**武汉大学**

**计算机学院实训**

**UClub软件系统设计说明书**

**项目成员： 李欣雨、张馨仪、李国峰、曹键、赵志远**

**撰 写 人：**  2025 年 6 月 26 日

**评 审 人：**  年 月 日

武汉凡诺软件技术有限公司

2025年

**修订记录**

| 日期 | 修订版本 | 修改章节 | 修改描述 | 修订人 |
| --- | --- | --- | --- | --- |
| 6.26 | V1.0 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1. 简介](#_Toc6698)

[1.1 目的](#_Toc4522)

[1.2 范围](#_Toc26915)

[一、软件名称](#_Toc21689)

[二、软件功能](#_Toc30755)

[三、软件应用](#_Toc26321)

[2. 第0层设计描述](#_Toc11701)

[2.1 软件系统上下文定义](#_Toc1841)

[2.2 设计思路](#_Toc31603)

[一、设计可选方案](#_Toc15102)

[二、 设计约束](#_Toc3097)

[三、遵循标准](#_Toc31621)

[四、硬件限制](#_Toc3899)

[五、技术限制](#_Toc20378)

[六、其他](#_Toc30628)

[3. 第一层设计描述](#_Toc28930)

[3.1 系统结构](#_Toc27043)

[一、系统结构描述](#_Toc26113)

[二、业务流程说明](#_Toc16596)

[3.2 分解描述](#_Toc30853)

[一、模块1/子系统1描述](#_Toc4393)

[二、模块2/子系统2描述](#_Toc32647)

[3.3 依赖性描述](#_Toc25918)

[3.4 接口描述](#_Toc21656)

[一、模块1/子系统1的接口描述](#_Toc19513)

[二、模块2/子系统2的接口描述](#_Toc24249)

[4. 第二层设计描述](#_Toc7111)

[4.1 模块1名称](#_Toc9439)

[一、模块设计描述](#_Toc12029)

[1、类名](#_Toc11749)

[2、类名](#_Toc28951)

[二、功能实现说明](#_Toc24755)

[4.2 模块2名称](#_Toc15222)

[一、模块设计描述](#_Toc28380)

[1、类名](#_Toc2123)

[类名](#_Toc25868)

[二、功能实现说明](#_Toc18380)

[5. 数据库设计](#_Toc6134)

[5.1 实体定义](#_Toc21482)

[一、分解描述](#_Toc6564)

[二、内部依赖性描述](#_Toc7906)

[5.2 行为定义](#_Toc599)

[一、分解描述](#_Toc23323)

[二、外部依赖性描述](#_Toc29017)

[三、内部依赖性描述](#_Toc3886)

[6. 原型或界面设计](#_Toc3441)

关键词：

摘 要：

缩略语清单：<对本文所用缩略语进行说明。>

| 缩略语 | 英文全名 | 中文解释 |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 

## 简介

## 目的

本文档旨在为UClub社团联盟系统的开发、测试、维护和使用提供全面的设计说明。主要读者包括开发人员、测试人员、项目管理者、系统管理员及最终用户。

## 范围

1. 软件名称

UClub

1. 软件功能

* 社团管理：社团创建、审核、查看、收藏、成员管理、角色分配、用户加入与退出
* 活动管理：活动发布、审核、报名、参与人员的查看
* 论坛交流：帖子发布、评论互动、内容审核、帖子查找、查看浏览历史
* 实时聊天：社团内部聊天室、大厅公共聊天室
* 公告系统：系统公告、社团公告
* 用户管理：注册、登录、权限控制、个人资料查看与管理
* 后台管理：社团/活动/用户审核，数据统计、公告发布
* AI智能助手：帮助没有头绪的用户找到适合的社团

1. 软件应用

本软件主要应用于高校内部，旨在为学生提供一个全面、便捷的社团信息获取与参与平台，帮助新生快速了解并加入社团及社团活动，论坛和聊天室能够让用户找到同好、扩展交际圈，同时为社团管理提供高效支持。以下是详细说明：

1. 社团信息获取与参与

* 新生引导：新生入学时，往往对学校的社团活动知之甚少。本软件通过提供详细的社团介绍、活动预告和成员评价，帮助新生快速了解各个社团的特色和活动内容，从而选择适合自己的社团加入。
* 社团展示：为每个社团提供专属页面，展示社团的基本信息（如宗旨、历史、特色活动等）、成员风采、举办活动，以及社团的最新动态。学生可以通过浏览这些信息，深入了解社团文化。
* 活动报名：社团可以通过平台发布活动信息，包括活动时间、地点、内容、报名方式等。学生可以直接在平台上报名参加感兴趣的活动，系统会自动记录报名信息并通知社团管理员。

1. 社团管理支持

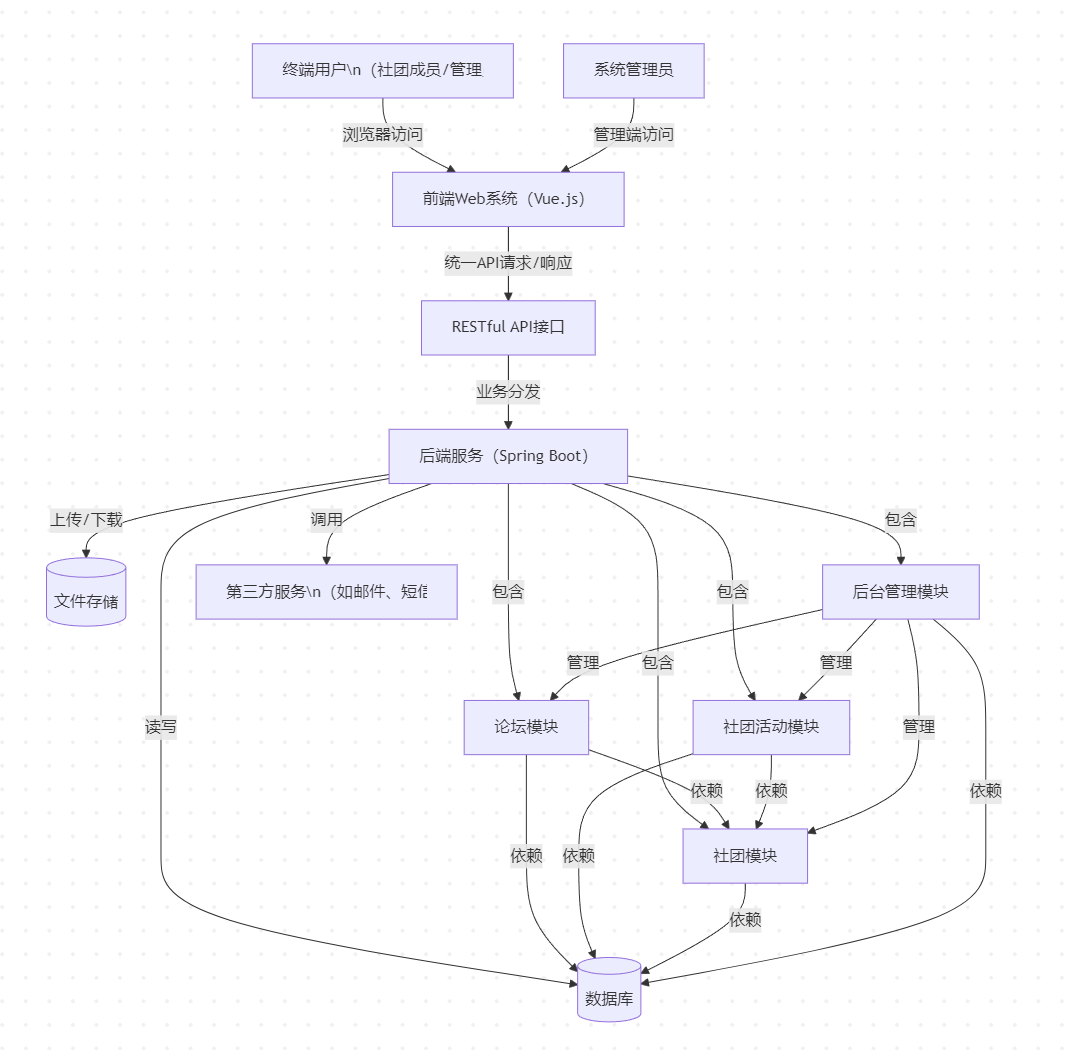
* 成员管理：社团管理员可以通过软件管理社团成员的基本信息、联系方式、参与活动记录和考勤情况。系统支持批量导入成员信息，方便社团管理员快速录入新成员。
* 活动管理：社团管理员可以在平台上发布、编辑和取消活动，设置活动的参与人数上限、报名截止时间等。活动结束后，管理员可以对活动进行总结和评价，收集参与者的反馈意见。

1. 社交互动与扩展交际圈

* 论坛交流：软件内置社团论坛，学生可以在论坛上发布帖子、回复评论，分享社团活动的体验、学习心得或生活趣事。论坛支持话题分类，方便学生查找感兴趣的内容。
* 聊天室功能：提供实时聊天室功能，学生可以与社团成员或感兴趣的人进行即时交流。聊天室支持文字、图片、表情等多种消息类型，增强交流的趣味性。

