课程管理后台系统实现

1. 用户登录

包含用户登录和登出功能；

登录参考代码：

public User adminLogin(@RequestBody UserVo userVo) throws Exception {  
 if (userVo == null) {  
 return new User();  
 }  
 User user = userRepository.findByAccountAndPasswordAndStatus(userVo.getAccount(), userVo.getPassword(), 1);  
 if (user == null) {  
 throw new ServiceException("账号或密码错误");  
 }  
 if (RoleEnum.STUDENT.getValue().equals(user.getRole())) {  
 throw new ServiceException("您不是教师或者管理员,无法登录后台,请联系管理员");  
 }  
 return user;  
}

1. 用户管理

完成用户的添加、修改、删除等功能，根据登录用户的角色实现不同的权限；管理员可以添加学生和老师，老师登录只能添加学生；

查询学生列表功能：

public PagerHelper<User> list(String key, String role, Integer pageNo, Integer pageSize) {  
 PagerHelper<User> result = new PagerHelper<>();  
 StringBuffer sql = new StringBuffer("select \* from s\_user where 1=1");  
 List<Object> params = new ArrayList<>();  
 if (StringUtils.isNotEmpty(key)) {  
 sql.append(" and (user\_name like ? or account like ?)");  
 params.add("%" + key + "%");  
 params.add("%" + key + "%");  
 }  
 if (StringUtils.isNotEmpty(role)) {  
 sql.append(" and role = ?");  
 params.add(role);  
 }  
  
 //判断是否为管理员,管理员不进行过滤; 如果是老师，则查询对应班级的学生  
 User tempUser = userRepository.getOne(GaeaContext.getAdminUserId());  
 if (!RoleEnum.ADMIN.getValue().equals(tempUser.getRole())) {  
 List<Classes> classesList = classesRepository.findByTeacherId(GaeaContext.getAdminUserId());  
 if (CollectionUtils.isEmpty(classesList)) {  
 return result;  
 }  
 sql.append(" and class\_id in (" + buildParam(classesList, params) + ")");  
 }  
 StringBuffer totalSql = new StringBuffer("select count(1) from (" + sql.toString() + ") a");  
 Integer total = jdbcTemplate.queryForObject(totalSql.toString(), params.toArray(), Integer.class);  
 int startIndex = (pageNo - 1) \* pageSize;  
 sql.append(" limit ?,?");  
 params.add(startIndex);  
 params.add(pageSize);  
 List<User> users = jdbcTemplate.query(sql.toString(), params.toArray(), new BeanPropertyRowMapper<User>(User.class));  
 if (CollectionUtils.isNotEmpty(users)) {  
 Map<Integer, Classes> classesMap = getAllClassesMap();  
 for (User user : users) {  
 user.setClassName(user.getClassId() == null || classesMap.get(user.getClassId()) == null ? "" : classesMap.get(user.getClassId()).getClassName());  
 }  
 }  
 result.setTotal(total);  
 result.setItems(users);  
 return result;  
}

创建用户参考代码：

public User create(User user) throws Exception {  
 User tempUser = userRepository.findByAccount(user.getAccount());  
 if (tempUser != null) {  
 throw new ServiceException("账号【" + user.getAccount() + "】已存在");  
 }  
 if (StringUtils.isEmpty(user.getPassword())) {  
 user.setPassword(MD5Utils.MD5Encode("123456", "utf8"));  
 }  
 if (StringUtils.isEmpty(user.getRole())) {  
 user.setRole(RoleEnum.STUDENT.getValue());  
 }  
 user.setCreateTime(new Date());  
 user.setStatus(1);  
 return userRepository.save(user);  
}

