数据库课程设计报告

软件学院

软件工程专业

2021 级 5 班

姓名：马贵亮

学号：202122202214

实验老师：钱进

目录

数据库课程设计实验报告

教室调度系统是一种用于有效管理和分配学校或教育机构内各个教室资源的软件系统。其主要目的是帮助学校或教育机构实现高效、合理地安排教室的使用，以满足不同教学和活动需求。

1 系统概述

教室调度系统是一种用于有效管理和分配学校或教育机构内各个教室资源的软件系统。其主要目的是帮助学校或教育机构实现高效、合理地安排教室的使用，以满足不同教学和活动需求。

1.1 目的与目标

- **优化资源利用**: 确保教室资源得到最大化利用，避免教室空闲时间过长或者出现资源浪费的情况。
- **简化调度流程**: 自动化教室调度流程，减少手动干预，提高效率。
- **满足多样化需求**: 支持不同类型的教学活动、会议、活动等的安排，考虑到各种资源的特殊要求。
- **提升教学效果**: 保证教室的使用符合教学计划，确保教学活动顺利进行。

1.2 主要功能模块

1.2.1 教室管理

- **教室信息管理**: 记录教室的基本信息，如名称、容纳人数、设备设施等。
- **教室状态监控**: 实时查看教室的占用情况，了解哪些教室正在使用，哪些是空闲的。

1.2.2 调度管理

- **教室预约**: 学校成员可以根据需求提前预约教室，包括时间、用途等信息。
- **冲突检测**: 系统能够自动检测和解决教室使用时间的冲突，防止重复预约或者时间冲突的情况发生。
- **调整与取消**: 允许用户在需要时修改或取消已经预约的教室。

1.2.3 用户权限管理

- **角色分配**: 区分不同用户的角色，如管理员、教师、学生等，不同角色拥有不同的系统权限。
- **权限控制**: 确保只有具有相应权限的用户才能进行调度和管理操作。

2 需求分析

2.1 功能性需求

教室信息管理：

- 管理教室基本信息，包括编号、容量、位置、设备设施等。
- 提供对教室信息的增加、修改和删除功能。

课程信息管理：

- 管理课程的基本信息，包括课程名称、课程编号、授课教师、课程容量等。
- 提供对课程信息的增加、修改和删除功能。

教师信息管理：

- 管理教师的基本信息，包括姓名、工号、联系方式等。
- 提供对教师信息的增加、修改和删除功能。

学生信息管理：

- 管理学生的基本信息，包括姓名、学号、联系方式等。
- 提供对学生信息的增加、修改和删除功能。

课程安排管理：

- 可以将课程分配到特定的教室、时间段和教师。
- 考虑到可能会出现重复课程、时间冲突等情况，需要提供冲突检测和解决机制。

教室预约管理：

- 学生或教师可以预约特定的教室进行活动或会议。
- 系统需要提供对预约的审核、修改和取消功能。

教室使用统计：

- 可以生成教室利用率、繁忙时段等统计报表，以便于学校管理者进行资源优化和规划。

通知和提醒：

- 向相关人员发送课程安排、教室预约等通知。

- 提供提前提醒功能，防止遗漏重要事项。

2.2 系统非功能性需求

性能（Performance）：

- 系统响应时间应该快速，以保证用户在输入信息后能够迅速得到结果。
- 能够支持同时多用户使用，保证系统的稳定性和效率。

可靠性（Reliability）：

- 系统应该具备高可靠性，避免因为故障或者错误导致系统崩溃或数据丢失。

安全性（Security）：

- 系统应该具备足够的安全保障，以防止未经授权的用户访问、篡改或者破坏系统数据。
- 用户的个人信息和敏感信息应当得到保护。

可用性（Availability）：

- 系统应该保证 24/7 的稳定运行，保证用户可以随时随地访问。

易用性（Usability）：

- 界面设计应该简单直观，使得用户能够容易地进行操作，不需要过多的培训。
- 提供必要的帮助信息和反馈，以便用户能够正确地使用系统功能。

3 系统设计

3.1 系统开发平台综述

操作系统： Windows 11

开发工具： Intellij IDEA Ultimate + Navicat

前端开发： Vue2

后端开发： Java+SpringBoot+Mybatis

数据库： mysql5.8

3.2 前后端简述

3.2.1 前端简述

在前端开发中我才用了 Vue2 进行前端开发，Vue2 将应用程序拆分成小的、独立的组件，使得代码更加模块化和可复用；Vue 使用了双向数据绑定，当数据发生变化时，视图会自动更新，无需手动操作 DOM；Vue 提供了一系列的指令（如 v-if、v-for 等），简化了对 DOM 元素的操作和控制；Vue 提供了 Vue Router，用于实现单页应用（SPA）的前端路由管理。并且相比于 Vue3 而言，Vue2 对组件以及图标的使用更加简单方便。

3.2.2 后端简述

后端的开发语言选择了 Java 语言，使用 SpringBoot，在数据库管理上采用 Mybatis（未采用 Mybatis plus，加强对 sql 语言的熟悉）。采用 SpringBoot 简化了开发部署运行的过程，采用 Mybatis 简化了一些对于数据库的操作。程序设计上设置 Controller 层、Service 层、Mapper 层。

Controller 层：

- **接收和处理用户请求：**Controller 层负责接收来自客户端（浏览器）的请求，并根据请求的内容进行相应的处理。
- **解耦视图与业务逻辑：**Controller 层将用户请求与业务逻辑分开，使得视图层（前端）可以独立变化而不影响业务逻辑。
- **统一请求处理：**可以在 Controller 层进行请求拦截、权限验证等统一处理。

Service 层：

- **业务逻辑处理：**Service 层负责处理业务逻辑，对业务需求进行具体的实现。
- **事务管理：**Service 层通常是事务的发起者，负责控制事务的边界，保证业务的一致性。
- **代码复用：**将业务逻辑封装在 Service 层，可以被多个 Controller 层共享，提高了代码的复用性。
- **单元测试：**Service 层的业务逻辑可以进行单元测试，保证业务功能的稳定性。

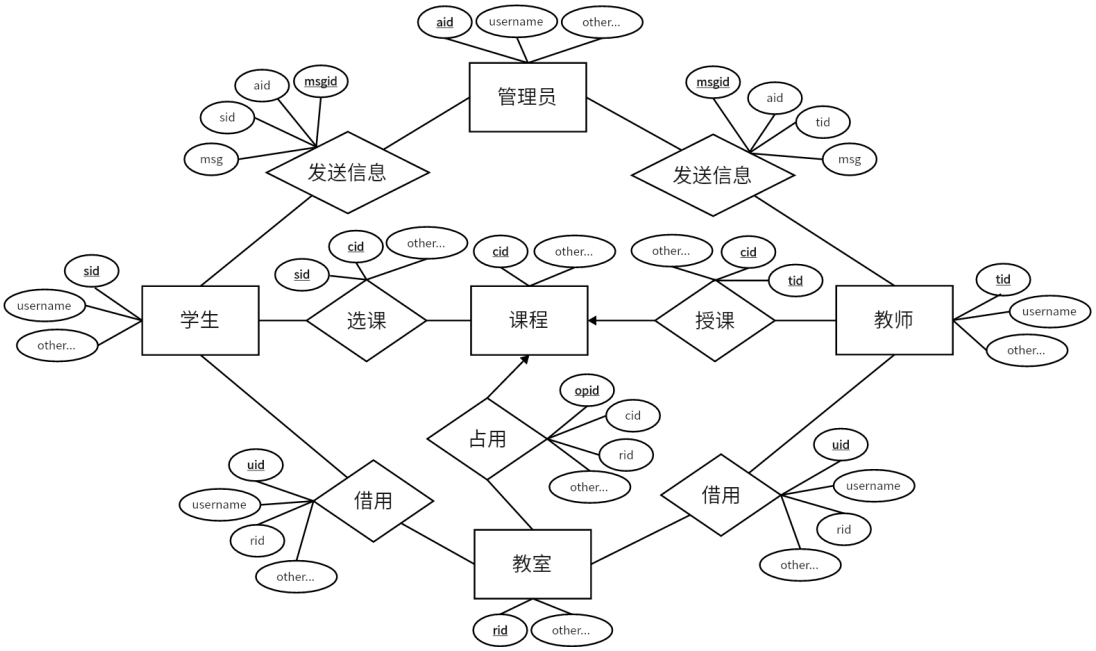
Mapper 层：

- **数据库操作：**Mapper 层负责与数据库进行交互，包括查询、插入、更新、删除等操作。
- **SQL 优化：**可以在 Mapper 层对 SQL 进行优化，提高数据库操作的效率。

- **与数据库解耦**：Mapper 层将数据库操作与业务逻辑分离，使得业务逻辑不依赖于特定的数据库实现，提高了系统的灵活性。
- **数据持久化**：负责将数据持久化到数据库中，保证数据的可靠性。

3.3 数据库逻辑设计

3.3.1 ER 模型



3.3.2 数据字典

学生实体：

属性	描述	键	类型及长度	是否空	是否多值	约束
sid	唯一表示学生	主键	varchar(20)	F	F	数字表示
name	学生姓名		varchar(255)	F	F	
gender	学生性别		varchar(255)	T	F	
username	学生对应的用户名		varchar(255)	F	F	外键约束