# 第0层设计描述

## 2.1 软件系统上下文定义



外部实体属性简述：

* 终端用户/系统管理员：具备唯一身份标识（如userId），通过前端界面与系统交互，权限不同，功能范围不同。
* 数据库：关系型数据库，存储所有核心业务数据，保证数据一致性和安全性。
* 文件存储：支持大文件、图片等非结构化数据的存储与访问。
* 第三方服务：如有集成，需保证接口安全、调用稳定。

## 2.2 设计思路

1. 设计可选方案
2. 方案一：前后端分离的架构

* 前端：Vue.js + Vite + Axios
* 后端：Spring Boot（Java）
* 数据库：MySQL
* 通信方式：RESTful API
* 部署：前端静态资源 + 后端服务分开部署

优点：

* 前后端职责清晰，开发效率高，易于团队协作。
* 前端界面响应快，用户体验好。
* 易于扩展和维护，适合后期功能迭代。
* 支持多终端接入（如Web、移动端）。
* 技术栈主流，社区资源丰富。

缺点：

* 需要处理前后端接口联调，开发初期沟通成本略高。
* 对于小型项目，初期搭建工作量略大。

1. 方案二：传统JSP/Servlet一体化架构

* 前端：JSP/HTML
* 后端：JSP（Java）
* 数据库：MySQL
* 通信方式：页面请求/响应
* 部署：单体Web应用（如Tomcat）

优点

* 架构简单，开发门槛低，适合小型项目。
* 部署方便，前后端一体化。

缺点

* 前后端耦合度高，界面交互体验较差。
* 不利于后期维护和扩展。
* 不适合多终端接入。
* 技术逐渐被淘汰，社区资源减少。

1. 方案三：移动端优先（App/小程序）+后端API

* 前端：微信小程序/Flutter/React Native
* 后端：Spring Boot（Java）
* 数据库：MySQL
* 通信方式：RESTful API
* 部署：移动端 + 后端服务

优点：

* 适合移动端用户，体验好。
* 后端API可复用，支持多端接入。
* 便于后期扩展App或小程序。

缺点：

* 开发成本高，需要多端适配。
* 需要专门的移动端开发能力。
* 对于以Web为主的场景不够友好。

本系统采用方案一，理由如下：

满足现代Web应用的开发需求，用户体验更好，易于维护和扩展。

团队技术掌握程度更高，前后端分离，开发效率也更高一些。

1. 设计约束
2. 业务约束

* 用户角色约束
  + 角色层级：社长> 副社长 > 干事 > 普通用户，权限逐级递减。
  + 权限继承：单个社团中高级角色自动拥有低级角色的所有权限。
  + 角色唯一性：同一用户在同一社团中只能拥有一个角色。
* 社团管理约束
  + 社团创建：普通用户可创建社团，但需系统管理员审核通过。
  + 社团名称：社团名称必须唯一，长度限制在2-20个字符。
  + 社团状态：社团状态包括"待审核"、"正常"、"已封禁"，状态变更需记录。
* 活动管理约束
  + 活动发布：只有社团管理员及以上角色可发布活动。
  + 活动时间：活动开始时间必须晚于当前时间，结束时间必须晚于开始时间。
  + 参与人数：活动参与人数不能超过设定的最大人数限制。
  + 活动状态：活动状态包括"报名中"、"进行中"、"已结束"，状态自动流转。
* 内容管理约束
  + 内容审核：所有用户发布的内容（帖子、评论）需要经过审核。
  + 内容长度：帖子标题限制50字符，内容限制5000字符。
  + 图片数量：单篇帖子最多可上传3张图片。

1. 技术约束

* 数据库约束
  + 字符集：必须使用utf8mb4字符集，支持emoji和特殊字符。
  + 存储引擎：必须使用InnoDB存储引擎，确保事务一致性。
  + 连接池：数据库连接池最大连接数不超过100，避免数据库压力。
  + 索引限制：单表索引数量不超过10个，避免影响写入性能。
* 文件存储约束
  + 文件大小：单文件上传大小限制为10MB。
  + 文件类型：只允许上传图片格式（jpg、jpeg、png、gif）。
  + 存储空间：单个用户上传文件总大小限制为100MB。
  + 文件命名：使用UUID生成文件名，避免文件名冲突。

1. 性能约束

* 响应时间约束
  + 页面加载：首屏加载时间不超过3秒。
  + API响应：90%的API请求响应时间不超过500ms。
  + 数据库查询：复杂查询响应时间不超过2秒。
  + 文件上传：10MB文件上传时间不超过30秒。
* 并发处理约束
  + 用户并发：系统必须支持很多并发用户同时在线。
  + 数据库并发：数据库连接池必须支持100个并发连接。
  + WebSocket连接：必须支持50个WebSocket并发连接。
  + 文件上传并发：必须支持50个用户同时上传文件。
* 资源使用约束
  + CPU使用率：正常负载下CPU使用率不超过70%。
  + 内存使用率：正常负载下内存使用率不超过80%。
  + 磁盘空间：预留30%的磁盘空间用于日志和临时文件。
  + 网络带宽：单个用户下载速度不低于1Mbps。

1. 安全约束

* 身份认证约束
  + 密码策略：密码必须包含大小写字母、数字和特殊字符，长度8-20位。
  + 会话超时：用户会话超时时间为30分钟，超时后需重新登录。
  + Token有效期：JWT Token有效期为24小时，过期需重新获取。
* 数据安全约束
  + 数据加密：用户密码必须使用BCrypt加密存储。
  + 传输加密：所有数据传输必须使用HTTPS协议。
  + 敏感信息：不得在日志中记录用户密码、token等敏感信息。
  + 数据备份：必须每日进行数据库备份，保留30天备份数据。
* 访问控制约束
  + API权限：所有API接口必须进行身份验证和权限检查。
  + 文件访问：用户只能访问自己上传的文件或公开文件。
  + 数据隔离：不同用户的数据必须严格隔离，防止越权访问。
  + 操作审计：所有敏感操作必须记录操作日志，便于审计。

1. 遵循标准
2. 前端开发标准

* Vue.js开发规范
  + 组件命名：使用PascalCase命名组件，如ClubDetails.vue、Activities.vue。
  + 文件结构：遵循Vue官方推荐的项目结构。
  + 代码风格：遵循Vue.js官方风格指南。
  + 组件名使用多词组合。
* JavaScript编码规范
  + ES6+标准：使用ES2015及以上版本的JavaScript特性。
  + 代码格式化：使用Prettier进行代码格式化。
  + 命名规范：
  + 变量和函数使用camelCase。
  + 常量使用UPPER\_SNAKE\_CASE。
  + 类名使用PascalCase。
  + 注释规范：使用JSDoc格式编写函数和类注释。
* CSS样式规范
  + 命名规范：使用BEM命名方法论。
  + 响应式设计：使用移动优先的响应式设计原则。
  + CSS变量：使用CSS自定义属性定义主题色彩和尺寸。

1. 后端开发标准

* Java命名规范
  + 类名使用PascalCase：UserController、ClubService。
  + 方法名使用camelCase：getUserById、createClub。
  + 常量使用UPPER\_SNAKE\_CASE：MAX\_FILE\_SIZE。
  + 包名使用小写：com.example.uclub\_backend.controller。
  + 代码结构：遵循Spring Boot官方推荐的项目结构。
* Spring Boot开发规范
  + 注解使用：
  + @RestController：REST API控制器。
  + @Service：业务逻辑层。
  + @Repository：数据访问层。
  + @Component：通用组件。
  + 依赖注入：使用构造器注入而非字段注入。
  + 异常处理：统一异常处理机制，使用@ControllerAdvice。
* RESTful API设计规范
  + HTTP方法使用：
  + GET：获取资源。
  + POST：创建资源。
  + PUT：更新资源。
  + DELETE：删除资源。

1. 数据库设计标准

* MySQL数据库规范：
  + 字符集：使用utf8mb4字符集，支持emoji和特殊字符。
  + 存储引擎：使用InnoDB存储引擎，支持事务和外键。
  + 表命名：使用小写字母和下划线，如club\_member、activity\_participant。
  + 字段命名：使用小写字母和下划线，如created\_at、user\_id。
* 数据库设计规范：
  + 主键设计：每个表必须有主键，使用自增ID。
  + 外键约束：建立适当的外键约束保证数据完整性。
  + 索引设计：为常用查询字段建立索引。
  + 字段类型：选择合适的字段类型和长度。
* SQL编写规范：
  + 查询优化：避免SELECT \*，只查询需要的字段。
  + 分页查询：使用LIMIT和OFFSET进行分页。
  + 事务处理：合理使用事务，避免长事务。
  + SQL注入防护：使用参数化查询，避免字符串拼接。

1. 安全标准

* OWASP安全指南：
  + 输入验证：所有用户输入必须进行验证和过滤。
  + 身份认证：使用安全的身份认证机制。
  + 会话管理：安全的会话管理，防止会话劫持。
  + 访问控制：基于角色的访问控制（RBAC）。
* JWT安全标准：
  + 密钥管理：使用足够长度的密钥，定期更换。
  + 过期时间：设置合理的token过期时间。
  + 传输安全：通过HTTPS传输JWT。
* 密码安全标准：
  + 加密算法：使用BCrypt等安全的密码哈希算法。
  + 密码存储：不存储明文密码，只存储哈希值。

