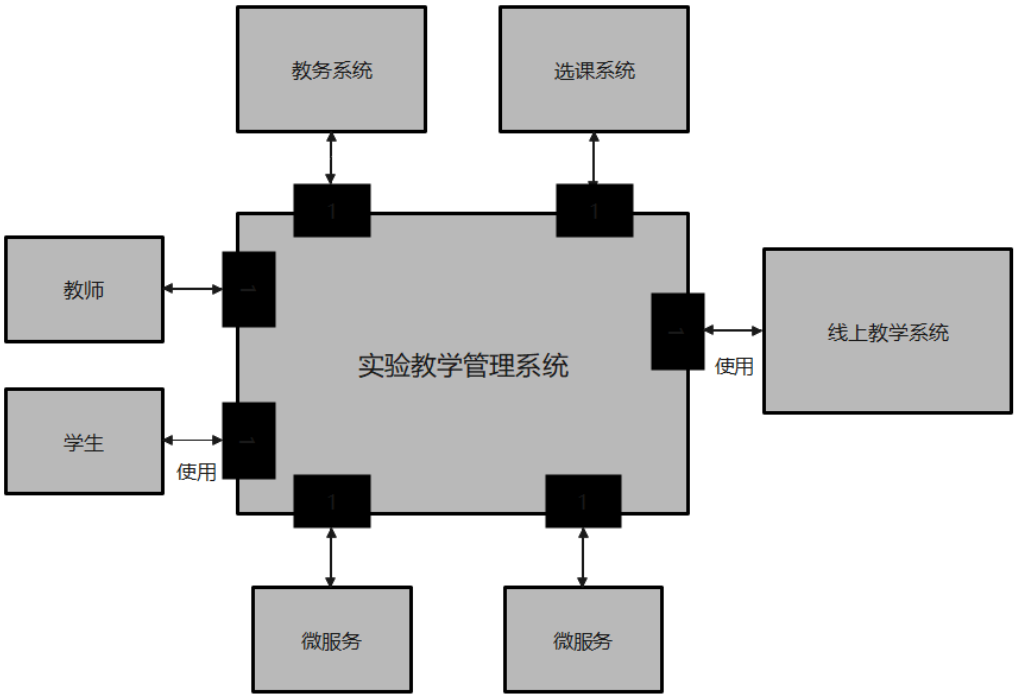
在本项目的开发中，采用 Eureka 进行服务的注册与发现，Feign 进行声明式服务调用，GateWay 作为 API 服务网关，为微服务的统一管理和相互调用提供支持。

2.2 微服务架构

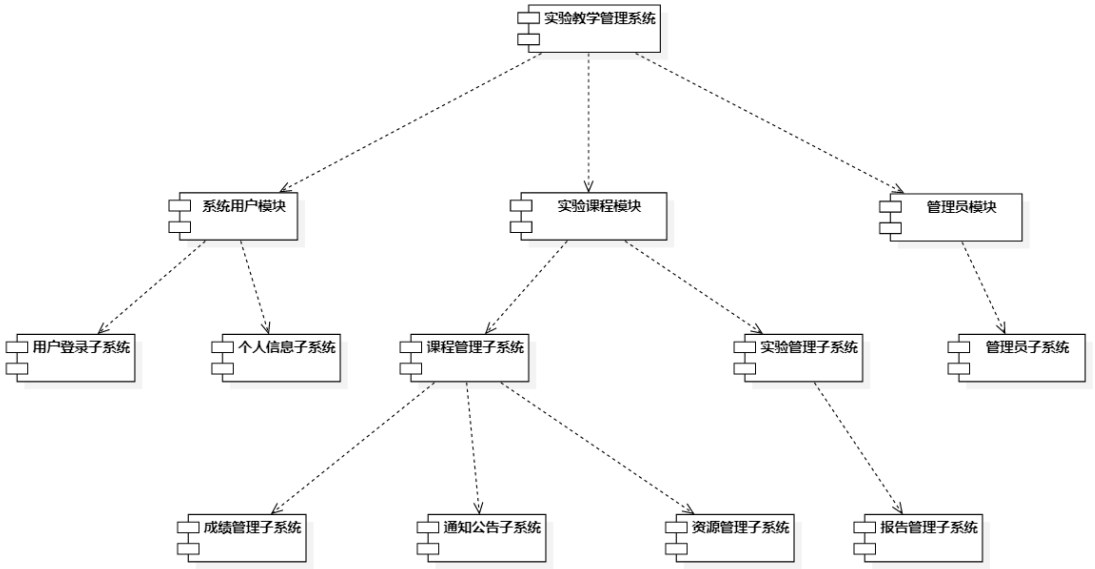
利用 SpringCloud 提供的工具链，我们搭建了适用于本套实验教学管理系统的微服务结构体系，具体包括用 Nacos 搭建的服务注册集群，以及服务配置集群，对于系统中可拆分的模块，我们另设一个微服务，并且在数据库层面上分表，拆分业务逻辑，尽可能降低各模块之间的耦合性，提高后期的可维护性，目前我们微服务架构体系可用下图简要描述。