教师实体

属性	描述	键	类型及长度	是否空	是否多值	约束

属性	描述	键	类型及长度	是否空	是否多值	约束
tid	唯一表示教师	主键	varchar(20)	F	F	数字表示
name	学生姓名		varchar(255)	F	F	
gender	学生性别		varchar(255)	T	F	
username	学生对应的用户名		varchar(255)	F	F	外键约束

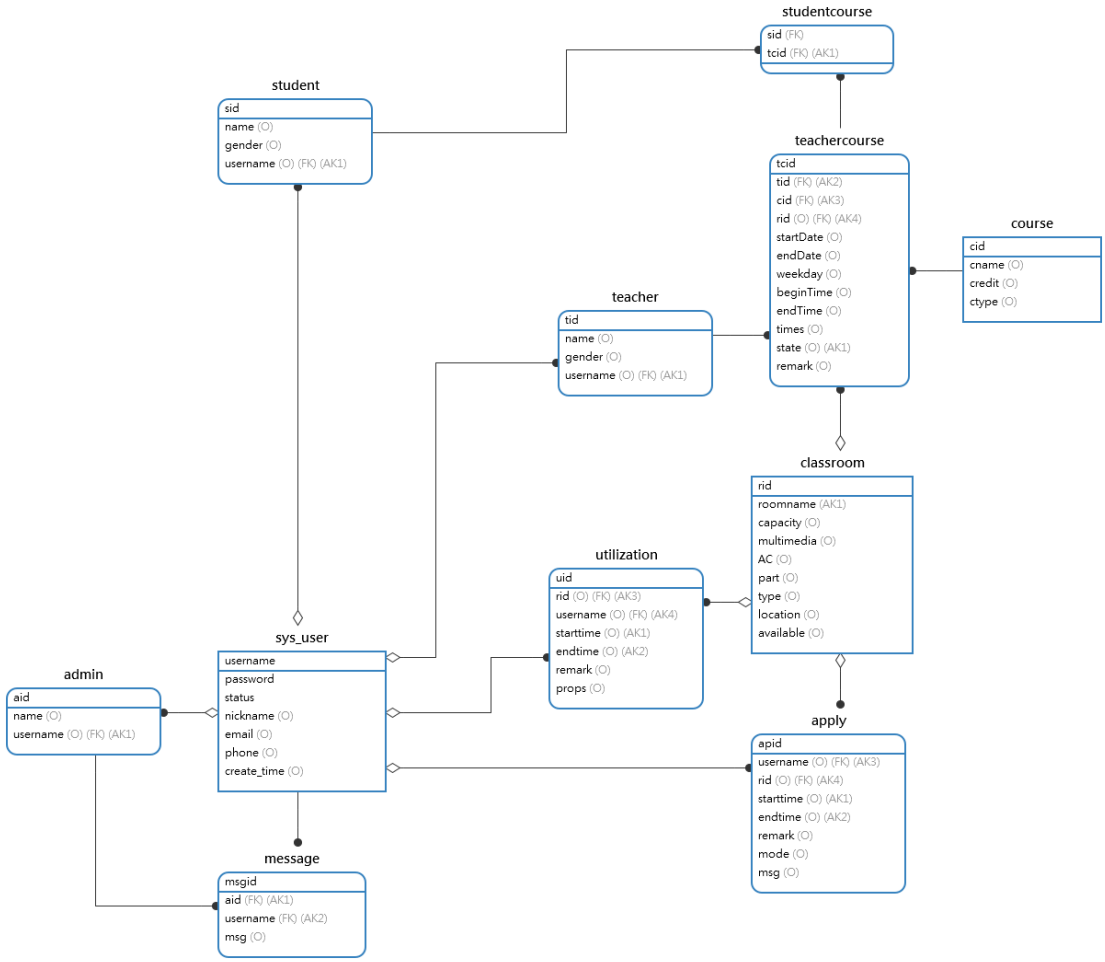
教室实体

属性	描述	键	类型及长度	是否空	是否多值	约束
rid	唯一表示教室	主键	varchar(20)	F	F	数字表示
roomname	教室名称		varchar(255)	F	F	
capacity	教室最大容量		int(11)	F	F	正整数
multimedia	是否有多媒体		bit(1)	F	F	1/0
ac	是否有空调		bit(1)	F	F	1/0
part	分区		varchar(255)	T	F	
type	教室类型		varchar(255)	F	F	阶梯/录播/普通
location	教室位置		varchar(255)	T	F	
available	教室是否可用		bit(1)	F	F	1/0

课程实体

属性	描述	键	类型及长度	是否空	是否多值	约束
cid	唯一表示课程	主键	varchar(255)	F	F	数字表示
cname	课程名称		varchar(255)	F	F	
credit	学分		double(10,1)	F	F	
ctype	课程类型		varchar(255)	F	F	必修/选修/限选

3.3.3 关系表



3.3.4 创建表

```
-- 创建 admin 表
CREATE TABLE `admin` (
  `aid` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `username` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`aid`),
  KEY `username` (`username`),
  CONSTRAINT `admin_ibfk_1` FOREIGN KEY (`username`) REFERENCES `sys_us
er` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb
4_unicode_ci;

-- 创建 apply 表
CREATE TABLE `apply` (
  `apid` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `rid` int(11) DEFAULT NULL,
```

```
  `starttime` datetime DEFAULT NULL,
  `endtime` datetime DEFAULT NULL,
  `remark` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `mode` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `msg` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`apid`),
  KEY `starttime` (`starttime`) USING BTREE,
  KEY `endtime` (`endtime`) USING BTREE,
  KEY `username` (`username`),
  KEY `rid` (`rid`),
  CONSTRAINT `apply_ibfk_1` FOREIGN KEY (`username`) REFERENCES `sys_us
er` (`username`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `apply_ibfk_2` FOREIGN KEY (`rid`) REFERENCES `classroom`
(`rid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=33 DEFAULT CHARSET=utf8mb4 COLLATE=utf8m
b4_unicode_ci;

-- 创建 asign 表
CREATE TABLE `asign` (
  `asgid` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `remark` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `capacity` int(11) DEFAULT NULL,
  `multiMedia` bit(1) DEFAULT NULL,
  `ac` bit(1) DEFAULT NULL,
  `type` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `starttime` datetime DEFAULT NULL,
  `endtime` datetime DEFAULT NULL,
  `state` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `lock` bit(1) DEFAULT b'0',
  PRIMARY KEY (`asgid`),
  KEY `starttime` (`starttime`) USING BTREE,
  KEY `endtime` (`endtime`) USING BTREE,
  KEY `username` (`username`),
  CONSTRAINT `asign_ibfk_1` FOREIGN KEY (`username`) REFERENCES `sys_us
er` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=23 DEFAULT CHARSET=utf8mb4 COLLATE=utf8m
b4_unicode_ci;

-- 创建 classroom 表
CREATE TABLE `classroom` (
  `rid` int(255) NOT NULL AUTO_INCREMENT COMMENT '教室号',
  `roomname` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT '
教室名称',
  `capacity` int(11) DEFAULT NULL COMMENT '教室最大容量',
  `multimedia` bit(1) DEFAULT NULL COMMENT '是否有多媒体(1/0)',
  `AC` bit(1) DEFAULT NULL COMMENT '是否有空调(1/0)',
  `part` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT '
分区',
```

```
  `type` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT '
教室类型(普通/阶梯/录播)',
  `location` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMME
NT '教室位置',
  `available` bit(1) DEFAULT NULL COMMENT '教室是否可用(1/0)',
  PRIMARY KEY (`rid`),
  UNIQUE KEY `roomname` (`roomname`) USING BTREE COMMENT '教室名'
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4 COLLATE=utf8m
b4_unicode_ci;

-- 创建 course 表
CREATE TABLE `course` (
  `cid` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `cname` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `credit` double(10,1) DEFAULT NULL,
  `ctype` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`cid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- 创建 message 表
CREATE TABLE `message` (
  `msgid` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `fromuser` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `touser` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `msg` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `sendtime` datetime DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`msgid`),
  KEY `message_ibfk_1` (`fromuser`),
  KEY `message_ibfk_2` (`touser`)
) ENGINE=InnoDB AUTO_INCREMENT=40 DEFAULT CHARSET=utf8mb4 COLLATE=utf8m
b4_unicode_ci;

-- 创建 sc 表
CREATE TABLE `sc` (
  `sid` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `cid` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  PRIMARY KEY (`sid`,`cid`) USING BTREE,
  KEY `cid` (`cid`),
  CONSTRAINT `sc_ibfk_1` FOREIGN KEY (`sid`) REFERENCES `student` (`sid
`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `sc_ibfk_2` FOREIGN KEY (`cid`) REFERENCES `course` (`cid`)
 ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- 创建 student 表
CREATE TABLE `student` (
  `sid` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT '学号',
  `name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT '
姓名',
```

```
  `gender` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT
 '性别',
  `username` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMME
NT '账户',
  PRIMARY KEY (`sid`),
  KEY `stu2user` (`username`),
  CONSTRAINT `stu2user` FOREIGN KEY (`username`) REFERENCES `sys_user`
(`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- 创建用户表
CREATE TABLE `sys_user` (
  `username` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `status` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `nickname` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `email` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `phone` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `create_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`username`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- 创建 tc 表
CREATE TABLE `tc` (
  `tid` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `cid` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  PRIMARY KEY (`tid`,`cid`),
  UNIQUE KEY `cid` (`cid`) USING BTREE,
  CONSTRAINT `tc_ibfk_1` FOREIGN KEY (`tid`) REFERENCES `teacher` (`tid
`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `tc_ibfk_2` FOREIGN KEY (`cid`) REFERENCES `course` (`cid`)
 ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- 创建 teacher 表
CREATE TABLE `teacher` (
  `tid` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT '教工
号',
  `name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `gender` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `username` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`tid`),
  KEY `teacher2user` (`username`),
  CONSTRAINT `teacher2user` FOREIGN KEY (`username`) REFERENCES `sys_us
er` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--创建 teachercourse 表
CREATE TABLE `teachercourse` (
```

```
  `tcid` int(11) unsigned NOT NULL AUTO_INCREMENT COMMENT '教师开课编号',
  `username` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT '
申请人',
  `cid` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT '课程
编号',
  `rid` int(11) DEFAULT NULL COMMENT '教室编号',
  `startDate` date DEFAULT NULL COMMENT '课程开课日期',
  `endDate` date DEFAULT NULL COMMENT '课程结课日期',
  `weekday` int(11) DEFAULT NULL,
  `beginTime` time DEFAULT NULL COMMENT '上课时间',
  `endTime` time DEFAULT NULL COMMENT '下课时间',
  `times` int(11) DEFAULT NULL COMMENT '频率',
  `state` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT
'当前状态',
  `remark` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT
 '备注',
  PRIMARY KEY (`tcid`),
  KEY `state` (`state`) USING BTREE,
  KEY `rid` (`rid`),
  KEY `username` (`username`),
  KEY `cid` (`cid`) USING BTREE,
  CONSTRAINT `teachercourse_ibfk_2` FOREIGN KEY (`cid`) REFERENCES `cou
rse` (`cid`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `teachercourse_ibfk_3` FOREIGN KEY (`rid`) REFERENCES `cla
ssroom` (`rid`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `teachercourse_ibfk_4` FOREIGN KEY (`username`) REFERENCES
 `sys_user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=23 DEFAULT CHARSET=utf8mb4 COLLATE=utf8m
b4_unicode_ci;

-- 创建 utilization 表
CREATE TABLE `utilization` (
  `uid` int(11) NOT NULL AUTO_INCREMENT,
  `rid` int(11) DEFAULT NULL,
  `username` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `starttime` datetime DEFAULT NULL,
  `endtime` datetime DEFAULT NULL,
  `remark` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `props` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT
'属性',
  `asgid` int(11) DEFAULT '0',
  PRIMARY KEY (`uid`),
  KEY `starttime` (`starttime`) USING BTREE,
  KEY `endtime` (`endtime`) USING BTREE,
  KEY `rid` (`rid`),
  KEY `username` (`username`),
  CONSTRAINT `utilization_ibfk_1` FOREIGN KEY (`rid`) REFERENCES `class
room` (`rid`) ON DELETE CASCADE ON UPDATE CASCADE,
```

```
  CONSTRAINT `utilization_ibfk_2` FOREIGN KEY (`username`) REFERENCES `
sys_user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=126 DEFAULT CHARSET=utf8mb4 COLLATE=utf8
mb4_unicode_ci;
```