描述本软件所遵循的标准、规范。

1. 硬件限制

服务器要求：

CPU：4核心以上，内存：8GB以上，存储：128GB以上，SSD网络：100Mbps以上

客户端要求：

内存：2GB以上，浏览器：支持ES6+网络：稳定的互联网连接

1. 技术限制
2. 前端技术限制

* Vue.js框架限制
  + 组件嵌套深度：建议不超过10层，避免性能问题。
  + 响应式数据限制：对象属性必须在初始化时定义，动态添加的属性无法响应式。
  + 内存限制：单页面组件数量过多可能导致内存泄漏。
* Element Plus组件库限制
  + 主题定制：只能通过CSS变量覆盖，无法深度定制组件内部结构。
  + 国际化：需要额外配置语言包，增加打包体积。
  + 组件数量：大量组件同时渲染会影响页面性能。
* Vite构建工具限制
  + 开发服务器：默认端口5173，需要配置代理解决跨域。
  + 热更新：某些情况下需要手动刷新页面。
  + 构建体积：生产环境打包后体积较大，需要代码分割优化。
* WebSocket通信限制
  + 连接数限制：浏览器对WebSocket连接数有限制。
  + 消息大小：单条消息建议不超过64KB。
  + 重连机制：网络断开需要手动处理重连逻辑。
  + 浏览器兼容：部分老旧浏览器不支持WebSocket。

1. 后端技术限制

* Spring Boot框架限制
  + 启动时间：应用启动需要10-30秒，影响开发效率。
  + 内存占用：JVM堆内存建议4GB以上，否则容易出现OOM。
  + 线程池限制：默认线程池大小有限，高并发时需要调优。
  + 热部署：生产环境不支持热部署，需要重启应用。
* Spring Security安全限制
  + 认证方式：当前只支持JWT认证，不支持OAuth2等第三方认证。
  + 权限粒度：基于角色的权限控制，无法实现细粒度权限。
  + 会话管理：无状态设计，无法实现会话管理功能。
  + 密码策略：密码复杂度验证相对简单。
* MyBatis限制：
  + XML配置繁琐，维护成本高。
  + 动态SQL复杂，容易出错。
  + 分页查询需要手动实现。
* WebSocket实时通信限制
  + 连接管理：需要手动管理连接池，避免内存泄漏。
  + 消息序列化：只支持JSON格式，不支持二进制数据。
  + 集群部署：多实例部署时消息无法共享。
  + 错误处理：连接异常处理机制不够完善。

1. 其他
2. 界面设计原则

* 一致性原则：整个系统使用统一的Element Plus组件库，保持视觉风格一致。
* 简洁性原则：界面布局简洁明了，避免信息过载。
* 响应式设计：支持PC端和移动端自适应，提升多设备访问体验。

1. 交互设计优化

* 操作反馈：所有用户操作都有明确的成功/失败提示。
* 加载状态：数据加载时显示loading动画，避免用户等待焦虑。
* 表单验证：实时表单验证，及时提示用户输入错误。
* 快捷操作：提供常用功能的快捷入口，如快速发布活动、一键加入社团。

1. 信息架构设计

* 导航结构：采用顶部导航栏，帮助用户了解当前位置。
* 搜索功能：支持社团、活动、帖子的模糊搜索。
* 分类标签：使用标签对社团进行分类，便于用户快速找到感兴趣的内容。