2.3 架构上下文



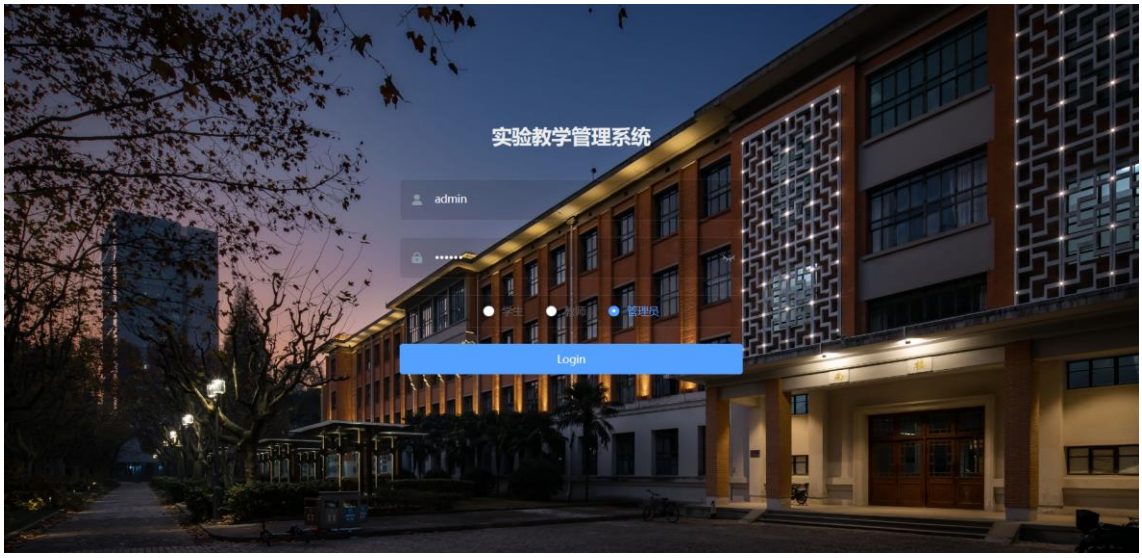
2.4 系统内部架构



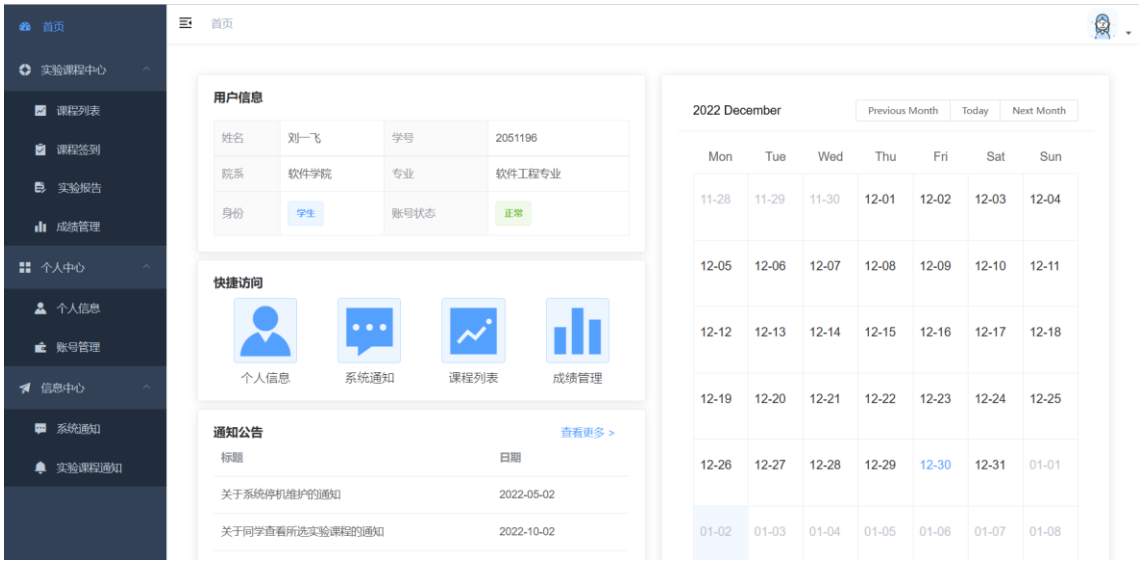
3 接口设计

3.1 图形用户界面

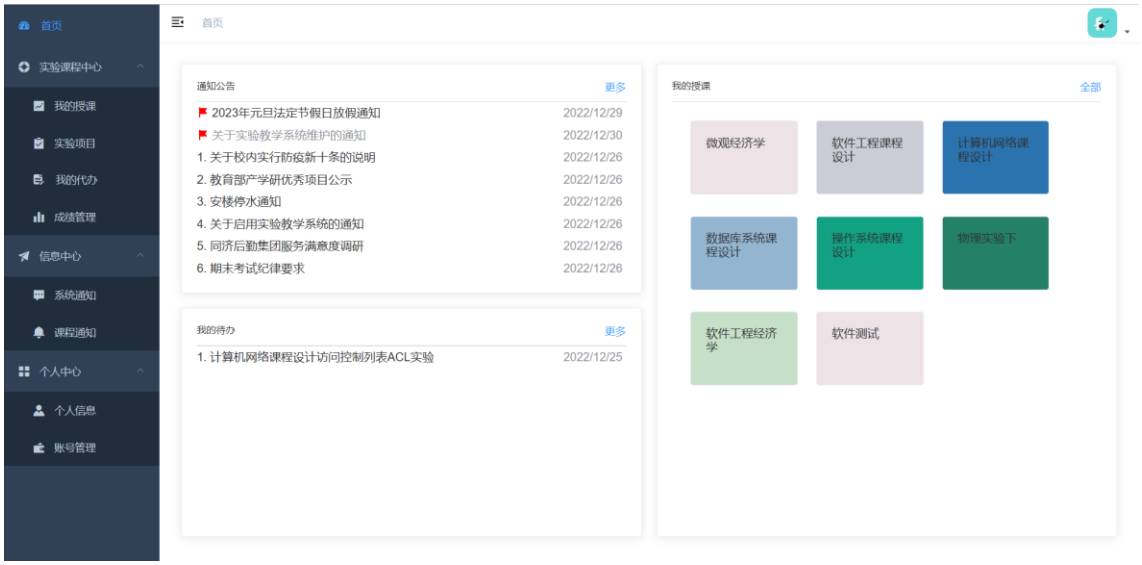
3.1.1 登录界面



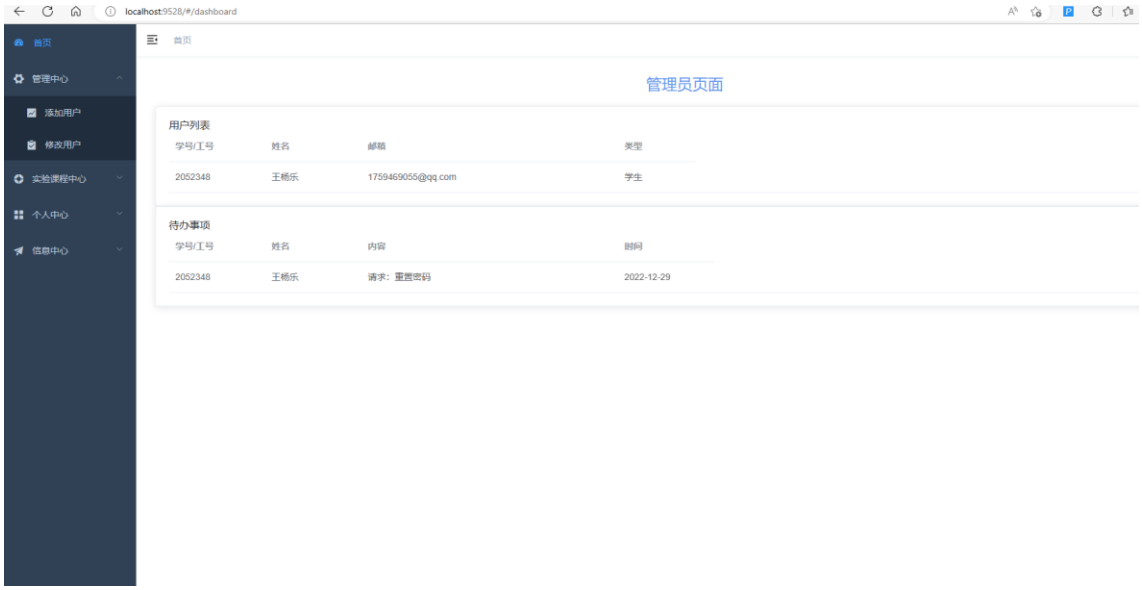
3.1.2 学生导航界面



3.1.3 教师导航界面

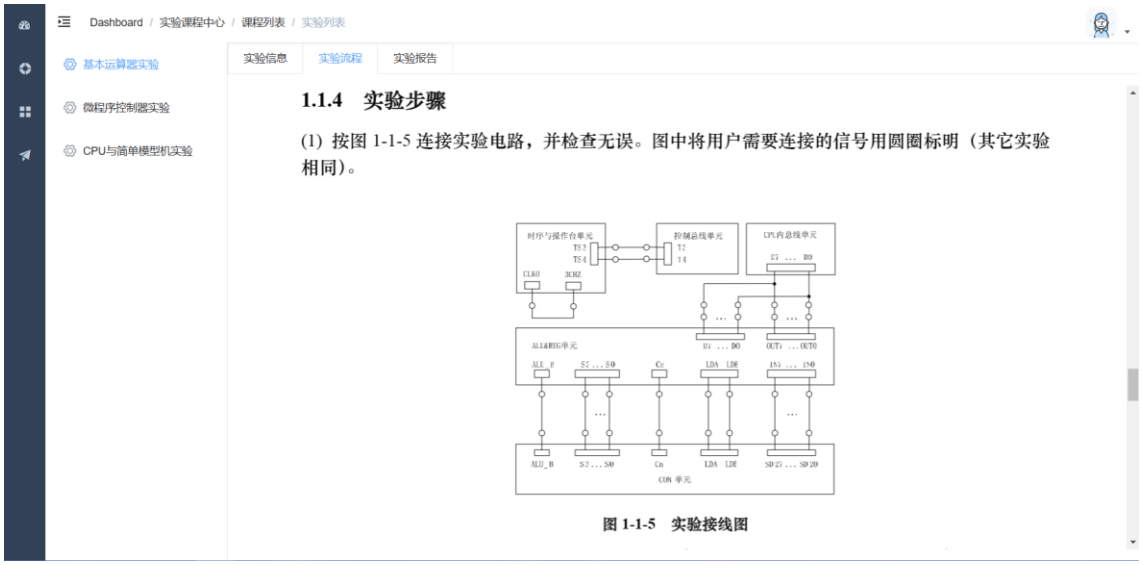


3.1.4 管理员导航界面



3.1.5 实验课程界面

3.1.6 实验项目界面



3.1.7 个人信息界面

3.2 内部接口

3.2.1 用户登录子系统

3.2.1.1 用户账户激活

接口地址	/api/login/activate
请求方式	POST
请求数据类型	application/x-www-formrmurlencoded,application/json
响应数据类型	*/*
接口描述	用于激活账号

- 请求示例:

```JavaScript
{
    "id":"",
    "email":"",
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注

id	学号或工号	true	bigint	
email	邮箱	true	string	用户输入的验证邮箱

- 响应示例

```JavaScript
{
    "status":1,
    "id":1,
    "name":"",
    "gender":"",
    "phone":""
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	status 返回 0 表示失败，1 表示成功
id	用户 ID	bigint	
name	用户姓名	string	
gender	用户性别	string	
phone	用户手机号码	string	

3.2.1.2 用户登录

接口地址	/api/login/login
请求方式	POST
请求数据类型	application/x-www-formrmurlencoded,application/json
响应数据类型	*/*

接口描述	用于验证登录请求

- 请求示例:

```JavaScript
{
    "id":"",
    "password":""
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
id	学号或工号	true	bigint	
password	账户密码	true	string	

- 响应示例

```JavaScript
{
    "id":1,
    "name":"",
    "type":"",
    "phone":""
}
```

- 响应参数

参数名称	参数说明	类型	备注
id	用户 ID	bigint	
name	用户姓名	string	
type	用户类型	string	
phone	用户手机号码	string	

3.2.1.3 忘记密码

接口地址	/api/login/login
请求方式	PUT
请求数据类型	application/x-www-formrmurlencoded,application/json
响应数据类型	*/*
接口描述	用于用户忘记密码时重置密码

- 请求示例:

```JavaScript
{
    "email":"",
    "new_passwd":""
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
email	邮箱名	true	string	系统会向邮箱中发送验证信息
new_passwd	用户新设置的密码	true	string	

- 响应示例

```JavaScript
{
    "status":1,
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	status 返回 0 表示失败，1 表示成功

3.2.2 课程管理子系统

3.2.2.1 用户获取课程列表

接口地址	/api/course/getCourseList
请求方式	GET
请求数据类型	application/x-www-formrmurlencoded,application/json
响应数据类型	*/*
接口描述	用于获取用户所有的课程列表

- 请求示例:

```
JavaScript
{
    "id":1,
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
id	用户 ID	true	bigint	

- 响应示例

```
JavaScript
{
    "courseList":[
        {
            "course_id":1,
            "name":"",
            "introduction":"",
            "create_time":yyyy-MM-dd hh:mm:ss
        }
    ]
}
```

- 响应参数

参数名称	参数说明	类型	备注
course_id	课程 ID	bigint	
name	课程名称	string	
introduction	课程简介	string	
create_time	课程创建时间	date	时间格式为 yyyy-MM-dd hh:mm:ss

3.2.2.2 责任教师添加课程

接口地址	/api/course/addCourse
请求方式	POST
请求数据类型	application/json
响应数据类型	*/*
接口描述	用于责任教师添加新的课程

- 请求示例:

```
JavaScript
{
    "teacher_id":1,
    "name":"",
    "introduction":"",
    "create_time":yyyy-MM-dd hh:mm:ss
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
teacher_id	责任教师 ID	true	bigint	

name	课程名称	true	string	
introduction	课程简介	false	string	
create_time	课程创建时间	true	date	时间格式为 yyyy-MM-dd hh:mm:ss

- 响应示例

```
JavaScript
{
    "status":1,
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	status 返回 0 表示失败，1 表示成功

3.2.2.3 责任教师修改课程

接口地址	/api/course/modifyCourse
请求方式	PUT
请求数据类型	application/json
响应数据类型	*/*
接口描述	用于责任教师修改现有的课程信息

- 请求示例:

```
JavaScript
{
    "teacher_id":1,
    "course_id":1,
    "name":"",
    "introduction":"",
```

```
    "create_time":yyyy-MM-dd hh:mm:ss
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
teacher_id	责任教师 ID	true	bigint	
course_id	课程 ID	true	bigint	
name	课程名称	true	string	
introduction	课程简介	false	string	
create_time	课程创建时间	true	date	时间格式为 yyyy-MM-dd hh:mm:ss

- 响应示例

```JavaScript
{
    "status":1,
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	status 返回 0 表示失败，1 表示成功

3.2.2.4 责任教师删除课程

接口地址	/api/course/removeCourse
请求方式	DELETE
请求数据类型	application/json

响应数据类型	*/*
接口描述	用于责任教师删除课程

- 请求示例:

```
JavaScript
{
    "teacher_id":1,
    "course_id":1,
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
teacher_id	责任教师 ID	true	bigint	
course_id	课程 ID	true	bigint	

- 响应示例

```
JavaScript
{
    "status":1,
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	status 返回 0 表示失败，1 表示成功

3.2.3 实验管理子系统

3.2.3.1 用户获取实验列表

接口地址	/api/course/getExperimentList

请求方式	GET
请求数据类型	application/json
响应数据类型	*/*
接口描述	用于用户获取实验项目列表

- 请求示例:

```JavaScript
{
    "id":1,
    "course_id":1,
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
id	用户 ID	true	bigint	
course_id	课程 ID	true	bigint	根据课程 ID 寻找该课程下属的所有实验

- 响应示例

```JavaScript
{
    "experimentList":[
        {
            "experiment_id":1,
            "name":"",
            "introduction":"",
            "create_time":yyyy-MM-dd hh:mm:ss,
            "deadline":yyyy-MM-dd hh:mm:ss
        }
    ]
}
```

- 响应参数

参数名称	参数说明	类型	备注
experiment_id	实验 ID	bigint	
name	实验名称	stirng	
introduction	实验简介	string	
create_time	实验创建时间	date	时间格式为 yyyy-MM-dd hh:mm:ss
dead_line	实验截止时间	date	时间格式为 yyyy-MM-dd hh:mm:ss

3.2.3.2 教师添加实验项目

接口地址	/api/course/addExperiment
请求方式	POST
请求数据类型	application/json
响应数据类型	*/*
接口描述	用于教师添加新的实验项目

- 请求示例:

```JavaScript
{
    "teacher_id":1,
    "name":"",
    "introduction":"",
    "create_time":yyyy-MM-dd hh:mm:ss,
    "dead_line":yyyy-MM-dd hh:mm:ss
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类	备注

			型	
teacher_id	教师 ID	true	bigint	
name	实验名称	true	string	
introduction	实验简介	false	string	
create_time	实验创建时间	true	date	时间格式为 yyyy-MM-dd hh:mm:ss
dead_line	实验截止时间	true	date	时间格式为 yyyy-MM-dd hh:mm:ss

- 响应示例

```JavaScript
{
    "status":1,
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	status 返回 0 表示失败，1 表示成功

3.2.3.3 教师修改实验项目

接口地址	/api/course/modifyExperiment
请求方式	PUT
请求数据类型	application/json
响应数据类型	*/*
接口描述	用于教师修改实验项目信息

- 请求示例:

```JavaScript
{
    "teacher_id":1,
    "experiment_id":1,
    "name":"",
    "introduction":"",
    "create_time":yyyy-MM-dd hh:mm:ss
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
teacher_id	责任教师 ID	true	bigint	
experiment_id	实验 ID	true	bigint	
name	实验名称	true	string	
introduction	实验简介	false	string	
create_time	实验创建时间	true	date	时间格式为 yyyy-MM-dd hh:mm:ss
dead_line	实验截止时间	true	date	时间格式为 yyyy-MM-dd hh:mm:ss

- 响应示例

```JavaScript
{
    "status":1,
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	status 返回 0 表示失败，1 表示成功

3.2.3.4 教师删除实验项目

接口地址	/api/course/removeExperiment
请求方式	DELETE
请求数据类型	application/json
响应数据类型	*/*
接口描述	用于用户获取实验项目列表

- 请求示例:

```JavaScript
{
    "teacher_id":1,
    "experiment_id":1,
}
```

- 请求参数:

参数名称	参数说明	是否必须	数据类型	备注
teacher_id	责任教师 ID	true	bigint	
experiment_id	实验 ID	true	bigint	

- 响应示例

```JavaScript
{
    "status":1,
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	status 返回 0 表示失败，1 表

			示成功

3.2.4 实验报告子系统

3.2.4.1 教师上传实验模板

接口地址	/api/report/uploadreporttemplate
请求方式	POST
请求数据类型	application/binary
响应数据类型	application/json
接口描述	用于上传实验报告模板文件(.docx)

请求示例:

```
JavaScript
{
    "teacher_id": 1,
    "course_id": 1,
    "experiment_id": 1,
    "template_file":"",
 }
```

请求参数:

参数名称	参数说明	类型	备注
teacher_id	教师 id	int	
course_id	课程编号	int	
experiment_id	实验编号	int	
template_file	模板文件内容	file	

- 响应示例

```
JavaScript
```

```
{
    "status":1
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	0 表示失败，1 表示成功

3.2.4.2 学生获取实验模板

接口地址	/api/report/downloadreporttemplate
请求方式	GET
请求数据类型	application/x-www-formrmurlencoded
响应数据类型	application/binary
接口描述	用于下载实验报告模板文件(.docx)

- 请求参数：无
- 响应示例：

```JavaScript
{
    "teacher_id": 1,
    "course_id": 1,
    "experiment_id": 1,
    "template_file":"",
 }
```

- 响应参数：

参数名称	参数说明	类型	备注
teacher_id	教师 id	int	
course_id	课程编号	int	

experiment_id	实验编号	int	
template_file	模板文件内容	file	

3.2.4.3 学生提交实验报告

接口地址	/api/report/reporthandin
请求方式	POST
请求数据类型	application/binary
响应数据类型	application/json
接口描述	用于学生提交实验报告至服务器

- 请求示例:

```JavaScript
{
    "student_id": 1,
    "course_id":1,
    "experiment_id":1,
    "file":"",
 }
```

- 请求参数:

参数名称	参数说明	类型	备注
student_id	学生 id	int	
course_id	课程编号	int	
experiment_id	实验编号	int	
file	报告文件内容	file	

- 响应示例

```JavaScript
```

```
{
    "status":1
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	0 表示失败，1 表示成功

3.2.4.4 教师/助教批阅实验报告

接口地址	/api/report/checkreport
请求方式	POST
请求数据类型	application/x-www-formrmurlencoded
响应数据类型	application/json
接口描述	用于给学生的实验报告打分

- 请求示例:

```
JavaScript
{
    "student_id": 1,
    "course_id": 1,
    "experiment_id": 1,
    "score": 1,
 }
```

- 请求参数:

参数名称	参数说明	类型	备注
student_id	学生 id	int	
course_id	课程编号	int	
experiment_id	实验编号	int	

score	实验报告分数	int	

- 响应示例

```
JavaScript
{
    "status":1
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	0 表示失败，1 表示成功

3.2.5 课程文件子系统

3.2.5.1 获取该课程的所有相关文件

接口地址	/api/resource
请求方式	GET
请求数据类型	application/json
响应数据类型	*/*
接口描述	获取课程的所有相关文件

- 请求示例

```
JavaScript
{
    "courseId":"",
}
```

- 响应示例

```
JavaScript
{
    "courseId":"",
```

```
    "courseName":"",
    "sources":[
        {
            "fileName":"",
            "fileSzie":"",
            "url":""
        },
    ]
}
```

3.2.5.2 教师上传文件

接口地址	/api/resource/postone
请求方式	POST
请求数据类型	application/json、application/octet-stream
响应数据类型	*/*
接口描述	上传文件至数据库

- 请求示例

