# 层次图

# 

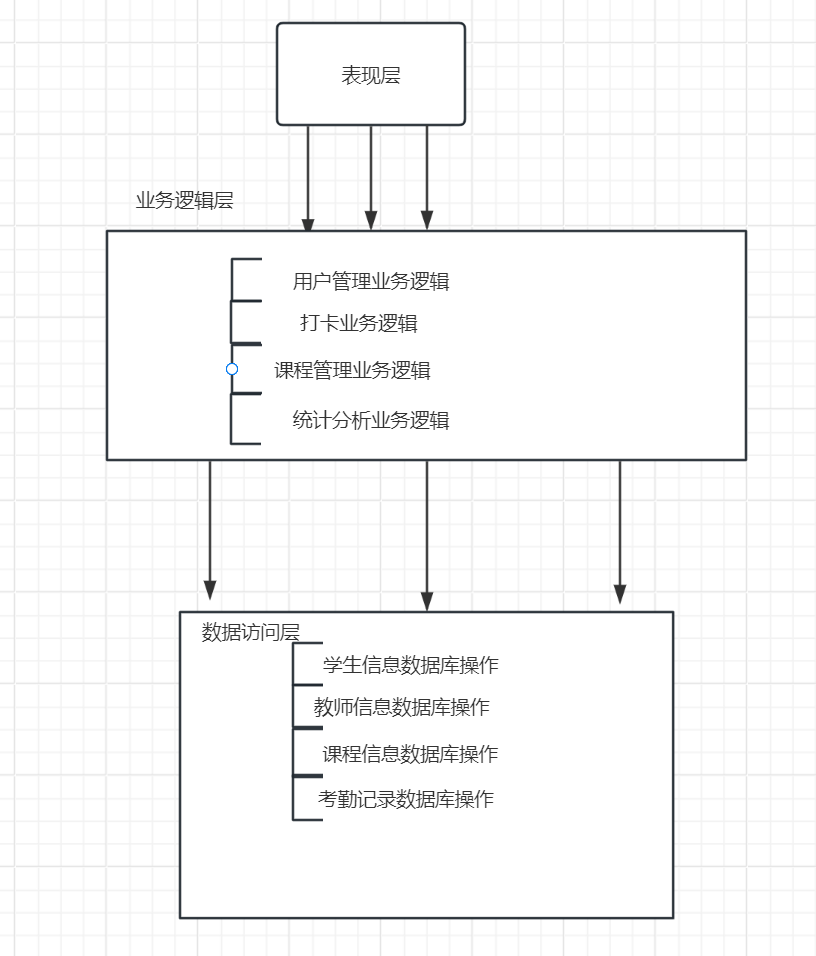
# ISO图



### **结构图**

#### **整体结构**

* ****表现层（UI 层）****
  + 学生端界面（登录、打卡操作界面、打卡记录查看等）
  + 教师端界面（课程管理操作界面、考勤查看与统计界面等）
* ****业务逻辑层****
  + 用户管理业务逻辑（处理各类用户相关操作逻辑）
  + 打卡业务逻辑（实现不同打卡方式的逻辑判断、记录等）
  + 课程管理业务逻辑（课程相关增删改查等操作逻辑）
  + 统计分析业务逻辑（进行考勤数据统计、报表生成的逻辑）
* ****数据访问层****
  + 学生信息数据库操作（增删改查学生基本信息、打卡记录关联等）
  + 教师信息数据库操作（教师账号等相关数据操作）
  + 课程信息数据库操作（课程数据的存储、读取、更新等）
  + 考勤记录数据库操作（专门用于处理打卡记录、统计结果等数据的存储和查询）



