- 仿郑州大学教务管理系统
  - 前端
    - Vue+Vite
  - 后端
    - Springboot
      - 登录模块
        - 根据不同的账号 登录进入不同的页面 涉及到的表结构 user【拦截器中可以去掉login页面】
        - 角色访问控制 (RBAC)【AuthController】

模型	全称	特点
RBAC0	基础模型	用户 ↔ 角色 ↔ 权限
RBAC1	层次模型	角色可以继承（如：管理员 > 教师 > 学生）
RBAC2	限制模型	加入约束（如时间、部门）
RBAC3	综合模型	结合层次与限制

          - 其他权限访问控制：基于对象的访问控制(OBAC)、基于角色的访问控制(RBAC)、基于任务的访问控制(TBAC)和基于属性的访问控制(ABAC)
          - 项目采用RBAC0基础模型 角色分为管理员（AdminController） 教师（TeacherControlller） 学生（StudentController）
            - admin：学生管理，教师管理，课程管理，公告管理，选课任务管理，缴费管理
            - 学生
              - 选课模块，缴费系统
            - 老师
              - 考试管理，课程管理，公告管理
          - 通过自定义注解 @RequireRole + Spring AOP实现细粒度角色访问控制实现解耦，在AOP中定义切点拦截使用自定义注解的方法 @Before 方法执行前进行权限检查和 @around

过滤器 拦截器 AOP

对比项	AOP	拦截器（Interceptor）
作用层级	作用于 方法级别（Service、Repository等）	作用于 请求级别（Controller层）
触发时机	Spring 容器管理的 Bean 方法执行前后	HTTP 请求进入 Controller 前后
底层原理	通过 动态代理（JDK / CGLIB）实现	基于 Servlet 过滤链 实现
典型用途	日志记录、事务管理、性能监控、权限注解	登录验证、请求日志、跨域处理
是否依赖 Spring MVC	❌ 不依赖	✅ 依赖 Spring MVC
可拦截的目标	任意被 Spring 管理的 Bean 方法	仅 Controller 层的请求
实现方式	注解 + 切面类（`@Aspect` + `@Around`）	实现接口 `HandlerInterceptor`

🧠 三、通俗理解（用个比喻）

AOP 是在"方法调用"的细胞加码子，
拦截器是在"请求进入 Controller"的层面加码子。↓

        - Spring Security + JWT 实现无状态认证 用户登录后服务器发送JWT TOKEN 客户端保存token 每次请求携带token

✻ 三、JWT 无状态认证的核心优势

对比点	Cookie + Session	JWT（无状态）
存储方式	服务器保存 Session 状态	客户端保存 Token
服务器压力	Session 占内存，用户多时压力大	不保存状态，扩展性强
分布式支持	Session 需要共享（Redis 或 Sticky Session）	任意节点可验证 Token，天然支持微服务
跨域支持	Cookie 默认同源策略，不方便前后端分离	Token 可放在 Header，跨域友好
性能	每次都查 Session（I/O 操作）	只做签名验证（纯计算）
安全性	Cookie 可能被伪造/窃取	JWT 有签名（可加密），可防篡改
注销/过期控制	可立即销毁 Session	需额外机制（如 Redis 黑名单）

          - 传统的是基于cookie和session进行认证。用户登录后 服务器发送sessionID存在浏览器cookie中，之后用户的每次请求携带cookie。 服务端需要保存session，无状态是指服务器不报存用户登录信息
        - 1.token签发之后不会主动吊销--redis黑名单机制可以实现动态吊销【也可也在数据库中设置用户活跃状态】 2.单token可能发生频繁登录问题【超时时间15min-2h】 ---双token机制，短token【15min-2h】和长token【7天】放入redis中，短token过期再找长token生成，长token过期用户重新登录
      - 高并发抢课模块
        - 管理员发布限时抢课，学生端进行抢课，首先提交预选课信息（购物车），定时任务扫描，当抢课时间开始时，进行抢课。
        - MD5+UUID动态路径加密 防止接口被恶意刷流量
          - MD5是一种hash加密算法 全局唯一标识符（Universally Unique Identifier），几乎不重复。常用于生成唯一文件名或路径
        - 令牌桶方法限流【固定窗口法，滑动窗口法...】创建一个工具类：SimpleTokenBucket
          - 桶：最大容量，令牌总数和生成速率，时间窗口 支持一定的突发流量
        - 检验path的合法性 将请求携带的path和redis中的path进行比较
        - 检查是否重复抢课【三级缓存 一级本地缓存caffeine/guava 二级缓存redis 三级缓存数据库】任何一级数据库发现重复 即抛出异常，更新缓存
        - redis预先减少库存 双重判断+库存预减【事务 redission分布式锁+lua原子操作保证数据一致性】
          - 双重判断是首先判断库存是否>0,然后执行库存减少操作 返回1表示成功 返回0表示失败 事务需要回滚
        - 通过RocketMQ异步入队 减少数据库压力 实现非阻塞抢课
          - 生产者生产半消息-执行本地事务-返回事务状态-broker根据状态判断是否往队列中投递-消费者受到并处理选课逻辑
      - 支付模块
        - 调用支付宝支付功能模块
      - AI选课助手
        - 调用Spring AI 中的chatClient接口 实现ollama本地大模型服务
          - 用户发送信息-调用OllamaChatClient--向本地ollama服务发送请求【路由分派】1.普通回复 2.选课相关问题回复（将业务数据作为上下文，与设计好的prompt进行组装成提示词，调用本地大模型进行专属业务回复）
      - 通过SpringDoc创建API文档方便团队合作 https://blog.csdn.net/xiyang_1990/article/details/140739948
  - 数据库
    - MySQL
      - MySQL设计
        - 登录模块users student teacher admin
          - users负责验证登录信息 student teacher admin进行数据分层
          - users.role属性实现RBAC角色访问控制
        - 抢课模块course，pre_course 【course.id是pre_course的外键】selections选课记录,sekills秒杀任务
          - course，selections分开设计实现多对多关系
          - 子主题
  - 中间件
    - RocketMQ，Redis