广东第二师范学院

2024-3110-001

本科毕业论文（设计）

一种学生在线实验平台设计与实现

学生姓名：郭浩蓝

学　　号：21551117077

院（系）：计算机学院

专　　业：网络工程

指导教师：张谦　副教授

提交日期：2024 年 04 月 20 日

类　　型：毕业论文☐毕业设计☑

本科毕业论文（设计）诚信声明

本人郑重声明：所呈交的本科毕业论文（设计），是本人在指导老师的指导下，独立进行研究工作所取得的成果，成果不存在知识产权争议，除论文（设计）中已经注明引用的内容外，本论文（设计）不含任何其他个人或集体已经发表或撰写过的作品成果。对本论文（设计）的研究做出重要贡献的个人和集体均已在论文（设计）中以明确方式标明。本声明的法律结果由本人承担。

学生签名：

年　　月　　日

本科毕业论文（设计）使用授权说明

本人完全了解广东第二师范学院关于收集、保存、使用毕业论文（设计）的规定，即：

1.按照学校要求提交毕业论文（设计）的印刷本和电子版本；

2.学校有权保存毕业论文（设计）的印刷本和电子版本，并提供目录检索与阅览服务，在校园网上提供服务；

3.学校可以采用影印、缩印、数字化或其它复制手段保存毕业论文（设计）。

本人同意上述规定。

学生签名：

年　　月　　日

摘 要

随着教育信息化的不断推进，在线实验平台已成为突破传统实验室时空限制、满足大规模教学需求的重要技术手段。然而，传统实验环境常依赖于物理硬件设备和固定场地，存在资源利用率低、扩展性差、管理复杂等问题，难以满足现代教育对灵活性、效率及资源优化的要求。为了解决这些问题，本文提出了一种基于 Kubernetes（K8S）和容器化技术的在线实验平台设计方案，设计基于 Kubernetes 的容器化在线实验平台架构，支持实验环境的动态创建、回收，提升了资源利用效率和平台的可维护性。并引入班级管理、课程管理、实验任务调度与权限控制等功能模块，形成了从用户操作到实验执行的完整流程。系统前后端部分采用分离架构，前端基于 React 实现动态交互界面，后端基于 Go 语言结合 Kafka 消息队列，实现任务异步处理和高并发响应。本文为在线实验教学平台的实现提供了一种可行的技术路径，具有一定的实践参考价值。

关键词：容器化技术；在线实验平台； Kubernetes

Abstract

With the advancement of educational informatization, online lab platforms have emerged as a crucial technology overcoming traditional labs' spatial and temporal constraints for large-scale teaching. However, traditional labs often depend on physical hardware and fixed locations, causing low resource utilization, poor scalability, and complex management, thus failing modern education’s demands for flexibility and efficiency. To address these issues, this paper proposes an online lab platform based on Kubernetes (K8s) and containerization. The design supports dynamic creation and recycling of experimental environments, enhancing resource utilization and maintainability. Integrated modules include class and course management, experiment scheduling, and permission control, providing a complete workflow from user operations to experiment execution. The system adopts a decoupled frontend-backend architecture, employing React for dynamic interaction and Go with Kafka message queues for asynchronous tasks and high concurrency. This paper offers a practical technical reference for implementing online experimental teaching platforms.

Key Words：Containerization Technology; Online Laboratory Platform; Kubernetes

目 录

一种学生在线实验平台设计与实现

在数字化转型浪潮的推动下，教育信息化正不断重塑传统教学的边界。远程教育作为典型代表，借助互联网突破了时间与空间的限制，让学习者能够以灵活多样的方式获取知识。早期远程教育主要面向非在校群体，依托电视和计算机等终端实现单向内容传播，其优势在于资源共享和学习自主性，却在实践教学环节暴露出短板：理论课程尚可轻松迁移线上，实验课程因缺乏动手操作场景难以落地。以计算机网络、物理、化学等学科为例，实验不仅验证理论，更关乎科学思维和实践能力的培养；但设备分散、投入不足等问题，让远程教育长期陷入“重理论、轻实践”的困境，制约了教学质量提升。

为突破瓶颈，虚拟实验成为研究热点。最初的软件模拟以预设流程呈现实验现象，交互性与真实性有限；随后出现的交互式虚拟实验通过实时反馈提升沉浸感，却因课程定制开发成本高、功能单一而难以推广。近年来，物联网与远程控制技术推动“真实设备云化”，允许用户在线操控实体仪器，虽然增强了真实感，却带来资源冲突、能耗高和维护复杂等挑战，尤其在大规模并发访问下，稳定性和利用率问题更加突出。

云计算与容器化技术的兴起为在线实验平台提供了全新思路：云平台的弹性扩展可动态满足多用户需求，Docker 等容器技术通过标准化封装与秒级部署，解决了传统虚拟机资源占用高、环境配置繁琐的难题。进一步借助 Kubernetes（K8S）进行容器编排，可实现实验资源的自动调度与负载均衡，大幅提升资源利用率和系统可扩展性，使在线实验平台向“高并发、低延迟、易维护”迈进。基于此，本课题聚焦于基于 K8S 的分布式在线实验平台设计，旨在通过技术创新实现资源高效管理、实验环境动态调度和用户体验优化，为教育信息化背景下实验教学的高质量发展提供有力支撑。

1.前言

1.1 研究背景

随着教育信息化的迅猛发展，在线教育与远程学习的需求呈现出指数级增长

趋势。近年来，在线课程与虚拟实验的兴起，使得虚拟实验平台成为教育技术领域的重要发展方向。然而，传统实验教学模式严重依赖实体实验室的硬件设备与固定场地，这不仅推高了学校基础设施的投入成本，也导致教学效率受制于时空限制。在计算机科学、物理、化学等实验密集型学科中，传统实验室在规模化教学环境下暴露出资源利用率低、学习方式单一和灵活性不足等问题，进一步加剧了教育资源分配不均衡现象。

在此背景下，构建一个具备高效性、灵活性与扩展性的在线实验平台已成为当前教育技术领域亟待解决的关键问题。现有虚拟实验平台普遍存在部署复杂、扩展能力不足、资源利用率不高等缺陷，严重制约了其在大规模场景下的广泛应用。随着虚拟化技术的快速演进，基于云端构建具备动态资源调度能力的实验环境逐渐成为现实。这类平台不仅能突破传统实验的时空限制，支持实验任务的即时创建与灵活部署，同时还能为教师提供便捷的实验设计与管理工具，大幅提升整体教学效率与学习体验。

作为一个开源的容器编排系统，Kubernetes（K8s）凭借其卓越的资源管理、负载均衡和弹性扩展能力，逐渐成为构建在线实验平台的首选技术框架。通过容器化技术将实验环境与底层基础设施解耦，K8s 实现了实验资源的高效打包和动态调度。与传统虚拟化解决方案相比，容器技术通过共享主机内核显著降低了资源使用，并通过二级启动功能提高了实验环境部署的效率。结合 K8s 的分布式架构，该平台能够在多用户场景下实现动态资源分配，确保高并发情况下服务的稳定性和响应速度，从而为规模化的实验教学提供了坚实的技术支撑。

表 1.1　虚拟实验平台发展历程

时间	软件可靠性研究发展状况	问题
20 世纪 80-90 年代	早期虚拟实验主要依赖计算机图形学和编程技术，通过软件模拟实验现象，例如使用 C++或 LabVIEW 开发单机版实验程序。	环境封闭、交互性差、难以适应复杂实验需求。
21 世纪初	物联网（IoT）与远程控制技术的成熟（如 MQTT 协议、WebSockets）推动了“远程操作真实实验设备”的模式。	设备共享冲突（如多用户同时操作）、能耗高、维护成本大，且扩展性不足。

2015 年至今	云计算与容器化技术（Docker、Kubernetes）的普及，为虚拟实验平台的规模化部署提供了技术支持。	资源冲突与成本：大规模并发访问下，动态资源调度仍需优化。实验真实性：虚拟环境与真实场景的差异(如传感器精度、操作复杂度）需进一步缩小。教育适配性：如何根据学科特点设计差异化的虚拟实验系统。

从表 1.1 可以看出，虚拟实验平台发展主要是从传统的软件模拟转变为容器化分布式技术中。

1.2 国内外研究现状

1.2.1 国外研究现况

在服务管理、资源调度及实验环境构建领域，Kubernetes（K8S）凭借其容器编排与自动化能力展现出显著的技术优势。Kharade[1]等人（2022）通过对比分析不同面向服务的系统管理方案在远程与虚拟实验环境中的应用效果，证实了 K8S 的突出价值。研究指出，K8S 的容器编排机制与自动化运维功能有效提升了实验平台的资源管理效率与系统稳定性，尤其在动态资源分配、故障自愈及弹性扩展方面表现优异，为复杂在线实验场景提供了可靠的技术保障（Kharade et al., 2022）。Xue[2]等人（2023）基于 K8S 开发了大数据实验教学平台“Kube-CC”。该平台通过容器化技术实现了实验环境的快速部署与资源隔离，支持学生在独立容器中完成大数据实验操作。研究结果显示，Kube-CC 通过自动扩展机制有效应对高负载场景下的性能需求，其并发处理能力和资源利用率均显著优于传统方案，为大规模在线实验平台的构建提供了可复用的技术范式（Xue et al., 2023）。Cai[3]等人（2023）聚焦数字校园背景下的教学资源管理，提出了一种基于 K8S 的容器化教学资源平台。该平台通过集成 K8S 的动态调度与自动扩缩容功能，实现了实验环境的灵活部署与智能化管理。研究验证了该平台在资源利用率和响应速度上的提升效果，证实了 K8S 在教育场景下对动态资源需求的高效适配能力（Cai et al., 2023）。

1.2.2 国内研究现状

在国内，容器化技术和 Kubernetes 在教育领域的研究也呈现出快速发展的趋势，部分高校和研究机构已在实验教学中开展相关探索。何羽等人[4]（2024）提出了一种基于 Kubernetes 的天基云平台可靠性方案，验证了 Kubernetes 在高性能计算和自动化管理中的优势。研究表明，通过 Kubernetes 的容器编排功能和自动扩展能力，可以有效解决实验环境在高并发场景下的资源瓶颈问题。蓝宗侦[5]（2023）研究了基于 Kubernetes 的融媒体云平台，展示了在复杂集群环境中 Kubernetes 资源调度的高效性和可扩展性。研究团队通过在平台中引入动态负载均衡和容器自动扩展技术，显著提升了平台在复杂环境下的运行效率和资源利用率。陈应虎[6]等人（2024）在跨平台应用开发的实践中，利用容器化技术实现了高灵活性的分布式实验环境部署，进一步提升了在线实验平台的适用性。通过在 Kubernetes 集群中引入分布式存储和多级缓存机制，研究团队解决了实验环境在多用户场景下的存储瓶颈和响应延迟问题。张沛[7]（2024）在高职院校的实验平台研究中，分析了 Docker 容器在实验环境中的应用优势，提出了在教育场景中优化资源配置和系统管理的策略。研究表明，基于 Docker 容器的实验环境在一致性和扩展性方面优于传统虚拟机环境，能够更好地满足学生的个性化实验需求。林德[8]祥等人 (2025) 在智能建筑机器人云平台设计中，利用 Kubernetes 的容器化特性，实现了实验环境的动态分配和高效管理。研究团队通过在 Kubernetes 集群中引入多层次权限管理和日志跟踪功能，提升了平台的安全性和系统管理能力。

1.2.3 研究存在的问题与挑战

尽管国内外在基于 Kubernetes 的在线实验平台研究中取得了积极进展，但当前研究仍存在以下问题和挑战还是有的，首先是资源调度与系统稳定性
在多用户并发场景下，实验平台的资源调度和负载均衡机制尚不完善。现有研究在实验环境的动态分配和集群扩展能力方面存在不足，导致在高并发访问场景下系统响应速度下降和资源利用率不均衡。其次，前端交互与用户体验

许多现有在线实验平台在用户界面和交互设计方面存在不足，学生在实验操作中的直观性和实时性需求尚未得到充分满足。特别是在实验环境的状态监控和实验结果展示方面，用户体验仍有较大提升空间。除此，安全性与管理性
部分平台在权限控制和日志管理等方面存在不足，难以满足多用户环境下的系统安全性和管理性需求。实验平台在多用户同时访问时，可能出现权限泄露和数据同步问题，影响平台的整体安全性和稳定性。

1.3 文章结构

本文围绕学生在线实验平台的设计与实现展开研究，全文共分为五个主体章节，具体结构安排如下：

第一章： 前言。本文描述了研究背景和主题的重要性，并梳理了虚拟实验平台的技术演变；通过分析国内外当前的研究现状，指出了现有平台在资源调度、用户体验和系统安全方面的不足之处，并进一步明确了本文的研究目标和技术路线。

第二章： 系统需求分析及相关技术介绍。基于教育信息化场景开展多维度需求分析，涵盖角色管理、班级管理、课程管理及实验管理等核心模块；系统论述 Kubernetes 容器编排、React 前端框架、Go 语言高并发处理等关键技术原理，完成技术选型论证。

第三章：系统设计方案。提出了一种基于 B/S 架构的分布式系统设计方案，以构建包括用户层、服务层和数据层的三级系统架构；强调了数据库的模块化设计策略、实验资源分配以及其他核心数据表结构及其关系。

第四章：系统实现。详细描述了各功能模块的实现过程，基于 Kubernetes 的容器化实验环境部署方案、React 动态交互界面的组件化开发、Go 语言与 Kafka 集成的异步任务处理机制，并通过系统测试验证平台在高并发场景下的性能表现。

第五章，系统测试。对整个系统的模块进行测试，说明测试方法和结论。

第六章，结论与展望。总结了本文的结果。本文最后附有参考文献、系统核心代码片段（附录）和致谢。

2. 系统需求分析及相关技术介绍

2.1 系统整体需求分析

2.1.1 角色管理

依据学生在线实验平台开发需求，并且结合相关业务操作明确了涉及到的系统的用户角色。系统用户如表 3-1 所示，本系统的用户主要是学生、教师、管理员等三类角色：

表 2.1 角色职责

角色	职责
学生	查看，修改个人班级信息，课程信息，以及根据课程完成对应实验
教师	查看，修改班级学生信息，课程信息，布置课程作业
管理员	查看，修改学生，教师班级信息，课程信息，实验信息