1. 课程管理

获取课程列表参考代码：

public PagerHelper<CourseVo> list(String key, Integer tagId, boolean isFront, Integer pageNo, Integer pageSize) throws Exception {  
 PagerHelper<CourseVo> result = new PagerHelper<>();  
 StringBuffer sql = new StringBuffer("select \* from s\_course where 1=1");  
 List<Object> params = new ArrayList<>();  
 if (StringUtils.isNotEmpty(key)) {  
 sql.append(" and course\_name like ?");  
 params.add("%" + key + "%");  
 }  
 if (tagId != null) {  
 sql.append(" and tag\_id like ?");  
 params.add("%" + tagId + "%");  
 }  
 if (!isFront) {  
 //判断是否为管理员,管理员不进行过滤  
 User user = userService.get(GaeaContext.getAdminUserId());  
 if (!RoleEnum.ADMIN.getValue().equals(user.getRole())) {  
 sql.append(" and teacher\_id = ?");  
 params.add(GaeaContext.getAdminUserId());  
 }  
 }  
 StringBuffer totalSql = new StringBuffer("select count(1) from (" + sql.toString() + ") a");  
 Integer total = jdbcTemplate.queryForObject(totalSql.toString(), params.toArray(), Integer.class);  
  
 int startIndex = (pageNo - 1) \* pageSize;  
 sql.append(" limit ?,?");  
 params.add(startIndex);  
 params.add(pageSize);  
 List<CourseVo> courseVos = jdbcTemplate.query(sql.toString(), params.toArray(), new BeanPropertyRowMapper<>(CourseVo.class));  
 if (CollectionUtils.isNotEmpty(courseVos)) {  
 for (CourseVo courseVo : courseVos) {  
 //获取课程章节信息  
 List<CourseChapter> courseChapters = courseChapterRepository.findByCourseId(courseVo.getId());  
 courseVo.setCourseChapterVos(resolveChapter(courseChapters, courseVo.getId(), false));  
 //解析课程的标签  
 courseVo.setTags(getTagNames(courseVo.getTagId()));  
 }  
 }  
 result.setTotal(total);  
 result.setItems(courseVos);  
 return result;  
}

**创建课程章节代码参考：**

public CourseChapterVo create(CourseChapterVo courseChapterVo) {  
 List<CourseChapter> courseChapters = new ArrayList<>();  
 CourseChapter courseChapter = new CourseChapter();  
 BeanUtils.copyProperties(courseChapterVo, courseChapter);  
 courseChapter.setCreateTime(new Date());  
 courseChapter = courseChapterRepository.save(courseChapter);  
 //保存子章节信息  
 if (CollectionUtils.isNotEmpty(courseChapterVo.getCourseChapterVos())) {  
 for (CourseChapterVo chapterVo : courseChapterVo.getCourseChapterVos()) {  
 CourseChapter chapter = new CourseChapter();  
 BeanUtils.copyProperties(chapterVo, chapter);  
 chapter.setChapterParentId(courseChapter.getId());  
 chapter.setCourseId(courseChapterVo.getCourseId());  
 courseChapters.add(chapter);  
 }  
 }  
 courseChapterRepository.saveAll(courseChapters);  
 return courseChapterVo;  
}

修改章节代码：

public CourseChapterVo update(Integer id, CourseChapterVo courseChapterVo) throws Exception {  
 List<CourseChapter> courseChapters = new ArrayList<>();  
 CourseChapter courseChapter = courseChapterRepository.getOne(id);  
 if (courseChapter == null) {  
 throw new ServiceException("不存在【" + id + "】对应的章节");  
 }  
 BeanUtils.copyProperties(courseChapterVo, courseChapter);  
 courseChapter.setId(id);  
 courseChapterRepository.save(courseChapter);  
 //保存子章节信息  
 if (CollectionUtils.isNotEmpty(courseChapterVo.getCourseChapterVos())) {  
 courseChapterRepository.deleteByChapterParentId(id);  
 for (CourseChapterVo chapterVo : courseChapterVo.getCourseChapterVos()) {  
 CourseChapter chapter = new CourseChapter();  
 BeanUtils.copyProperties(chapterVo, chapter);  
 chapter.setChapterParentId(courseChapter.getId());  
 chapter.setCourseId(courseChapterVo.getCourseId());  
 courseChapters.add(chapter);  
 }  
 }  
 courseChapterRepository.saveAll(courseChapters);  
 return courseChapterVo;  
}