# 第一层设计描述

## 3.1 系统结构

一、系统结构描述

1. 系统结构分为三层：

表现层（前端）：

职责：负责用户交互与页面展示，调用后端API获取和提交数据。

技术栈：Vue.js + Axios

业务逻辑层（后端Service/Controller）：

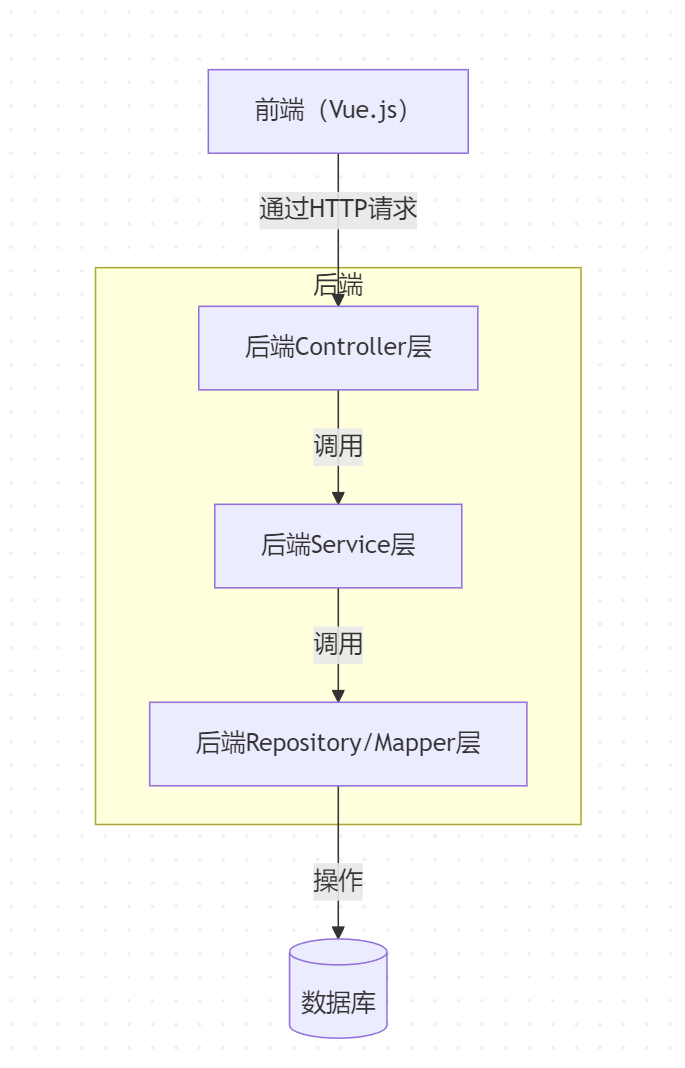
职责：处理业务逻辑，接收前端请求，调用数据访问层，返回处理结果。

技术栈：Spring Boot + Spring MVC。

数据访问层（后端Repository/Mapper）：

职责：与数据库进行交互，完成数据的增删改查。

技术栈：MyBatis。



2. 结构划分原则

2.1 遵循MVC分层思想

解耦前后端：前后端分离，前端专注于用户界面和交互，后端专注于业务逻辑和数据处理。

便于维护和扩展：各层之间职责分明，便于独立开发和维护，支持灵活的功能扩展。

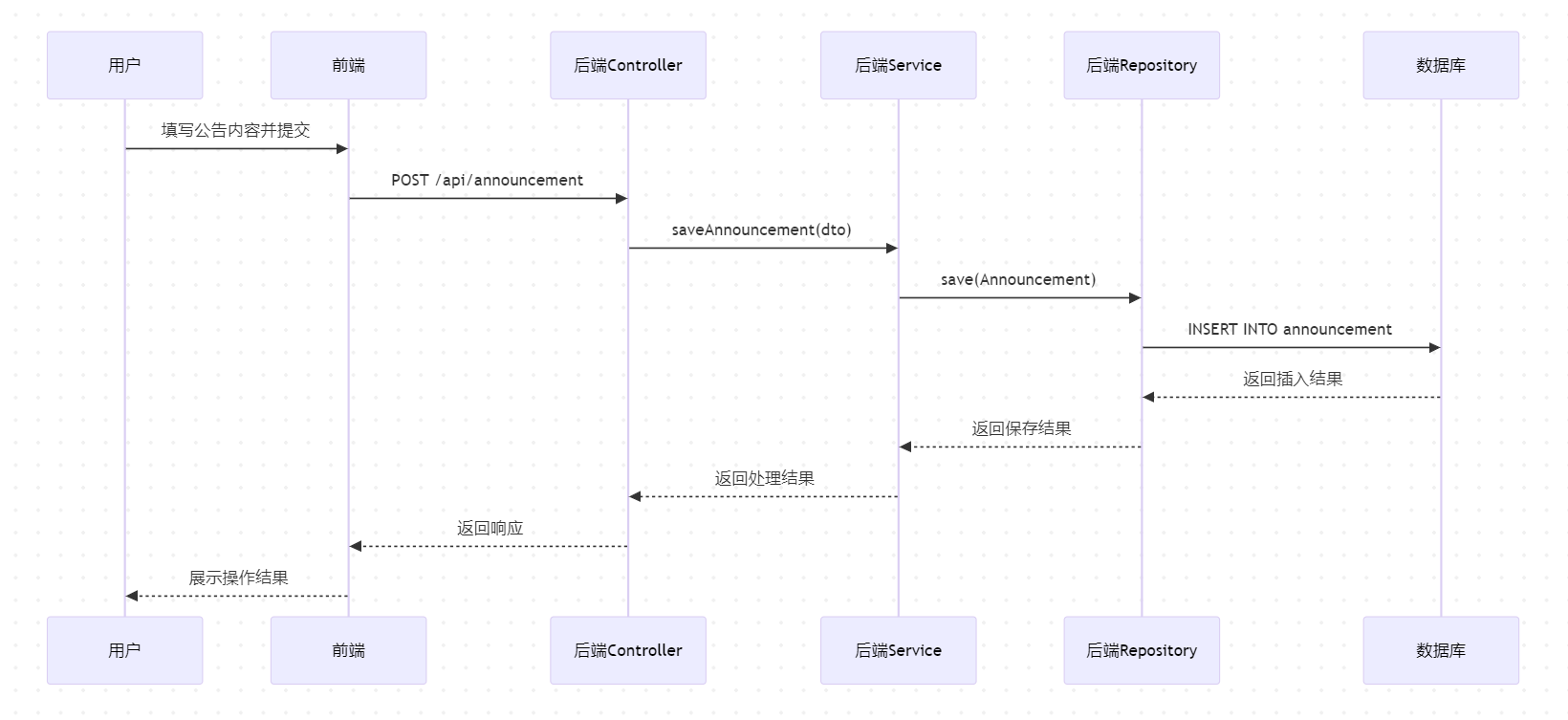
2.2 后端接口遵循RESTful规范

接口设计：后端接口采用RESTful风格，通过HTTP方法（GET、POST、PUT、DELETE等）实现资源的增删改查。

数据交互：前端通过 Axios 等工具调用后端API，实现数据的获取和提交。

二、业务流程说明

以发布公告为例说明前后端动态交互



## 3.2 分解描述

一、**社团模块**

1. Overview简介

社团模块负责社团的基本信息管理、成员管理、社团主页展示等，是社团运营的核心模块。

2．Functions功能列表

社团信息管理：创建、编辑社团基本信息。

社团成员管理：成员加入、退出社团，管理员审核成员。

社团主页展示：展示社团介绍、活动、公告等内容。

社团公告：发布社团内部公告。

社团风采展示：上传社团风采图片、编辑社团信息等资料。

二、**社团活动模块**

1. Overview简介

社团活动模块负责社团活动的全流程管理，包括活动的创建、报名、审核、查询和详情展示。该模块服务于社团管理员和普通成员，支持活动的线上组织与参与。

2．Functions功能列表

活动创建：社团管理员发起新活动。

活动报名：成员报名参加活动。

活动审核：对报名成员进行资格审核（如有需要）。

活动列表查询：按条件检索活动信息。

活动详情查看：展示活动详细内容与参与情况。

活动状态管理：支持活动的发布、取消、结束等状态变更。

**三、论坛模块**

1. Overview简介

论坛模块为社团成员提供交流互动平台，支持发帖、评论、点赞、举报等功能，促进成员间的信息共享与讨论。

2．Functions功能列表

帖子发布：成员可发布主题帖。

帖子浏览：浏览、检索所有帖子。

评论回复：对帖子进行评论、回复。

点赞功能：对帖子、评论点赞。

举报功能：对不当内容进行举报。

帖子/评论管理：管理员可删除违规内容。

**四、后台管理模块**

1. Overview简介

后台管理模块为系统管理员和社团管理员提供统一的管理入口，涵盖公告管理、用户管理、数据统计等功能，保障系统的有序运行。

2．Functions功能列表

公告管理：发布、编辑、删除系统公告。

用户管理：查看、禁用、审核用户账号。

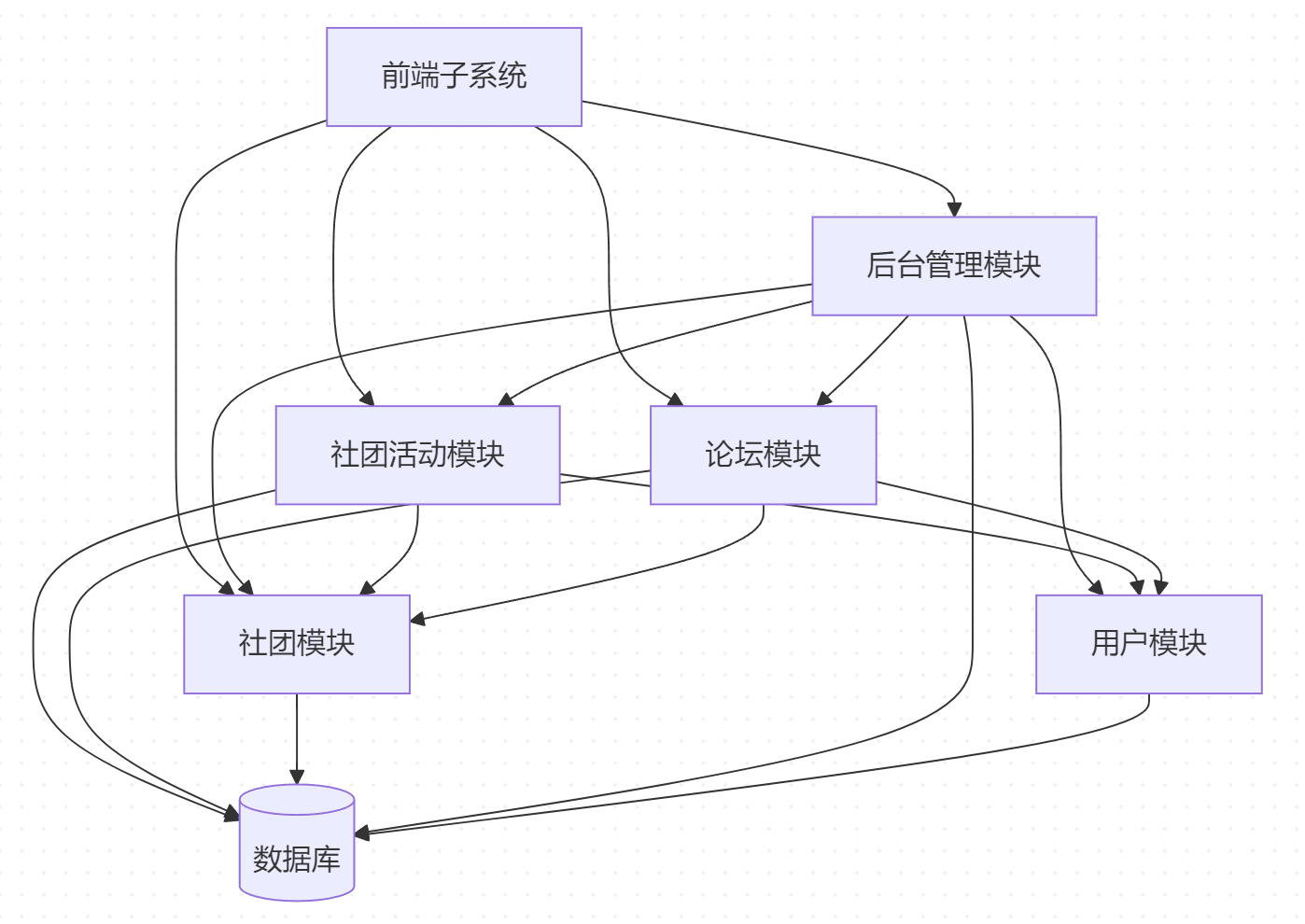
权限分配：分配和调整用户、社团的权限。

数据统计：统计社团、活动、用户等各类数据。

文件上传：支持图片、文档等资源的上传与管理。

## 3.3 依赖性描述

## 



## 3.4 接口描述

一、社团活动模块

1.1 创建活动接口

名称：创建活动

说明：社团管理员发起新活动

定义：URL：POST /api/activity；请求参数（JSON）：name：活动名称（String）；description：活动描述（String）；clubId：所属社团ID（Long）；startTime：开始时间（String/DateTime）；endTime：结束时间（String/DateTime）；返回值：操作结果（如 { code, message, data }）

1.2 活动报名接口

名称：报名活动

说明：用户报名参加指定活动

定义：URL：POST /api/activity/join；请求参数（JSON）：activityId：活动ID（Long）；userId：用户ID（Long）；返回值：操作结果

1.3 活动列表查询接口

名称：获取活动列表

说明：查询所有或指定条件的活动信息

定义：URL：GET /api/activity/list；请求参数：可选（如 clubId、status 等，Query 参数）；返回值：活动列表（List<ActivityVO>）

二、后台管理模块

2.1 发布公告接口

名称：发布公告

说明：管理员发布系统公告

定义：URL：POST /api/admin/announcement；请求参数（JSON）：title：公告标题（String）；content：公告内容（String）；type：公告类型（如全局/社团）（String）；返回值：操作结果

2.2 用户管理接口

名称：获取用户列表

说明：管理员查询所有用户信息

定义：URL：GET /api/admin/user/list；请求参数：可选（如分页、筛选条件，Query 参数）；返回值：用户列表（List<UserVO>）

2.3 文件上传接口

名称：上传文件

说明：上传图片、文档等资源

定义：URL：POST /api/admin/upload；请求参数：file（Multipart 文件）；返回值：文件URL或上传结果

三、论坛模块

3.1 发布帖子接口

名称：发布帖子

说明：用户在论坛发布新主题帖

定义：URL：POST /api/forum/post；请求参数（JSON）：title：帖子标题（String），content：帖子内容（String），clubId：所属社团ID（Long），userId：发帖人ID（Long）；返回值：操作结果