3.4 数据库物理设计

3.4.1 索引 与 外键

以下具体的构建与实现都在上述构建表的 mysql 代码中体现。

外键方面：

username 外键：对于 admin、apply、asign、student、teacher、utilization、teachercourse 表中设立 username 外键，来保证数据的完善性。

sid 外键： 对于 sc 表中设立 sid 外键

tid 外键： 对于 tc 表中设立 tid 外键

cid 外键： 对于 sc、tc 中设立 cid 外键

rid 外键： 对于 apply、utilization、teachercourse 表中设立 rid 外键

除去以上外键之外，在如下位置设立索引，来提高查询速率：

apply 表： 设立 starttime、endtime 两个 BTREE 索引来提高表中对于时间的查询速度

asign 表： 设立 starttime、endtime 两个 BTREE 索引来提高表中对于时间的查询速度

classroom 表： 设立 roomname 这个 UNIQUE 的 BTREE 索引，来保证 roomname 无重名，且便于查询

message 表： 设立 fromuser、touser 两个 BTREE 索引，提高对于信息发送与收取的信息查询速度

teachercourse 表： 设立 state 一个索引，提高查询速度

apply 表： 设立 starttime、endtime 两个 BTREE 索引来提高表中对于时间的查询速度

3.4.2 安全机制

数据安全：

- 程序启动需要经过登陆认证，登陆 ID 和密码正确才能启动程序。

- 程序内部的任何会引起数据库改动的操作（增删改）均经过了前端和后端两次严密的审查判定，以确保数据库的准确性和一致性。

系统安全：

- 登陆需经过身份认证，即身份口令密码一致才可登录。
- 不同的用户有不同的权限，用户通过返回的 token 明确权限，用户只能使用分配给他的权限，无法越权操作。

4 功能设计

4.1 基础用户登录设计

4.1.1 基础设计

该系统对于每一个用户设置一个账号，每一个学生/教室/管理员绑定一个账号。

账号设置不可重复的账号，并设立一个密码采用 MD5 加密后存储在数据库中。

每一个账号绑定一个身份（包含学生、教师、管理员），通过注册只可以注册临时教师和临时学生账号，不可注册管理员账号。

管理员通过用户管理可以将某个账号的权限修改。（详见后文用户管理）

管理员在用户管理中新增学生与教师的过程中根据其学号/教工号，自动注册一个账号。（详见后文用户管理）

```
// MD5 加密
public String passwordMD5(String userName, String userPassword) {
    // 需要加密的字符串
    String src = userName + userPassword;
    try {
        // 加密对象，指定加密方式
        MessageDigest md5 = MessageDigest.getInstance("md5");
        // 准备要加密的数据
        byte[] b = src.getBytes();
        // 加密：MD5 加密一种被广泛使用的密码散列函数，
        // 可以产生出一个 128 位（16 字节）的散列值（hash value），用于确保信
息传输完整一致
        byte[] digest = md5.digest(b);
        // 十六进制的字符
        char[] chars = new char[]{'0', '1', '2', '3', '4', '5',
                                  '6', '7', 'A', 'B', 'C', 'd', 'o', '*
', '#', '/'};
        StringBuffer sb = new StringBuffer();
        // 处理成十六进制的字符串(通常)
        // 遍历加密后的密码，将每个元素向右位移 4 位，然后与 15 进行与运算(byt
```

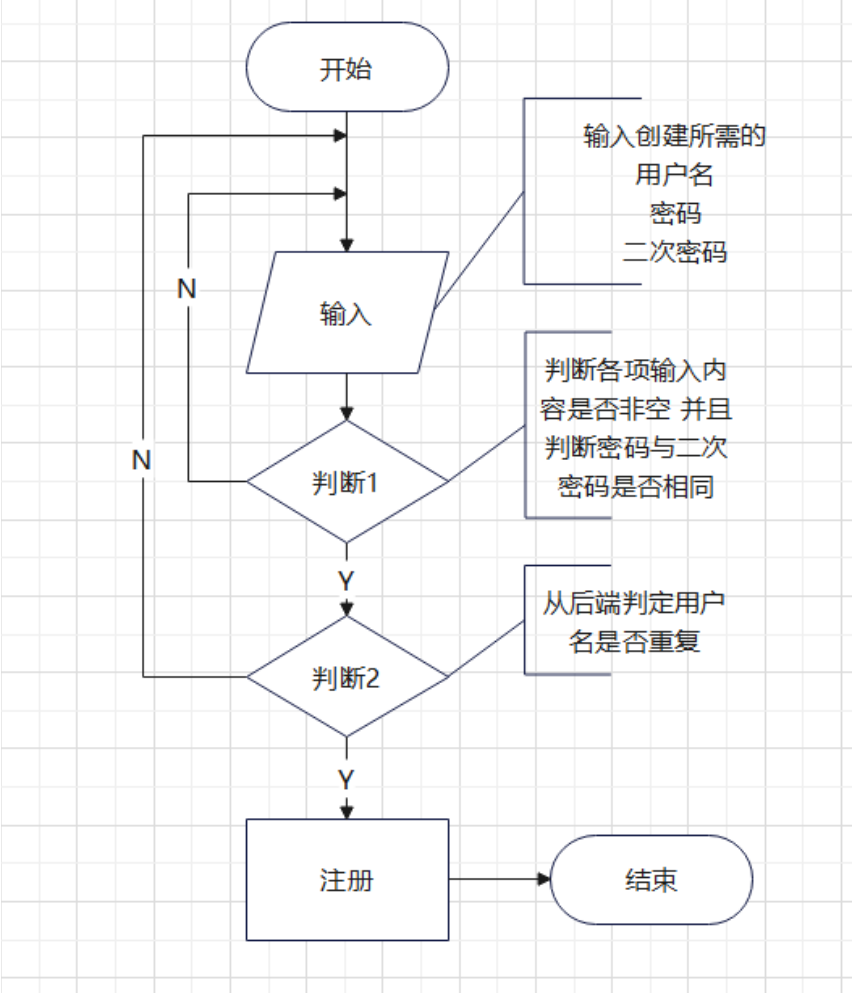
```
e 变成数字)
        for (byte bb : digest)
        {
            sb.append(chars[(bb >> 4) & 15]);
            sb.append(chars[bb & 15]);
        }
        return sb.toString();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return null;
}
```