2.1.2 班级管理需求

班级管理是实验教学平台的基础功能模块，主要涉及学生和教师之间的组织管理关系。系统需提供完整的班级信息管理功能，以支持动态调整和跨班级的教学活动。

2.1.2.1 学生班级管理

为了满足学生在实验教学中的个性化需求，系统需要具备以下功能：支持学生班级信息的添加、删除、修改和查询，并确保班级信息的动态更新和完整性。系统应能够根据学生的班级属性自动加载相关课程和实验内容，并确保课程和班级信息的同步与一致性。学生需要能够通过平台界面查看他们的班级、相关课程及实验安排，以提高他们对课程体系的整体理解和学习规划能力。学生班级管理的要求如图 2.1 所示。

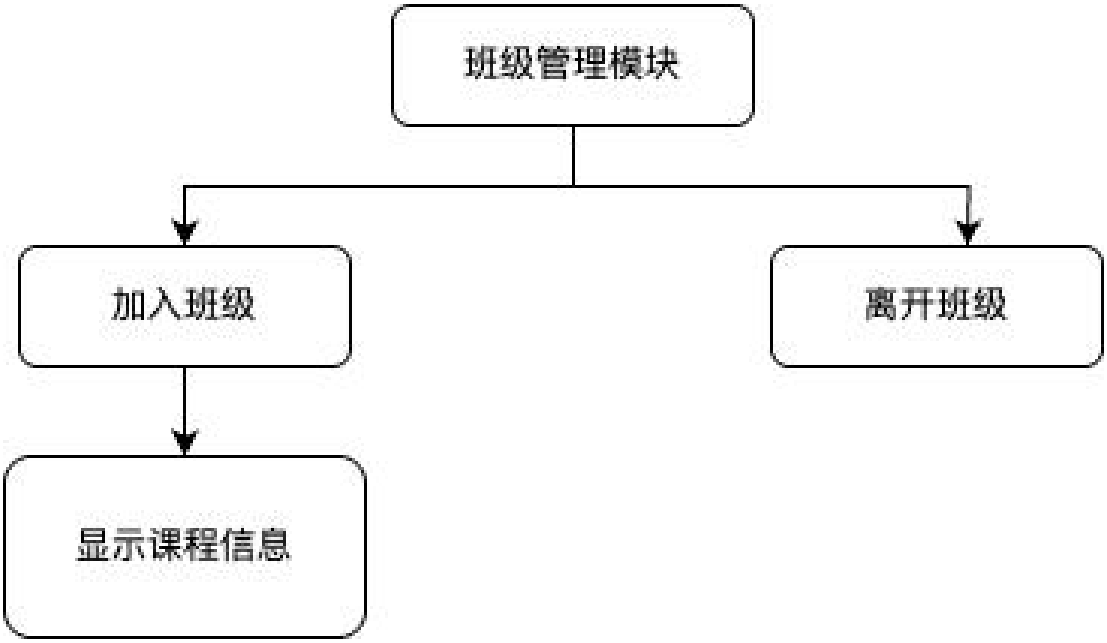

图 2.1 班级管理模块（学生）需求

2.1.2.2 教师班级管理

为了提高教师在班级管理和教学资源配置方面的灵活性，系统需要支持以下功能：教师应该能够通过系统查看和管理他们负责的班级信息，包括调整班级成员、修改班级属性等。系统需要支持教师和班级之间的动态绑定，以保证教学资源配置的灵活性和适应性。教师可以根据课堂特点和学生情况调整实验教学内容，提高教学的针对性和有效性。教师班级管理要求如图 2.2 所示。

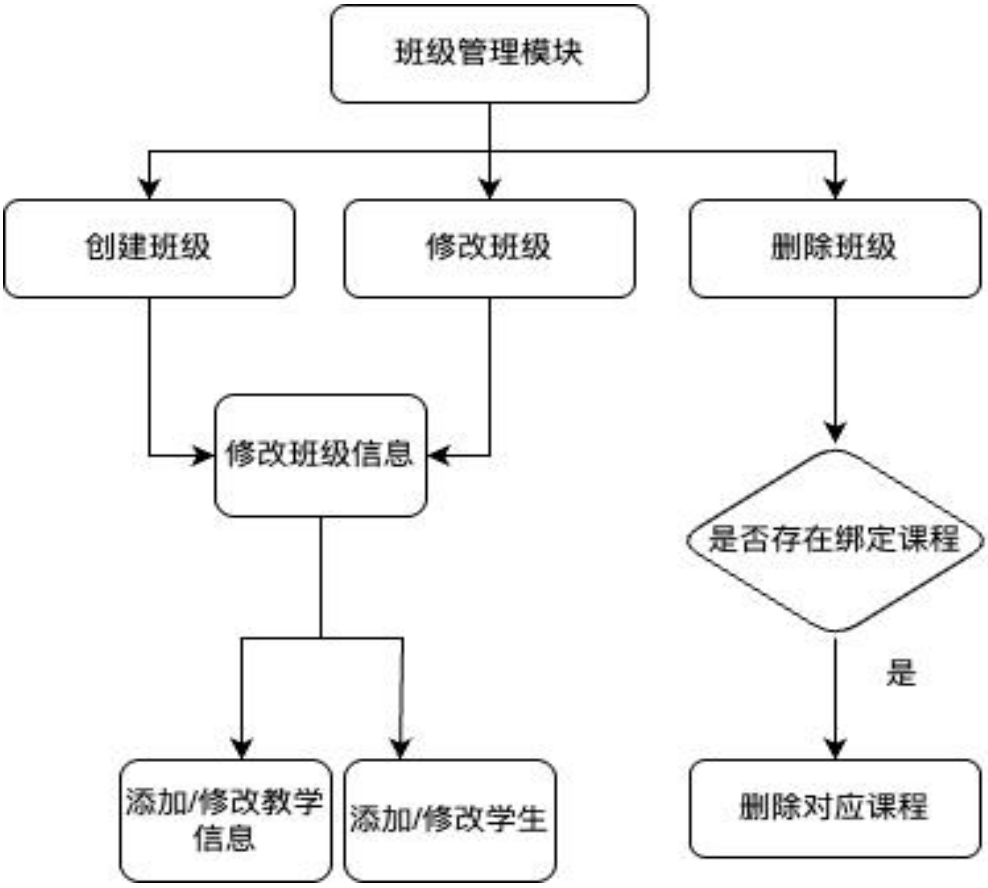

图 2.2　班级管理模块（教师）需求

2.1.3 课程管理需求

课程管理是实验平台的核心模块，涉及课程内容配置、课程与班级的关联管理以及学生的学习路径管理。系统需提供动态的课程管理功能，确保课程配置的灵活性和一致性。

2.1.3.1 课程配置管理

为保证教学内容的动态适配，系统需要满足以下功能需求：系统需要支持课程的增删、修改、查询，保证课程信息的动态更新。教师可以根据不同班级的特点和教学目标灵活配置课程内容，并与班级和学生动态绑定。系统应能够根据学生的学习进度和课程配置自动推荐实验内容，以提高课程资源的利用效率。课程配置管理要求如图 2.2 所示。

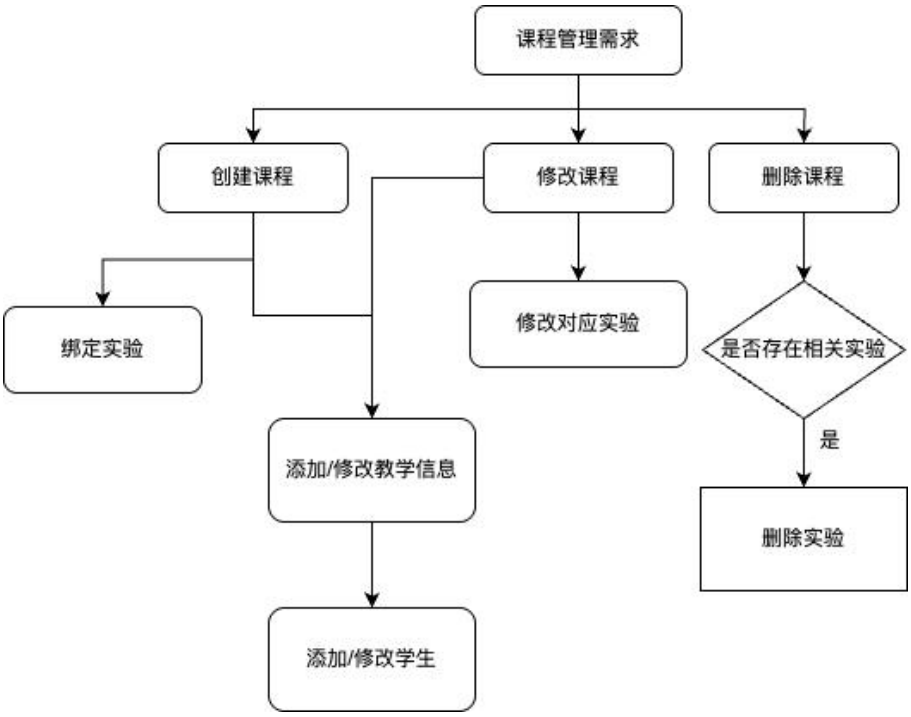

图 2.2 课程配置模块需求

2.1.3.2 学生课程管理

为了满足学生的个性化学习需求，系统需要具备以下功能：学生可以通过系统界面查看所选课程和相应的实验内容，了解学习进度和实验安排。系统需要支持根据学生的班级属性自动配置相应的课程资源，以保证课程内容和班级信息的动态匹配。系统需要记录学生的学习轨迹和实验进度，以方便教师分析教学效果并进行动态调整。

2.1.4 实验管理需求

实验管理是实验平台的关键模块，涉及实验环境配置、实验任务调度和权限管理。系统需依托 Kubernetes（K8S）的容器编排能力，构建高效、灵活和可扩展的实验环境。

2.1.4.1 实验内容管理需求

为了满足实验教学中多样化的实验内容配置需求，系统需要具备以下功能：系统需要支持不同实验内容的动态配置和管理，以满足不同课程和学生的实验需求。教师可以根据课程内容灵活配置实验内容和实验参数，保证实验内容与教学目标的一致性。系统应支持实验内容的快速更新和扩展，以利于根据教学需要实时调整实验计划。

学生仅可查看所属课程的已开放实验基础信息（名称、描述、课程关联、开放时间），访问实验文档及视频资源，实时查看个人实验进度与成绩反馈，禁止进行实验创建、修改、删除及权限配置等管理操作，系统自动隐藏非关联实验及未开放内容。其操作流程如图 2.3 所示。

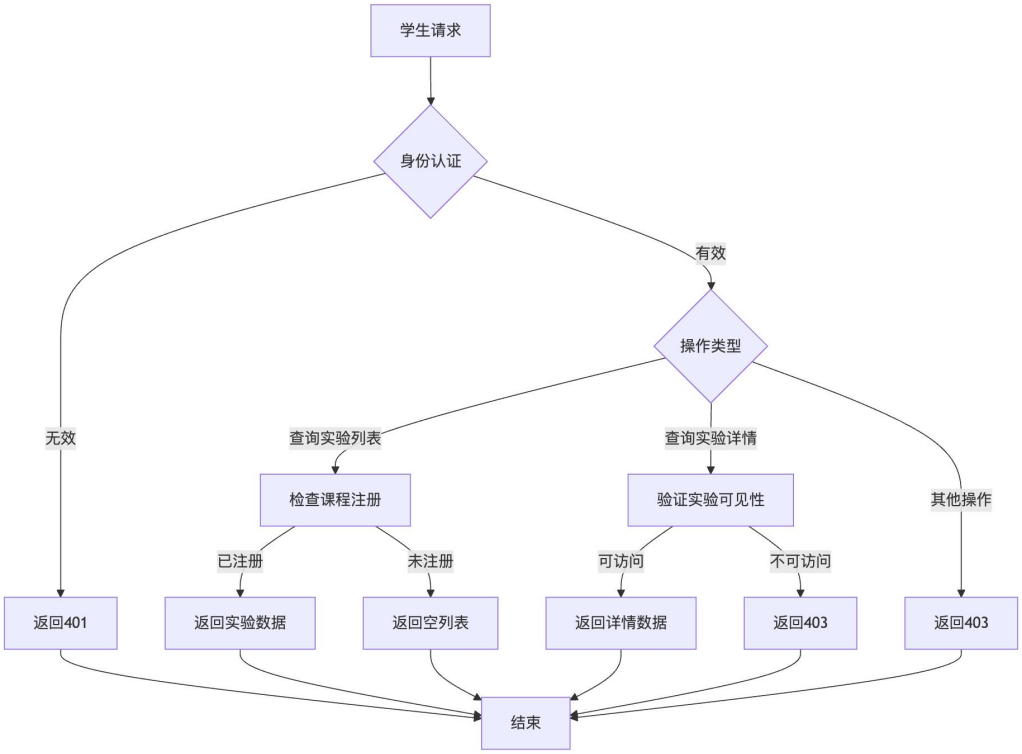

图 2.3　实验管理（学生）流程图

而教师可自主创建、编辑和删除所属课程的实验内容，配置实验关联资源（虚拟机模板、指导文档），设置实验开放时间与访问权限，实时查看学生实验参与进度及成绩分布，支持批量操作与实验状态管理（暂停/恢复/归档），系统自动关联课程数据并同步更新教学资源。其操作流程如图 2.4 所示。