### **用户接口设计**

* ****学生端接口****：
  + ****登录接口****：学生通过学号和密码登录系统，接口验证用户身份信息，若正确则返回登录成功及学生相关课程信息，包括课程名称、上课时间、上课地点等，以方便学生后续打卡操作及查看课程安排。例如，采用 RESTful API 设计，接口路径为 /student/login，请求方法为 POST，请求参数为 { "studentId": "学号", "password": "密码" }，成功响应数据为 { "status": "success", "studentInfo": { "name": "学生姓名", "courses": [ { "courseName": "课程 1", "classTime": "周一 9:00 - 10:30", "classroom": "教学楼 301" },... ] } }。
  + ****打卡接口****：支持多种打卡方式，如扫码打卡、定位打卡和手动打卡。扫码打卡时，学生扫描课程专属二维码，接口获取二维码信息并验证课程与学生当前所选课程的匹配性；定位打卡则获取学生的地理位置信息，判断是否在上课教室的有效范围内；手动打卡需教师在教师端操作后，学生端显示相应的手动打卡确认信息。以扫码打卡为例，接口路径为 /student/scanCheckIn，请求方法为 POST，请求参数为 { "qrCodeData": "二维码数据", "studentId": "学号" }，成功响应数据为 { "status": "success", "message": "打卡成功" }，若打卡失败则返回相应错误信息，如课程不匹配或已过打卡时间等。
  + ****打卡记录查询接口****：学生可查询自己的历史打卡记录，可按时间段（如本周、本月、本学期等）进行筛选查询。接口路径为 /student/checkInHistory，请求方法为 GET，可带查询参数如 { "startDate": "开始日期", "endDate": "结束日期" }，响应数据为 { "status": "success", "checkInRecords": [ { "courseName": "课程名称", "checkInTime": "打卡时间", "checkInMethod": "打卡方式" },... ] }。
* ****教师端接口****：
  + ****登录接口****：与学生端登录类似，教师通过账号和密码登录，接口验证成功后返回教师所授课程信息、班级学生名单等数据，方便教师进行课程管理和考勤操作。接口路径为 /teacher/login，请求方法和参数格式类似学生端登录接口，响应数据为 { "status": "success", "teacherInfo": { "name": "教师姓名", "courses": [ { "courseName": "课程 1", "classStudents": [ { "studentId": "学号 1", "studentName": "学生姓名 1" },... ] } ] } }。
  + ****课程管理接口****：包括课程信息的创建、编辑和删除操作。创建课程时，教师需填写课程名称、上课时间、上课地点、课程周期等信息，接口将数据保存到数据库，并返回课程创建成功信息及课程 ID。例如创建课程接口路径为 /teacher/createCourse，请求方法为 POST，请求参数为 { "courseName": "课程名称", "classTime": "上课时间", "classroom": "上课地点", "courseDuration": "课程周期" }，响应数据为 { "status": "success", "courseId": "课程唯一标识" }。编辑课程接口（/teacher/editCourse）和删除课程接口（/teacher/deleteCourse）类似，根据课程 ID 对相应课程信息进行修改或删除操作，并返回操作结果信息。
  + ****考勤管理接口****：教师可实时查看当前课程的学生打卡情况，包括已打卡人数、未打卡人数及学生名单，还可对考勤数据进行统计分析，如按周、月、学期统计学生的出勤天数、缺勤次数等。查看考勤接口路径为 /teacher/checkAttendance，请求方法为 GET，可带课程 ID 参数（{ "courseId": "课程 ID" }），响应数据为 { "status": "success", "attendanceData": { "checkedInStudents": [ { "studentId": "学号 1", "studentName": "学生姓名 1", "checkInTime": "打卡时间" },... ], "uncheckedInStudents": [ { "studentId": "学号 2", "studentName": "学生姓名 2" } ], "totalStudents": "班级总人数" } }。统计分析考勤接口（/teacher/analyzeAttendance）可根据教师选择的统计时间段和维度，返回相应的统计报表数据，如 { "status": "success", "attendanceReport": { "weekSummary": { "student1": { "presentDays": 3, "absentDays": 2 },... }, "monthSummary": { "averageAttendanceRate": "80%" },... } }。