4.1.2 用户注册

与传统的用户注册相似，通过输入账号与密码与二次密码确认，进行账号注册。在前端与后端进行两次对于账号和密码非空的判断来保证保存到数据库中的数据为合法的数据。

而对于学生与教师在创建过程中生成的账号，整体代码逻辑与注册过程一致，账号采用教工号/学号，密码采用默认 123456。

流程图如下：



后端部分代码：

```
// from com/mgl/server/controller/UserController.java
public Result register(UserDTO userDTO)
{
    String username = userDTO.getUsername();
    String password = userDTO.getPassword();
    String status = userDTO.getStatus();
    String nickname=userDTO.getNickname();
    if(StrUtil.isBlank(username)||StrUtil.isBlank(password)||StrUtil.is
Blank(status)||StrUtil.isBlank(nickname)){
        return Result.error("500","信息存在空值");
    }
    if(Objects.equals(userMapper.mapperUsername(username), "1")){
        return Result.error("500","当前账号已存在");
    }
    if(StrUtil.length(password)<=5||StrUtil.length(password)>20){
        return Result.error("500","密码长度错误");
    }
```

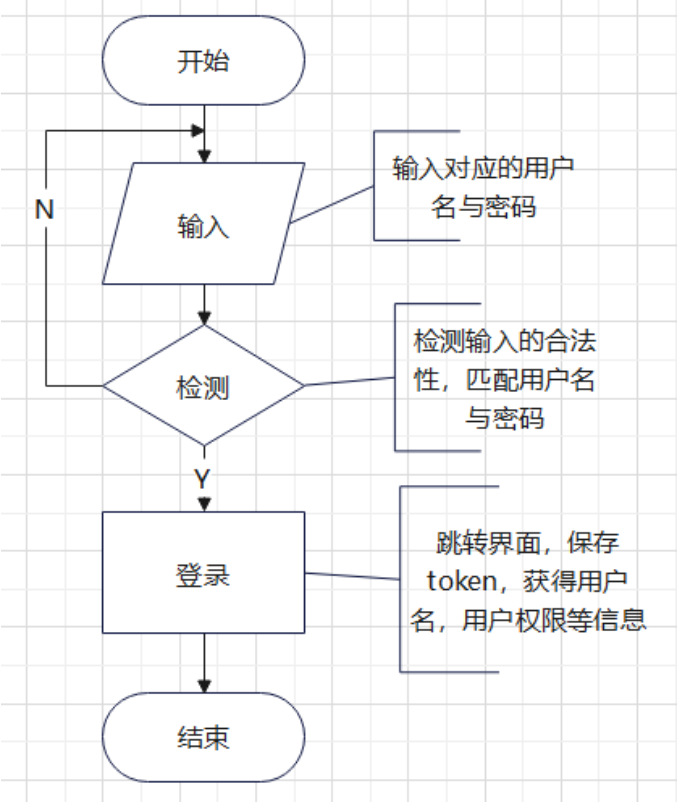
```
    if(! status.equals("学生") && ! status.equals("教师")){
        return Result.error("500","该身份不存在");
    }
    userDTO.setPassword(passwordMD5(username,password));

    userMapper.register(userDTO);
    userDTO.setPassword(null);
    return Result.success(userDTO);
}
```

4.1.3 用户登录与用户信息存储

与正常的登录相似，通过输入账号与密码，然后进行后台检测。检测成功后将对应的部分用户信息与 token 返回给前台，并跳转页面。若检测失败，则登录失败。

流程图：



部分代码：

```
// vue 前端代码
async login(formName) {
    this.$refs['userForm'].validate((valid) => {
        if (valid){
            this.request.post("http://localhost:9090/user/login", this.
user).then(async res => {
```

```
                await console.log(res)

                let loadingInstance = Loading.service({
                    text: "请稍等片刻",
                    background: 'rgba(0,0,0,.5)'
                })
                await delay(200) // 同步设置

                loadingInstance.close();
                if (res.code==="200") {

                    sessionStorage.setItem("user",JSON.stringify(res.da
ta))

                    this.$router.push('/home')
                    this.$message.success("登录成功")
                    if(sessionStorage.getItem("registerUser")){
                        sessionStorage.removeItem("registerUser")
                    }

                } else {
                    this.$message.error(res.msg)
                    this.user.password = ''
                }
            })
        }
    })
},

// 后端代码
public Result login(UserDTO userDTO)
{
    String username = userDTO.getUsername();
    String password = userDTO.getPassword();
    if (StrUtil.isBlank(username) || StrUtil.isBlank(password)){
        return Result.error("500","账号或密码为空");
    }
    if(userMapper.mapperUsername(username)==null){
        return Result.error("500","当前账号不存在");
    }
    password=passwordMD5(username,password);
    if(userMapper.mapperUser(username,password)==null)
    {
        return Result.error("500","密码错误");
    }
    else{
        UserDTO data=userMapper.getUser(username);
```

```
        data.setToken(genToken(username, password));
        return Result.success(data);
    }
}
```

对于 token 的处理，用简单的用户名与密码作为 token 的信号，将用户名作为载荷，具体生成与提取的的代码如下：

```
public static String genToken(String userId,String sign){
    return JWT.create().withAudience(userId)//将 userId 保存至 token 中作
为载荷
        .withExpiresAt(DateUtil.offsetHour(new Date(),2))// 2 小时有效
        .sign(Algorithm.HMAC256(sign));
}

public static User getCurrentUser(){
    try{
        HttpServletRequest request = ((ServletRequestAttributes) Reques
tContextHolder.getRequestAttributes()).getRequest();
        String token =request.getHeader("token");
        if(StrUtil.isNotBlank(token)){
            String username=JWT.decode(token).getAudience().get(0);
            return staticUserMapper.getUserByUsername(username);
        }
    }catch (Exception e){
        return null;
    }
    return null;
}
```

4.2 基础信息管理

注：该部分基本为大量的增删改查的过程，不展示流程图，代码仅展示部分。

4.2.1 用户管理

用户管理主要设计以下对于用户的增删改查，以及对于用户权限的修改。

新增：通过给定信息，增加一些临时不含有学号/教工号的临时学生/教师账号。整个过程通过前端对数据的输入，再检测数据的完整性，数据完整之后向数据库中插入数据，整个过程为线性过程。这种临时账号不具备上课与授课的功能，只具备一些对教室的简单申请的功能，同样的不能申请课程教室。

插入的简易 sql 注解语句：

```
@Insert("INSERT into sys_user(username,password,status,nickname,email,p
hone) " +
            "VALUES (#{username},#{password},#{status},#{nickname},#{em
ail},#{phone})")
```

查找：该过程主要通过筛选信息，利用 sql 的分页语句，对展示的内容进行筛选与分页，与本项目其他任何功能模块的查找基本相似，大致利用类似以下的 sql 语句（.xml 文件）来对需要内容进行分页与筛选。其中的筛选采用模糊查询。

```
<select id="searchPage">
    select *
    from sys_user
    <where>
        <if test="username!=''">
            username like concat('%',#{username},'%')
        </if>
        <if test="email!=''">
            and email like concat('%',#{email},'%')
        </if>
        <if test="phone!=''">
            and phone like concat('%',#{phone},'%')
        </if>
    </where>
    ORDER BY create_time DESC
    limit #{pageNum},#{pageSize}
</select>
```

修改：修改过程与正常的项目修改相同，即在提交修改内容以后检测该账号是否存在，再进行修改，该过程的修改的 xml 文件内代码如下

```
<update id="update">
    update sys_user
    <set>
        <if test="username!=null">
            username = #{username},
        </if>
        <!--            <if test="password!=null">-->
        <!--                username = #{username}-->
        <!--            </if>-->
        <if test="status!=null">
            status=#{status},
        </if>
        <if test="nickname!=null">
            nickname=#{nickname},
        </if>
        <if test="email!=null">
            email=#{email},
        </if>
        <if test="phone!=null">
            phone=#{phone}
        </if>
    </set>
    <where>
        username = #{username}
```

```
    </where>
</update>
```

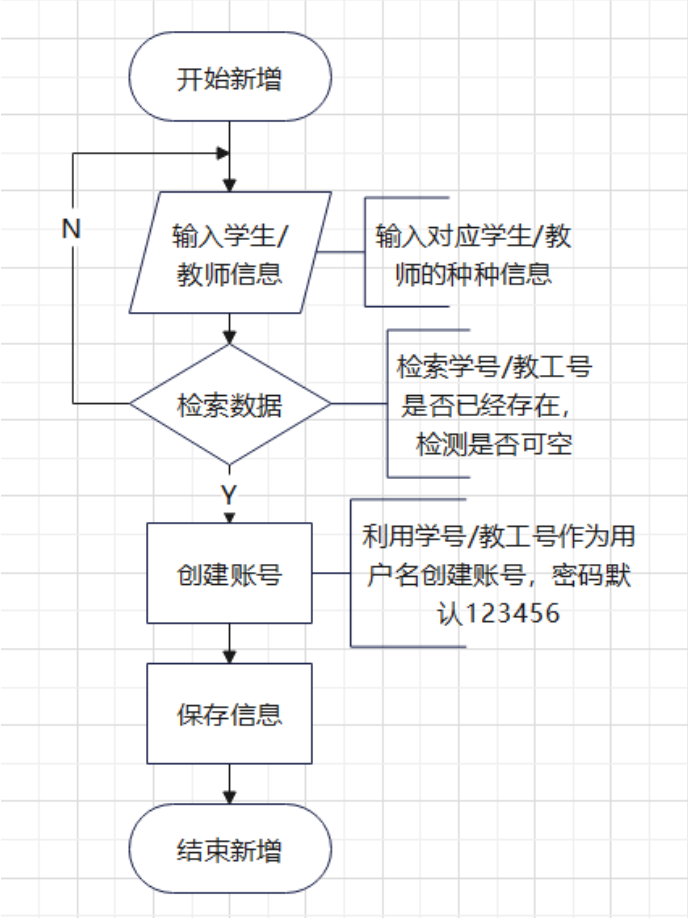
删除：同样与其他项目类似，不过该项目的删除为强行删除，在删除一个用户的过程中会删除一部分与该用户有关的信息，但会保存一些特殊的记录性信息，这个过程又数据库的外键模式保证。删除的注解语句与函数如下：

```
@Delete("delete from sys_user where username = #{username}")
public Integer deleteByUserName(@Param("username") String username);
```