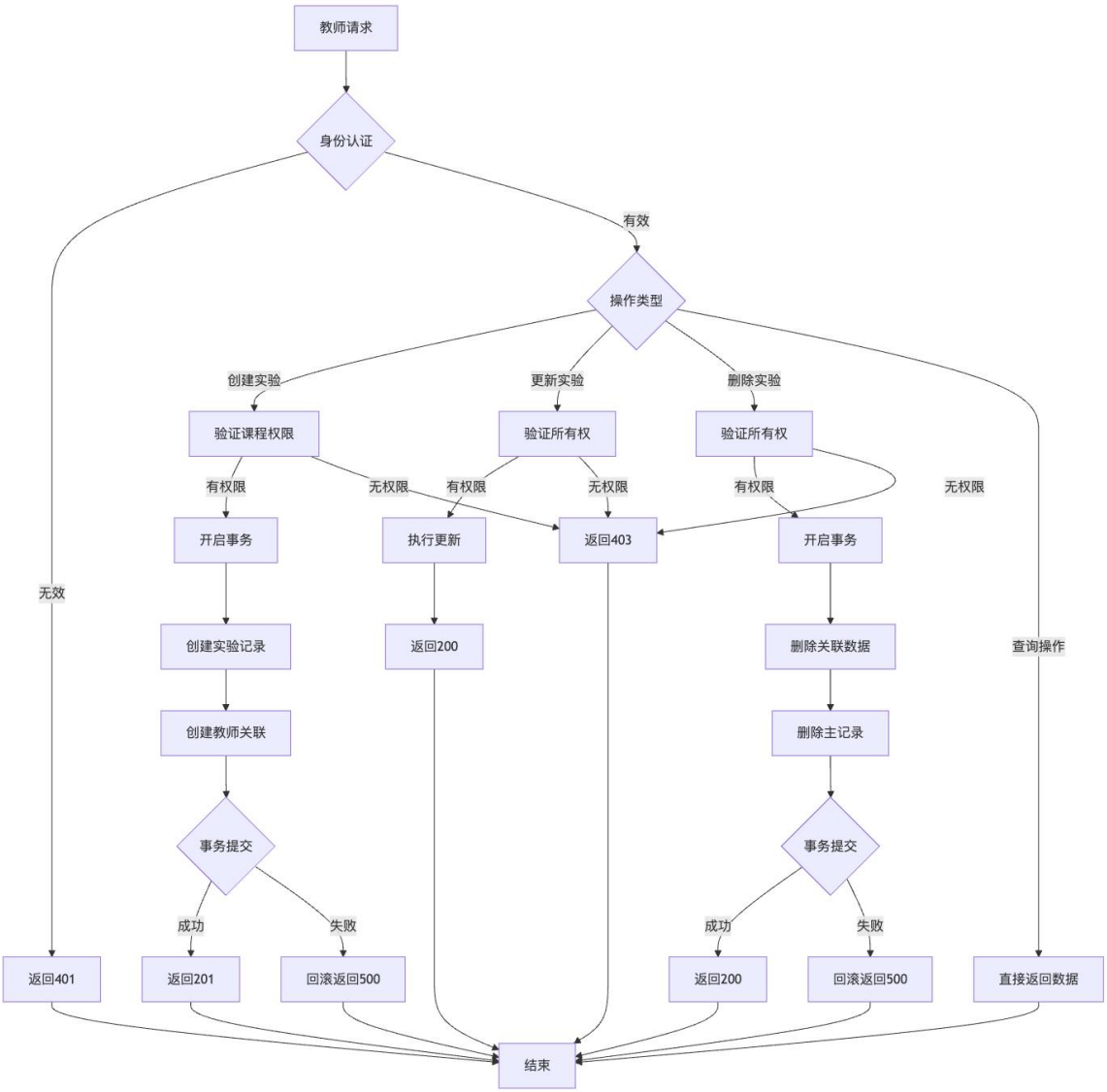

图 2.4　实验管理（教师）流程图

2.1.4.2 实验环境需求

为了保证实验环境的动态创建和高效调度，系统需要具备以下能力：基于 Kubernetes（K8S）技术，系统需要能够动态创建、销毁和扩展实验环境，以满足高并发访问场景下的资源调度需求。通过容器化技术，将实验环境封装为独立的容器镜像，保证实验环境的隔离性和一致性。系统需要具备实验环境的监控和日志管理功能，支持实时查看实验状态，提高实验环境的可作性和可维护性。为了满足上述需求，建议的主机配置要求如表 2.2 所示。

表 2.2 Docker 宿主机配置需求表

配置项	最低要求	推荐配置	说明
CPU	4 核	≥ 8 核	多容器并发调度，需保证调度与负载均衡效率
内存	8 GB	≥ 16 GB	容器运行需占用内存，实验环境并发数越高内存需求越大
磁盘容量	100 GB SSD	≥ 500 GB SSD	存储容器镜像、实验数据及日志，推荐使用 SSD 提高 I/O 性能
网络带宽	1 Gbps	≥ 10 Gbps	确保高并发实验访问下的网络稳定性与传输效率
操作系统	Ubuntu 20.04 / CentOS 7	Ubuntu 22.04 LTS / Rocky Linux 9	推荐使用支持长期维护的系统版本
容器运行时	Docker / containerd	containerd	推荐轻量、兼容性强的运行时
虚拟化支持	支持 VT-x / AMD-V 虚拟化指令集	已启用虚拟化	支持底层虚拟化容器创建
Kubernetes 版本	≥ v1.24	最新稳定版	保证兼容最新容器调度与监控功能
其他组件	-	配置 cAdvisor、Prometheus、Grafana	用于监控、可视化与资源审计

2.1.4.3 权限管理与安全性

为保证实验环境和课程内容的安全性，系统需要根据用户角色（学生、教师、管理员）设置不同的访问权限，以保证用户作权限的精细化管理。通过日志管理和权限控制，系统应该能够记录用户的作轨迹，以保证系统运行的可追溯性和安全性。平台需要支持多级权限设置，以保证实验内容和教学资源的安全共享和访问控制。

2.1.4.4 高可用

为确保平台持续对外提供稳定可靠的服务，在容器调度与服务编排方面，本平台使用 Kubernetes 对容器集群进行统一管理，通过 ReplicaSet 机制保证各类服务始终维持预定的副本数量，即使节点发生故障也能够快速恢复并持续提供服务。Kubernetes 的高可用性归功于 Etcd 提供的分布式一致存储。etcd 本身具有

无单点故障的分布式架构，保证了控制面的稳定运行。在消息传输和事件流向层面，平台集成 Kafka 消息队列作为核心异步通信组件，通过分布式分区机制实现高吞吐量的数据处理任务。此外，为保证 Kafka 集群的可靠性，平台引入了 Zookeeper 作为协调服务来管理 Kafka 节点的选举和状态同步。Zookeeper 也部署在多节点集群模式，有效避免了单点故障的风险，从而建立了一个高可用、高容错的消息传输和事件处理系统。

2.2 系统相关开发技术

2.2.1 概述

本课题在线实验平台采用前后端分离架构，整合 React、Go、Kubernetes、MySQL、Zookeeper 与 Kafka 等关键技术，构建高性能、高可用的实验服务系统。前端基于 React 实现组件化开发与高效渲染，提升用户界面的交互性与响应速度；后端使用 Go 语言开发分布式虚拟机管理系统，保障高并发场景下的稳定运行。Kubernetes 提供容器调度与自动伸缩能力，支持实验环境的动态创建与管理。MySQL 用于存储用户数据与实验信息，确保数据一致性与高效访问。Kafka 负责实验任务的数据传输，Zookeeper 提供集群协调与节点选举支持，共同构建可靠的消息通信机制，增强平台整体的扩展性与稳定性。

2.2.2 前后端技术概述

前端使用 Facebook 开源的 JavaScript 库 React 来实现界面的动态渲染和高效交互。React 倡导组件化的开发模式，将复杂的用户界面拆分为多个独立且可复用的功能模块，大大提高了开发效率和代码可维护性。此外，React 引入了虚拟 DOM 技术，有效降低了更新数据状态时页面的性能开销，保证了实验平台交互的流畅性和快速响应性。在本项目中，基于 React 构建的前端界面使用户能够实时查看实验状态并方便地进行实验控制作，显著提高了平台的易用性和用户体验。后端是使用 Google 推出的 Go （Golang） 语言开发的。Go 语言具有简洁明了的语法设计、出色的并发处理能力、高效的内存管理机制，特别适合构建高并发、分布式的系统架构。该平台使用 Go 实现分布式虚拟机管理系统。通

过 Goroutine 和 Channel 等并发编程原语，高效处理用户请求和资源调度任务，保证系统在高负载条件下的稳定运行，有效提升实验环境的自动化管理能力和整体性能。

2.2.3 K8S 技术概述

Kubernetes（K8S）是一个开源容器编排平台，旨在自动化应用容器的部署、扩展与管理。通过容器化技术，K8S 提供了高效的资源调度、自动扩展和故障恢复机制，是构建大规模分布式系统的理想选择。K8S 可以动态调整容器实例数量，根据系统负载保障平台的稳定性。结合 Go 语言和 React 框架，K8S 在本课题中为实验平台提供了强大的资源管理能力和灵活性。

2.2.4 数据库技术概述

MySQL 是广泛使用的开源关系型数据库管理系统，由美国 Sun 公司开发，后被甲骨文公司收购。因其可靠性和易用性而被广泛应用于各类应用平台。MySQL 支持事务管理、数据完整性保障以及高效的数据检索。在本课题中，MySQL 用于存储和管理实验平台中的用户数据、实验配置及实验日志，保证了数据的一致性和高效访问。

2.2.5 分布式协调与消息通信技术

Zookeeper 是一个开源的分布式协调服务，由广泛应用于配置管理、命名服务及分布式系统中的同步控制等场景。其高效的协调机制在本课题中被用于管理 Kafka 集群的 Leader 节点选举，确保系统各节点间的一致性与高可用性。

Kafka 是一个开源的分布式消息队列系统，最初由美国领英公司开发，后捐献给了 Apache 软件基金会，其具备高吞吐量、低延迟、强持久性和容错能力，适用于处理大规模并发数据流，也是当下最热门的分布式消息队列系统。Kafka 承担了平台中实验任务与数据的异步通信任务，保障了消息在高并发场景下的稳

定可靠传递。通过 Zookeeper 的集群协调与 Leader 管理机制，Kafka 系统在运行过程中实现了更高的稳定性与数据一致性。

2.3 本章小结

本章重点介绍了学生在线实验平台的设计目标，从角色管理、班级管理、课程管理、实验管理等方面系统分析了平台的功能需求，明确了不同用户角色的核心作权限和平台应具备的业务能力。同时，结合当前虚拟化和分布式架构的发展趋势，提出了一种基于 Kubernetes 的实验环境解决方案，详细阐述了实验内容配置、容器化打包、权限控制、系统高可用性等关键需求，并给出了主机的硬件配置建议。

在技术选择方面，平台采用前后端分离的系统架构。前端使用 React 提供高效流畅的用户交互界面，后端使用 Go 构建稳定高效的分布式服务体系。此外，结合 Kubernetes 实现容器的自动编排和弹性扩展，使用 MySQL 实现可靠的数据持久化存储，使用 Kafka 和 Zookeeper 构建高可用的消息通信机制。这些技术的有机集成有效地保证了平台的高效运行、高并发处理能力和良好的可扩展性。

3. 系统设计方案

3.1 系统架构设计

本文采用分层架构设计，以模块化组件构建教育实验场景的技术支撑能力。基于 B/S 架构的访问模式，用户无需安装客户端，便可通过主流浏览器完成实验操作。在前端架构中，系统引入 MVVM 模式，实现接口逻辑和业务数据的解耦：视图层依赖 React 框架构建高度可复用的组件库，通过虚拟 DOM 提高用户交互响应的效率;视图模型层采用数据绑定技术，实现实验状态的实时同步，保证用户作和界面反馈的一致性;模型层由用 Go 语言编写的后端服务支撑，负责虚拟机编排、实验数据持久化等核心业务逻辑处理。后端系统使用云原生技术栈，实现灵活的资源调度和高可用管理。通过 Kubernetes API 连接容器化实验环境，构建涵盖虚拟机创建、实验过程监控、资源回收的全生命周期管理体系。Kubernetes 的 Pod 水平扩展机制可以根据实时负载自动分配计算资源，其内置的负载均衡策略保证了高并发访问下的服务稳定性。引入 Kafka 消息队列，将虚拟机启动、关机等耗时任务转换为异步作流程，避免阻塞前端请求，提升整体系统响应能力。在数据管理层面，使用 MySQL 关系型数据库，借助事务机制和索引优化来保证实验数据的完整性和高效查询。架构设计的亮点在于技术组件之间的高效协同和开放扩展能力。前后端分离模式使系统能够在交互体验和计算性能之间实现良好的平衡;Kubernetes 提供的声明式 API 支持基础设施的模板配置以及环境的快速复制和版本回滚;Kafka 在业务逻辑和资源调度之间构建了缓冲通道，显著提高了系统的吞吐量和异常隔离级别。如图 3.1 所示，各个模块通过标准化的接口协同工作，形成稳定高效的技术闭环，不仅增强了系统的可维护性，还为未来实验资源的弹性扩展、多模态设备的接入等场景提供了坚实的技术支持，为教育信息化的持续演进构建了开放的平台基础。