```
JavaScript
{
    "courseId":"",
    "courseName":"",
    "source":
        {
            "fileName":"",
            "fileSzie":"",
            "filetype":"",
            "content":"",
        },
}
```

- 响应示例

```
JavaScript
{
    "status": 1,
```

```
}
```

3.2.5.3 用户下载文件

接口地址	/api/resource/getone
请求方式	GET
请求数据类型	application/json
响应数据类型	application/octet-stream
接口描述	获取文件下载链接

- 请求示例

```
JavaScript
{
    "courseId":"",
    "courseName":"",
    "sourcepath":"",
}
```

- 响应示例

```
JavaScript
{
     "source":
        {
            "fileName":"",
            "fileSzie":"",
            "filetype":"",
            "content":"",
        },
}
```

3.2.6 学生成绩子系统

3.2.6.1 教师提交成绩

接口地址	/api/grade/submit

请求方式	POST
请求数据类型	application/json
响应数据类型	*/*
接口描述	教师提交学生课程成绩总评

- 请求参数

```JavaScript
{
    "courseId":"",
    "teacherId":"",
    "grade":[
        {
            "studentId":"",
            "grade":""
        }
    ]
}
```

3.2.6.2 学生查看成绩

接口地址	/api/grade/selfGrade
请求方式	GET
请求数据类型	application/json
响应数据类型	*/*
接口描述	学生查看自己各个课程的成绩

- 请求示例

```JavaScript
{
    "studentId":""
}
```

- 响应示例

```JavaScript
{
    "Id":"",
    "grade":[
        {
            "courseId":"",
            "courseName":"",
            "score":"",
            "grade":""
        },
        。。。
    ]
}
```

3.2.7 通知公告子系统

3.2.7.1 获取公告列表

接口地址	/api/notice/getNoticeList
请求方式	GET
请求数据类型	application/x-www-formrmurlencoded
响应数据类型	*/*
接口描述	用于获取公告列表

响应示例:

```JavaScript
{
    "content": "",
    "course_id": "",
    "create_time": "",
    "notice_id": "",
    "teacher_id": "",
    "title": "",
 }
```

响应参数:

参数名称	参数说明	请求类型	是否必须	数据类型	备注
notice	Notice	POST	true	body	
content	公告内容		false	string	
course_id	课号		false	int	
create_time	发布时间		false	date	
notice_id	公告 id		false	int	数据库在创建时生成
teacher_id	发布者 id		false	int	
title	公告标题		false	string	

- 响应示例
- 响应参数

3.2.7.2 新增公告信息

接口地址	/api/notice/newNotice
请求方式	POST
请求数据类型	application/x-www-formrmurlencoded,application/json
响应数据类型	*/*
接口描述	新增公告

请求示例:

```JavaScript
{
    "content": "",
    "course_id": 0,
    "create_time": "",
    "teacher_id": "",
```

```
    "title": "",
 }
```

请求参数:

参数名称	参数说明	类型	备注
notice			
content	公告内容	string	
course_id	课号	int	
create_time	日期	date	date 格式
publisher_id	发布者 ID	int	
title	公告标题	string	

- 响应示例
- 响应参数

3.2.7.3 更新公告状态

接口地址	/api/admin/updateUser
请求方式	POST
请求数据类型	application/x-www-formrmurlencoded
响应数据类型	*/*
接口描述	用于更新某个特定公告

- 请求示例:

```JavaScript
{
    "notice_id": 0,
    "state":0,
 }
```

- 请求参数:

参数名称	参数说明	类型	备注
notice_id	公告 ID	int	
state	公告状态	int	

- 响应示例

```JavaScript
{
    "state":1
}
```

- 响应参数

参数名称	参数说明	类型	备注
state	公告当前状态	int	

3.2.8 个人信息子系统

3.2.8.1 个人信息获取

接口地址	/api/person/getInformation/
请求方式	GET
请求数据类型	application/x-www-formrmurlencoded
响应数据类型	*/*
接口描述	用于请求个人信息

请求示例:

```JavaScript
{
    "id":"",
}
```

请求参数:

参数名称	参数说明	请求类型	是否必须	数据类型	备注
id	学号或工号		false		

响应示例:

```JavaScript
{
    "id":"",
    "name":"",
    "gender":1,
    "email":"",
    "phone":"",
    "avatar":"",
    "schoolNum":"",
}
```

响应参数:

参数名称	参数说明	类型	备注
user	用户信息结构体	body	
id	学号或工号	int	
name	用户姓名	string	
gender	性别	boolean	
email	邮箱	string	
phone	电话	string	
avatar	头像	string	

3.2.8.2 修改个人信息

接口地址	/api/person/setInformation

请求方式	PUT
请求数据类型	application/x-www-formrmurlencoded,application/json
响应数据类型	*/*
接口描述	用于更新个人信息

请求示例:

```JavaScript
{
    "id":"",
    "name":"",
    "gender":1,
    "email":"",
    "phone":"",
}
```

请求参数:

参数名称	参数说明	请求类型	是否必须	数据类型	备注
user	用户信息结构体	body	true	body	
id	学号或工号	int	false	int	
name	用户姓名	string	false	string	
gender	性别	boolean	false	int	
email	邮箱	string	false	string	
phone	电话	string	false	string	

- 响应示例
- 响应参数

3.2.9 管理员子系统

3.2.9.1 录入账号列表

接口地址	/api/admin/uploadUser
请求方式	POST
请求数据类型	application/x-www-formrmurlencoded
响应数据类型	application/json
接口描述	用于上传包含账户信息的 excel 文件(.xlsx)

请求示例:

```JavaScript
{
    "Userlist":[
        "ID": 1,
        "name":"",
        "gender":1,
        "email":"",
        "type":"",
    ]
 }
```

请求参数:

参数名称	参数说明	类型	备注
Userlist	存储在 excel 文件中上传		
ID	学号/工号	int	
name	用户姓名	string	
gender	性别	int	
email	邮箱	string	
type	类型	string	

- 响应示例

```JavaScript
{
    "status":1
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	0 表示失败，1 表示成功

3.2.9.2 获取账号列表

接口地址	/api/admin/downloadUser
请求方式	GET
请求数据类型	application/x-www-formrmurlencoded
响应数据类型	application/json
接口描述	用于获取账户信息

- 请求参数：无
- 响应示例：

```JavaScript
{
     "Userlist":[
        "ID": 1,
        "name":"",
        "gender":1,
        "email":"",
        "type":"",
    ]
 }
```

- 响应参数：

参数名称	参数说明	类型	备注

ID	学号/工号	int	
name	用户姓名	string	
gender	性别	int	
email	邮箱	string	
type	类型	string	

3.2.9.3 更新账号信息

接口地址	/api/admin/updateUser
请求方式	POST
请求数据类型	application/x-www-formrmurlencoded
响应数据类型	application/json
接口描述	用于更新某个特定用户信息

- 请求示例:

```
JavaScript
{
    "ID": 1,
    "name":"",
    "gender":"",
    "email":"",
    "type":"",
 }
```

- 请求参数:

参数名称	参数说明	类型	备注
ID	学号/工号	int	禁止修改，只能注销后新增

name	用户姓名	string	
gender	性别	string	
email	邮箱	string	
type	类型	string	

- 响应示例

```JavaScript
{
    "status":1
}
```

- 响应参数

参数名称	参数说明	类型	备注
status	操作成功与否	int	0 表示失败，1 表示成功

3.3 外部接口

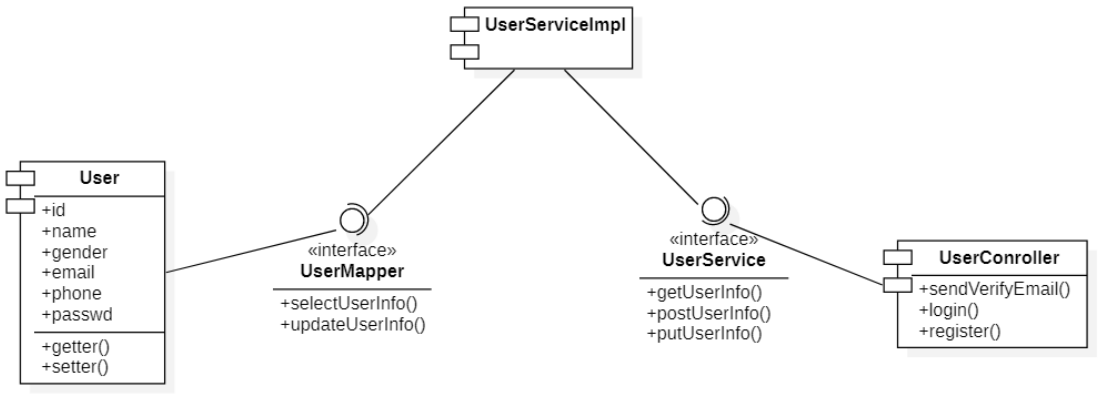
3.3.1 JavaMail 邮件发送

通过调用 JavaMail API 实现用户在登录注册时需要完成的邮件验证功能，使用 SMTP 协议将邮件发送给对应用户。

4 构件设计

4.1 用户管理构件

4.1.1 构件图



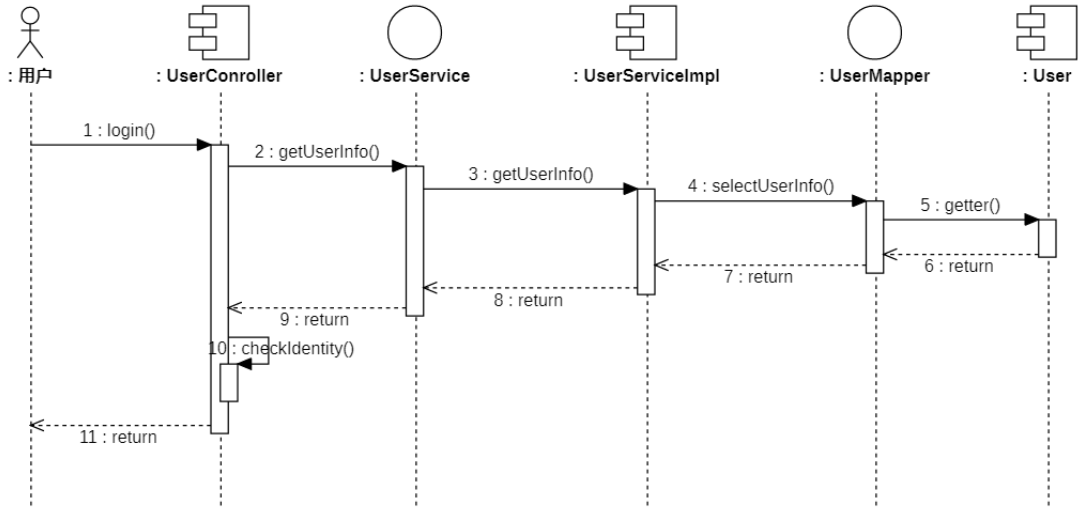
4.1.2 核心操作

4.1.2.1 login

4.1.2.1.1 操作简述

该操作对应用户正常登录，调用 Controller 中对外的 login 方法后，会调用业务层 Service 中的 getUserInfo 获取传入的登录信息中的用户 ID，接着调用 DAO 层 Mapper 中的 selectUserInfo 根据用户 ID 在数据库中查找到对应的用户信息并返回，UserController 会比对用户输入的信息和数据库中保存的信息，并根据结果判断是否登录成功。