3.2 评论接口

名称：发表评论

说明：用户对帖子发表评论

定义：URL：POST /api/forum/comment；请求参数（JSON）：postId：帖子ID（Long）content：评论内容（String）userId：评论人ID（Long）；返回值：操作结果

3.3 点赞接口

名称：点赞

说明：用户对帖子或评论点赞

定义：URL：POST /api/forum/like；请求参数（JSON）：targetId：目标ID（帖子或评论ID，Long）；type：类型（帖子/评论，String）；userId：用户ID（Long）；返回值：操作结果

四、社团模块

4.1 创建社团接口

名称：创建社团

说明：用户或管理员创建新社团

定义：URL：POST /api/club；请求参数（JSON）：name：社团名称（String）；description：社团简介（String）；creatorId：创建人ID（Long）；返回值：操作结果

4.2 加入社团接口

名称：加入社团

说明：用户申请加入指定社团

定义：URL：POST /api/club/join；请求参数（JSON）：clubId：社团ID（Long）；userId：用户ID（Long）；返回值：操作结果

4.3 社团信息查询接口

名称：获取社团信息

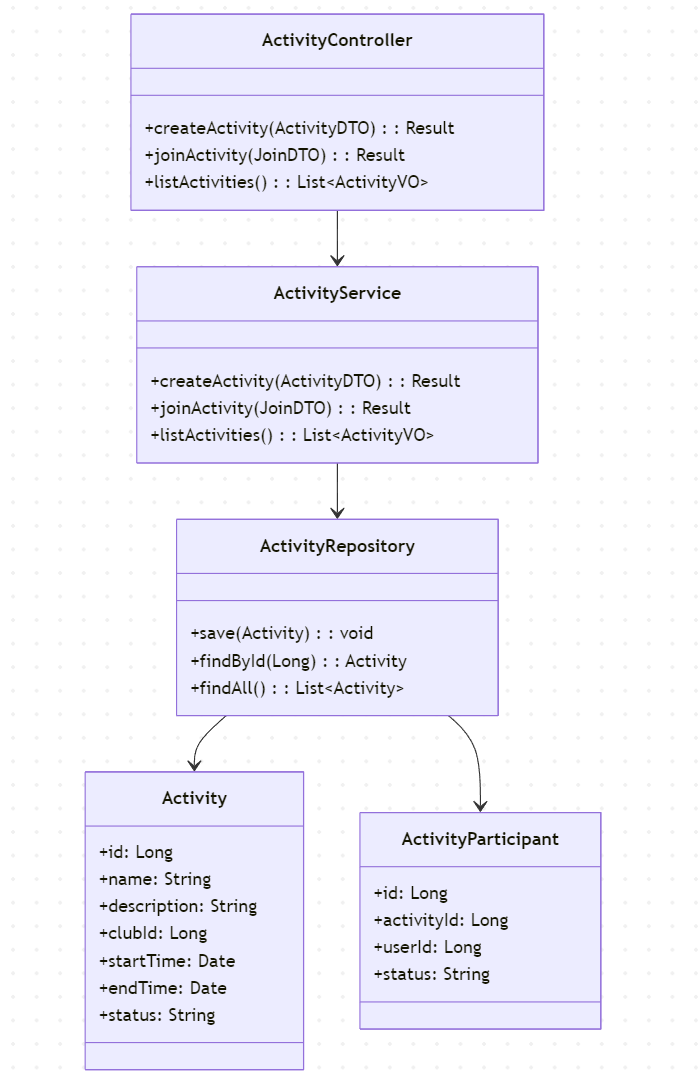
说明：查询指定社团的详细信息

定义：URL：GET /api/club/{clubId}；请求参数：clubId（Path 参数）；返回值：社团详情（ClubDetailVO）

# 第二层设计描述

## 4.1 社团活动模块

一、模块设计描述



活动管理：活动的创建、编辑、删除、查询。

活动报名：用户报名、取消报名、报名审核。

活动状态管理：活动发布、结束、取消等状态变更。

**1、ActivityController**

（1）标识：UClub\_Activity\_ActivityController

（2）简介：处理与活动相关的HTTP请求，负责参数校验和结果返回。

（3）类定义：

public Result createActivity(ActivityDTO dto)

public Result joinActivity(JoinDTO dto)

public List<ActivityVO> listActivities()

**2、ActivityService**

（1）标识：UClub\_Activity\_ActivityService

（2）简介：封装活动业务逻辑，处理活动的创建、报名、查询等。

（3）类定义：

public Result createActivity(ActivityDTO dto)

public Result joinActivity(JoinDTO dto)

public List<ActivityVO> listActivities()

**3、ActivityRepository**

（1）标识：UClub\_Activity\_ActivityRepository

（2）简介：负责活动相关的数据持久化操作。

（3）类定义：

public void save(Activity activity)

public Activity findById(Long id)

public List<Activity> findAll()