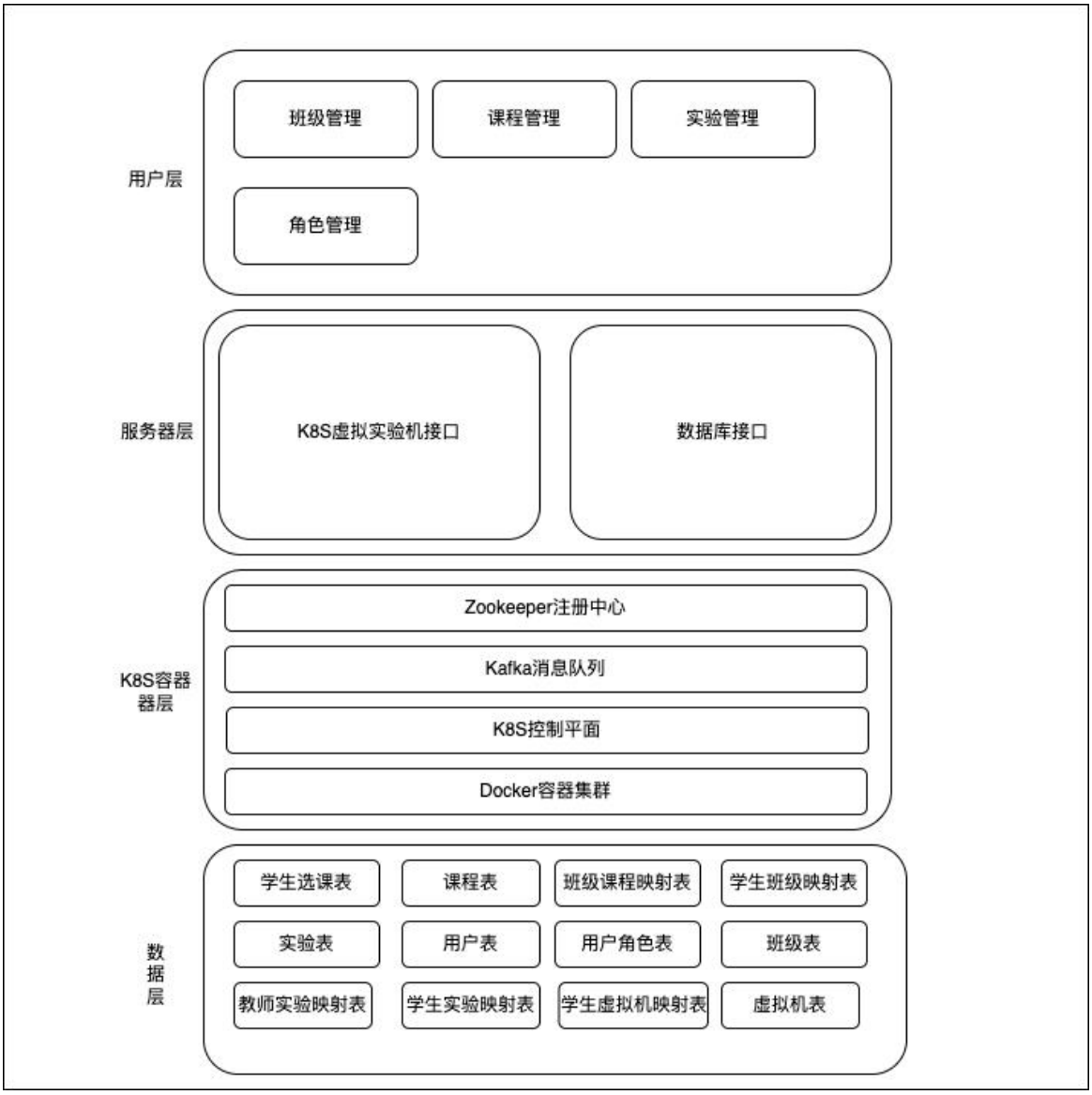

图 3.1 系统架构

3.2 数据库结构设计

3.2.1 数据库模块设计

本文的数据库设计围绕在线教育平台的核心业务场景，采用模块化设计思路，构建课程管理、课堂管理、实验管理、用户管理四大核心模块。遵循第三范式设计原则，通过合理的外键约束和索引优化来保证数据的一致性和查询性能。整体数据库结构设计如图 3.1 所示，相关模块分表设计如表 3.1，具体表结构设计如表 3.2-3.21 所示。

表 3.1 模块数据结构

模块	关键表	主要功能
用户体系	users、user_roles、user_profiles、student_informations	实现用户身份管理，区分学生、教师角色，支持扩展字段（如学号、联系方式）
课程管理	courses、course_chapters、course_assessments	管理课程基本信息、章节内容（含视频）及考核任务（考试/作业）
班级管理	classes、student_classes、teacher_classes	支持班级创建、学生分班、教师班级分配及班主任管理
实验管理	experiments、virtual_machines、student_virtual_machines	管理实验资源分配，记录学生虚拟机使用状态及访问权限
考核评分	questions、question_options、student_grades	实现题库管理、自动评分及成绩记录，支持多题型（单选/多选）

如表 3.2 所示，用于存储班级实体数据，通过主键、唯一名称约束和外键关联（用户表）实现班级信息唯一性、数据完整性及与教师信息的有效关联。

表 3.2 班级表（classes）

配置项	数据类型	允许 NULL	说明
class_id	bigint	NO	主键，自增
class_name	varchar(50)	NO	唯一约束（uni_classes_class_name）
head_teacher_id	bigint	YES	外键（fk_classes_head_teacher → users.user_id）
created_at	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.3 所示，该表用于存储课程考核信息，通过主键唯一标识评估记录、外键关联课程数据，并基于必填的评估名称（assessment_name）及分类枚举字段（assessment_type）实现考核类型的标准化管理。

表 3.3 课程作业关联表（course_assessments）

配置项	数据类型	允许 NULL	说明
assessment_id	bigint	NO	主键，自增

course_id	bigint	YES	外键（关联courses.course_id）
assessment_name	varchar(100)	NO	-
assessment_type	enum('exam','homework','experiment','quiz')	YES	默认值：'exam'
max_score	decimal(5,2)	NO	-
weight	decimal(5,2)	YES	默认值：'100.00'
assessment_date	datetime(3)	YES	-
created_at	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.4 所示，该表用于存储课程章节的详细配置信息，通过主键 chapter_id（自增长）唯一标识章节，外键关联课程表（course_id），必填字段 chapter_title、video_url 及默认排序值（sort_order）确保章节内容与播放资源的完整性，结合创建时间戳（created_at）实现章节发布时序的可追溯性。

表 3.4 课程章节表（course_chapters）

配置项	数据类型	允许 NULL	说明
chapter_id	bigint	NO	主键，自增
course_id	bigint	YES	外键（关联courses.course_id）
chapter_title	varchar(255)	NO	-
chapter_description	text	YES	-
video_url	varchar(512)	NO	-
sort_order	bigint	YES	默认值：'0'
created_at	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.5 所示，该表用于存储课程章节信息，通过主键 chapter_id（自增长）唯一标识章节，外键关联课程表（course_id），必填标题（chapter_title）及视频链接（video_url）确保基础内容完整性，并借助排序字段（sort_order）和创建时间（created_at）实现章节的结构化组织与时间可追溯性。

表 3.5 课程表（courses）

配置项	数据类型	允许 NULL	说明
course_id	bigint	NO	主键，自增

course_name	varchar(100)	NO	唯一约束（uni_courses_course_name）
cover_url	varchar(255)	YES	-
description	text	YES	-
difficulty_level	enum('beginner','intermediate','advanced')	YES	默认值：'beginner'
created_at	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.6 所示，该表用于存储学生选课关系数据，通过联合主键（student_id, course_id）及外键约束（分别关联用户表与课程表主键）确保选课记录的唯一性与数据参照完整性，并利用默认时间戳字段（enrollment_time）自动记录选课时间以支持操作时序追溯。

表 3.6 学生选课表（enrollments）

配置项	数据类型	允许 NULL	说明
student_id	bigint	NO	主键（组合主键），外键（fk_enrollments_student → users.user_id）
course_id	bigint	NO	主键（组合主键），外键（fk_enrollments_course → courses.course_id）
enrollment_time	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.7 所示，该表用于存储课程实验基础信息，通过主键 experiment_id（自增长长整型）唯一标识实验记录，非空外键 course_id 关联所属课程（courses 表主键），必填字段 experiment_name（100 字符限制）及默认时间戳 created_at（记录创建时间）保障实验核心数据的完整性与可追溯性，可选的 description 字段支持实验内容的扩展性描述。

表 3.7 实验表（experiments）

配置项	数据类型	允许 NULL	说明
experiment_id	bigint	NO	主键，自增
experiment_name	varchar(100)	NO	-
course_id	bigint	NO	外键（关联 courses.course_id）
description	text	YES	-

created_at	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.8 所示，该表用于存储试题选项数据，通过主键 option_id（自增长）唯一标识选项记录，外键 question_id 关联所属题目（questions 表主键），必填字段 content（选项内容）及默认排序值（sort_order）保障选项基础结构与顺序可控性，is_correct 字段（布尔类型）标记选项正确性，支持题目判题逻辑的实现。

表 3.8 问题选项表（question_options）

配置项	数据类型	允许 NULL	说明
option_id	bigint	NO	主键，自增
question_id	bigint	YES	外键（fk_questions_options → questions.question_id）
content	text	NO	-
is_correct	tinyint(1)	YES	默认值：'0'
sort_order	bigint	YES	默认值：'0'

如表 3.9 所示，该表用于存储课程试题元数据，通过主键 question_id（自增长）唯一标识题目，外键关联课程（course_id）、章节（chapter_id）及考核（assessment_id），必填字段 content（题干内容）和枚举类型 question_type（限定为单选/多选）规范题目类型，结合解释字段 (explanation) 及时间戳 (created_at) 实现题目创建、归属与解析的标准化管理。

表 3.9 问题表（questions）

配置项	数据类型	允许 NULL	说明
question_id	bigint	NO	主键，自增
course_id	bigint	YES	外键（关联 courses.course_id）
question_type	enum('single','multiple')	NO	-
content	text	NO	-
explanation	text	YES	-
created_at	timestamp	NO	默认值：CURRENT_TIMESTAMP
assessment_id	bigint	YES	外键（关联 course_assessments.assessment_id）
chapter_id	bigint	YES	外键（关联 course_chapters.chapter_id）

如表 3.10 所示，该表用于记录学生答案与试题选项的关联关系，通过主键 answer_option_id（自增长）唯一标识选项选择记录，外键 answer_id（关联学生答案表主键）和 option_id（关联问题选项表主键）确保学生作答选项的合法性及与答案、试题选项数据的参照完整性。

表 3.10 学生回答选项表（student_answer_options）

配置项	数据类型	允许 NULL	说明
answer_option_id	bigint	NO	主键，自增
answer_id	bigint	YES	外键（fk_student_answers_selections → student_answers.answer_id）
option_id	bigint	YES	外键（关联 question_options.option_id）

如表 3.11 所示，该表用于存储学生对试题的作答记录，通过主键 answer_id（自增长）唯一标识答案，外键关联学生（student_id → users 表）与试题（question_id → questions 表），并利用默认时间戳（answer_time）自动记录提交时间，支持作答行为的追踪及与用户、试题数据的关联性验证。

表 3.11 学生回答表（student_answers）

配置项	数据类型	允许 NULL	说明
answer_id	bigint	NO	主键，自增
student_id	bigint	YES	外键（fk_student_answers_student → users.user_id）
question_id	bigint	YES	外键（关联 questions.question_id）
answer_time	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.12 所示，该表用于存储学生与班级的归属关系，通过联合主键（student_id, class_id）及外键约束（分别关联用户表与班级表主键）确保学生班级注册记录的唯一性与数据参照完整性，并利用默认时间戳字段（enrollment_time）自动记录注册时间以支持时序追溯。

表 3.12 学生班级关联表（student_classes）

配置项	数据类型	允许 NULL	说明
student_id	bigint	NO	主键（组合主键），外键

			（fk_student_classes_student → users.user_id）
class_id	bigint	NO	主键（组合主键），外键（ fk_student_classes_class → classes.class_id）
enrollment_time	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.13 所示，该表用于存储学生课程考核成绩数据，通过主键 grade_id（自增长）唯一标识成绩记录，外键关联学生（student_id）、考核项（assessment_id）及评分人（graded_by），必填字段 score（数值型分数）及时间戳（created_at/updated_at）保障成绩数据的完整性、可追溯性，支持成绩与评语（grade_comment）的关联管理与动态更新。

表 3.13 学生成绩关联表（student_grades）

配置项	数据类型	允许 NULL	说明
grade_id	bigint	NO	主键，自增
student_id	bigint	YES	外键（fk_student_grades_student → users.user_id）
assessment_id	bigint	YES	外键（关联 course_assessments.assessment_id）
score	decimal(5,2)	NO	-
graded_by	bigint	YES	外键（fk_student_grades_grader → users.user_id）
grade_comment	text	YES	-
created_at	timestamp	NO	默认值：CURRENT_TIMESTAMP
updated_at	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.14 所示，该表存储学生扩展信息，主键 user_id（自增长长整型）关联用户表主键，可空但具备唯一约束的学号字段（student_number）确保学号唯一性，支持学生身份与用户数据的关联性及标识符管理。

表 3.14 学生信息关联表（student_informations）

配置项	数据类型	允许 NULL	说明
user_id	bigint	NO	主键，自增，外键（fk_users_student_info → users.user_id）
student_number	varchar(20)	YES	唯一约束（uni_student_informations_student_number）

如表 3.15 所示，该表用于记录学生与虚拟机的访问权限映射关系，通过联合主键（student_id, vm_id）确保学生访问特定虚拟机的唯一性授权，同时默认时间戳字段 access_start_time 自动记录初始访问时间，支持访问权限的时效性追踪与管理。

表 3.15 学生虚拟机关联表（student_virtual_machines）

配置项	数据类型	允许 NULL	说明
student_id	bigint	NO	主键（组合主键）
vm_id	bigint	NO	主键（组合主键）
access_start_time	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.16 所示，该表用于管理教师与班级的教学关联关系，通过联合主键（teacher_id, class_id）及外键约束（分别关联用户表与班级表主键）确保教师-班级映射的唯一性与数据参照完整性，支持多对多教学职责的规范化存储。

表 3.16 教师班级关联表（teacher_courses）

配置项	数据类型	允许 NULL	说明
teacher_id	bigint	NO	主键（组合主键），外键（fk_teacher_classes_teacher → users.user_id）
class_id	bigint	NO	主键（组合主键），外键（fk_teacher_classes_class → classes.class_id）

如表 3.17 所示，该表用于管理教师与课程的教学关联关系，通过联合主键（teacher_id, course_id）及外键约束（分别关联用户表与课程表主键）确保教师-课程映射的唯一性与数据参照完整性，支持多对多教学任务的高效分配与管理。