4.1.2.1.2 时序图



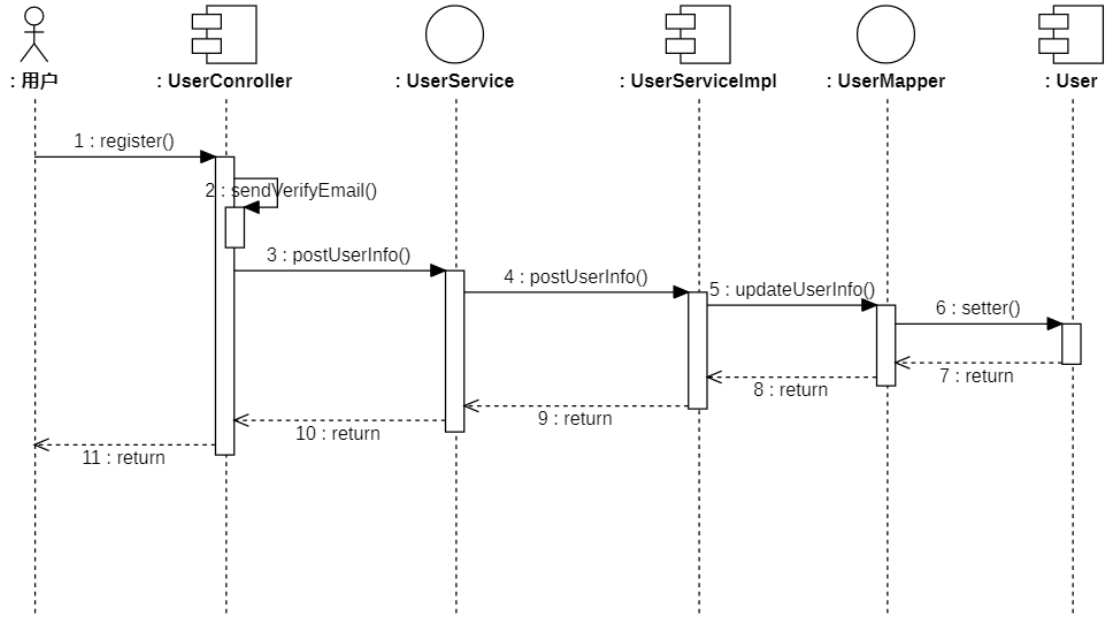
4.1.2.2 register

4.1.2.2.1 操作简述

该操作对应用户的账户注册，调用 Controller 中对外的 register 方法后，会调用业

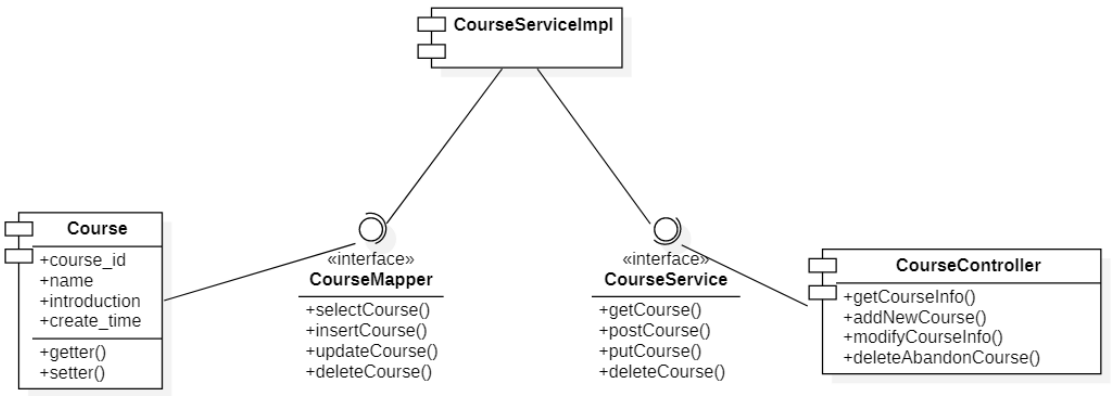
务层 Service 中的 sendVerifyEmail 根据用户输入的邮箱信息向用户发送验证邮件，用户完成验证后，会调用 Service 中的 postUserInfo 将用户输入的信息组装成用户对象，接着调用 DAO 层 Mapper 中的 updateUserInfo 根据用户 ID 在数据库中更新对应的用户信息，从而完成账户注册流程。

4.1.2.2.2 时序图



4.2 课程管理构件

4.2.1 构件图



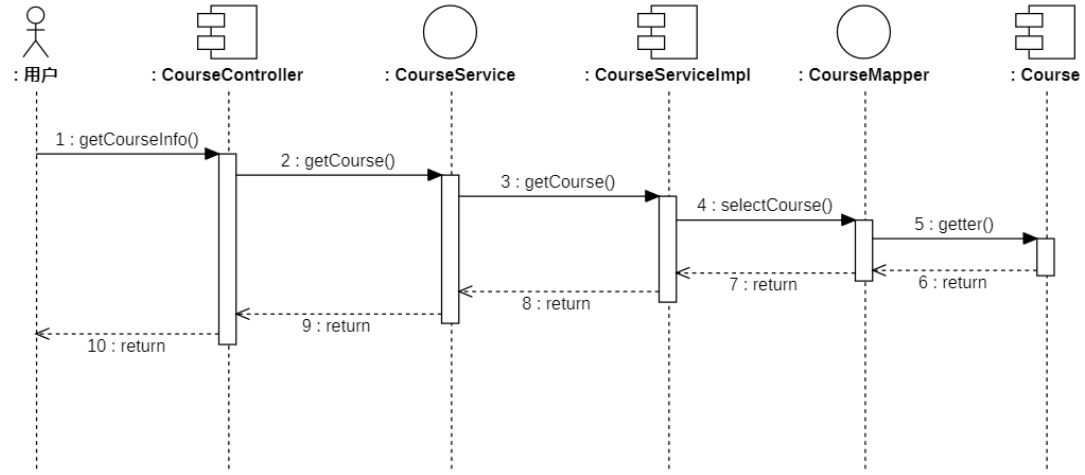
4.2.2 核心操作

4.2.2.1 getCourseInfo

4.2.2.1.1 操作简述

该操作对应用户获取课程信息，调用 Controller 中对外的 getCourseInfo 方法后，会调用业务层 Service 中的 getCourse 获取传入的课程 ID，接着调用 DAO 层 Mapper 中的 selectCourse 根据课程 ID 在数据库中查找到对应的课程信息并返回。

4.2.2.1.2 时序图

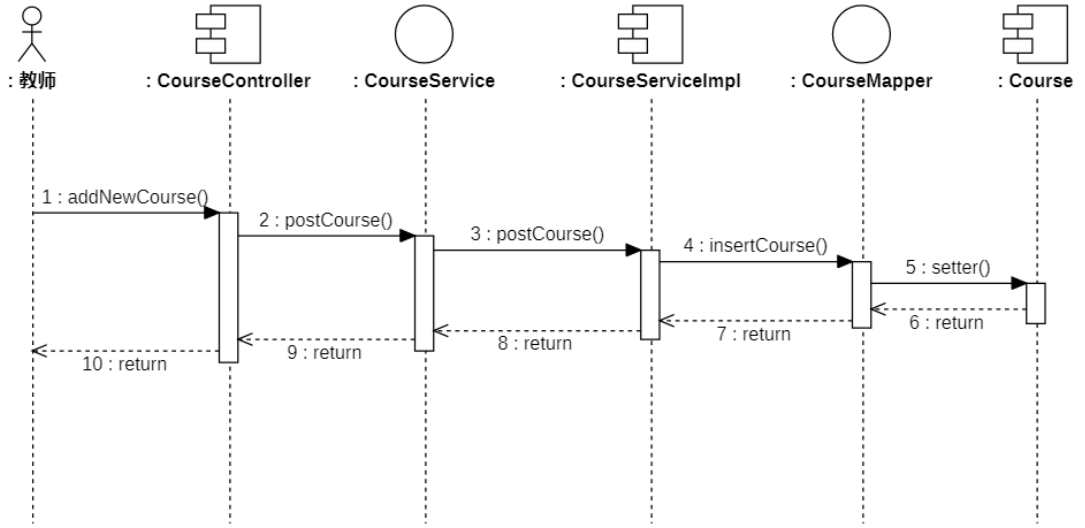


4.2.2.2 addNewCourse

4.2.2.2.1 操作简述

该操作对应教师添加新的实验课程，调用 Controller 中对外的 addNewCourse 方法后，会调用业务层 Service 中的 postCourse 对传入的课程信息封装为课程对象，接着调用 DAO 层 Mapper 中的 insertCourse 完成课程信息在数据库中的插入。

4.2.2.2.2 时序图

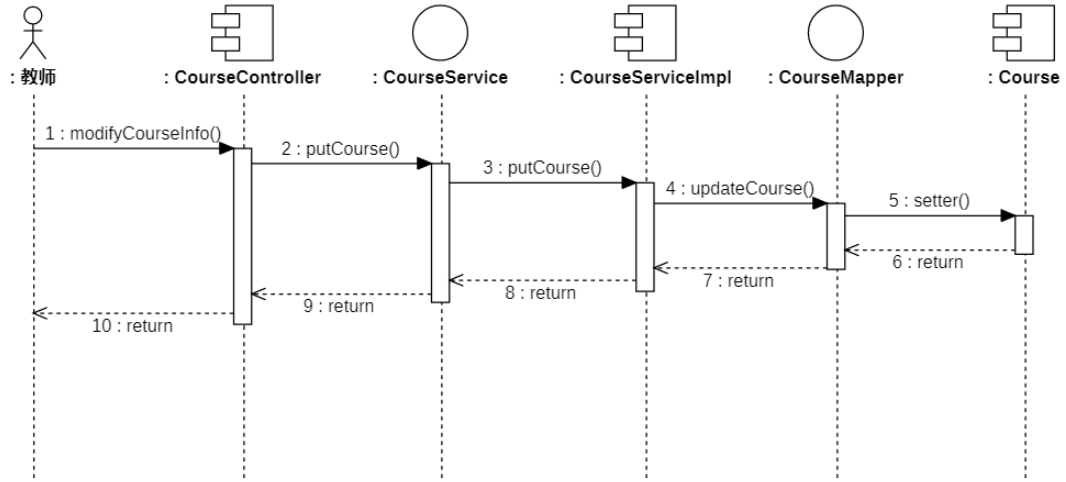


4.2.2.3 modifyCourseInfo

4.2.2.3.1 操作简述

该操作对应教师修改已有实验课程的信息，调用 Controller 中对外的 modifyCourseInfo 方法后，会调用业务层 Service 中的 putCourse 对传入的课程信息封装为课程对象，接着调用 DAO 层 Mapper 中的 updateCourse 完成对课程信息的更新。

4.2.2.3.2 时序图

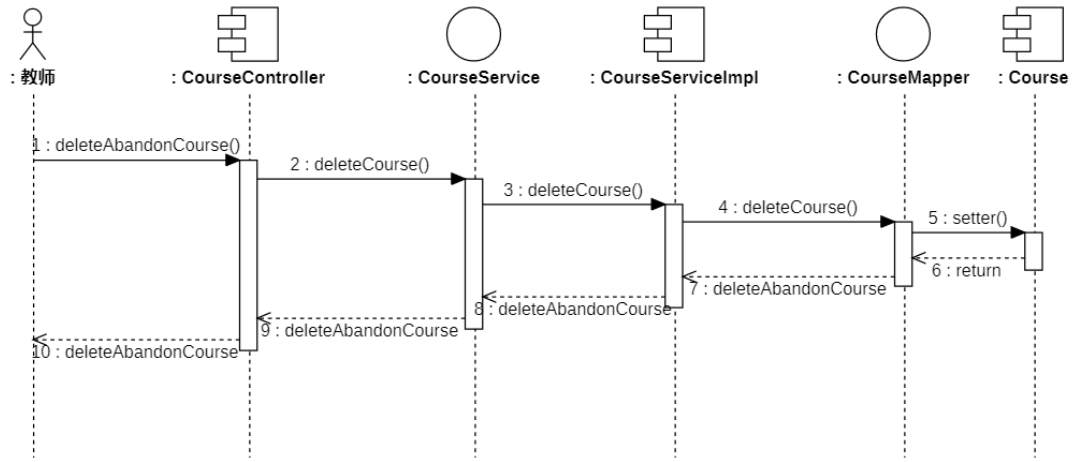


4.2.2.4 deleteAbandonCourse

4.2.2.4.1 操作简述

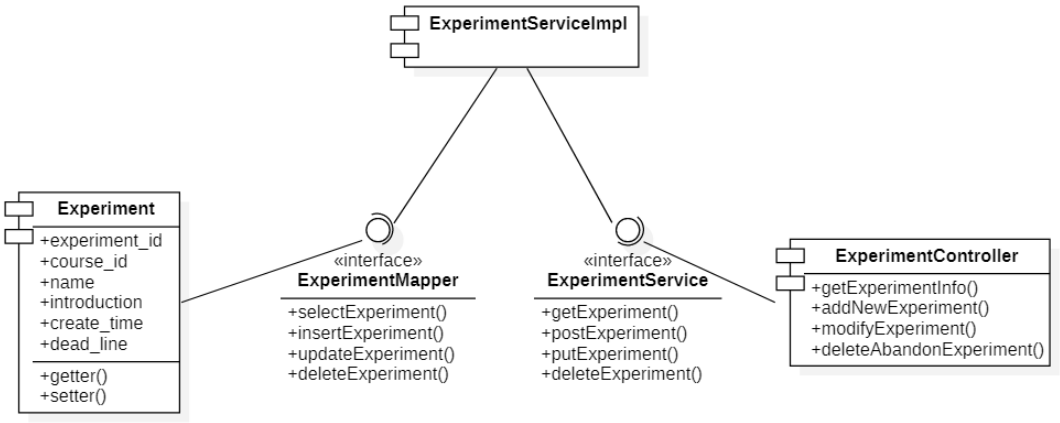
该操作对应教师删除已经废弃的实验课程，调用 Controller 中对外的 deleteAbandonCourse 方法后，会调用业务层 Service 中的 deleteCourse 获取要删除的实验课程的 ID，接着调用 DAO 层 Mapper 中的 deleteCourse 根据指定的课程 ID 完成课程信息在数据库中的删除。

4.2.2.4.2 时序图



4.3 实验管理构件

4.3.1 构件图



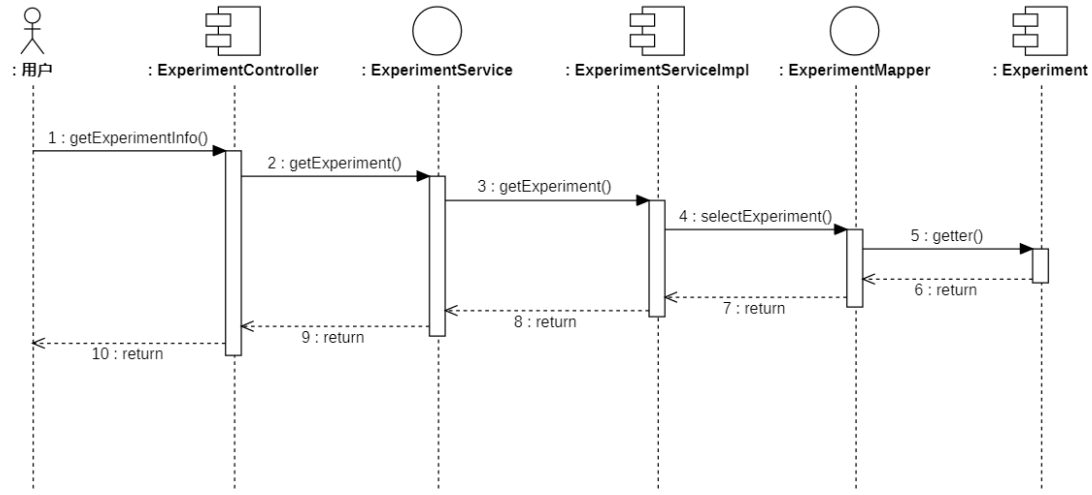
4.3.2 核心操作

4.3.2.1 getExperimentInfo

4.3.2.1.1 操作简述

该操作对应用户获取实验项目信息，调用 Controller 中对外的 getExperimentInfo 方法后，会调用业务层 Service 中的 getExperiment 获取传入的实验项目 ID，接着调用 DAO 层 Mapper 中的 selectExperiment 根据实验项目 ID 在数据库中查找到对应的实验信息并返回。