1. 班级管理

班级查询：

public PagerHelper<Classes> list(String key, Integer pageNo, Integer pageSize) throws Exception {  
 PagerHelper<Classes> result = new PagerHelper<>();  
 StringBuffer sql = new StringBuffer("select \* from s\_classes where 1=1");  
 List<Object> params = new ArrayList<>();  
 if (StringUtils.isNotEmpty(key)) {  
 sql.append(" and class\_name like ?");  
 params.add("%" + key + "%");  
 }  
 //判断是否为管理员,管理员不进行过滤  
 User user = userService.get(GaeaContext.getAdminUserId());  
 if (!RoleEnum.ADMIN.getValue().equals(user.getRole())) {  
 sql.append(" and teacher\_id = ?");  
 params.add(GaeaContext.getAdminUserId());  
 }  
 StringBuffer totalSql = new StringBuffer("select count(1) from (" + sql.toString() + ") a");  
 Integer total = jdbcTemplate.queryForObject(totalSql.toString(), params.toArray(), Integer.class);  
 int startIndex = (pageNo - 1) \* pageSize;  
 sql.append(" limit ?,?");  
 params.add(startIndex);  
 params.add(pageSize);  
 List<Classes> classesList = jdbcTemplate.query(sql.toString(), params.toArray(), new BeanPropertyRowMapper<Classes>(Classes.class));  
 if (CollectionUtils.isNotEmpty(classesList)) {  
 Map<Integer, User> userMap = userService.getAllUserMap();  
 for (Classes classes : classesList) {  
 classes.setProgress(classes.getProgress() == null ? 0 : classes.getProgress());  
 classes.setTeacherName(classes.getTeacherId() == null || userMap.get(classes.getTeacherId()) == null ? "" : userMap.get(classes.getTeacherId()).getUserName());  
 classes.setMonitorName(classes.getMonitor() == null || userMap.get(classes.getMonitor()) == null ? "" : userMap.get(classes.getMonitor()).getUserName());  
 }  
 }  
 result.setTotal(total);  
 result.setItems(classesList);  
 return result;  
}

查看学生签到代码：

public List<StudentVo> findClassesStudent(Integer classId) {  
 List<User> users = userRepository.findByClassId(classId);  
 if (CollectionUtils.isEmpty(users)) {  
 return Collections.emptyList();  
 }  
 List<StudentVo> studentVos = new ArrayList<>();  
 Classes classes = classesRepository.getOne(classId);  
 for (User user : users) {  
 Sign sign = signRepository.findByStudentIdAndTeacherIdAndCreateTime(user.getId(), classes.getTeacherId(), simpleDateFormat.format(new Date()));  
 StudentVo studentVo = new StudentVo();  
 studentVo.setUserName(user.getUserName());  
 studentVo.setAccount(user.getAccount());  
 studentVo.setSign(sign == null ? 0 : 1);  
 studentVos.add(studentVo);  
 }  
 return studentVos;  
}

导出班级学生代码：

/\*\*  
 \* 导出班级学生  
 \*  
 \* @param response  
 \* @param classId  
 \*/  
@Override  
public void exportStudent(HttpServletResponse response, Integer classId) {  
 List<UserExportVo> exportVos = new ArrayList<>();  
 Classes classes = classesRepository.getOne(classId);  
 List<User> users = userRepository.findByClassId(classId);  
 ExcelUtils<UserExportVo> exportExcelUtil = new ExcelUtils<>();  
 Map<String, String> operationMap = new HashMap<>();  
 Map<String, Object> valueMap = new HashMap<>();  
 String fileName = classes == null ? "学生" : classes.getClassName() + "学生";  
 String[] headers = new String[]{"学生名称", "学生账号", "班级名称", "身份证", "性别", "手机号码", "状态", "创建时间"};  
 for (User user : users) {  
 UserExportVo exportVo = new UserExportVo();  
 BeanUtils.copyProperties(user, exportVo);  
 exportVo.setClassName(classes.getClassName());  
 exportVo.setSex(user.getSex() == 1 ? "男" : "女");  
 exportVo.setStatus(user.getStatus() == 1 ? "有效" : "失效");  
 exportVos.add(exportVo);  
 }  
 exportExcelUtil.exportExcel(fileName, headers, exportVos, "yyyy-MM-dd HH:mm:ss", fileName, response, operationMap, valueMap);  
}