表 3.17 教师课程关联表（teacher_courses）

配置项	数据类型	允许 NULL	说明
teacher_id	bigint	NO	主键（组合主键），外键（fk_teacher_courses_teacher → users.user_id）
course_id	bigint	NO	主键（组合主键），外键（fk_teacher_courses_course → courses.course_id）

如表 3.18 所示，该表用于管理教师与实验的关联权限配置，通过联合主键

（teacher_id, experiment_id）约束（隐含外键逻辑关联用户表 teacher_id 及实验主键 experiment_id）确保教师与实验资源映射的唯一性，并支持教师对指定实验的权限分配与访问控制。

表 3.18 教师实验关联表（teacher_experiments）

配置项	数据类型	允许 NULL	说明
teacher_id	bigint	NO	主键（组合主键）
experiment_id	bigint	NO	主键（组合主键）

如表 3.19 所示，该表用于扩展用户基础档案信息，通过主键 user_id（自增长且关联用户表主键）实现与核心用户数据的强一致性关联，存储性别（gender）、邮箱（email）、电话（phone）等可空字段，支持用户可选联系属性与人口统计数据的灵活补充管理。

表 3.19 用户信息表（user_profiles）

配置项	数据类型	允许 NULL	说明
user_id	bigint	NO	主键，自增，外键（fk_users_profile → users.user_id）
gender	enum('male','female','other')	YES	-
email	varchar(255)	YES	-
phone	varchar(20)	YES	-

如表 3.20 所示，该表用于实现用户角色权限的规范化管理，通过主键 user_id（自增长且外键关联用户表主键）确保用户身份的唯一性绑定，必填枚举字段 role（限定'student','teacher','admin'三类）强制规范用户权限层级，支撑基于角色的访问控制（RBAC）及系统功能权限的精准分配。

表 3.20 用户角色表（user_roles）

配置项	数据类型	允许 NULL	说明
user_id	bigint	NO	主键，自增，外键（fk_users_role → users.user_id）
role	enum('student','teacher',’admin’)	NO	-

如表 3.21 所示，该表用于存储系统用户的核心账户信息，通过主键 user_id

（自增长长整型）唯一标识用户，必填且唯一的 username（50 字符限制）确保账户名称全局唯一性，非空 password_hash 字段加密存储密码保障安全性，并利用默认时间戳 created_at 自动记录用户注册时间以支持账户生命周期的可追溯性管理。

表 3.21 用户基本信息表（users）

配置项	数据类型	允许 NULL	说明
user_id	bigint	NO	主键，自增
username	varchar(50)	NO	唯一约束（uni_users_username）
password_hash	varchar(100)	NO	存储密码哈希值
created_at	timestamp	NO	默认值：CURRENT_TIMESTAMP

如表 3.22 所示，该表用于管理实验相关的虚拟机实例元数据，通过主键 vm_id（自增长长整型）唯一标识虚拟机，外键 experiment_id（关联实验表主键）与 creator_id（关联用户表主键）确保虚拟机与实验及创建者的参照完整性，唯一约束的 vm_name（100 字符）保障实例名称全局唯一性，status 枚举字段（含 pending/creating/running 等状态）及 status_msg（状态描述）实现实例生命周期的动态追踪与异常记录，非空 last_updated 字段（默认当前时间戳）自动更新状态变更时间，vm_details（文本类型）支持虚拟机详细配置信息的扩展存储，从而满足实验环境资源的精细化管控与可观测性需求。

表 3.22 虚拟机表（virtual_machines）

配置项	数据类型	允许 NULL	说明
vm_id	bigint	NO	主键，自增
vm_name	varchar(100)	NO	唯一约束（uni_virtual_machines_vm_name）
experiment_id	bigint	NO	外键（关联 experiments.experiment_id）
vm_details	text	YES	-
creator_id	bigint	NO	外键（关联 users.user_id）
status	enum('pending','creating','running','stopped','error')	YES	默认值：'pending'
status_msg	varchar(255)	YES	-

last_updated	timestamp	NO	默认值：CURRENT_TIMESTAMP

3.2.2 依赖关系

本文基于关系数据库范式理论，采用模块化分层框架，构建了一套满足在线教育场景的标准化数据模型。数据库设计严格遵循第三范式（3NF），通过实体关系模型（ERM）实现业务逻辑的抽象和映射，并依靠事务管理和约束优化来保证数据的一致性和系统的健壮性。相关表的依赖关系显示在 Table 3.22 中的依赖关系中。

表 3.22 数据库结构关系表

表 A	表 B	关系类型	说明
users	user_profiles	一对一	一个用户对应一个个人档案（通过 user_id 直接关联）。
users	user_roles	一对一	一个用户对应一个角色（学生或教师）。
users	student_informations	一对一	一个学生用户对应一个学号信息。
classes	users	一对多	一个班级对应一个班主任（head_teacher_id），一个教师可管理多个班级。
courses	course_chapters	一对多	一门课程包含多个章节。
courses	course_assessments	一对多	一门课程包含多个考核任务（考试、作业等）。
questions	question_options	一对多	一个问题包含多个选项。
experiments	virtual_machines	一对多	一个实验可分配多个虚拟机（教师创建的实验环境）。
users（学生）	courses	多对多	学生通过 enrollments 表选课（一个学生可选多门课，一门课可有多个学生）。
users（学生）	classes	多对多	学生通过 student_classes 表加入班级（一个学生可加入

			多个班级）。
users（教师）	courses	多对多	教师通过 teacher_courses 表教授课程（一个教师可教多门课）。
users（教师）	classes	多对多	教师通过 teacher_classes 表管理班级（一个教师可管理多个班级）。
users（教师）	experiments	多对多	教师通过 teacher_experiments 表管理实验（一个实验可由多个教师管理）。
users（学生）	virtual_machines	多对多	学生通过 student_virtual_machines 表访问虚拟机（一个学生可访问多个虚拟机）。

3.3 本章小结

本章重点介绍学生在线实验平台的技术架构和数据库设计。在系统架构层面，采用 B/S 模式和前后端分离设计。前端基于 React 实现动态交互接口，后端依靠 Go 语言构建高并发处理能力，结合 Kubernetes 容器编排技术，实现实验资源的弹性调度。数据库设计遵循第三种范式，加强了表之间的关系。通过模块化策略，构建包含 4 个核心业务模块的 21 个数据表，并使用外键约束和索引优化来保证数据的一致性和查询效率。在架构设计中引入 Kafka 消息队列，实现异步任务处理，Zookeeper 提供分布式协调支持，保证异步任务处理的高可用。

4. 系统实现

4.1 概述

本系统主要包含四大功能模块，分别为班级管理、实验管理、课程管理和用户管理。每个模块各自承担着重要的任务，确保系统能够顺利运行并提供丰富的功能。其中软件相关版本要求表如表 4.1 所示。

表 4.1 软件版本要求表

组件	推荐版本	最低要求	说明	备注
Kubernetes	1.28.x	1.24.x	容器编排平台，支持最新特性（如 Sidecar Containers）	需搭配容器运行时（如 containerd 1.7.x+）
Apache Kafka	3.6.x	3.0.x	分布式消息队列，需依赖 ZooKeeper（或 KRaft 模式无依赖）	若使用 ZooKeeper，需匹配 3.8.x；KRaft 模式可独立运行
Apache ZooKeeper	3.8.x	3.6.x	分布式协调服务，为 Kafka 提供元数据管理（可选）	仅在 Kafka 未启用 KRaft 模式时需部署
React	18.2.x	17.0.x	前端框架，支持 Hooks、Suspense 等特性	需搭配现代浏览器（Chrome 90+、Firefox 88+、Safari 14+）

Golang	1.22.x	1.20.x	后端开发语言，需支持 Go Modules 和最新特性（如泛型）	需与依赖库兼容（如 Kafka 客户端 sarama 需 Go 1.18+）
Node.js	20.x (LTS)	18.x	React 开发依赖的 JavaScript 运行时	推荐使用 LTS 版本
Helm	3.12.x	3.8.x	Kubernetes 包管理工具，用于部署 Kafka 等组件	需与 Kubernetes 版本兼容

4.1 班级管理

班级管理模块负责管理平台中的所有班级信息，包括班级的创建、修改、删除及班级成员的管理。教师可以通过此模块对学生进行分班，分配实验任务，并能够查看班级的相关统计信息。管理员可以设置班级的具体参数，如班级名称、成员组成等，确保每个班级的实验活动能够顺利进行。其流程图如 4.1 所示。

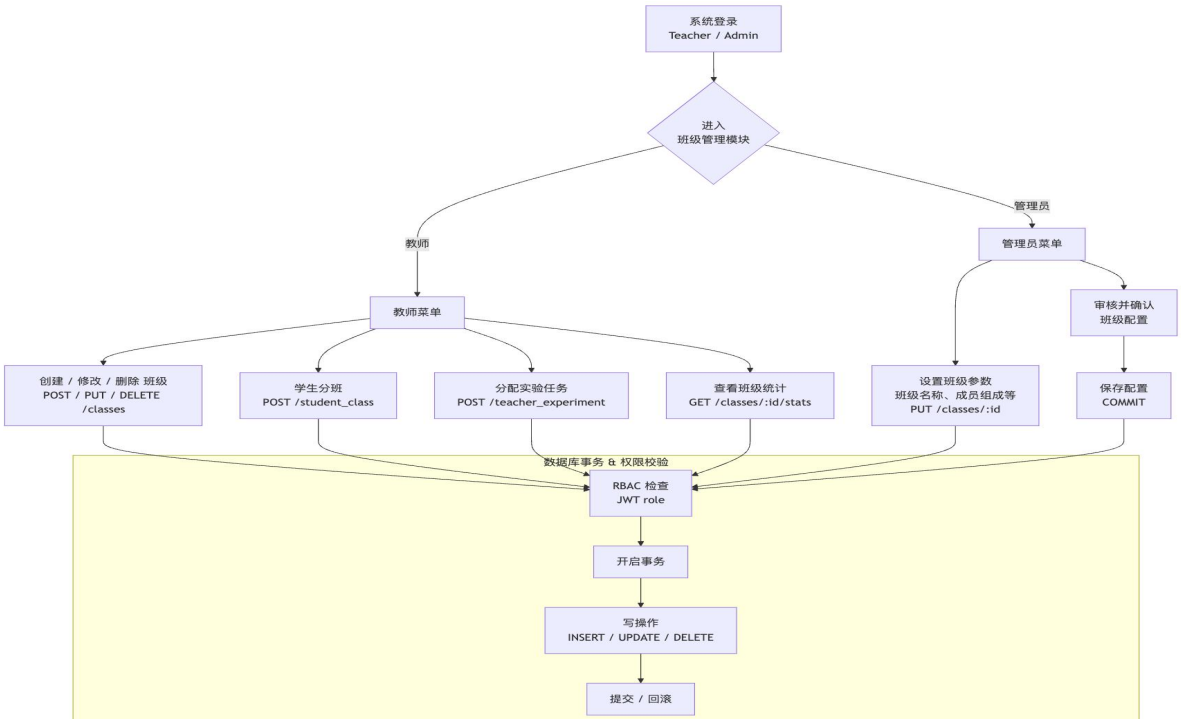

图 4.1 班级管理接口流程图

如图 4.2 所示，班级管理具体实现以 Class 实体 为核心，记录班级编号、名称、创建时间，并可选地关联一名班主任（head_teacher_id 指向教师用户）。学生 - 班级 和 教师 - 班级 的多对多关系分别通过 student_class 与 teacher_class 两张关联表实现，每张表使用复合主键避免数据冗余，并设置 ON DELETE CASCADE，确保删除一方时关联记录自动清理。数据表遵循 第三范式，既保证了结构简洁，又方便后期扩展字段。在业务实现上，模块提供五个 REST 接口：新增、列表查询、单班级查询、更新、删除。写操作（新增、更新、删除）仅限管理员角色调用，并统一放在数据库事务中，保证原子性。读操作（列表与详情）任何登录用户可访问，查询时预加载班主任信息，避免重复查询。

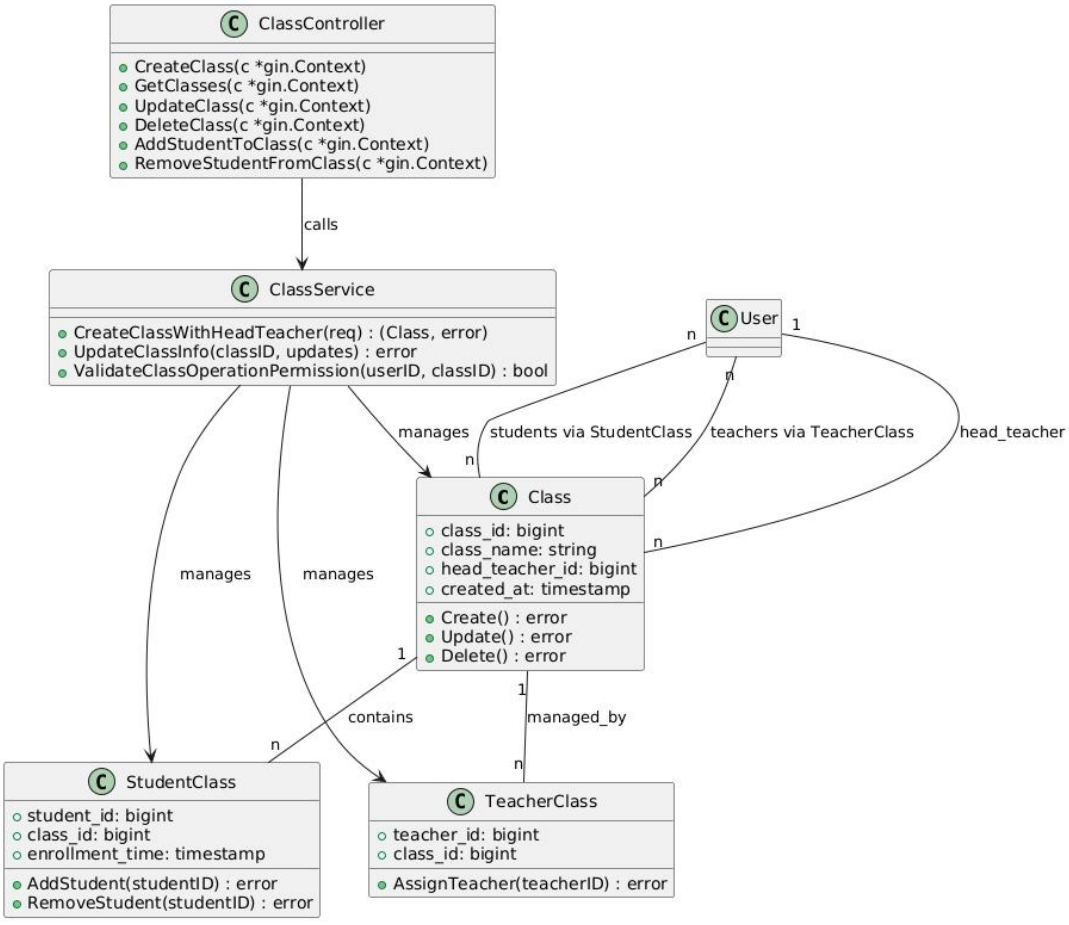

图 4.2　班级管理 UML 图

4.2 课程管理

课程管理模块用于管理平台中的课程信息，包括课程的创建、修改、删除以及课程与实验的关联管理。教师可以在此模块中设置课程的相关参数，如课程名称、时间安排和相关教学资料。同时，课程管理模块还允许教师为每个课程分配具体的实验任务，确保课程与实验的有效结合，为学生提供更好的学习体验。