4.3.2.1.2 时序图

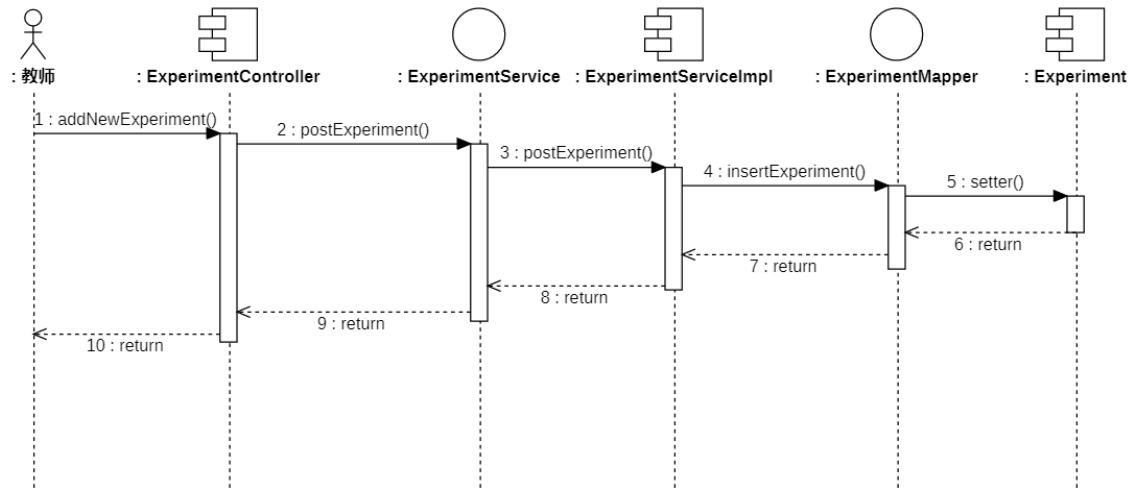


4.3.2.2 addNewExperiment

4.3.2.2.1 操作简述

该操作对应教师添加新的实验项目，调用 Controller 中对外的 addNewExperiment 方法后，会调用业务层 Service 中的 postExperiment 对传入的实验信息封装为实验对象，接着调用 DAO 层 Mapper 中的 insertExperiment 完成实验项目信息在数据库中的插入。

4.3.2.2.2 时序图



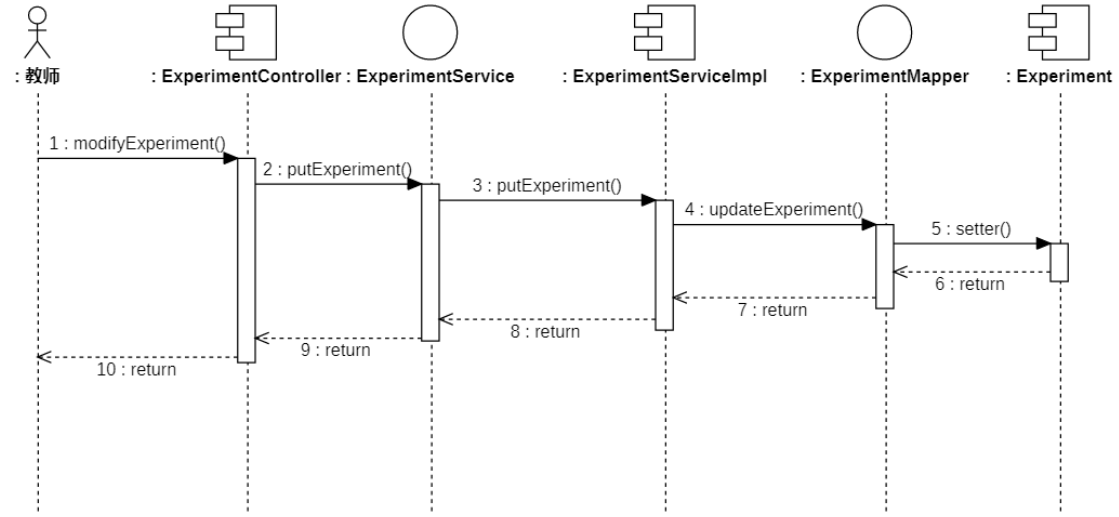
4.3.2.3 modifyExperiment

4.3.2.3.1 操作简述

该操作对应教师修改已有实验项目的信息，调用 Controller 中对外的 modifyExperimentInfo 方法后，会调用业务层 Service 中的 putExperiment 对传入的实验信息封装为实验对象，接着调用 DAO 层 Mapper 中的 updateExperiment 完成对实

验项目信息的更新。

4.3.2.3.2 时序图

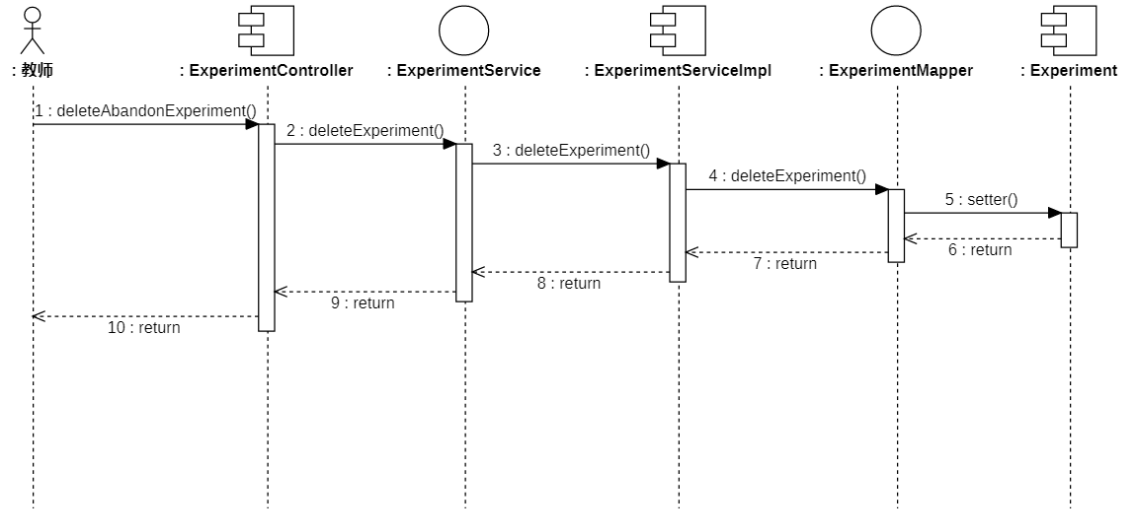


4.3.2.4 deleteAbandonExperiment

4.3.2.4.1 操作简述

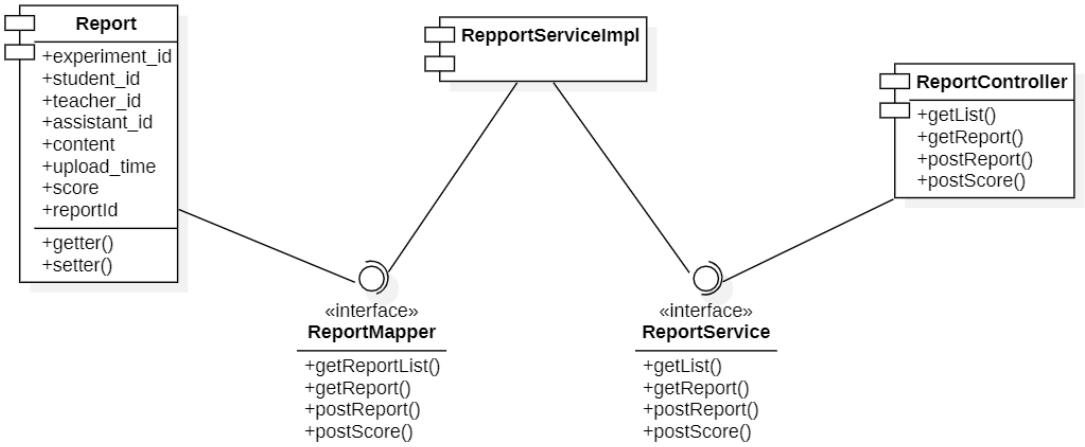
该操作对应教师删除已经废弃的实验项目，调用 Controller 中对外的 deleteAbandonExperiment 方法后，会调用业务层 Service 中的 deleteExperiment 获取要删除的实验项目的 ID，接着调用 DAO 层 Mapper 中的 deleteExperiment 根据指定的实验 ID 完成实验项目信息在数据库中的删除。

4.3.2.4.2 时序图



4.4 实验报告构件

4.4.1 构件图



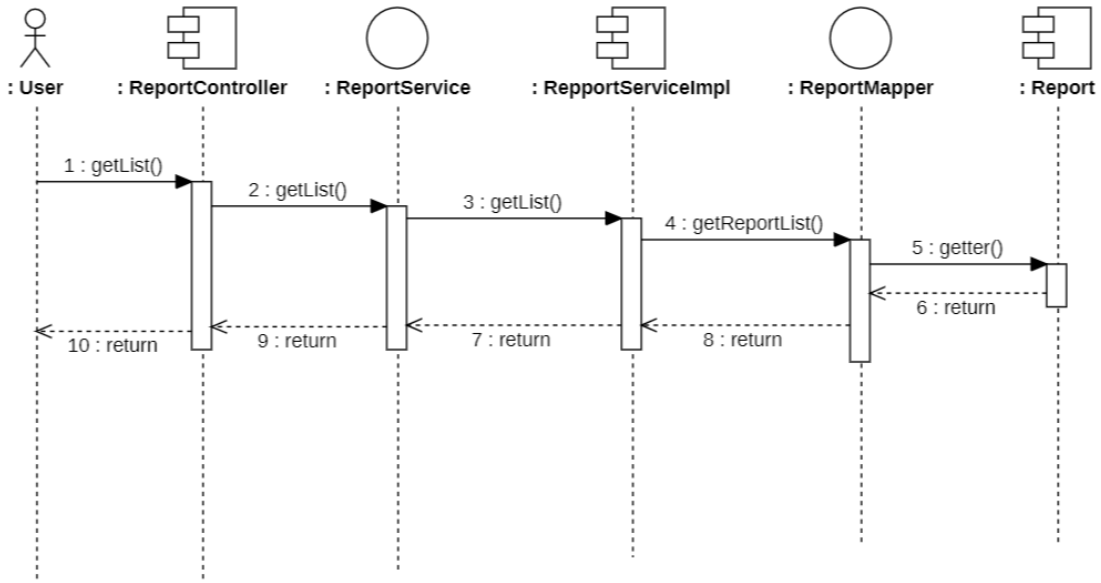
4.4.2 核心操作

4.4.2.1 getList

4.4.2.1.1 操作说明

该操作用于获取学生提交的实验报告列表，在 controller 层中触发该操作后，getList()方法会调用 service 层中的 getList()方法，getList()方法检查参数合法性调用 DAO 层中的 getReportList()方法访问数据库，将查询结果返回给上一层，在 controller 层中将结果封装为 json 返回给前端。

4.4.2.1.2 时序图

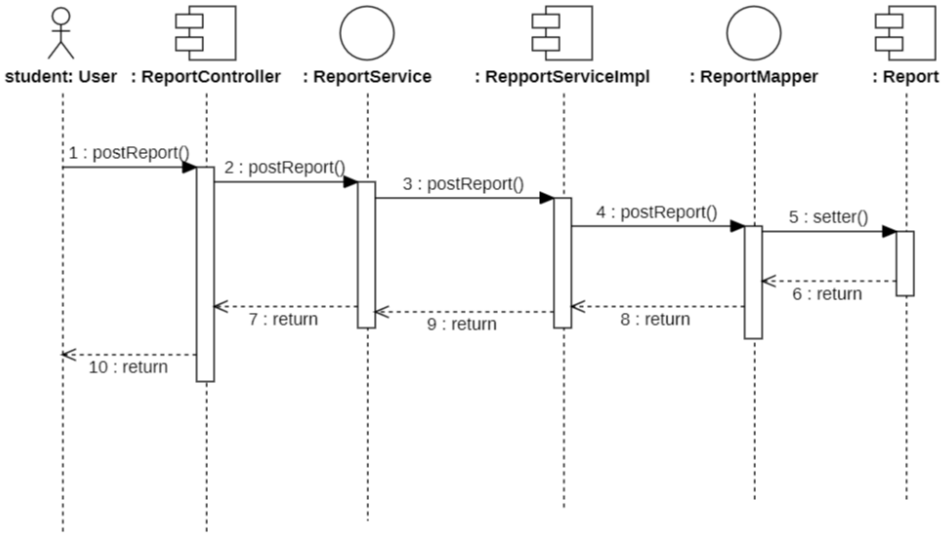


4.4.2.2 postReport

4.4.2.2.1 操作说明

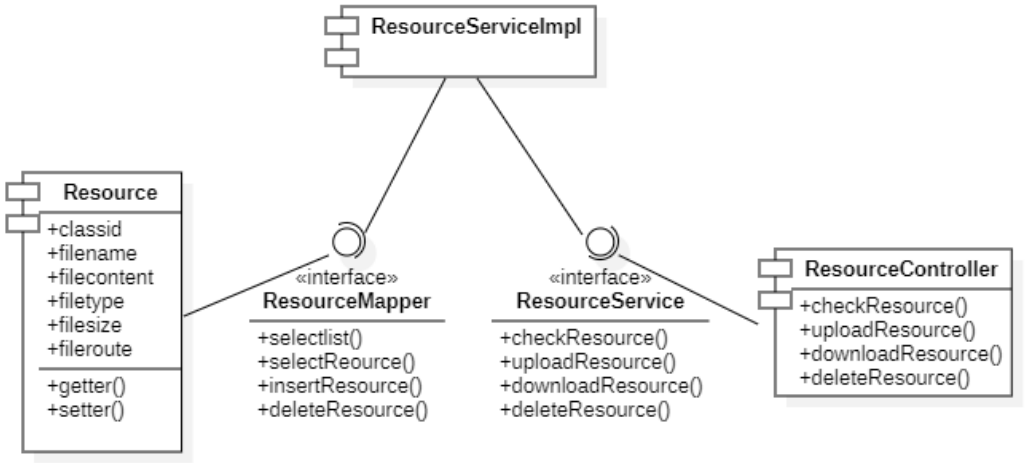
该操作用于学生提交实验报告，controller 层从前端请求方法中获取到 Report 实体，依次向下传递给 service 层的 postReport()，在 service 层中对字段进行非空检查再传递到 DAO 层中，DAO 层中的 postReport()方法访问数据库，对该 Report 进行持久化。