4.2.2 教师/学生管理

该过程主要对于教师和学生的处理基本一致，而其中查找、修改和删除的过程与上述用户管理的过程基本一致，在此不在此处贴附上 xml 语句或者 mybatis 注解，仅作语言描述，而在新增过程中涉及对一个全新账户的创建，这一点是与前面用户管理有所不同的。当然这两个队教师的管理和对学生的管理是两个不同的界面，由于功能基本相同下文阐述中一块阐述。

新增：该过程与上述用户管理中的新增功能最大的不同在于在新增过程中，依托学工号与教工号建立一个所属的账号，该账号有着学生/教师的所有权限。该新增的过程大致如下流程图：



相应的代码，以新增学生为例：

后端 Service 层代码：

```
@Transactional(readOnly = false,isolation = Isolation.REPEATABLE_READ)
public Result save(Student student)
{
    if(studentMapper.mapperSid(student.getSid())==null){
        User user = new User(student.getSid(),null,student.getName(),"
学生" ,student.getEmail(),student.getPhone());
        userService.save(user);
        student.setUsername(student.getSid());
        studentMapper.save(student);
        return Result.success(true);
    }
    else{
        return Result.error("500","该学号已经存在");
    }
}
```

其中关于 userService.save 语句已在上面展示而 studentMapper.save 的相关内容与上述相似。

查找：查找的基本流程与上述用户管理的过程相同，也是通过 xml 语句来筛选与分页进行实现。其 sql 的写法与上述基本一致。以学生为例，学生的权限是与用户绑定的，是通过 sql 的并语句实现的，在实际存储中学生表值只存储学生用户对应的账号，而并不存储学生的权限，学生的权限的查询是依托对应用户的权限的，是通过 `student natural join (select username,email,phone,create_time from sys_user) as user` 这样的 sql 语句确定其是否为学生的。

修改：在该内容模块的修改过程中，修改过程中存在一些不可修改的值，比如学生的学号，学生的用户名以及学生的权限。其余的修改过程与用户管理中修改用户信息是一致的。

删除：与修改相同，整体过程与用户管理中删除操作相同，仍通过外键来保证部分重要信息在删除的过程中保存在数据库中，部分数据随着学生/教师的删除而删除。

4.2.3 课程管理

这个模块基本没有什么非常特殊的操作，都是简单的增删改查操作，同样在新增或者修改的过程中要检测信息的合法性。

新增：与上述用户管理的操作基本相同，通过确定的信息来保证课程的新增。

查找：该模块的查找功能同样使用 xml 语句来对约束条件进行约束。

修改：与用户管理相同，通过先检测，再修改的方式来保证修改的合法性。

删除：该过程仍旧通过外键来保证信息的存在与完整性。

4.2.4 教师授课/学生上课管理

该模块为一个简易的的教师授课与分配学生课程的过程，是为了满足教室调度而实现的特殊需求。而教室授课与学生上课这个过程基本相似，在数据库层面都是一个双主键的模式，两者基本相似，因此同时放在此处展示。

新增：通过输入两种不同管理相应的内容，进行绑定，首先进行对输入内容（课程号、教工/学生号）进行检索，查看是否已经存在绑定以及是否存在输入内容，再通过类似用户管理的 sql 语句将其保存在数据库中。

查找：该模块的查找功能同样使用 xml 语句来对约束条件进行约束。

删除：该过程删除和以上若干功能相似。

4.2.5 教室管理

该模块为新增教室的模块，将教室数字化，将其教室的有关信息如空调、多媒体、教室类型、地理位置抽象成具体信息。其增删改查与上述的各种管理模块一致。

新增：与上述用户管理的操作基本相同，通过确定的信息来实现教室新增。

查找：该模块的查找功能同样使用 xml 语句来对约束条件进行约束。

修改：与用户管理相同，通过先检测，再修改的方式来保证修改的合法性。

删除：该过程仍旧通过外键来保证信息的存在与完整性。

4.3 教室调度管理

该部分为教室调度项目重点的内容简述，首先我将会现进行简述，对整个功能做一个较粗略的描述，其次描述各功能的具体过程。

4.3.1 教室调度过程简述

对于教室的使用，主要分为以下两种：**临时使用**（单次使用），**课程使用**（周期性使用）。

- **临时使用：**主要针对学生和教师在日常生活中对于教室的临时使用。例如学生在一段时间内需要某教室开展活动；教师需要在一段时间内需要某教室答疑等对于教室的单次使用。
- **课程使用：**主要针对学生和教师在教学生活中对于教室的周期性使用，主要为对周期性使用设计。

而对于不同的用户权限，对于学生、教师、管理员应当有不同的事务。因此在此为不同的权限分配不同的事务：

- **对于学生而言：**仅仅涉及教室的临时使用。其所需的功能有：
- ①对空闲教室的查询与申请。②查询自己教室使用信息与 教室申请信息。③对于提出申请但为通过的申请信息进行撤回。
- **对于教师而言：**涉及到临时使用和周期性使用。其所需的功能有：①对空闲教室的查询与申请。②查询自己教室使用信息与 教室申请信息。③对于提出申请但为通过的申请信息进行撤回。④为自己开设的课程申请一个周期性使用教室。⑤查询课程申请教室与课程使用教室。
- **对于管理员而言：**对于管理员而言，不仅包含上述两种身份的所有功能，还需要以下功能：①对所有申请的信息进行审批。②对于一些信息冲突提供一些处理方案。

在此基础上临时使用存在两种申请形式，一为申请性使用、二为调度性使用。

- **申请性使用（申请）：**指通过对于在指定时间内对于指定教室进行申请，等待管理员批准后，方可使用。
- **调度性使用（请求）：**指定使用时间与具体的使用信息内容，如空调多媒体人数等，由系统直接从空闲教室和未被申请的教室里满足需求的教室调度一个分配给申请者，如果此时没有任何可使用的教室，则等待其他对于教室的取消，若到使用时间前一段时间仍未被分配成功，则为申请者返回请求失败的消息。

优先级设计：

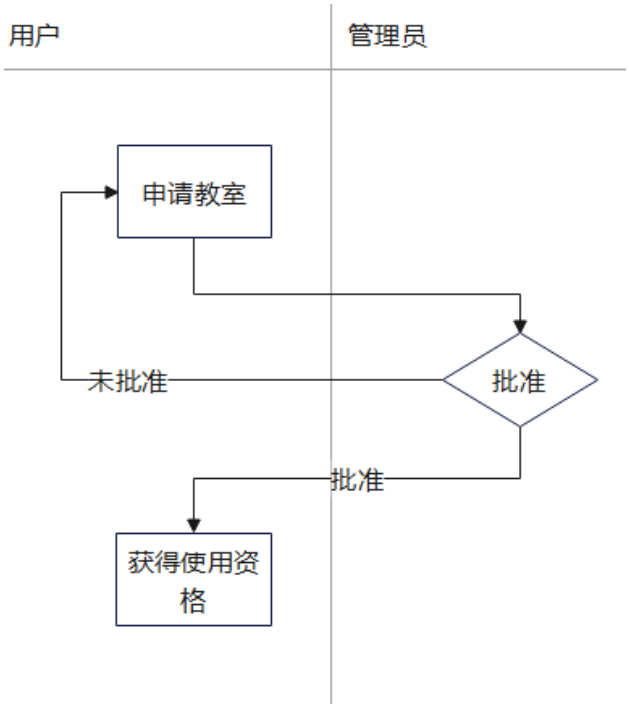
对于以上描述，一共设计了三种对于教室的使用，分别为：临时教室申请、临时教室调度、课程教室，为了便于操作，整个教室调度的优先级为：**课程教室 >> 临时教室申请 > 临时教室申请**。其中课程教室享有最高的优先级，其在申请过程中一旦被申请成功将无条件剥夺临时教室的使用，即使临时教室已经被占用。而两种临时教室的优先级则在于选择上，调度教室只会从确定不会有人使用的教室中去寻找合适的教室，而不会去调度那些申请还为批准的教室。

而有关于不同功能的界面设计，对于临时使用，可以通过查询查询到指定时间内未使用的教室来进行申请，因此查询未使用与申请为一个界面，对于自己的申请的信息存在专属的界面来保证可以方便查询。对于周期使用，通过提交使用申请，来实现教室的周期性申请。