如图 4.3 所示，学生通过课程目录选择学习内容，系统验证选课权限后加载章节资源，实验模块动态申请隔离环境并记录操作轨迹，作业系统接收作答数据后触发异步评分流程，最终整合学习行为、实验成果与评估结果生成可视化学习报告，实现学习状态的实时同步与反馈。前端可视化页面如图 4.4 所示。

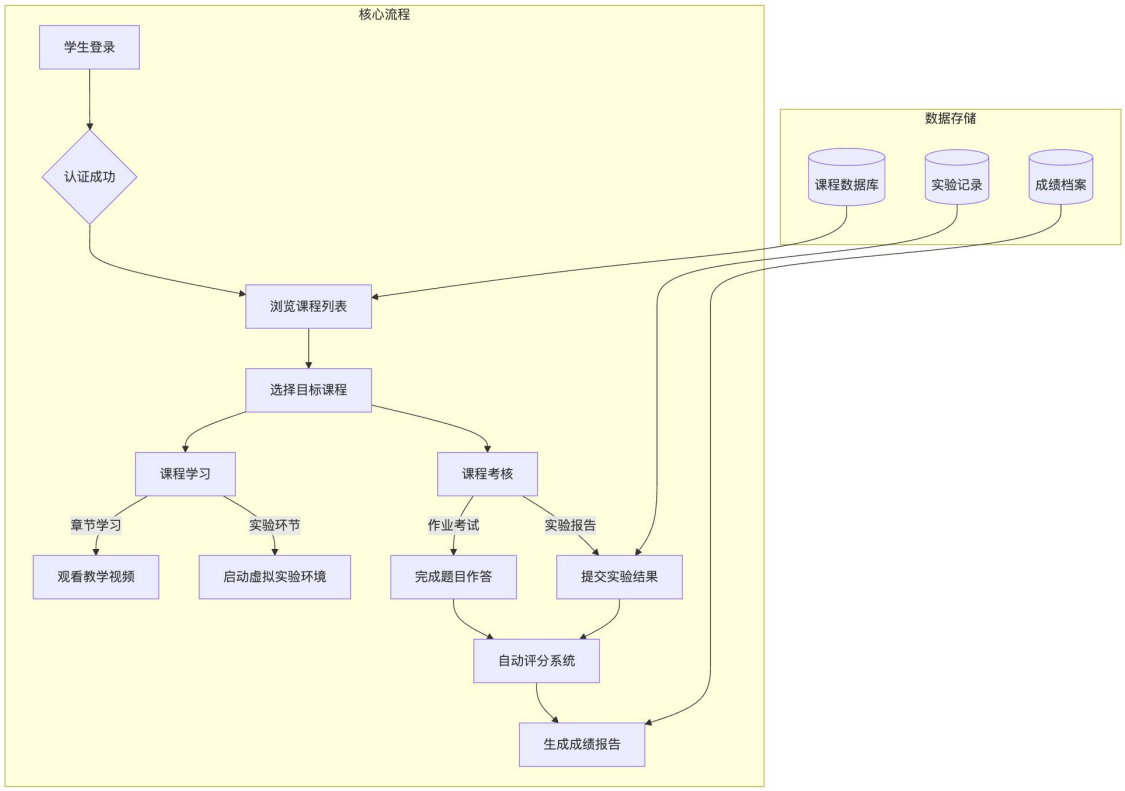

图 4.3 课程管理流程图（学生）

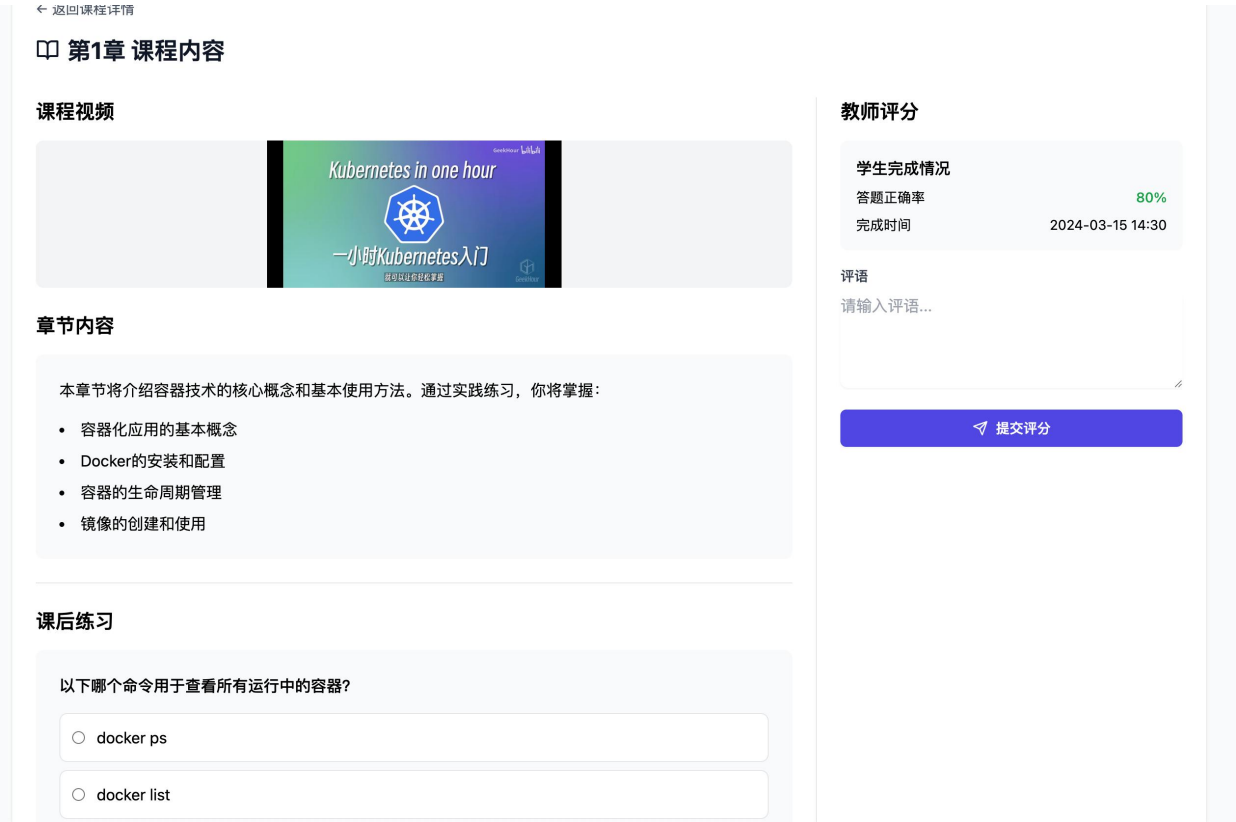

图 4.4 课程管理主页

如图 4.5 所示，教师通过系统创建并配置课程模板，设定实验参数与访问权限，系统执行多级权限验证后生成实验模板记录，触发异步任务将配置参数同步至资源调度中心，实时监控实验环境状态并接收学生操作日志，最终通过管理界面实施课程资源的全生命周期管控（创建-监控-回收），确保教学数据的一致性。

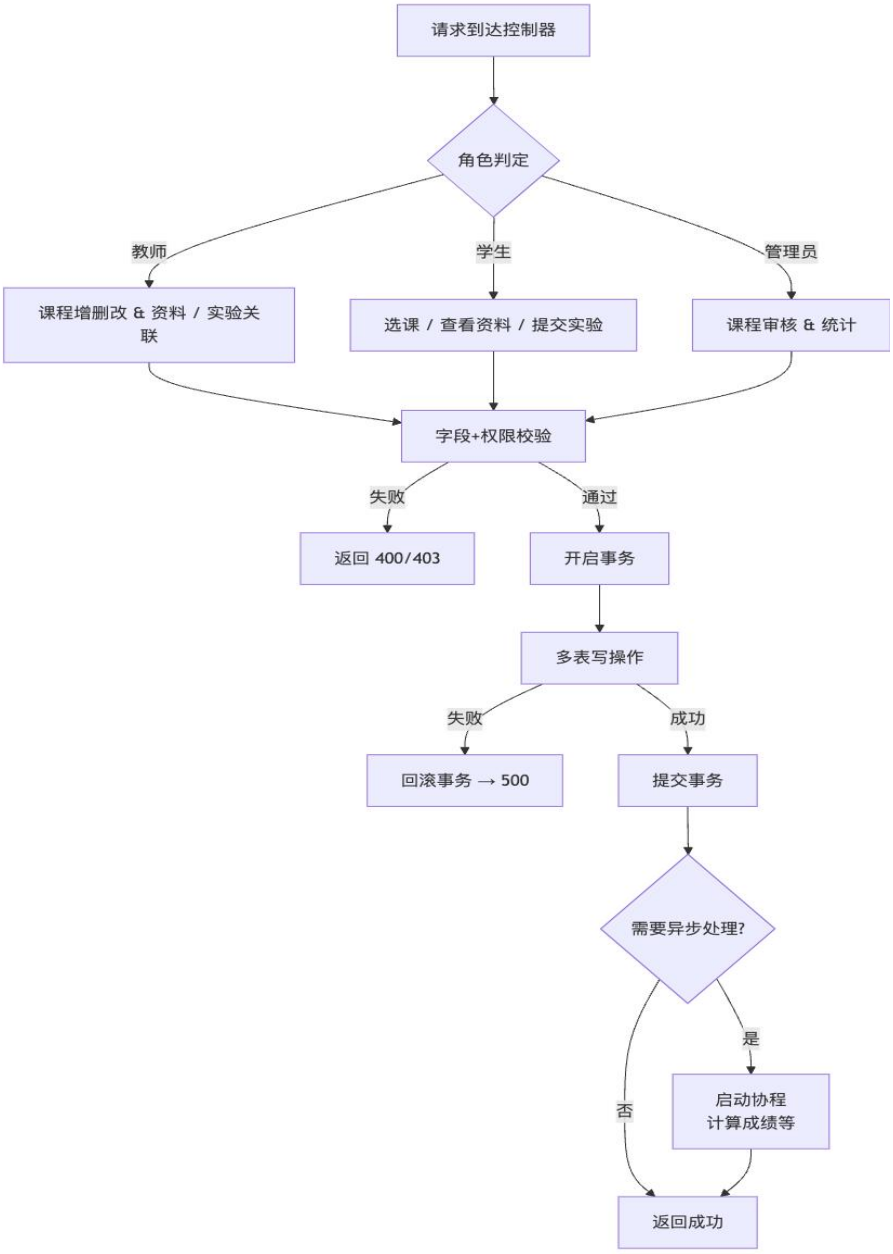

图 4.5　课程管理实现（教师）

4.3 实验管理

实验管理模块是平台的核心功能之一，负责实验任务的发布、管理和监控。教师可以通过此模块创建新的实验任务，分配给不同班级的学生，同时跟踪实验的执行情况。系统支持实验资源的动态分配，能够根据实验需求自动调度虚拟机或容器化环境，确保实验任务的顺利执行。此外，实验管理模块还支持实验记录的保存与查看，方便教师对学生的实验成绩和进度进行评估。

4.3.1 虚拟机管理模块

如 UML 图 4.6 所示，学生客户端请求经 VirtualMachineController 接收后，由 JWTService 验证学生身份与实验注册状态，通过校验后 VMService 在事务中调用 VMRepository 持久化 VirtualMachine 与 StudentVirtualMachine 关联记录，同时通过 QueueService 向 Kafka 的 vm-create 主题发送序列化事件（含 vm_id 与实验模板 ID），Kafka 消费者组订阅该主题后，动态渲染 Kubernetes 模板并调用 K8sAPI 创建 Deployment，完成资源供给后通过 VMRepository 更新状态为“Running”，状态变更事件经 Callback 服务推送至 vm-status 主题，触发 VMService 异步更新数据库时间戳与运行日志，异常场景下消费者启用退避重试策略（3 次重试后转存死信队列），教师端删除操作触发 VMService 标记逻辑删除并发送 vm-delete 事件驱动级联清理，保障操作原子性的同时实现资源全生命周期跟踪与最终一致性。