1. 作业管理

**查看提交作业学生参考代码：**

public List<HomeworkUserVo> listUsers(Integer id) {  
 List<HomeworkSubmit> homeworkSubmits = homeworkSubmitRepository.findByHomeworkId(id);  
 if (CollectionUtils.isEmpty(homeworkSubmits)) {  
 return Collections.EMPTY\_LIST;  
 }  
 List<HomeworkUserVo> homeworkUserVoList = new ArrayList<>();  
 Map<Integer, User> userMap = userService.getAllUserMap();  
 for (HomeworkSubmit homeworkSubmit : homeworkSubmits) {  
 HomeworkUserVo homeworkUserVo = new HomeworkUserVo();  
 homeworkUserVo.setUserId(homeworkSubmit.getUserId());  
 homeworkUserVo.setHomeworkId(homeworkSubmit.getHomeworkId());  
 homeworkUserVo.setUserName(homeworkSubmit.getUserId() == null || userMap.get(homeworkSubmit.getUserId()) == null ? "" : userMap.get(homeworkSubmit.getUserId()).getUserName());  
 homeworkUserVo.setSubmitTime(homeworkSubmit.getCreateTime());  
 homeworkUserVo.setSubmitHomeworkFilePath(homeworkSubmit.getHomeworkFilePath());  
 homeworkUserVo.setScore(homeworkSubmit.getScore());  
 homeworkUserVoList.add(homeworkUserVo);  
 }  
 return homeworkUserVoList;  
}  
 老师对用户提交作业评分代码：

public void score(Integer id, Integer userId, Integer score) {  
 HomeworkSubmit homeworkSubmit = homeworkSubmitRepository.findByHomeworkIdAndUserId(id, userId);  
 homeworkSubmit.setScore(score);  
 homeworkSubmitRepository.save(homeworkSubmit);  
}

导出提交作业学生（如果有选择班级，则导出该班级下所有学生的提交；没有选择班级，则导出该作业下面的所有提交学生），参考代码：

public void exportSubmitStudent(HttpServletResponse response, Integer id, Integer classId) {  
 Homework homework = homeworkRepository.getOne(id);  
 Classes classes = null;  
 List<Integer> userIds = new ArrayList<>();  
 List<HomeworkSubmit> homeworkSubmits = new ArrayList<>();  
 if (classId != null) {  
 classes = classesRepository.getOne(classId);  
 List<User> users = userRepository.findByClassId(classId);  
 if (CollectionUtils.isNotEmpty(users)) {  
 for (User user : users) {  
 userIds.add(user.getId());  
 }  
 }  
 homeworkSubmits = homeworkSubmitRepository.findByHomeworkIdAndUserIdIn(id, userIds);  
 } else {  
 homeworkSubmits = homeworkSubmitRepository.findByHomeworkId(id);  
 }  
 List<HomeworkUserExportVo> userExportVos = new ArrayList<>();  
 Map<Integer, User> userMap = userService.getAllUserMap();  
 for (HomeworkSubmit homeworkSubmit : homeworkSubmits) {  
 User user = userRepository.getOne(homeworkSubmit.getUserId());  
 Classes tempClasses = null;  
 if (user != null) {  
 tempClasses = classesRepository.getOne(user.getClassId());  
 }  
 HomeworkUserExportVo userExportVo = new HomeworkUserExportVo();  
 userExportVo.setClassName(tempClasses == null ? "" : tempClasses.getClassName());  
 userExportVo.setHomeworkName(homework.getHomeworkTitle());  
 userExportVo.setScore(homeworkSubmit.getScore() == null ? "" : "" + homeworkSubmit.getScore());  
 userExportVo.setUserName(homeworkSubmit.getUserId() == null || userMap.get(homeworkSubmit.getUserId()) == null ? "" : userMap.get(homeworkSubmit.getUserId()).getUserName());  
 userExportVo.setSubmitHomeworkFilePath(homeworkSubmit.getHomeworkFilePath());  
 userExportVo.setSubmitTime(homeworkSubmit.getCreateTime());  
 userExportVos.add(userExportVo);  
 }  
 ExcelUtils<HomeworkUserExportVo> exportExcelUtil = new ExcelUtils<>();  
 Map<String, String> operationMap = new HashMap<String, String>();  
 Map<String, Object> valueMap = new HashMap<String, Object>();  
 String fileName = classes == null ? "所有提交作业学生" : classes.getClassName() + "提交作业学生";  
 String[] headers = new String[]{"用户名称", "班级名称", "作业名称", "评分", "提交作业文件路径", "提交时间"};  
 exportExcelUtil.exportExcel(fileName, headers, userExportVos, "yyyy-MM-dd HH:mm:ss", fileName, response, operationMap, valueMap);  
}