4.4.2.2.2 时序图



4.5 课程文件构件

4.5.1 构件图



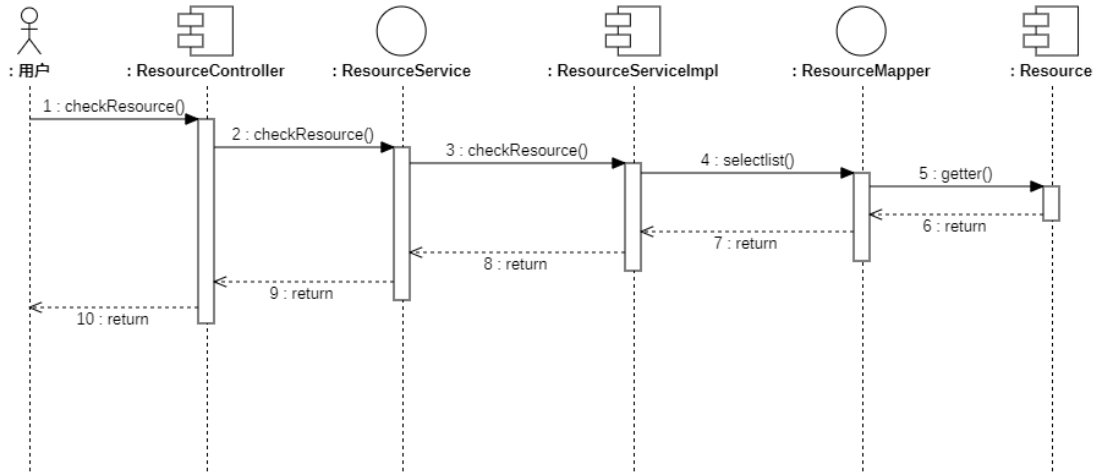
4.5.2 核心操作

4.5.2.1 checkResource

4.5.2.1.1 操作简述

该操作对应老师或学生在课程页面中查看课程包含的文件，调用 Controller 中对外的 checkResource 方法后，会调用业务层 Service 中的 checkResource 对传入的课程信息进行封装，接着调用 DAO 层 Mapper 中的 selectResource 完成对课程文件的查询，并返回文件的信息列表 json 文件。

4.5.2.1.2 时序图

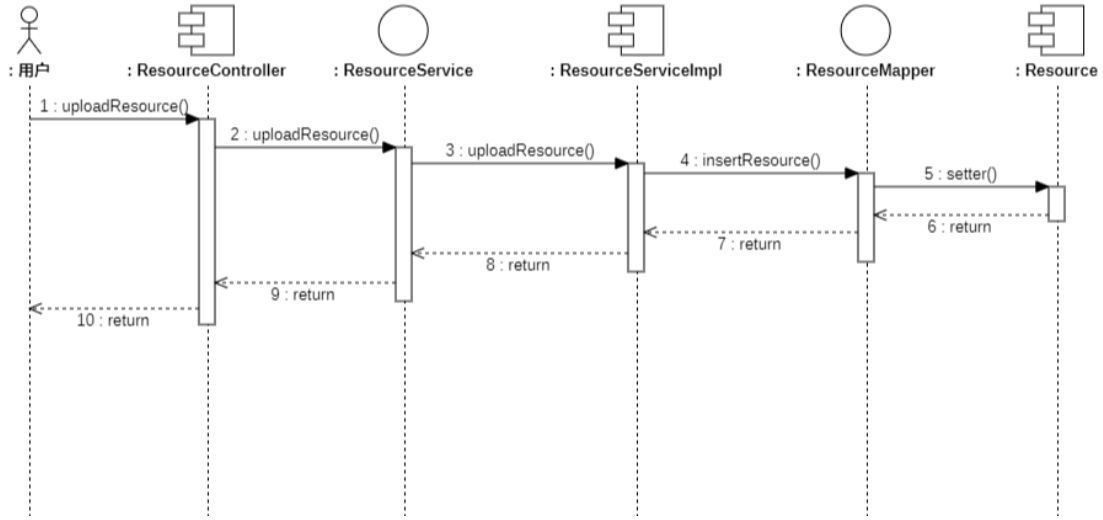


4.5.2.2 uploadResource

4.5.2.2.1 操作简述

该操作对应老师在课程文件页面中上传本地文件，调用 Controller 中对外的 uploadResource 方法后，会调用业务层 Service 中的 uploadResource 对传入的文件信息封装为课程文件对象，接着调用 DAO 层 Mapper 中的 insertResource 完成对课程文件的插入。

4.5.2.2.2 时序图

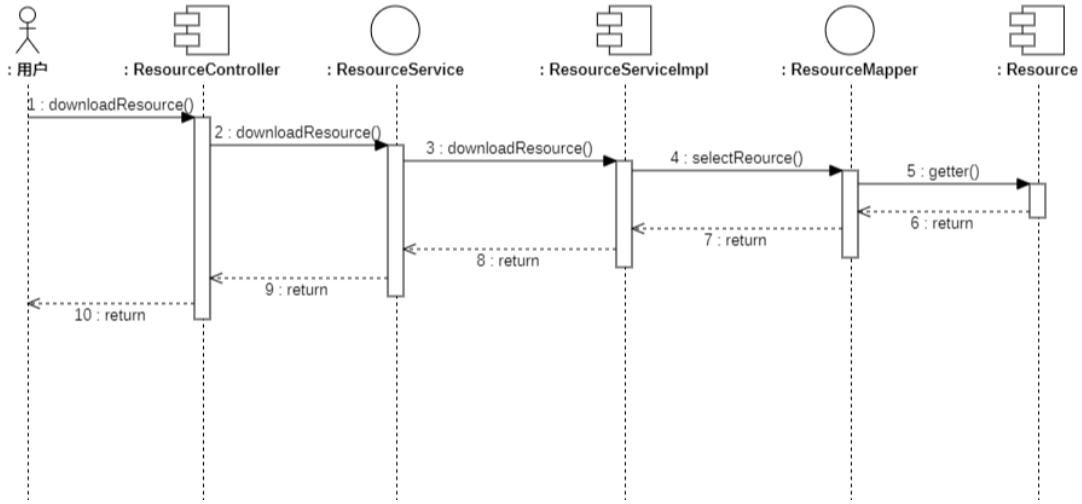


4.5.2.3 downloadResource

4.5.2.3.1 操作简述

该操作对应老师或学生在课程文件页面中上传本地文件，调用 Controller 中对外的 downloadResource 方法后，会调用业务层 Service 中的 downloadResource 对传入的文件信息封装为文件对象，接着调用 DAO 层 Mapper 中的 selectResource 完成对课程文件的查询，并返回对应的课程文件。

4.5.2.3.2 时序图



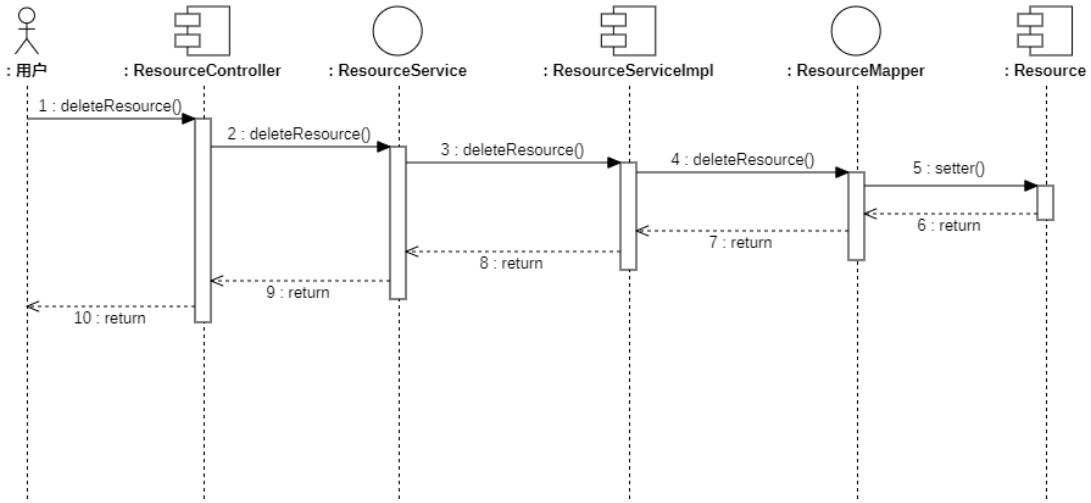
4.5.2.4 deleteResource

4.5.2.4.1 操作简述

该操作对应老师在课程文件页面中删除服务器上的文件，调用 Controller 中对外的 deleteResource 方法后，会调用业务层 Service 中的 deleteResource 对传入的文件信息封装为文件对象，接着调用 DAO 层 Mapper 中的 deleteResource 完成对课程

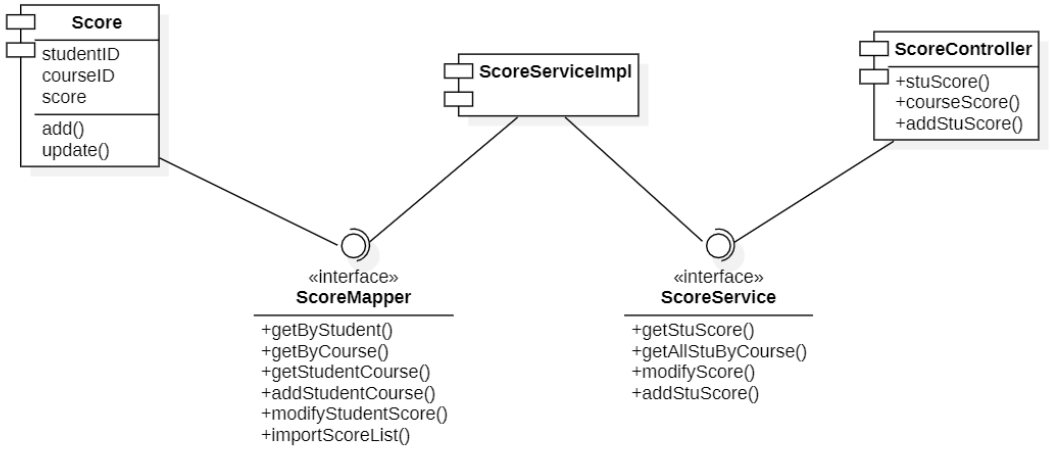
文件的删除。

4.5.2.4.2 时序图



4.6 学生成绩构件

4.6.1 构件图



4.6.2 核心操作

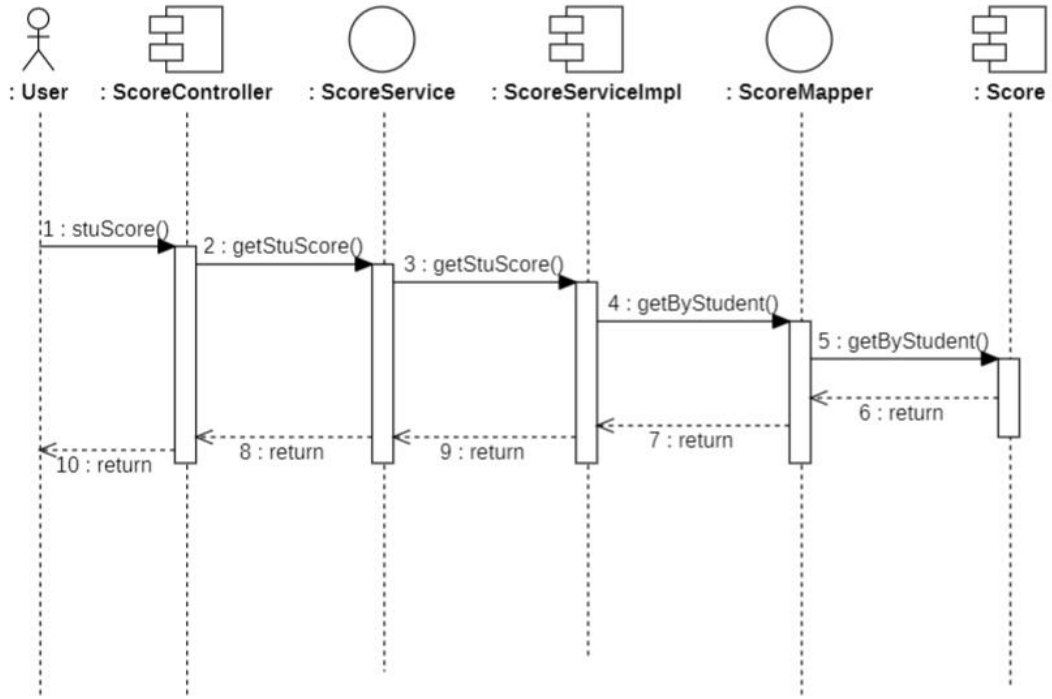
4.6.2.1 stuScore

4.6.2.1.1 操作简述

该操作用于获取学生各已修读课程的成绩，当前端请求该接口时，stuScore()方法会调用 service 层中的 getStuScore()方法，service 层中 getStuScore 方法去调用