4.3.2 临时教室申请性使用

这个功能由用户提出申请，被管理员批准后即可使用。

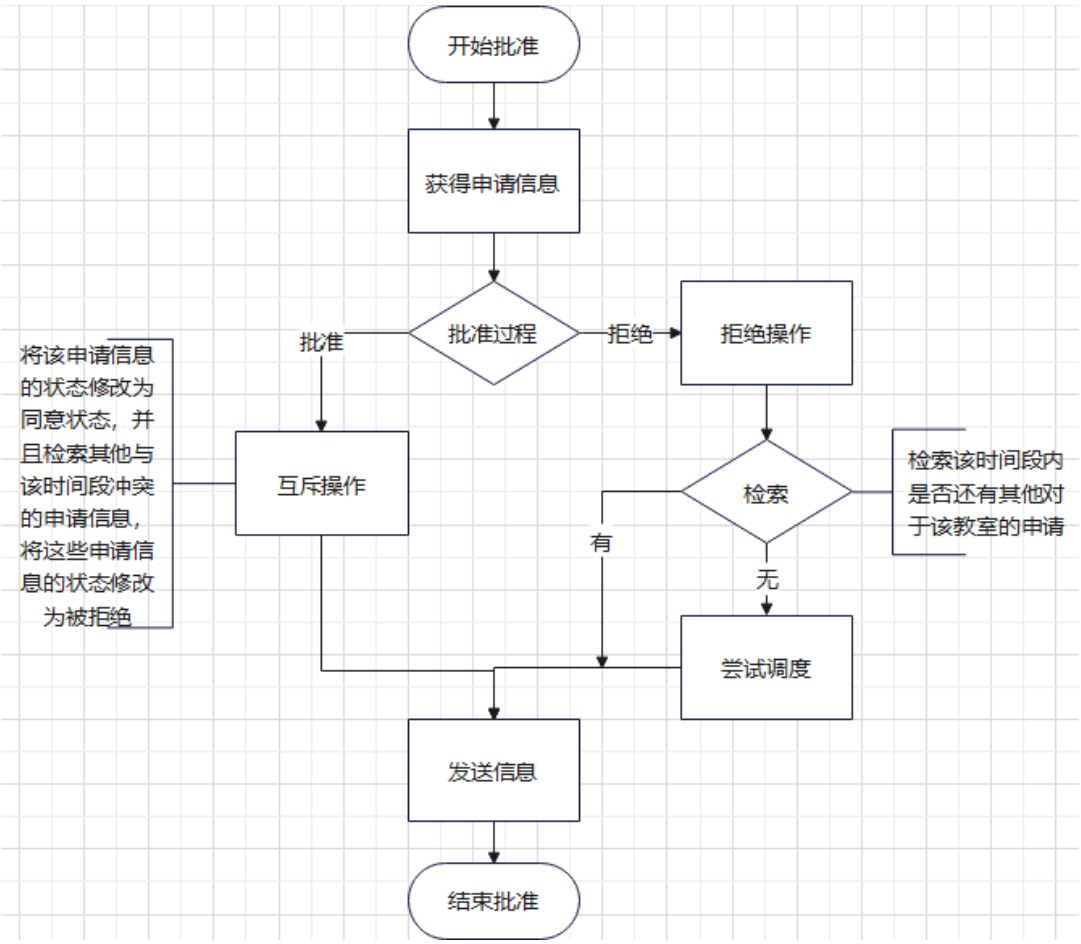
教室的临时使用的大致流程如下图：



对于申请，我们规定只能申请未被占用的教室，对于已经被申请的教室是可以继续被申请的，也就是说教室的使用申请必须申请空闲，申请教室中不是互斥的，使用教室是互斥的。

在用户的申请过程中，其代码与流程与用户管理等管理模块的新增类似，因此在此不贴放代码。

而在此过程中，批准的过程中会存在着同意与拒绝，整个批准过程还会依次进行若干操作，大致流程如下：



此过程的 Service 层代码如下，其中有关尝试调度（asignService.tryBatchReassign(0)）以及消息发送（applyMapper.sendMessage()）的代码在下文中会提及：

```
@Transactional(readOnly = false, isolation = Isolation.SERIALIZABLE)
public Result accepte(@NotNull Apply apply)
{
    // ①检索该申请是否仍为待批准
    String nowMode = applyMapper.findMode(apply.getApid());
    if(!StrUtil.equals(nowMode,"待批准")){
        return Result.error("500","该信息已被处理");
    }
    // ②修改其为已同意
    int update = applyMapper.accept(apply.getApid());
    // ③修改使用表
    utilsTableMapper.insert(apply.getUsername(),apply.getRid(),
                            apply.getStartTime(),apply.getEndTime(),app
ly.getRemark(),"临时");
    // ④将其他与该时段冲突的申请设为拒绝
    // applyMapper.modify(apply.getRid(),apply.getStartTime(),apply.get
```

```
EndTime());
    List<ApplyDTO> temp =applyMapper.modifyList(apply.getRid(),apply.ge
tStartTime(),apply.getEndTime());
    for(ApplyDTO ap:temp)
    {
        applyMapper.refuseByApid(ap.getApid(),"该时段此教室被其他人占用");

        applyMapper.sendMessage(apply.getAdmin(),ap.getUsername(),"您申
请的"+ap.getRoomName()+"教室，已拒绝 "+" 该时段此教室被其他人占用",DateUtil.
date());
    }
    applyMapper.sendMessage(apply.getAdmin(),apply.getUsername(),"您申
请的"+apply.getRoomName()+"教室，已同意",DateUtil.date());
    asignService.tryBatchReassign(0);

    return Result.success(update);
}

@Transactional(readOnly = false, isolation = Isolation.SERIALIZABLE)
public Result refuse(Apply apply)
{
    // ①检索该申请是否仍为待批准
    String nowMode = applyMapper.findMode(apply.getApid());
    if(!StrUtil.equals(nowMode,"待批准")){
        return Result.error("500","该信息已被处理");
    }
    // ②修改其为已拒绝
    int res = applyMapper.refuse(apply);
    applyMapper.sendMessage(apply.getAdmin(),apply.getUsername(),"您申
请的"+apply.getRoomName()+"教室，已拒绝 "+apply.getMsg(),DateUtil.date
());
    asignService.tryBatchReassign(0);
    return Result.success(res);
}
```

同时，提出的申请还有撤回过程，撤回过程的流程与被管理员批准中拒绝的流程相同，在此不再放撤回的流程图，在撤回的过程中仍要进行尝试调度的操作。

此处同样附上撤回的 Service 层代码：

```
@Transactional(readOnly = false,isolation = Isolation.REPEATABLE_READ)
public Result revoke(Integer apid)
{
    String state = applyMapper.findState(apid);
    if(! Objects.equals(state, "待批准"))
        return Result.error("500","该信息已被处理");
    applyMapper.revoke(apid);
    asignService.tryBatchReassign(0);
```

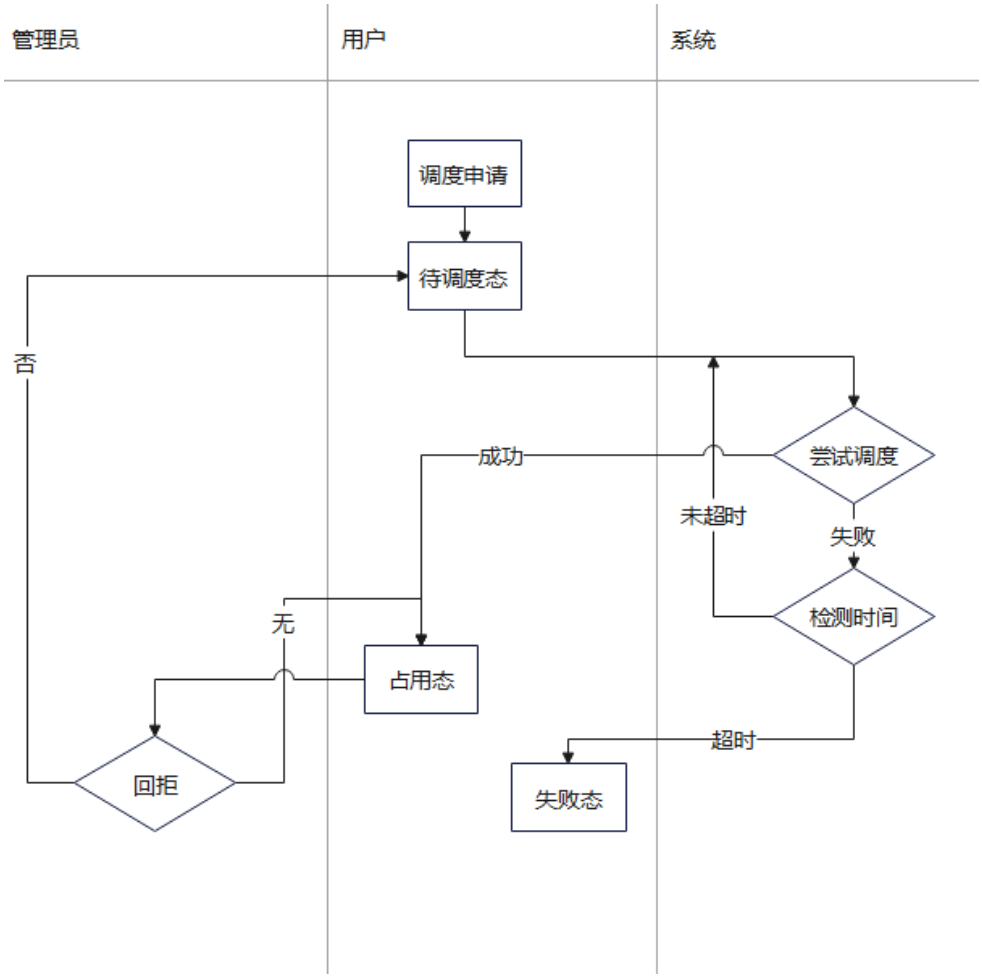
```
    return Result.success("success");
}
```