1. 统计相关

**班级学生签到统计：可以选择当前老师登录的班级，图表只展示近七天的签到情况**

**public List<SignStatisticVo> getSignStatistic(Integer days, Integer classId) {  
 List<SignStatisticVo> result = new ArrayList<>();  
 List<Object> params = new ArrayList<>();  
 StringBuffer sql = new StringBuffer("SELECT DATE\_FORMAT(si.create\_time,\"%Y-%m-%d\") date,count(si.student\_id) sign from s\_sign si ");  
 sql.append(" LEFT JOIN s\_user us on us.id = si.student\_id");  
 sql.append(" where 1=1");  
 if (classId != null) {  
 sql.append(" and us.class\_id = ?");  
 params.add(classId);  
 }  
 sql.append(" GROUP BY DATE\_FORMAT(si.create\_time,\"%Y-%m-%d\")");  
 List<SignStatisticVo> signStatisticVos = jdbcTemplate.query(sql.toString(), params.toArray(), new BeanPropertyRowMapper<SignStatisticVo>(SignStatisticVo.class));  
 Map<String, Integer> signMap = new HashMap<>();  
 for (SignStatisticVo signStatisticVo : signStatisticVos) {  
 signMap.put(signStatisticVo.getDate(), signStatisticVo.getSign());  
 }  
 //获取近几天数据  
 List<String> dates = DateUtils.getDayBetween(days);  
 //获取班级人数  
 List<User> users = userRepository.findByClassId(classId);  
 for (String date : dates) {  
 SignStatisticVo signStatisticVo = new SignStatisticVo();  
 signStatisticVo.setSign(signMap.get(date) == null ? 0 : signMap.get(date));  
 signStatisticVo.setNotSign(users.size() - signStatisticVo.getSign());  
 signStatisticVo.setDate(date);  
 result.add(signStatisticVo);  
 }  
 return result;  
}**

**作业提交情况统计：（统计老师发布作业的提交情况，如果是老师角色登录，则只能看到自己发布的作业统计，无法选择查看别的老师发布的作业提交情况）**

**public List<HomeworkStatisticVo> getHomeworkStatistic(Integer teacherId) {  
 List<Object> params = new ArrayList<>();  
 StringBuffer sql = new StringBuffer("SELECT home.homework\_title title,COUNT(sub.user\_id) user from s\_homework\_submit sub ");  
 sql.append(" LEFT JOIN s\_homework home on sub.homework\_id = home.id");  
 sql.append(" where 1=1");  
 if (teacherId != null) {  
 sql.append(" and home.publish\_teacher = ?");  
 params.add(teacherId);  
 }  
 sql.append(" GROUP BY sub.homework\_id");  
 List<HomeworkStatisticVo> homeworkStatisticVos = jdbcTemplate.query(sql.toString(), params.toArray(), new BeanPropertyRowMapper<>(HomeworkStatisticVo.class));  
 return homeworkStatisticVos;  
}**

**班级学生学习情况统计（统计班级下学生学习课程数）**

**public List<CourseStatisticVo> getCourseStatistic(Integer classId) {  
 //获取班级下所有用户信息  
 List<User> users = userRepository.findByClassId(classId);  
 if (CollectionUtils.isEmpty(users)) {  
 return Collections.EMPTY\_LIST;  
 }  
 List<CourseStatisticVo> courseStatisticVos = new ArrayList<>();  
 Map<String, Integer> studyCourseMap = getUserStudy();  
 //设置用户学习课程数  
 for (User user : users) {  
 CourseStatisticVo courseStatisticVo = new CourseStatisticVo();  
 courseStatisticVo.setUserName(user.getUserName());  
 courseStatisticVo.setStudyCourse(studyCourseMap.getOrDefault(user.getUserName(), 0));  
 courseStatisticVos.add(courseStatisticVo);  
 }  
 return courseStatisticVos;  
}**