**4、Activity**

（1）标识：UClub\_Activity\_Activity

（2）简介：活动实体类，映射数据库活动表。

（3）类定义：

public Long id

public String name

public String description

public Long clubId

public Date startTime

public Date endTime

public String status

**5、ActivityParticipant**

**（1）标识：UClub\_Activity\_ActivityParticipant**

**（2）简介：活动参与者实体类，映射数据库活动报名表。**

**（3）类定义：**

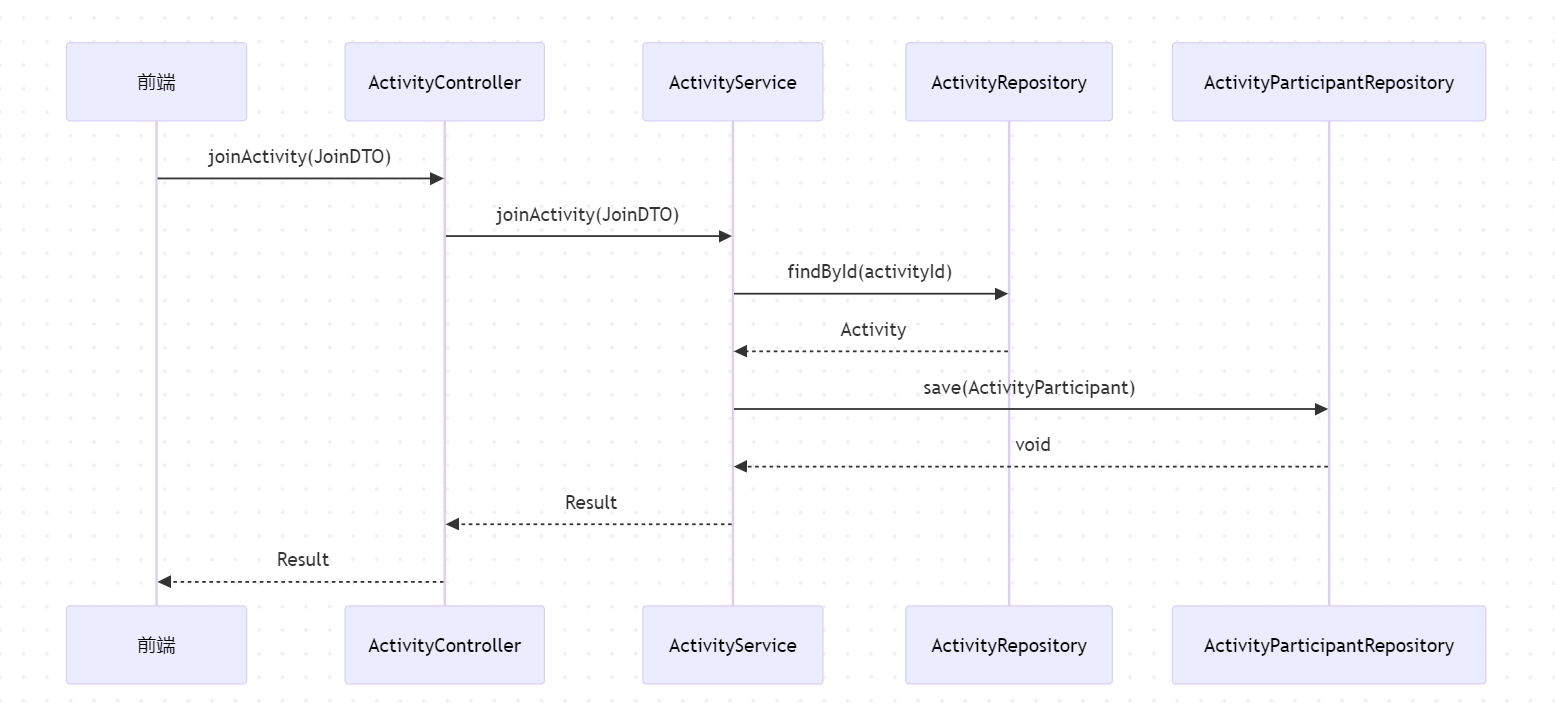
**public Long id**

**public Long activityId**

**public Long userId**

**public String status**

二、功能实现说明

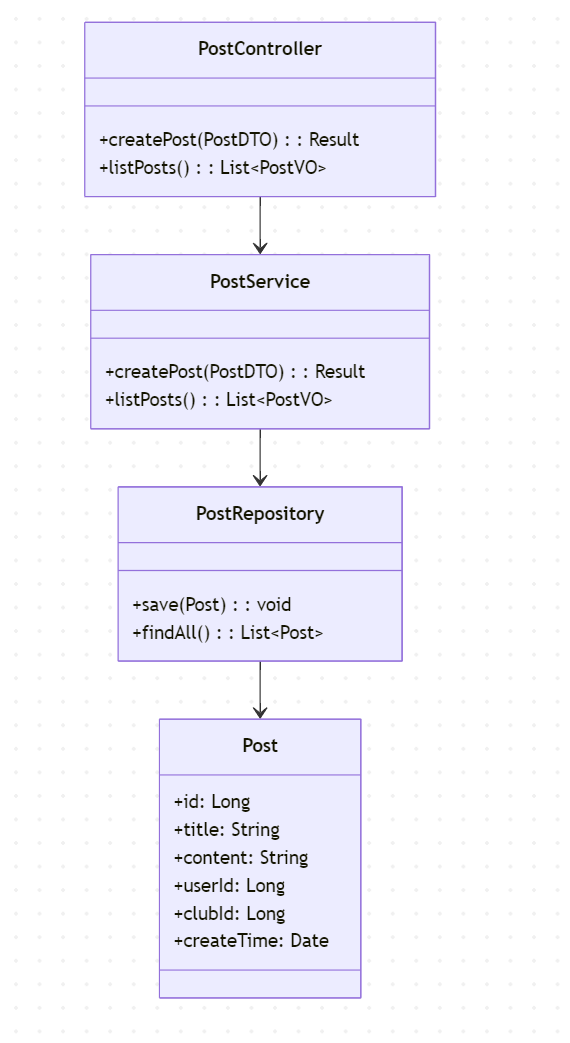


参与活动

## 4.2 论坛模块

**一、模块设计描述**

**1. 子模块分解**



帖子管理：发帖、删帖、查帖。

评论管理：评论、回复、删除。

点赞举报：点赞、举报功能。

**2. 主要设计类说明**

**1、PostController**

（1）标识：UClub\_Forum\_PostController

（2）简介：处理帖子相关的HTTP请求。

（3）类定义：

public Result createPost(PostDTO dto)

public List<PostVO> listPosts()

**2、PostService**

（1）标识：UClub\_Forum\_PostService

（2）简介：帖子业务逻辑。

（3）类定义：

public Result createPost(PostDTO dto)

public List<PostVO> listPosts()

**3、PostRepository**

（1）标识：UClub\_Forum\_PostRepository

（2）简介：帖子数据持久化操作。

（3）类定义：

public void save(Post post)

public List<Post> findAll()

**4、Post**

（1）标识：UClub\_Forum\_Post

（2）简介：帖子实体类。

（3）类定义：

public Long id

public String title

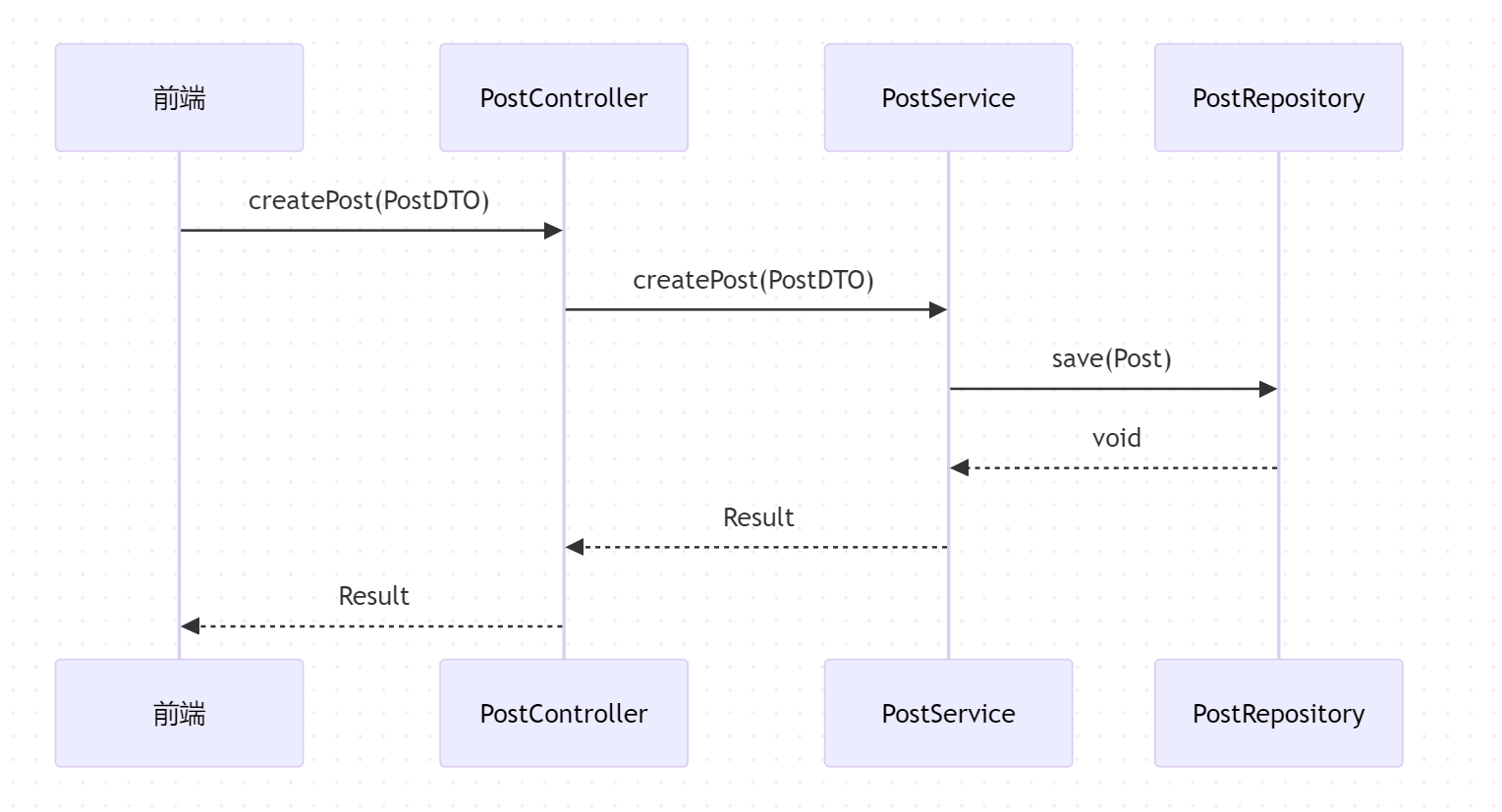
public String content

public Long userId

public Long clubId

public Date createTime

二、功能实现说明

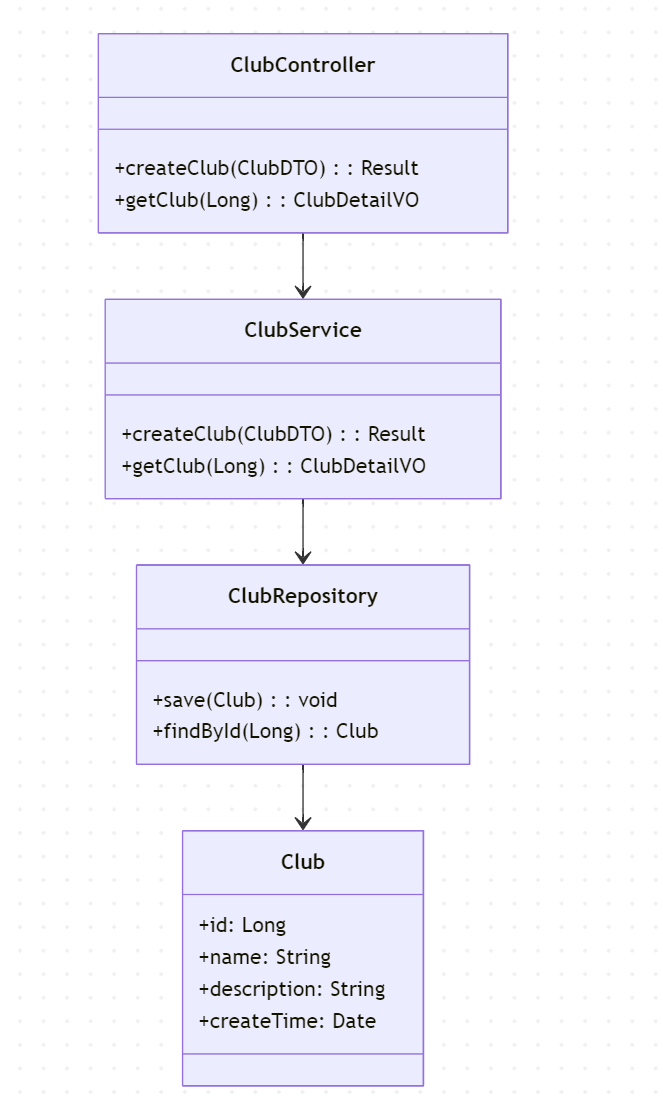


发帖功能

#### 4.3 社团模块

**一、模块设计描述**

**1. 子模块分解**



社团信息管理：创建、编辑、查询社团信息。

社团成员管理：成员加入、退出、审核。

社团主页展示：社团风采、公告等。

**2. 主要设计类说明**

**1、ClubController**

（1）标识：UClub\_Club\_ClubController

（2）简介：处理社团相关的HTTP请求。

（3）类定义：

public Result createClub(ClubDTO dto)

public ClubDetailVO getClub(Long id)