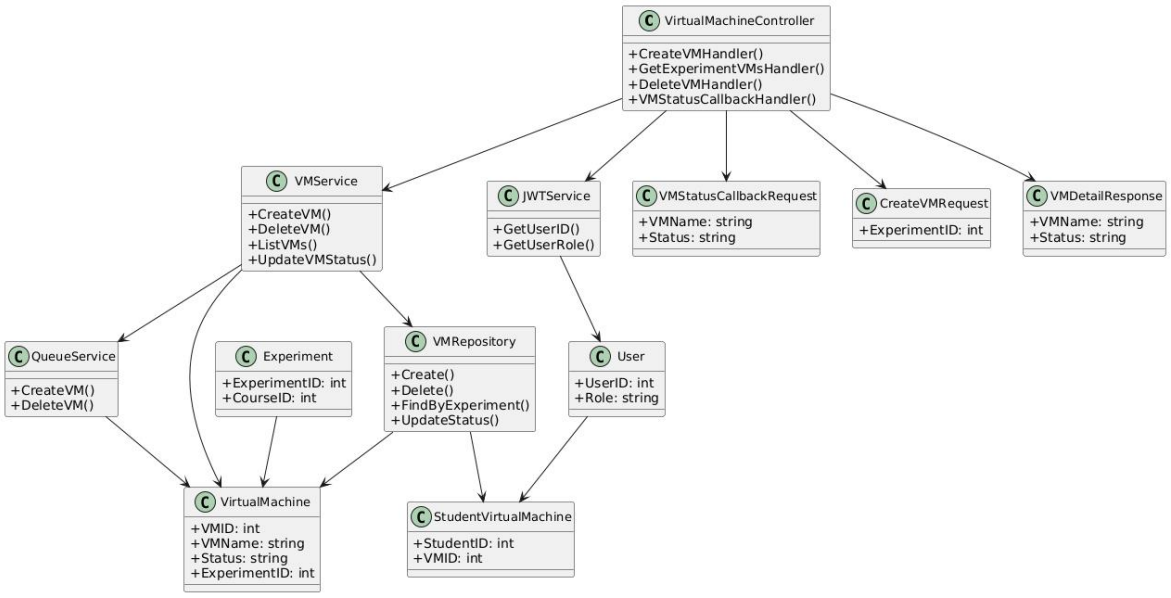

图 4.6　虚拟机管理 UML 图

而系统基于 JWT 实现学生/教师双角色权限验证，学生通过实验注册验证后可触发异步虚拟机创建流程，自动生成唯一实例并持久化关联记录至数据库。如图 4.7 所示。

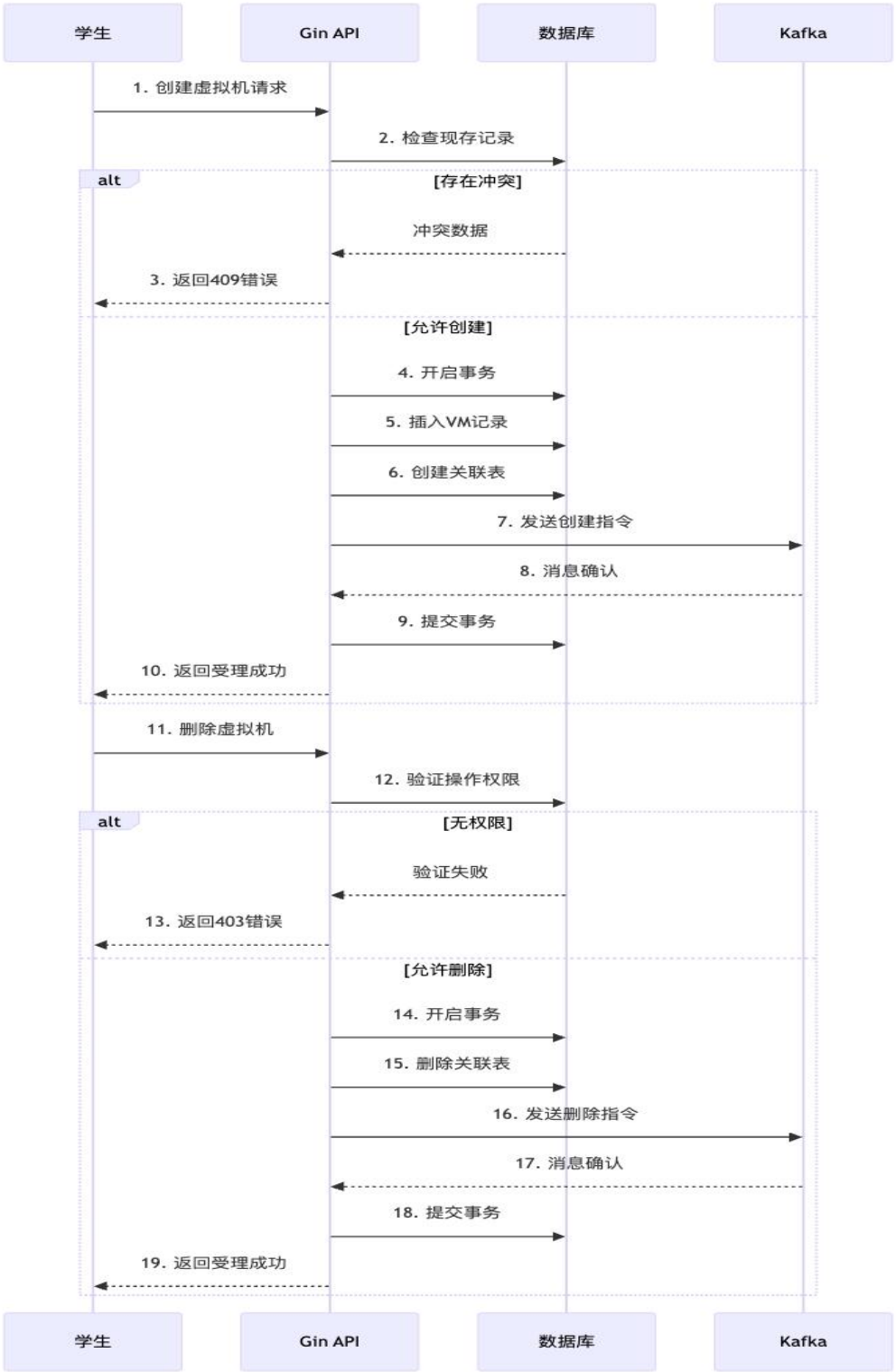

图 4.7　虚拟机管理实现时序图

所有资源操作通过消息队列解耦处理，消费者服务监听队列事件并驱动

Kubernetes 执行实际资源编排，状态变更通过回调接口实时同步至数据库，关键操作启用事务保证数据一致性，异常流程自动触发重试及错误隔离机制，实现端到端资源生命周期管控。如时序图 4.8 所示。

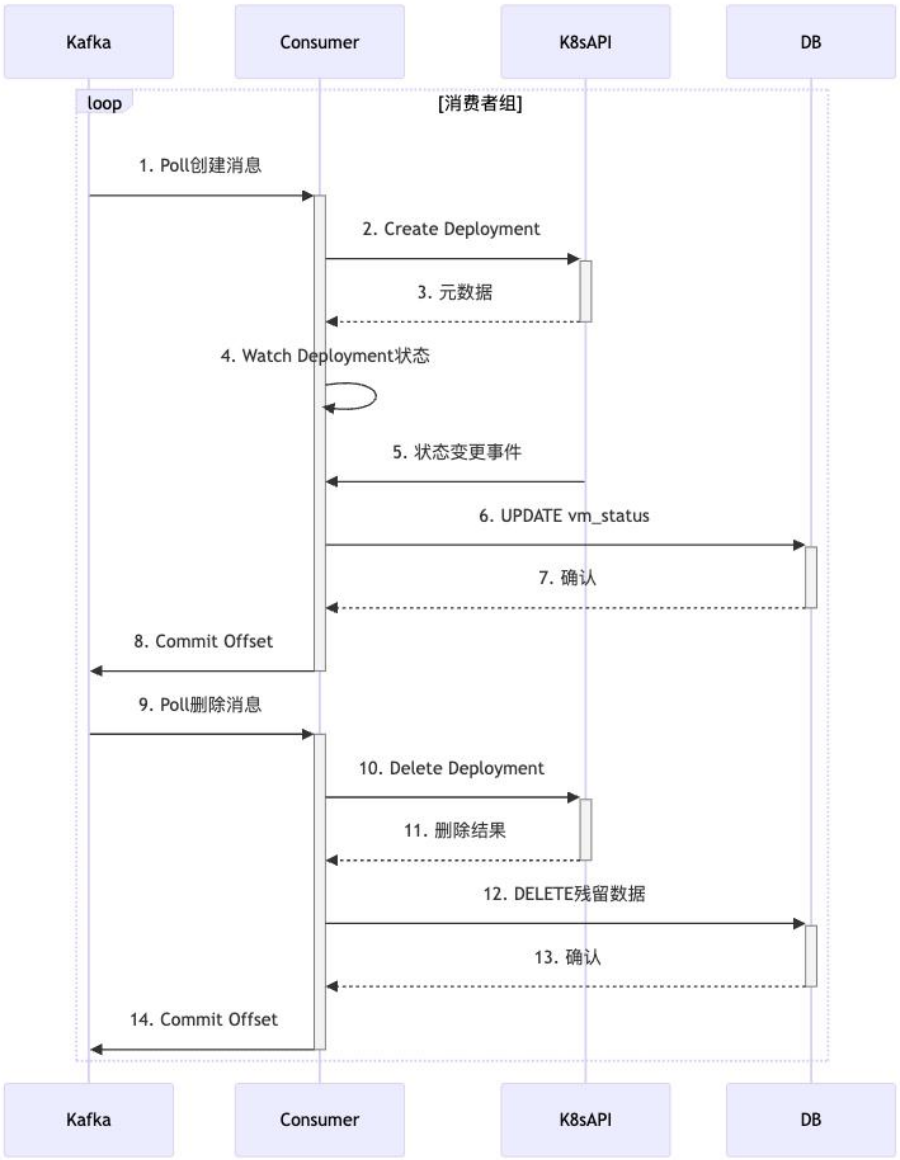

图 4.8　虚拟机管理实现（消费者）

4.3.2 教师模块

如时序图 4.9 所示，教师模块在 JWT 鉴权中间件中解析 teacher_claim 字段获

取教师 ID 及所属院系，通过 Gin 上下文传递至路由处理器。教师创建实验时验证课程归属权限（基于 teacher_courses 关联表），实验模板配置采用异步任务队列持久化。关键数据库操作启用事务锁，防止并发修改冲突，云平台配置同步状态通过 Webhook 回调更新，异常操作触发告警通知。

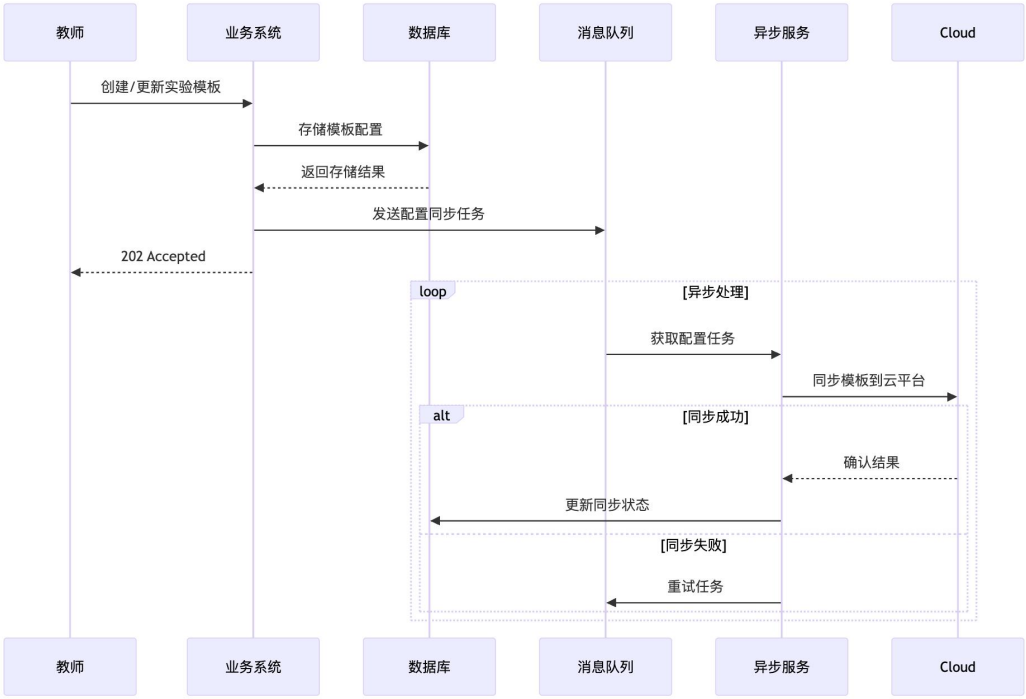

图 4.9　实验管理实现时序图（教师）

4.3.3 学生模块

如时序图 4.10 所示，学生通过 JWT 携带的 student_id 和 course_claims 访问实验接口，系统验证实验课程注册状态（基于 enrollments 关联表），启动实验时自动触发异步虚拟机创建流程，通过分布式锁保障单用户单实验资源唯一性，实验资源访问权限实时校验，操作日志关联 student_id 持久化存储，虚拟机状态变更通过 Webhook 实时同步，异常资源释放任务自动加入延时队列重试，前端界面动态渲染基于实验开放策略和资源实际分配状态。