同样对于使用教室信息的删除，与上文中各种管理模块中的删除方式相同，在此不多赘述。

4.3.3 临时教室调度性使用

而相比于申请式使用，调度式使用的过程较为繁琐，其过程充斥在整个工程运行的过程中。

其大致流程图如下：



同样提交调度的过程与新增一项内容相似，在此不再赘述。

而整个调度过程基于以下原则：①选择未被申请且未被使用进行调度②在提出申请的时刻，进行调度申请③管理员在拒绝其他类型的教室使用、已经其他用户自

行撤回的过程后，需要进行调度④管理员可以回拒调度成功的信息，该过程后仍需重新调度，而在该过程中不再将该请求分配教室。

其中 Service 层的代码如下：

```
// 单个信息的调度
@Transactional(readOnly = false,isolation = Isolation.SERIALIZABLE)
public Boolean tryReassign(int asgid){
    AsignDTO assign = asignMapper.getEntity(asgid);
    if(! Objects.equals(assign.getState(), "未分配"))
        return false;
    DateTime startTime=new DateTime(assign.getStartTime());
    DateTime endTime=new DateTime(assign.getEndTime());
    List<ClassroomDTO> list = asignMapper.findList(startTime,endTime);

    int newid=0;

    for(ClassroomDTO room :list){
        if(room.getCapacity()>=assign.getCapacity()&&
           Objects.equals(room.getMultiMedia(), assign.getMultiMedia())
&&
           Objects.equals(room.getAC(), assign.getAc())&&
           Objects.equals(room.getType(), assign.getType())){
            newid=Integer.parseInt(room.getRid());
            break;
        }
    }

    if(newid==0) return false;

    asignMapper.reassign(newid,assign.getUsername(),startTime,endTime,a
ssign.getRemark(),"临时",asgid);
    asignMapper.success(asgid);
    asignMapper.sendMessage("admin",assign.getUsername(),"您请求教室信息
 "+asgid+" 现已经分配至教室编号 "+newid+"，请查看",DateUtil.date());

    return true;
}

// 批量多个申请的调度
@Transactional(readOnly = false,isolation = Isolation.SERIALIZABLE)
public void tryBatchReassign(int old){
    Date now = DateUtil.date();
    List<Integer> passList=asignMapper.passList(now);
    for(int asgid:passList){
        asignMapper.updatepass(asgid);
        asignMapper.sendMessage("admin",asignMapper.getUsername(asgid),
"您请求教室信息 "+asgid+" 现已经过期请查看",DateUtil.date());
```

```
        }

        List<Integer> assignList=asignMapper.List();
        for(int asgid:assignList){
            if(asgid==old) continue;
            tryReassign(asgid);
        }
    }

    // 回拒
    @Transactional(isolation = Isolation.SERIALIZABLE)
    public Result delete(Integer asgid)
    {
        AsignDTO assign = asignMapper.getEntity(asgid);
        String state=assign.getState();
        if(Objects.equals(state, "已分配")){
            asignMapper.deleteUidByAsgid(asgid);
            asignMapper.sendMessage("admin",assign.getUsername(),"您的请求教
室信息，信息号为 "+asgid+" 已被管理员删除",DateUtil.date());
            asignMapper.deleteAsgid(asgid);
            tryBatchReassign(asgid);
        }
        else{
            asignMapper.sendMessage("admin",assign.getUsername(),"您的请求教
室信息，信息号为 "+asgid+" 已被管理员删除",DateUtil.date());
            asignMapper.deleteAsgid(asgid);
        }
        return Result.success("success");
    }
```

4.3.4 课程教室申请性使用

课程教室为周期性使用，为了保证安全，其整体流程与临时申请教室相似，但是整个过程中寻找合适教室成为非常重要的一步。

通过该界面，先通过各项约束，进行教室搜索，然后在右侧显示可用信息，再从可用教师中选择一个进行申请。

由于课程教室可以剥夺临时教室的使用，因此在右侧显示的这个过程的查找的 java 代码如下，本质式从所有教室中抛出已经被其他课程冲突占用的教室：

```
@Transactional(readOnly = false, isolation = Isolation.REPEATABLE_READ)

public Result findRoom(Integer pageNum, Integer pageSize,
                       String startDate, String endDate, String weekday,

                       String beginTime, String endTime, Integer times)

{
    if (StrUtil.isBlank(startDate) || StrUtil.isBlank(endDate) || StrUt
il.isBlank(weekday)
        || StrUtil.isBlank(beginTime) || StrUtil.isBlank(endTime))
        return Result.error("500", "参数不全");
    pageNum = (pageNum - 1) * pageSize;


    Date dayStart = DateUtil.parse(dateFormat(startDate) + " " + beginT
ime);
    Date dayEnd = DateUtil.parse(dateFormat(startDate) + " " + endTime);

    int nowWeekday = DateUtil.dayOfWeek(dayStart) - 1;
    int offset = 0;
    if (StrUtil.isBlank(weekday))
        offset = 0;
    else
    {
        int Weekday = Integer.parseInt(weekday) - 1;
        if (Weekday >= nowWeekday)
            offset = Weekday - nowWeekday;
        else
            offset = Weekday + 7 - nowWeekday;
    }
    dayStart = DateUtil.offsetDay(dayStart, offset);
    dayEnd = DateUtil.offsetDay(dayEnd, offset);

    List<Classroom> tempRoomList = teacherCourseMapper.classroomList();

    List<Classroom> Rooms = new ArrayList<>();
    for (Classroom classroom : tempRoomList)
    {
        int rid = classroom.getRid();
        Date classStart = dayStart;
        Date classEnd = dayEnd;
        Date LastDay = DateUtil.parse(endDate);
        boolean check = true;
        while (DateUtil.compare(classEnd, LastDay) < 0)
```

```
        {
            List<Integer> temp = teacherCourseMapper.check(rid, classSt
art, classEnd);
            if (temp.size() != 0)
            {
                check = false;
                break;
            }
            for (int i = 1; i <= times; i++)
            {
                classStart = DateUtil.offsetDay(classStart, 7);
                classEnd = DateUtil.offsetDay(classEnd, 7);
            }
        }
        if (check)
            Rooms.add(classroom);
    }
    if (Rooms.size() == 0)
        return Result.error("500", "无可用教室");

    List<ClassroomDTO> data = new ArrayList<>();

    for (int i = 0; i < Math.min(pageSize, Rooms.size()); i++)
    {
        data.add(new ClassroomDTO(Rooms.get(i)));
    }

    Map<String, Object> res = new HashMap<>();
    res.put("data", data);
    res.put("total", Rooms.size());
    System.out.println("~~~~~~~~~~");
    return Result.success(res);
}
```