**2、ClubService**

（1）标识：UClub\_Club\_ClubService

（2）简介：社团业务逻辑。

（3）类定义：

public Result createClub(ClubDTO dto)

public ClubDetailVO getClub(Long id)

**3、ClubRepository**

（1）标识：UClub\_Club\_ClubRepository

（2）简介：社团数据持久化操作。

（3）类定义：

public void save(Club club)

public Club findById(Long id)

**4、Club**

（1）标识：UClub\_Club\_Club

（2）简介：社团实体类。

（3）类定义：

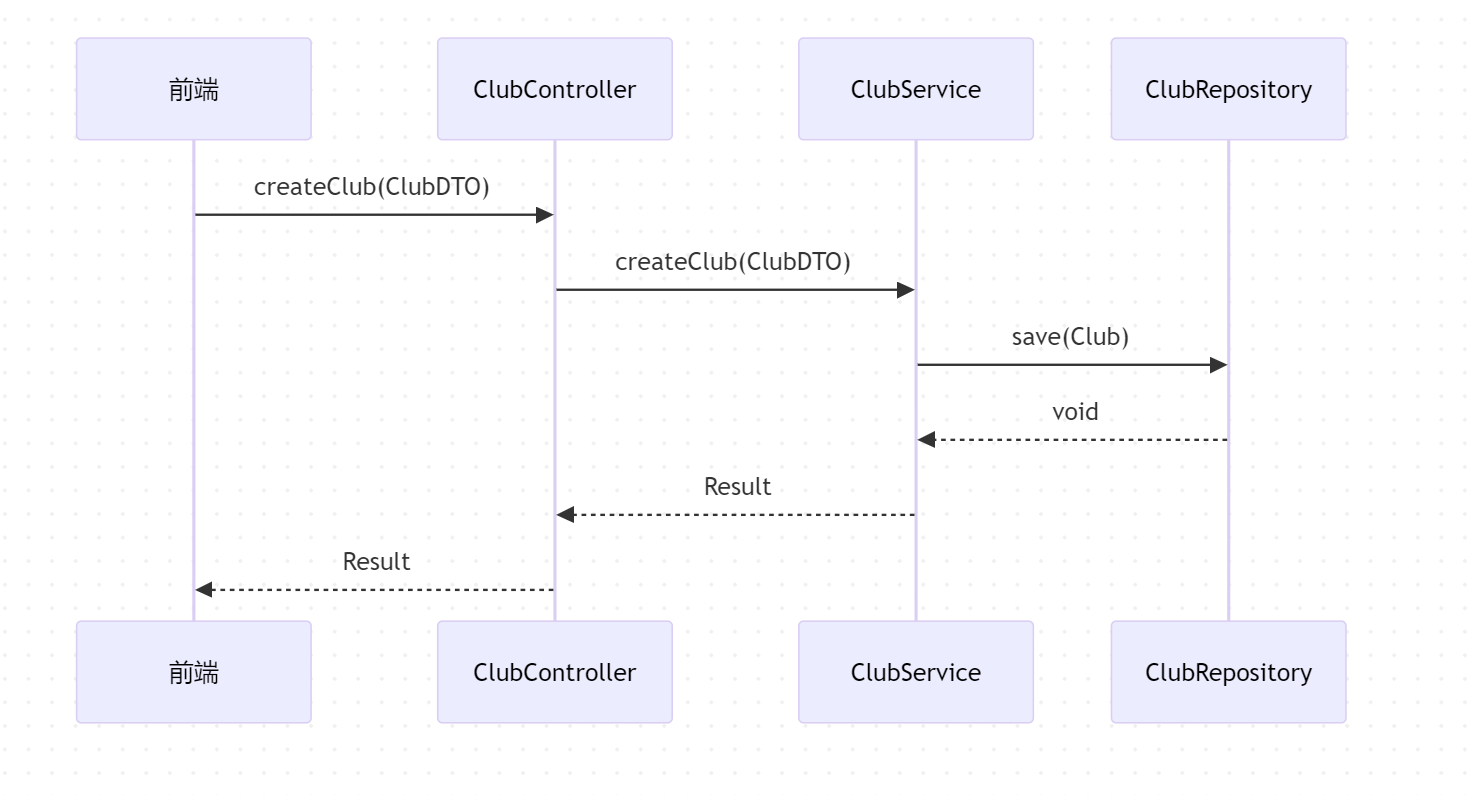
public Long id

public String name

public String description

public Date createTime

二、功能实现说明



创建社团功能

# 数据库设计

## 5.1 实体定义

User表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 约束 | 说明 |
| id | int | 主键，自增 | 用户ID |
| email | varchar(100) | 唯一，非空 | 邮箱地址 |
| password | varchar(100) | 非空 | 加密密码 |
| nickname | varchar(50) | 可空 | 昵称 |
| avatar\_url | varchar(255) | 可空 | 头像URL |
| role | enum('普通用户', '社团管理员', '系统管理员') | 默认'普通用户' | 用户角色 |
| status | enum('正常', '禁言', '封禁') | 默认'正常' | 账号状态 |
| created\_at | datetime | 默认当前时间 | 创建时间 |

Club表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 约束 | 说明 |
| id | int | 主键，自增 | 社团ID |
| name | varchar(100) | 非空 | 社团名称 |
| logo\_url | varchar(255) | 可空 | 社团Logo |
| tags | varchar(100) | 可空 | 标签（逗号分隔） |
| creator\_id | int | 外键，非空 | 创建人用户ID |
| description | text | 可空 | 社团简介 |
| status | enum('正常', '待审核', '已封禁') | 默认'待审核' | 社团状态 |
| created\_at | datetime | 默认当前时间 | 创建时间 |

Club\_member表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **约束** | **说明** |
| id | int | 主键，自增 | 记录ID |
| user\_id | int | 外键，非空 | 用户ID |
| club\_id | int | 外键，非空 | 社团ID |
| role | enum('成员', '干事', '副社长', '社长') | 默认'成员' | 社内角色 |
| join\_status | enum('待审核', '已通过', '已拒绝') | 默认'待审核' | 加入状态 |
| joined\_at | datetime | 默认当前时间 | 加入时间 |

Club\_activity表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **约束** | **说明** |
| id | int | 主键，自增 | 活动ID |
| club\_id | int | 外键，非空 | 所属社团ID |
| title | varchar(100) | 非空 | 活动标题 |
| description | text | 可空 | 活动内容简介 |
| location | varchar(100) | 可空 | 活动地点 |
| start\_time | datetime | 非空 | 开始时间 |
| end\_time | datetime | 非空 | 结束时间 |
| max\_participants | int | 可空 | 最大参与人数 |
| apply\_status | enum('待审核', '通过', '拒绝') | 默认'待审核' | 活动审核状态 |
| creator\_id | int | 外键，非空 | 发起人ID |
| created\_at | datetime | 默认当前时间 | 创建时间 |

Announcement表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **约束** | **说明** |
| id | int | 主键，自增 | 公告ID |
| club\_id | int | 外键，可空 | 社团ID（为NULL表示系统公告） |
| title | varchar(100) | 非空 | 公告标题 |
| type | enum('系统', '社团') | 非空 | 公告类型 |
| content | text | 非空 | 公告内容 |
| creator\_id | int | 外键，非空 | 发布人ID |
| created\_at | datetime | 默认当前时间 | 发布时间 |

Post表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **约束** | **说明** |
| id | bigint | 主键，自增 | 帖子ID |
| club\_id | bigint | 非空 | 社团ID |
| user\_id | bigint | 非空 | 用户ID |
| title | varchar(100) | 非空 | 标题 |
| content | text | 非空 | 内容 |
| image\_urls | json | 可空 | 图片URL数组 |
| status | enum('active', 'deleted', 'hidden', 'violated') | 默认'active' | 状态 |
| like\_count | int | 默认0 | 点赞数 |
| comment\_count | int | 默认0 | 评论数 |
| created\_at | datetime | 默认当前时间 | 创建时间 |

Comment表(评论)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 约束 | 说明 |
| id | bigint | 主键，自增 | 评论ID |
| post\_id | bigint | 非空 | 帖子ID |
| user\_id | bigint | 非空 | 用户ID |
| content | text | 非空 | 内容 |
| status | enum('active', 'deleted', 'hidden', 'violated') | 默认'active' | 状态 |
| like\_count | int | 默认0 | 点赞数 |
| created\_at | datetime | 默认当前时间 | 创建时间 |

Like表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 约束 | 说明 |
| id | bigint | 主键，自增 | 点赞ID |
| user\_id | bigint | 非空 | 用户ID |
| target\_type | enum('post', 'comment') | 非空 | 目标类型 |
| target\_id | bigint | 非空 | 目标ID |
| created\_at | datetime | 默认当前时间 | 点赞时间 |
| 唯一约束 | user\_id + target\_type + target\_id |  | 防止重复点赞 |