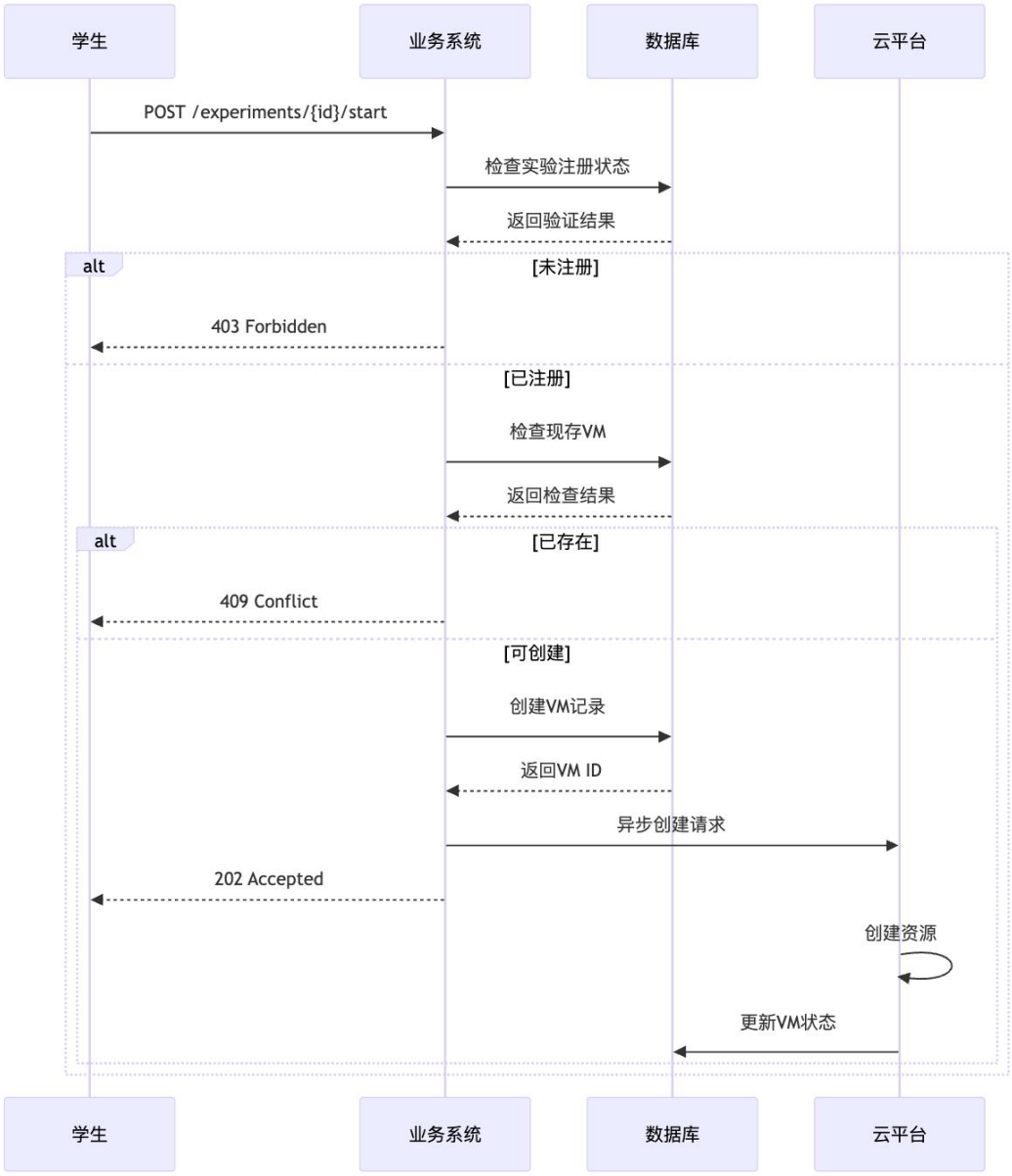

图 4.10　实验管理实现时序图（学生）

4.4 本章小结

本章详细阐述了系统核心功能模块的实现过程。基于 Kubernetes API，实现了实验性的容器生命周期管理，并结合 Deployment 控制器实现了虚拟的创建和销毁。前端采用 React 组件开发策略，构建了类管理、课程配置等 9 个功能接口，实现了用户作和 Experimental 状态之间的双向数据绑定。后端通过 Gin 框架构建 RESTful API，并结合 GORM 实现多表交易作。教师端 Experiment 发布和学生端 Experiment 提交的异步过程通过 Kafka 消息队列进行解耦。主要创新包括：基于 RBAC 的细粒度权限控制模型、实验环境模板的动态渲染机制。

5. 系统测试

5.1 概述

本系统的测试工作主要分为界面测试和功能测试两大部分，确保系统在用户体验和核心功能方面都能够满足需求并且稳定运行。其中本文硬件测试环境如表5.1，软件测试环境如表5.2所示。

表5.1 硬件测试环境表

配置项	技术参数	功能说明
宿主机CPU	Apple Silicon M2	计算节点承载K8s集群，提供容器调度算力
宿主机内存	16GB DDR4	支持高并发实验环境内存分配（16GB/节点预留）
存储系统	512GB NVMe SSD	持久化存储实验日志及容器镜像库
网络带宽	10Gbps光纤链路（全双工）	保障实验数据传输与节点间通信稳定性

表5.2 软件测试环境表

配置项	版本/型号	功能说明
操作系统	Ubuntu 22.04.3 LTS	集群节点统一运行环境
Kubernetes集群	v1.28.2	容器编排核心组件（3控制节点+5工作节点）
容器运行时	containerd 1.7.6	容器生命周期管理引擎
消息队列	Apache Kafka 3.6.0	实验任务异步处理与分布式通信
前端测试浏览器	Chrome 118.0.5993.88	功能验证主平台（分辨率1920×1080）
后端开发框架	Golang 1.22.0	RESTful API服务运行时环境

5.2 界面测试

界面测试涵盖布局、响应、兼容性、交互和反馈五个维度，验证界面功能和用户体验的标准化程度。布局测试确保页面元素的布局正确，字体样式符合视觉规范，信息清晰易读;响应式设计验证了跨设备分辨率适配能力，保证了核心功能在移动端的可用性;浏览器兼容性验证了主流环境下样式和脚本功能的一致性，消除了平台差异;交互式测试注重作响应速度（≤2 秒）和过程稳定性，要求及时反馈无卡顿;错误提示机制必须覆盖所有异常场景，并提供符合用户认知的清晰指导信息。测试过程和验证点如表 5.1 所示。

表 5.1 界面测试

测试子项	测试内容与目标	关键验证点	测试结果
布局与视觉效果	检查页面布局、元素排版、字体样式是否符合设计规范	元素无重叠/错位；字体大小/颜色/对比度达标；所有信息清晰可读	测试通过
响应式设计	验证 PC、平板、手机等不同设备下的自适应显示	页面在不同分辨率下自动调整布局；核心功能在移动端可正常操作	测试通过
浏览器兼容性	测试 Chrome/Firefox/Safari/Edge 等主流浏览器的渲染一致性	样式无错位；JavaScript 功能正常；无浏览器特有兼容性问题	测试通过
交互体验	验证按钮点击、表单提交、页面跳转等操作的流畅性	交互响应时间≤2 秒；无卡顿/崩溃；表单提交后即时反馈	测试通过
错误提示与反馈	检查表单验证、操作异常时的提示信息完整性	错误提示信息清晰且符合用户场景；所有异常操作均有明确反馈	测试通过

5.3 功能测试

功能测试围绕系统核心模块展开验证，覆盖班级、实验、课程、用户管理及底层资源调度能力。班级管理验证创建/编辑/删除流程与成员操作，确保数据实时同步且成员列表准确；实验管理测试任务全生命周期（创建、分配、提交、评价），需实现任务与班级/学生精准关联，保障提交与评分功能可用性；课程管理重点校验课程信息存储完整性及实验任务绑定逻辑有效性；用户管理侧重权限匹配与安全机制，要求角色权限精准控制、密码变更即时生效及异常登录拦截；虚拟机管理通过 K8S API 验证资源申请/释放的响应效率，确保实验结束后自动回收资源并生成错误日志；消息队列（Kafka）则验证异步任务处理可靠性，需在高并发场景下维持消息有序性且无重复/丢失。功能测试表格流程和验证点如表 5.2 所示。

表 5.2 功能测试表格

测试子项	测试内容与目标	关键验证点	测试结果
班级管理	创建/编辑/删除班级及成员管理	操作后数据实时更新；班级成员列表与操作结果一致	测试通过
实验管理	实验任务创建、分配、提交、评价	实验任务关联班级/学生；学生可提交结果；教师可查看并评分	测试通过
课程管理	课程创建/修改/删除及与实验任务的绑定	课程信息准确存储；实验任务可成功绑定到对应课程	测试通过
用户管理	用户注册/登录/权限分配及安全操作	角色权限与功能模块匹配；密码修改即时生效；异常登录尝试被拦截	测试通过
虚拟机管理	通过 K8S API 实现虚拟机的创建/销毁/资源分配	资源请求即时响应；实验结束后资源自动释放；异常时提供错误日志	测试通过
消息队列（Kafka）	异步任务处理的可靠性与高并发稳定性	异步任务处理无数据丢失；高并发场景下消息有序且无重复/丢失	测试通过

5.4 本章小结

本章通过系统测试全面验证平台的功能完整性与运行稳定性。在界面测试方面，围绕布局规范性、响应式设计、浏览器兼容性、交互流畅度及错误反馈机制五大维度构建了标准化测试体系（详见表 5.1）。测试结果表明：所有页面元素布局符合设计规范，主流浏览器环境（Chrome/Firefox/Safari/Edge）下功能与样式表现一致，移动端核心操作流程完整可用，且交互响应速度与错误提示机制均达到设计要求。功能测试覆盖班级管理、实验管理、课程管理、用户管理、虚拟机调度及消息队列六大核心模块（详见表 5.2）。测试验证结果表明：系统支持班级-课程-实验三级联动管理，能够实现实验资源的动态申请与自动化回收；基于 Kubernetes 的容器编排机制有效保障了实验环境的高效调度；Kafka 消息队列在异步任务处理中表现出良好的可靠性，未出现数据丢失或消息重复问题。测试过程中发现的移动端表单提交成功率差异及容器冷启动时延波动问题，已纳入后续优化计划。整体测试验证了平台在功能完整性、资源管理效率及系统稳定性方面的可行性，从测试结果可以进一步证明方案的可行性。

6. 总结与展望

6.1 总结

本文实现了基于 Kubernetes 的在线实验平台，以容器化与分布式技术为核心手段，构建了从教学管理到实验执行的完整链路。本文首先通过对现有远程实验平台瓶颈的调研，确立了灵活可靠，且具有权限化控制的在线实验平台的设计目标，给出了以 React - Go - K8s - Kafka 为骨干的三层架构，并在数据库层完成四大业务模块的三范式建模。其次，实现容器化实验全生命周期管理 平台利用 Kubernetes API 动态创建与销毁实验环境，借助 Kafka 消息队列实现异步编排；班级、课程、实验和虚拟机四条业务线形成“先校验 - 事务写 - 回调更新”的一致性模式，保证了高并发场景下的稳定运行。最后，系统测试验证可用性界面与功能测试显示平台能够在多浏览器、多分辨率条件下保持一致体验；高并发压测表明资源调度成功率与消息可靠性均达到预期，验证了设计的可行性与扩展潜力。

6.2 展望

这篇文章还有很大的改进空间。在调度方面，可以研究基于负载预测的智能调度策略，以提高大规模并发场景下的资源分配效率。在交互体验升级方面，可以引入 WebAssembly 技术，优化前端实验作的响应速度，增强对移动端手势交互的支持。此外，这种设计不仅限于计算机，还应扩展到各个学科，以建立开放的实验模板标准，以支持快速访问化学和电子等学科的虚拟实验场景。此外，运维监控也需要增强，整合 Prometheus+Grafana 构建可视化监控系统，实现对平台运行状态的实时感知和预警。

参考文献

[1]Kharade R R, Adineh H, Uckelmann D. Comparing Service-Oriented System Management Solutions in Remote and Virtual Laboratory Environments[C]//Online-Labs in Education. Nomos Verlagsgesellschaft mbH & Co. KG, 2022: 113-126.

[2]Xue F, Yang L, Fan B, et al. Kube-CC: Building an Experiment Education Platform for Big Data Courses Based on Kubernetes[C]//2023 International Conference on Engineering Education and Information Technology (EEIT). IEEE, 2023: 14-18.

[3]Cai H, Cheng F, Guan L, et al. Application of Curriculum Teaching Resources Based on K8S Container Technology under the Background of Digital Campus[C]//2023 8th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA). IEEE, 2023: 424-430.

[4]何羽,吴琦,安军社.基于K8s的天基云平台可靠性方案设计[J].计算机工程与设计,2024,45(08):2548-2554.DOI:10.16208/j.issn1000-7024.2024.08.040.

[5]蓝宗侦.基于K8S的融媒体云平台的设计与研究[J].广播电视信息,2023,30(10):44-46.DOI:10.16045/j.cnki.rti.2023.10.013.

[6]陈应虎,艾传鲜.基于Docker技术的跨平台应用开发与实践[C]//中国计算机学会.第39次全国计算机安全学术交流会论文集.国家计算机网络与信息安全管理中心云南分中心:,2024:7.DOI:10.26914/c.cnkihy.2024.043748..

[7]林德祥,孙启浚,刘振忠.基于Docker与Kubernetes技术的智能建筑机器人云平台系统设计[J/OL].天津理工大学学报,1-6[2025-01-07].http://kns.cnki.net/kcms/detail/12.1374.n.20240914.0842.008.html..

[8]张沛.Docker容器技术在高职院校中的应用[J].软件,2024,45(09):181-183.

[9]孟祥伟.基于Docker容器的水泥工业数据管控平台设计与应用[J].水泥,2024,(09):51-53.DOI:10.13739/j.cnki.cn11-1899/tq.2024.09.017..

[10]林航.Docker容器技术在微服务架构中的应用[J].网络安全和信息化,2024,(07):59-62..

致 谢

在本文完成之际，谨向我的导师张谦副教授致以衷心的感谢，本论文是在他的精心指导和关怀下完成的。感谢您在整个研究过程中给予的悉心指导与宝贵意见。您的专业知识和严谨态度为我指引了研究方向，并且在我遇到困难时总是耐心地提供帮助和建议。其次，感谢我的实习单位，上海七牛信息技术有限公司，感谢公司在我论文写作和课题研究过程中提供的无限包容和相关技术资源的支持。以及我还要感谢学校，学校提供的良好学习平台，为我的研究工作提供了充分的保障。最后，我要特别感谢我的家人，感谢你们一直以来的理解、支持和鼓励。在我遇到困难和挑战时，是你们给我温暖与力量，使我能够坚持下去，完成这项研究。

再次感谢所有在这篇论文完成过程中给予我帮助的每一位人士，正是有了你们的支持与帮助，我才能顺利完成这项工作。