在此后的申请过程与审核过程与临时的课程申请区别不大，只是在同意的情况下需要进行对临时使用强行剥夺，对其他冲突申请课程设为拒绝，而在被拒绝的情况下也需要尝试调度。

```
@Transactional(readOnly = false, isolation = Isolation.SERIALIZABLE)
public Result accept(iDTO dto)
{
    int tcid=dto.getTcid();
    String admin= dto.getUsername();

    System.out.println("~~~~~~~~~~~~~~");
    System.out.println(tcid);
    System.out.println(admin);
    System.out.println("~~~~~~~~~~~~~~");
```

```
    TeacherCourse tc = teacherCourseMapper.getEntity(tcid);
    // ①检查该信息是否已经被审批
    if (! Objects.equals(tc.getState(), "待批准"))
        return Result.error("500", "该信息已被处理");
    // ②修改审批状态为已同意
    teacherCourseMapper.accept(tcid);
    teacherCourseMapper.sendMessage(admin,tc.getUsername(),"您申请的课程
号为: "+tc.getCid()+" 的教室已申请成功",DateUtil.date());

    // ③对该信息的每一次使用教室日期时间，临时 待批准的拒绝，临时 已同意的拒
绝（与课程冲突）,增加当天使用

    // // 1）把临时使用中与该课程使用冲突的 待批准的设置为已拒绝，remark=与
课程安排冲突
    // // 2）把临时使用中与该课程使用冲突的 已同意的设置为已拒绝，remark=与
课程安排冲突
    // // 3）在临时使用表中加入该课程使用

    int rid = tc.getRid();
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    String startDate = sdf.format(tc.getStartDate());
    String endDate = sdf.format(tc.getEndDate());
    int weekday = tc.getWeekday();
    String beginTime = tc.getBeginTime();
    String endTime = tc.getEndTime();
    int times = tc.getTimes();
    Date dayStart = DateUtil.parse(startDate + " " + beginTime);
    Date dayEnd = DateUtil.parse(startDate + " " + endTime);
    int nowWeekday = DateUtil.dayOfWeek(dayStart) - 1;
    int offset = 0;
    int Weekday = weekday - 1;
    if (Weekday >= nowWeekday)
        offset = Weekday - nowWeekday;
    else
        offset = Weekday + 7 - nowWeekday;
    dayStart = DateUtil.offsetDay(dayStart, offset);
    dayEnd = DateUtil.offsetDay(dayEnd, offset);
    Date classStart = dayStart;
    Date classEnd = dayEnd;
    Date LastDay = DateUtil.parse(endDate);
    while (DateUtil.compare(classEnd, LastDay) < 0)
    {
        // 1）把临时使用中与该课程使用冲突的 待批准的设置为已拒绝，remark=与
课程安排冲突
        List<ApplyDTO> temp =applyMapper.modifyList(rid,classStart,clas
sEnd);
        for(ApplyDTO ap:temp)
        {
```

```
            applyMapper.refuseByApid(ap.getApid(),"该时段此教室被其他人占
用");
            applyMapper.sendMessage(admin,ap.getUsername(),"您申请的"+ap.
getRoomName()+"教室，已拒绝 "+" 该时段与课程冲突",DateUtil.date());
        }
        // 2）把临时使用中与该课程使用冲突的 已同意的设置为重新调度已拒绝，re
mark=与课程安排冲突

        List<Integer> conflictList=teacherCourseMapper.conflictAccepteT
able(rid,classStart,classEnd);
        for(int uid:conflictList){
            int asgid=teacherCourseMapper.getAsgid(uid);
            if(uid!=0){
                asignMapper.drop(asgid);
                asignMapper.sendMessage("admin",asignMapper.getUsername
(asgid),"您请求教室信息 "+asgid+" 获得的教室，现已经被收回",DateUtil.date
());
            }
            teacherCourseMapper.deleteByUid(uid);
        }


        List<ApplyDTO> agreeList = teacherCourseMapper.agreelist(rid,cl
assStart,classEnd);
        for(ApplyDTO apply:agreeList){
            System.out.println("~~~~~~");
            System.out.println(apply);
            Apply allocate = new Apply(apply);

            int newid=0;
            String newName=null;

            Classroom old = teacherCourseMapper.findClassroom(allocate.
getRid());
            List<Classroom> allocateClassroom = teacherCourseMapper.fin
dAllocate(allocate);
            for(Classroom room:allocateClassroom){
                if(Classroom.isSimlar(old,room)){
                    newid=room.getRid();
                    newName=room.getRoomName();
                    break;
                }
            }
            System.out.println("~~~~~~~~");
            System.out.println(newid);
            System.out.println("~~~~~~~~");
            if(newid!=0){
                teacherCourseMapper.conflictChange(apply.getApid(),"您
```

```
之前申请的教室先被课程占用，先调换为"+newName);
                teacherCourseMapper.sendMessage(admin,apply.getUsername
(),"您之前申请的教室"+old.getRoomName()+"先被课程占用，先调换为"+newName,D
ateUtil.date());
                teacherCourseMapper.usage(newid,apply.getUsername(),app
ly.getStartTime(),apply.getEndTime(),apply.getRemark()+"(被调换)","临时
");
            }
            else{
                teacherCourseMapper.conflictAccepte(apply.getApid());
                teacherCourseMapper.sendMessage(admin,apply.getUsername
(),"您之前申请的教室"+old.getRoomName()+"与课程使用冲突，并且暂时没有合适的
教室可以分配给您，请重新申请",DateUtil.date());
            }
        }



        // 3）在临时使用表中加入该课程使用
        String username = teacherCourseMapper.getUsername(tc.getTcid());

        System.out.println(username);
        teacherCourseMapper.accpetInsert(rid,username,classStart,classE
nd,"课程使用","课用");


        for (int i = 1; i <= times; i++)
        {
            classStart = DateUtil.offsetDay(classStart, 7);
            classEnd = DateUtil.offsetDay(classEnd, 7);
        }
    }

    // ④对于时段冲突且时间段冲突且工作日冲突 的检查可用性，若不再可用设为拒绝


    List<TeacherCourse> conflictList = teacherCourseMapper.conflictList
(tc);
    System.out.println(conflictList);
    for (TeacherCourse thc : conflictList)
    {
        rid = thc.getRid();
        startDate = sdf.format(thc.getStartDate());
        endDate = sdf.format(thc.getEndDate());

        weekday = thc.getWeekday();
        beginTime = thc.getBeginTime();
        endTime = thc.getEndTime();
```

```
            times = thc.getTimes();
            dayStart = DateUtil.parse(startDate + " " + beginTime);
            dayEnd = DateUtil.parse(startDate + " " + endTime);
            nowWeekday = DateUtil.dayOfWeek(dayStart) - 1;
            offset = 0;
            Weekday = weekday - 1;
            if (Weekday >= nowWeekday)
                offset = Weekday - nowWeekday;
            else
                offset = Weekday + 7 - nowWeekday;
            dayStart = DateUtil.offsetDay(dayStart, offset);
            dayEnd = DateUtil.offsetDay(dayEnd, offset);
            classStart = dayStart;
            classEnd = dayEnd;
            LastDay = DateUtil.parse(endDate);
            boolean check = true;
            while (DateUtil.compare(classEnd, LastDay) < 0)
            {
                List<Integer> temp = teacherCourseMapper.check(rid, classSt
art, classEnd);
                if (temp.size() != 0)
                {
                    check = false;
                    break;
                }
                for (int i = 1; i <= times; i++)
                {
                    classStart = DateUtil.offsetDay(classStart, 7);
                    classEnd = DateUtil.offsetDay(classEnd, 7);
                }
            }
            if(!check){
                teacherCourseMapper.refuse(thc.getTcid(),"与其他课程时间冲突,
请再次申请");
                teacherCourseMapper.sendMessage(admin,thc.getUsername(),"您
为申请的课程号为："+thc.getCid()+" 的教室与其他课程时间冲突,请再次申请",Date
Util.date());
            }
        }
        asignService.tryBatchReassign(0);
        return Result.success("success");
    }
```

而其撤回与删除的功能与临时申请的过程基本一致，在此不多赘述。

4.3.5 空闲教室查找与时间冲突检测

对于不同的时间上互斥的申请，可能有不同时间的错位，这种情况在此项目中会经常出现，因此在判断互斥的过程中用以下 sql（xml）语句：`endTime > #{startTime} and #{endTime} > startTime`。来维护互斥的时间操作。

而对于空闲教室，采用类似以下的代码来实现查询：

```
<select id="findFreePage">
    select *
    from classroom
    <where>
        rid not in
        (select rid
        from utilization
        where endTime > #{startTime} and #{endTime} > startTime) and
        capacity >= #{capacity}
        <if test="multiMedia!=2">
            and multimedia=#{multiMedia}
        </if>
        <if test="AC!=2">
            and AC=#{AC}
        </if>
        <if test="roomName!=''">
            and roomname like concat('%',#{roomName},'%')
        </if>
        <if test="type!=''">
            and type like concat('%',#{type},'%')
        </if>
    </where>
    limit #{pageNum},#{pageSize}
</select>
```

4.4 信息管理

当管理员批准或拒绝用户的一些请求的时，会给用户发送一些信息来通知用户。

其代码如下：

```
@Insert("insert into message(fromuser, touser, msg,sendtime) values (#
{admin},#{username},#{s},#{now})")
void sendMessage(String admin, String username, String s,Date now);
```

而用户的主页即为消息列表，可以实时关注信息。

5 总结与反思

5.1 系统不足

- 对于消息的通知方面，只完成了在系统主页内的信息发送，并没有实现发送手机短信或邮件的功能，使得用户在不登陆系统时不能第一时间知道自己的申请信息的进度。
- 在前端界面中对于教室的使用表，本意是对于每天每个教室制作一个使用时间线或者忙碌时间表来展示，使得用户更清晰的看到教室的使用情况，但最后并没有成功实现。
- 参数分配与抢占后的重分配过程中，采用的基本为暴力搜索与匹配的算法，除了个别如容量一样的弹性参数，只能对应匹配。不仅效率不高，而且可扩展性小。
- 存在大量的自主指定教室申请，在大量批准后，导致教室利用率不高，后期应当增加一个此前教室使用情况来作为参数，去提高教室使用率。

5.2 系统升级方向

- 可以在系统申请教室的位置显示当前对于该教室的申请数量，来减少教室申请的同时拥堵性。
- 可以在展示空闲信息以及其他展示界面加入更多的图形化模块，使观感更好。
- 在消息界面设置已读、未读等不同状态，使得消息界面不那么单调。
- 进一步打磨调度算法，使其变得更加优秀稳定。
- 设计对于课程教室的调度使用算法，让整体功能变得完善。
- 优化整个界面的 ui，使其 vue 的”味道“变淡。

5.3 个人总结

在进行数据库课程设计的过程中，我收获颇丰，也有了一些深刻的体会和反思。首先，我意识到了数据库设计的重要性。在课程设计中，我深入学习了数据库的基本原理，包括关系型数据库的设计范式、索引优化等方面的知识。通过合理的数据库设计，可以提高数据的存储效率和查询速度，从而更好地支持应用程序的运行。其次，我注意到了实践性的重要性。在课程设计中，我通过实际的项目练习，将理论知识应用到实际中。这使我更加深入地理解了数据库设计的原理，并提升了我在实际项目中处理数据库的能力。在另一些方面本次实验我采用 mybatis 而没有采用用 mybatis-plus，因此自己写了大量的 sql 语句，提高了自己对于数据库的认识，在对于一些事务进行分析的过程中重新思考了事务已经事务隔离级别这个知识点，提高了我对这些方面的认知。这次课程设计制作的并不完美，有些功能通过无数次的 debug 到现在也没有成功配置上，体现了自己的能力有所欠缺，还需要提高自

己的代码能力。自己一开始对模型的构建不够完善，导致后期工作在“面多了加水水多了加面”，整体效率不高。

总的来说，通过这次数据库课程设计，我不仅学到了丰富的数据库知识，也提高了实践能力。在未来的学习和工作中，我将会继续加强对数据库方面知识的学习，并将其运用到实际项目中，为提升自己的能力而不懈努力。