### **外部系统接口设计**

* ****教务管理系统接口****：
  + ****课程数据同步接口****：为保证上课打卡系统中的课程信息与学校教务管理系统中的课程数据保持一致，定时从教务管理系统获取最新的课程安排信息，包括新增课程、课程调整（如时间、地点变更）、课程取消等数据。接口采用定时任务触发，例如每天凌晨 2 点执行一次同步操作。接口路径为 /sync/courseData，请求方法为 GET，响应数据为教务管理系统中更新后的课程信息列表，格式为 { "courses": [ { "courseId": "课程 ID", "courseName": "课程名称", "classTime": "上课时间", "classroom": "上课地点", "teacherId": "教师 ID", "isActive": "是否有效（新增或修改为 true，取消为 false）" },... ] }。上课打卡系统根据 isActive 字段对本地课程数据进行相应的新增、更新或删除操作，确保数据的准确性和及时性，避免因课程信息不一致导致的打卡混乱问题，提高教学管理的协同性和效率。
  + ****学生信息同步接口****：定期同步学生的基本信息（如学号、姓名、专业、班级等）和学籍状态（入学、休学、退学等）变化情况，确保上课打卡系统能够准确识别学生身份并进行相应的权限管理和考勤记录。同步接口同样采用定时任务方式，每周执行一次，接口路径为 /sync/studentData，请求方法为 GET，响应数据为 { "students": [ { "studentId": "学号", "studentName": "学生姓名", "major": "专业", "class": "班级", "status": "学籍状态" },... ] }。系统根据同步的学生信息更新本地数据库，保证学生信息的完整性和准确性，为精准的考勤管理提供数据支持，同时也有助于学校对学生的全面管理，提升教育管理的信息化水平。

### **内部模块接口设计**

* ****打卡模块与用户管理模块接口****：
  + ****用户身份验证接口****：当学生或教师进行打卡操作时，打卡模块调用用户管理模块的身份验证接口，验证用户登录状态的合法性，防止非法打卡行为。接口接收用户输入的账号和密码（或其他身份标识信息），返回验证结果（成功或失败）及用户相关信息（如用户类型、权限级别、所属班级或课程等）。接口定义在用户管理模块中，打卡模块通过内部服务调用方式访问，例如在 Java 项目中可以使用 Spring Cloud 的 Feign 客户端进行服务间的远程调用，接口方法签名为 boolean authenticateUser(String username, String password)，返回 true 表示验证成功，false 表示验证失败。
  + ****用户信息获取接口****：打卡模块在打卡过程中需要获取用户的详细信息，如学生的班级信息用于考勤统计和报表生成，教师的授课课程信息用于确认打卡权限等。用户管理模块提供用户信息获取接口，根据用户 ID 返回用户的完整信息数据结构。接口方法签名为 User getUserById(String userId)，其中 User 为包含用户各种属性的实体类，如 { "userId": "用户唯一标识", "userName": "用户名", "userType": "用户类型（学生/教师）", "relatedData": { "studentClass": "学生班级", "teacherCourses": [ "课程 1 ID", "课程 2 ID",... ] } }。这样的接口设计实现了模块之间的低耦合和高内聚，提高了系统的可维护性和扩展性，使得打卡功能能够准确、高效地依赖用户信息进行业务逻辑处理，保障整个上课打卡系统的稳定运行。
* ****打卡模块与课程管理模块接口****：
  + ****课程信息查询接口****：在打卡操作时，打卡模块需要查询当前课程的详细信息，如课程名称、上课时间、上课地点等，以验证打卡的有效性和准确性（例如判断打卡时间是否在课程规定的时间段内，打卡位置是否与上课地点相符等）。课程管理模块提供课程信息查询接口，根据课程 ID 返回课程的详细信息对象。接口方法签名为 Course getCourseById(String courseId)，Course 实体类包含课程的各种属性，如 { "courseId": "课程 ID", "courseName": "课程名称", "classTime": "上课时间", "classroom": "上课地点", "teacherId": "授课教师 ID" }。打卡模块通过调用该接口获取课程信息，与打卡操作中的实际数据进行比对，从而确保打卡数据的合理性和真实性，有效防止学生的误打卡或作弊打卡行为，同时也为后续的考勤统计和分析提供了准确的数据基础，有助于学校对教学秩序的规范管理和教学质量的提升。
  + ****课程状态更新接口****：当课程信息发生变化（如课程时间调整、地点变更等）时，课程管理模块通过此接口通知打卡模块更新本地缓存的课程相关数据，以保证打卡操作依据的是最新的课程信息。接口方法签名为 void updateCourseStatus(Course updatedCourse)，其中 updatedCourse 为包含更新后课程信息的对象。打卡模块接收到课程状态更新通知后，及时更新本地的课程数据缓存，确保后续打卡流程不受课程信息变更的影响，维持系统的一致性和稳定性，避免因课程数据不一致导致的打卡错误和管理混乱，提高了系统对课程信息变化的适应性和响应能力，保障了学校教学活动的正常开展。