DAO 层中 getByStudent()方法去访问数据库，获取学生成绩逐层返回到 stuScore()方法中，以 json 形式返回给前端。

4.6.2.1.2 时序图

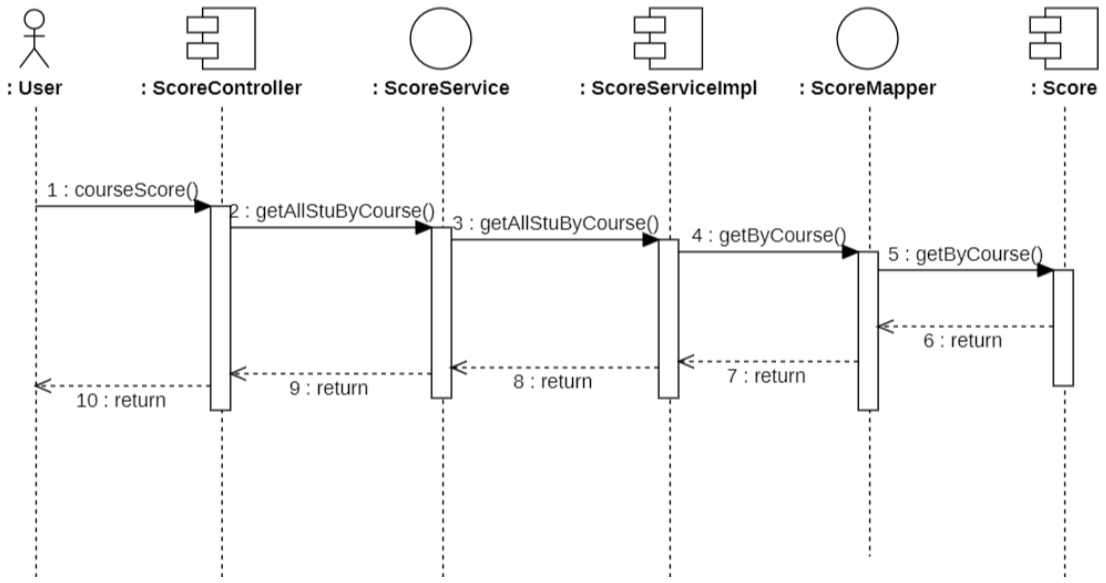


4.6.2.2 courseScore

4.6.2.2.1 操作描述

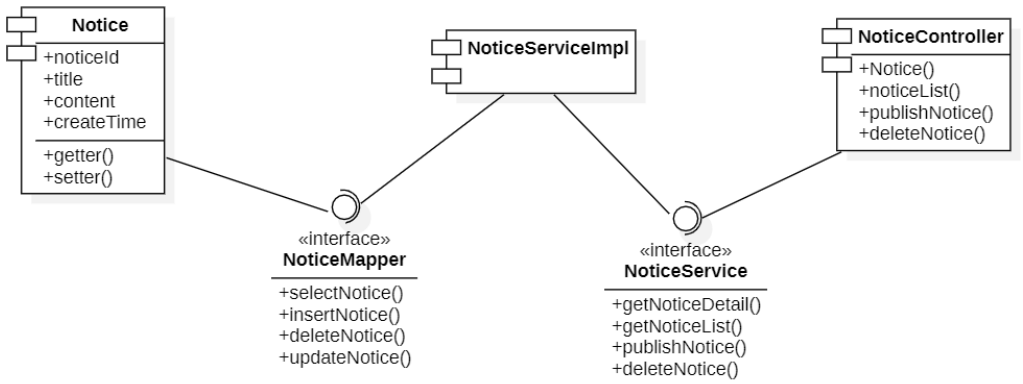
该操作是获取课程内所有学生的成绩，仅有相应的授课教师可调用，前端请求相应接口触发该操作后，controller 层中的 courseScore 方法，该方法继而调用 service 层中的 getAllStuByCourse()方法，该方法在检查参数之后会调用 DAO 层中的 getByCourse()方法访问数据库获取相应数据返回给上一层，在 getAllStuByCourse()方法中对结果封装为 Map，返回给 controller 层，再以 json 形式返回给前端。

4.6.2.2.2 时序图



4.7 通知公告构件

4.7.1 构件图



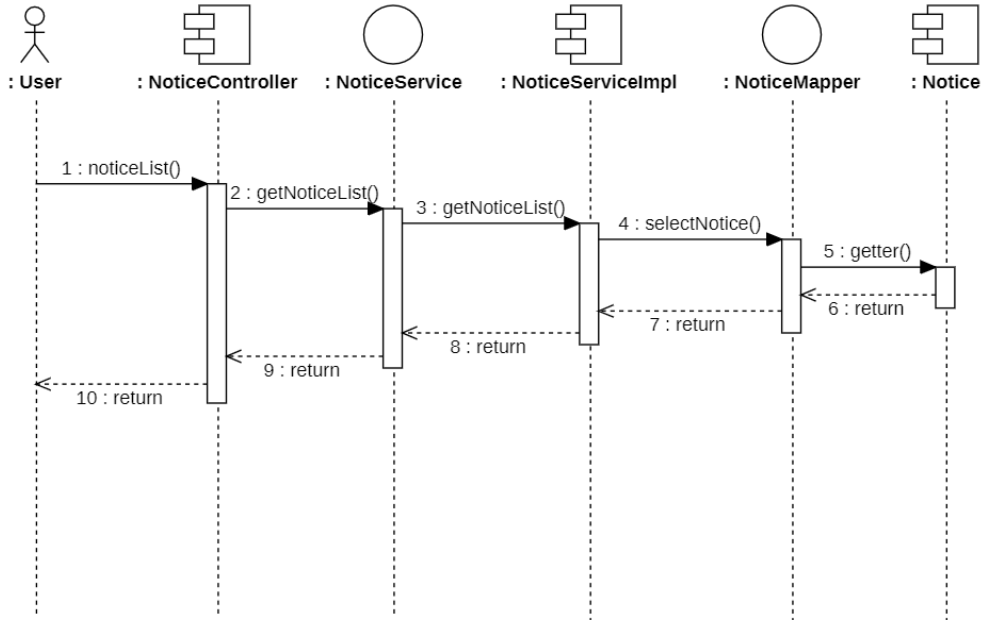
4.7.2 核心操作

4.7.2.1 noticeList

4.7.2.1.1 操作简述

该操作是获取当前课程中的所有通知公告，通过 parameter 可以实现分页展示，用户交互界面请求该 API 接口时，系统匹配该方法，该方法首先调用在 service 层中 getNoticeList()方法，getNoticeList()方法会首先检查所给参数，然后调用 DAO 层中 selectNotice()方法从数据库中查询结果，getNoticeLIst()方法对查询结果封装为 Map，在 Controller 层中由 noticeList()方法对结果封装为 json 返回给请求端。

4.7.2.1.2 时序图

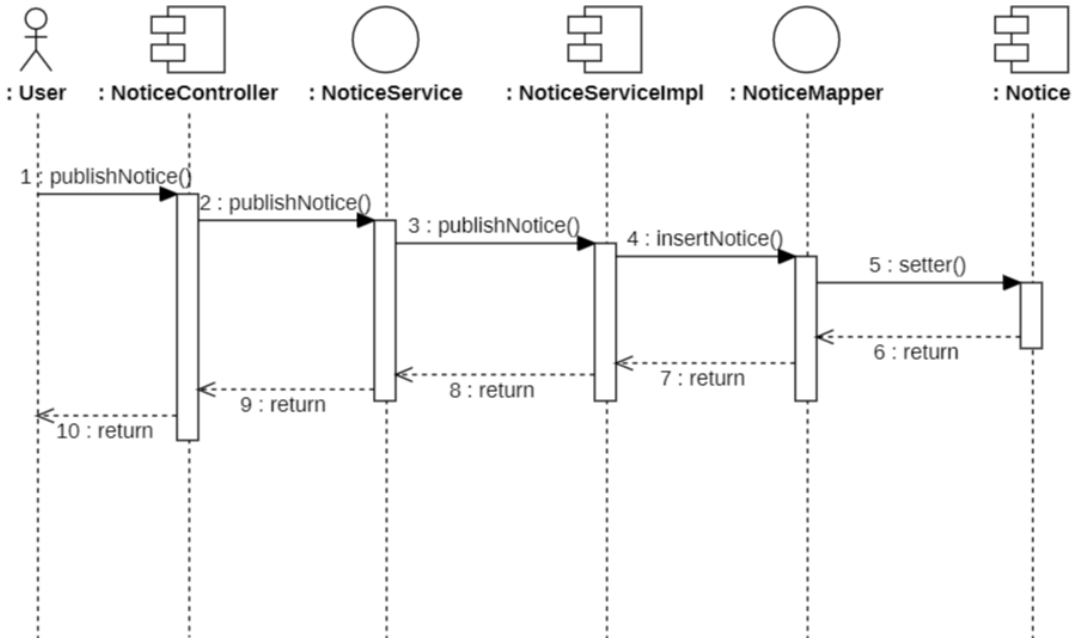


4.7.2.2 publishNotice

4.7.2.2.1 操作简述

该操作用于教师发布通知公告，前端请求相应 URL 接口，匹配到该操作后，controller 层中的 publishNotice()方法会接收到前端传送的 Notice 对象，传递给 service 层中的 publishNotice()方法，service 层中的该方法会对相应字段进行合法性检查，检查无误后，将其传递给 DAO 层中 insertNotice 方法()，该方法对 Notice 对象进行持久化。

4.7.2.2.2 时序图

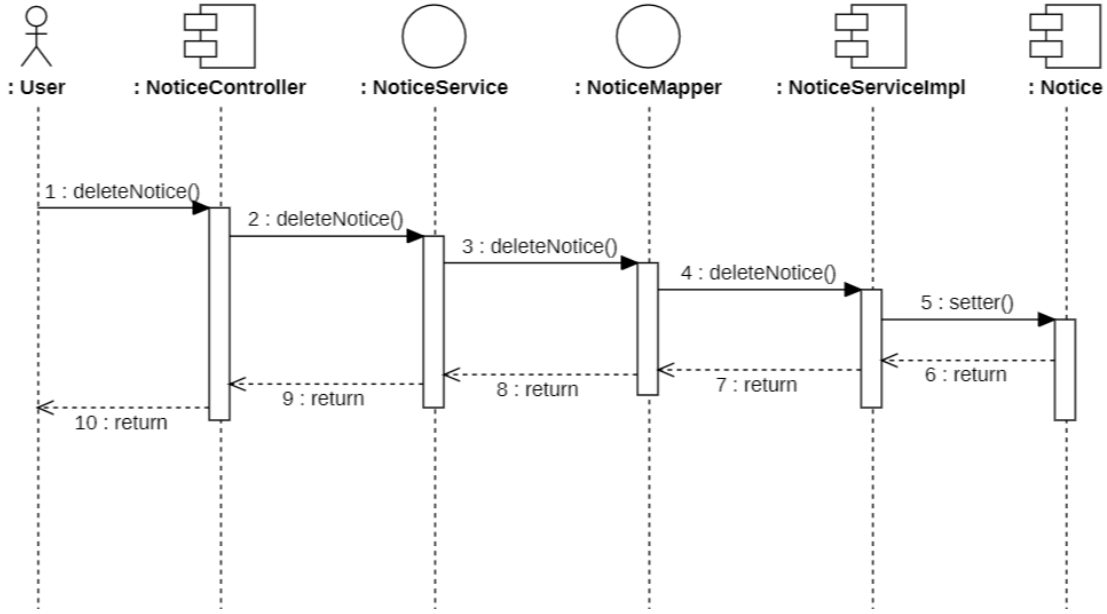


4.7.2.3 deleteNotice

4.7.2.3.1 操作简述

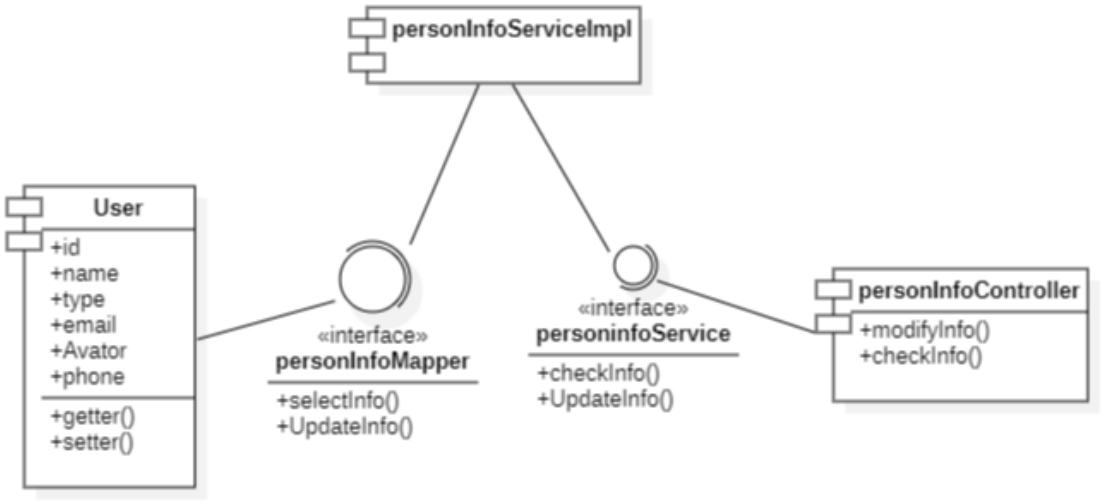
该操作用于删除已发布的通知公告，在 controller 层中触发该操作后，deleteNotice()方法会调用 service 层中的 deleteNotice()方法，在 service 层中完成基本的业务逻辑，然后调用 DAO 层中的 deleteNotice()方法对数据库中的数据进行删除。

4.7.2.3.2 时序图



4.8 个人信息构件

4.8.1 构件图



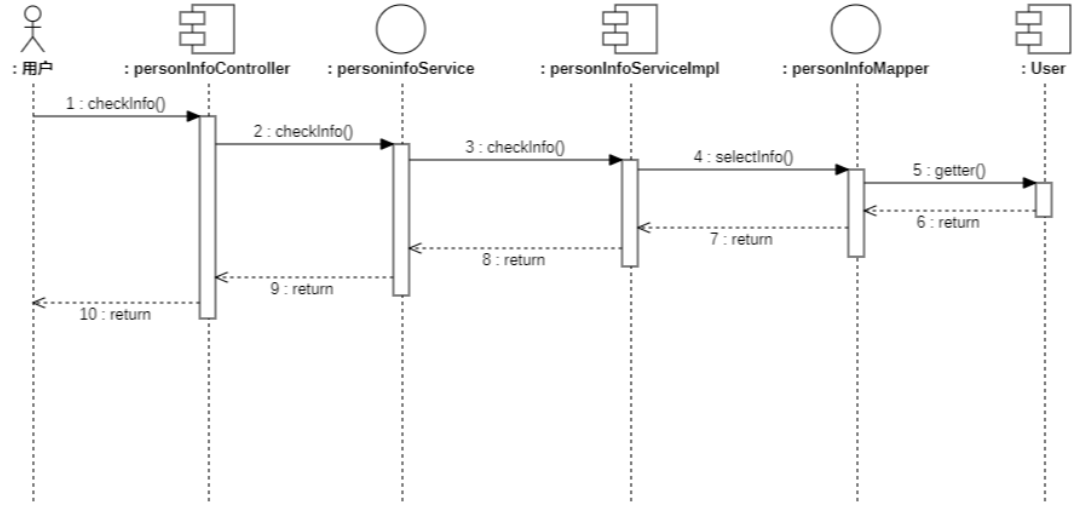
4.8.2 核心操作

4.8.2.1 checkInfo

4.8.2.1.1 操作简述

该操作对应用户对自己的个人信息进行查询，调用 Controller 中对外的 checkInfo 方法后，会调用业务层 Service 中的 checkInfo 对传入的用户信息封装为用户对象，接着调用 DAO 层 Mapper 中的 selectInfo 完成用户信息在数据库中的查询，并返回 json 格式的用户信息。

4.8.2.1.2 时序图

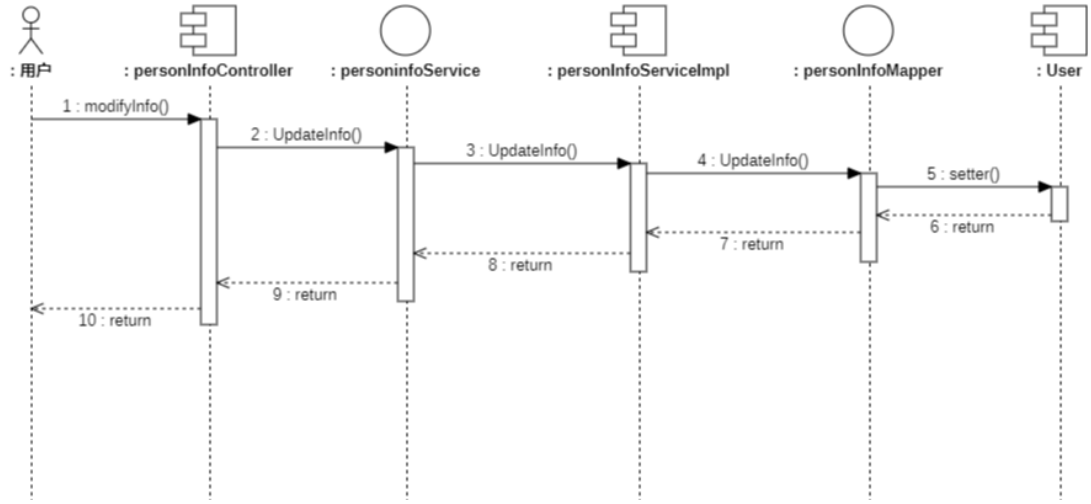


4.8.2.2 modifyInfo

4.8.2.2.1 操作简述

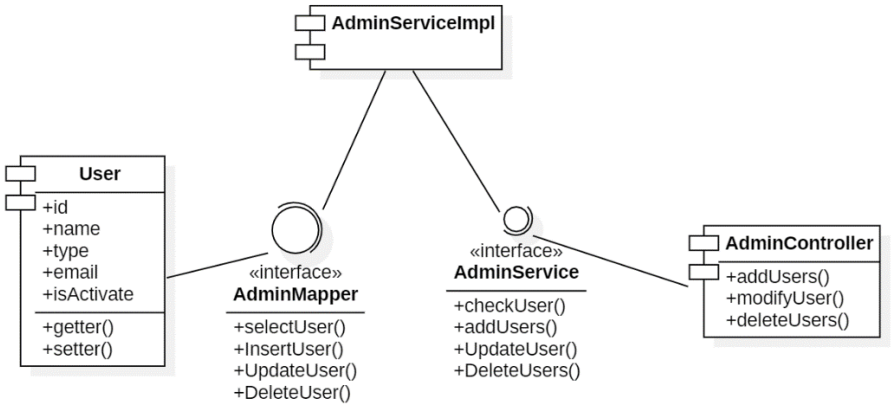
该操作对应用户对自己的个人信息进行修改，调用 Controller 中对外的 modifyInfo 方法后，会调用业务层 Service 中的 UpdateInfo 对传入的用户信息封装为用户对象，接着调用 DAO 层 Mapper 中的 UpdateInfo 完成用户信息在数据库中的查询，并返回 json 格式的用户信息。

4.8.2.2.2 时序图



4.9 管理员构件

4.9.1 构件图



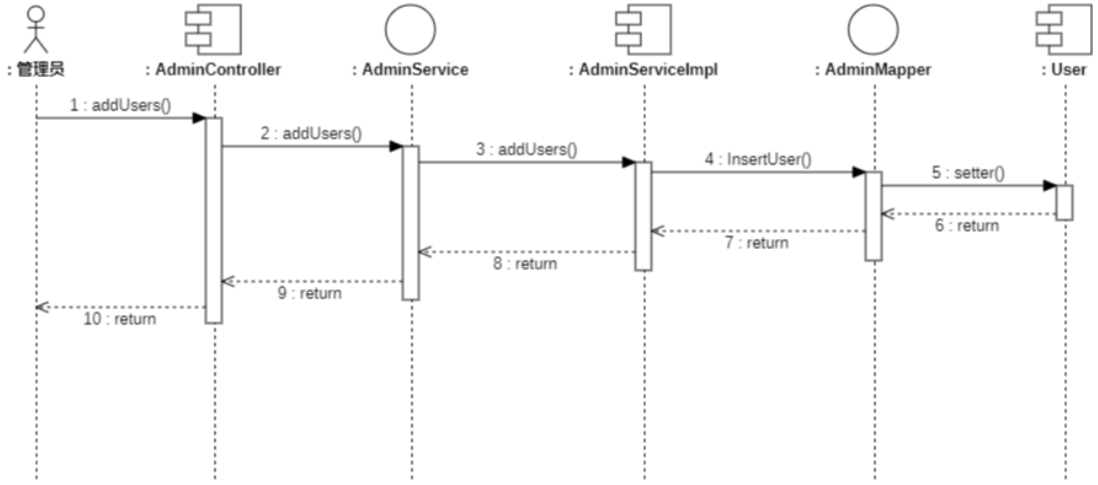
4.9.2 核心操作

4.9.2.1 addUsers

4.9.2.1.1 操作简述

该操作对应管理员添加新的系统用户（包括学生和老师），调用 Controller 中对外的 addUsers 方法后，会调用业务层 Service 中的 addUsers 对传入的用户信息表格封装为多个用户对象，接着调用 DAO 层 Mapper 中的 insertUser 完成用户信息在数据库中的插入。

4.9.2.1.2 时序图

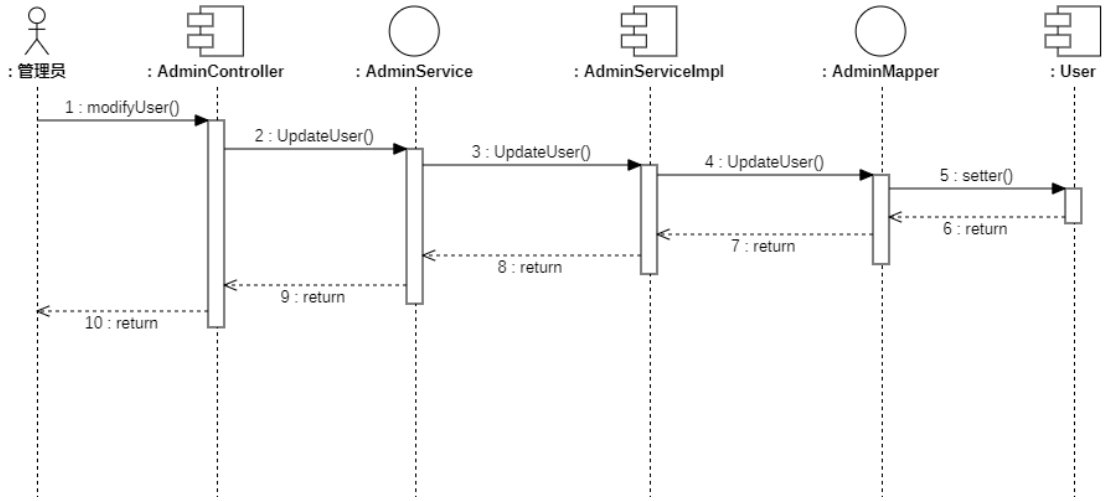


4.9.2.2 modifyUser

4.9.2.2.1 操作简述

该操作对应管理员修改系统用户信息，包括个人信息和账户状态，调用 Controller 中对外的 modifyUsers 方法后，会调用业务层 Service 中的 UpdateUser 对传入的用户信息封装为用户对象，接着调用 DAO 层 Mapper 中的 UpdateUser 完成用户信息在数据库中的更新。

4.9.2.2.2 时序图



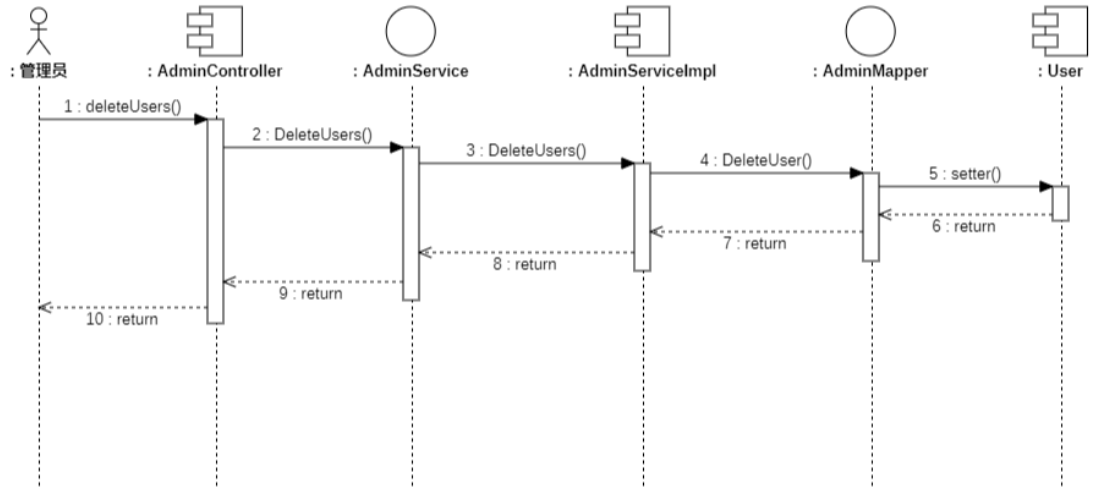
4.9.2.3 deleteUsers

4.9.2.3.1 操作简述

该操作对应管理员注销系统用户，调用 Controller 中对外的 deleteUsers 方法后，会调用业务层 Service 中的 DeleteUsers 对传入的用户信息表格文件封装为多个用户

对象，接着调用 DAO 层 Mapper 中的 DeleteUser 完成用户在数据库中的注销。

4.9.2.3.2 时序图



5 项目代码

前端仓库：[happy-someone-wang/EMS-front (github.com)]

后端仓库：[happy-someone-wang/EMS-back (github.com)]