Report表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 约束 | 说明 |
| id | int | 主键，自增 | 举报ID |
| reporter\_id | int | 外键，非空 | 举报人ID |
| target\_type | enum('帖子', '评论', '用户', '活动', '公告') | 非空 | 举报对象类型 |
| target\_id | int | 非空 | 对应对象ID |
| reason | text | 可空 | 举报理由 |
| status | enum('待处理', '已处理') | 默认'待处理' | 处理状态 |
| created\_at | datetime | 默认当前时间 | 举报时间 |

Activity\_participant表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 约束 | 说明 |
| id | int | 主键，自增 | 记录ID |
| activity\_id | int | 非空 | 活动ID |
| join\_time | datetime(6) | 可空 | 参与时间 |
| status | enum('已参加', '已退出') | 默认'已参加' | 参与状态 |
| user\_id | int | 非空 | 用户ID |

Chat\_message表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 约束 | 说明 |
| id | Bigint | 主键，自增 | 消息ID |
| room | varchar(50) | 非空 | 聊天室ID，如"public"或"club-1" |
| sender\_id | bigint | 非空 | 发送者用户ID |
| sender\_name | varchar(100) | 可空 | 发送者昵称 |
| avatar\_url | varchar(255) | 可空 | 发送者头像URL |
| content | text | 非空 | 消息内容 |
| role | varchar(20) | 可空 | 发送者角色，如"社长" |
| time | datetime | 非空，默认当前时间戳 | 消息发送时间 |

1. 分解描述

设计思路：

1. 以用户为核心，围绕社团业务展开

整体数据库以 user（用户表） 为核心，所有社团、活动、公告、帖子、评论、消息等业务均与用户相关联，保证了数据的可追溯性和权限管理的灵活性。

2. 实体分明，职责清晰

每个表只负责一类业务数据，避免冗余。例如：

club：只存储社团基本信息。

club\_member：只记录成员与社团的关系。

club\_activity：只记录活动。

3. 采用枚举类型管理状态

例如：

用户角色（普通用户、社团管理员、系统管理员）

账号状态（正常、禁言、封禁）

社团状态（正常、待审核、已封禁）

活动审核状态（待审核、通过、拒绝）

均采用 enum 类型，便于业务逻辑判断和数据一致性校验。

4. 外键约束保证数据完整性

关键字段（如用户ID、社团ID、活动ID等）均设置外键，保证引用数据的存在性，防止“脏数据”产生。例如：

club.creator\_id → user.id

club\_member.user\_id → user.id

5. 索引优化查询性能

对高频查询字段（如 user\_id、club\_id、post\_id、room 等）建立索引，提升数据检索效率，适应高并发场景。

6. 约束规则

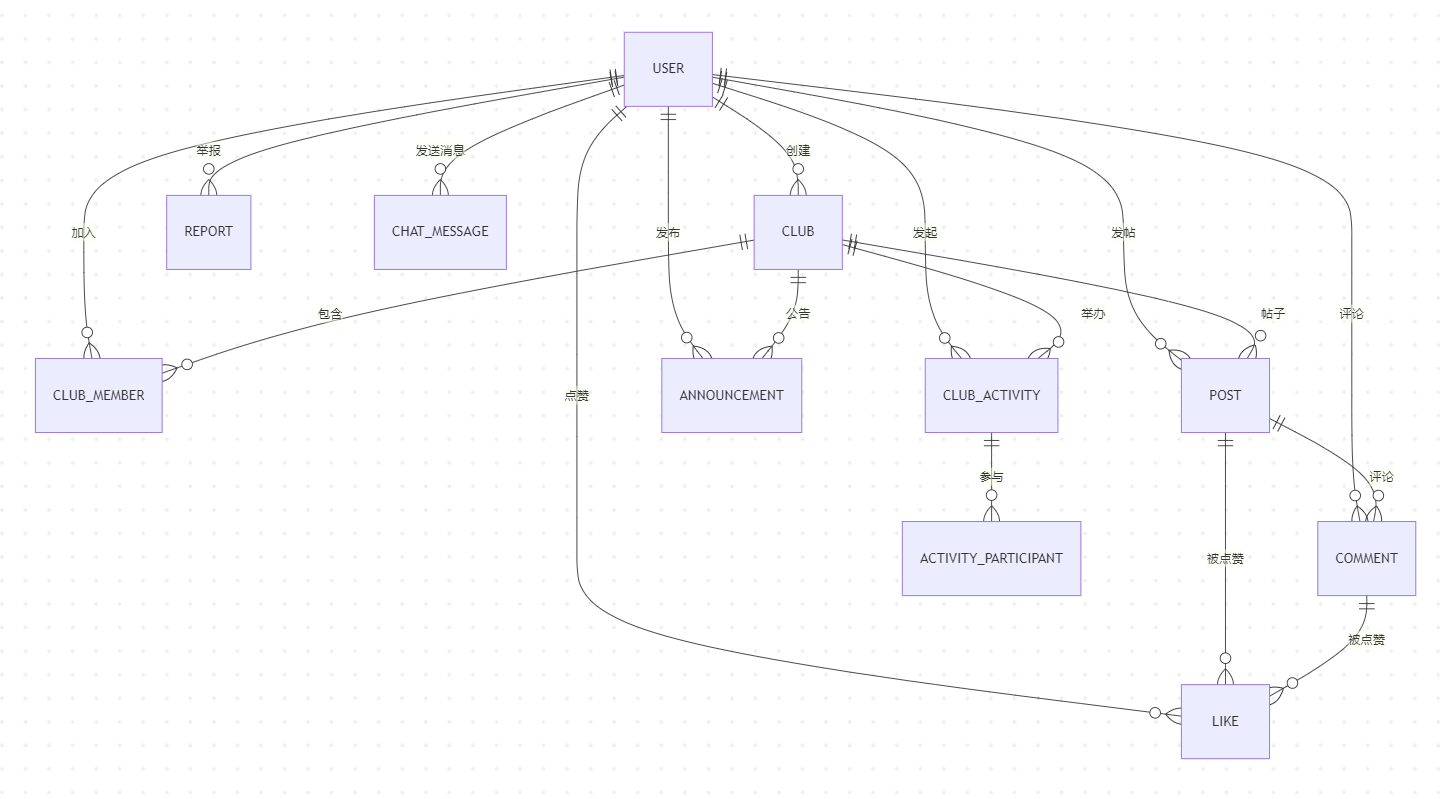
主键约束：每个表均有自增主键，保证记录唯一性。

唯一约束：user.email 唯一；like 表对 (user\_id, target\_type, target\_id) 设唯一索引，防止重复点赞

外键约束：关键业务表均有外键，保证数据引用的有效性，并设置级联删除/更新或置空，防止数据孤岛

非空约束：业务关键字段（如 user\_id、club\_id、content、room 等）均设为非空，保证数据完整。

二、内部依赖性描述



## 5.2 行为定义

一、分解描述

1. 存储过程/触发器分类

* 活动相关存储过程/触发器：主要负责活动的创建、报名、删除等操作。
* 公告相关存储过程/触发器：主要负责公告的发布、删除等操作。
* 社团相关存储过程/触发器：主要负责社团的创建、成员管理、删除等操作。
* 论坛相关存储过程/触发器：主要负责帖子、评论的发布、删除等操作。

1. 典型存储过程/触发器定义

1. 活动相关存储过程/触发器

1.1 活动报名存储过程

* 功能：为指定用户报名指定活动，若已报名则返回提示。
* 输入参数：
* IN p\_activity\_id BIGINT：活动ID
* IN p\_user\_id BIGINT：用户ID
* 输出参数：
* OUT p\_result INT：操作结果（0=成功，1=已报名，2=活动不存在等）
* 返回值/记录集：无
* 依赖数据表：activity\_participant, activity
* 依赖存储过程：无
* 特殊要求：需启用事务，保证报名操作的原子性。

1.2 活动删除触发器

* 功能：删除活动时，自动删除与该活动相关的报名记录。
* 触发时机：AFTER DELETE on activity
* 输入输出参数：无
* 返回值/记录集：无
* 依赖数据表：activity\_participant
* 依赖存储过程：无
* 特殊要求：需保证级联删除，防止数据残留。

2. 公告相关存储过程/触发器

2.1 公告发布存储过程

* 功能：发布新公告，插入公告表并可选插入附件表。
* 输入参数：
* IN p\_title VARCHAR(255)：公告标题
* IN p\_content TEXT：公告内容
* IN p\_type VARCHAR(50)：公告类型
* 输出参数：
* OUT p\_announcement\_id BIGINT：新公告ID
* 返回值/记录集：无
* 依赖数据表：announcement, announcement\_attachment
* 依赖存储过程：无
* 特殊要求：需启用事务，保证主表和附件表数据一致。

3. 社团相关存储过程/触发器

3.1 创建社团存储过程

* 功能：创建新社团，插入社团表并添加创建人为社团管理员。
* 输入参数：
* IN p\_name VARCHAR(255)：社团名称
* IN p\_description TEXT：社团简介
* IN p\_creator\_id BIGINT：创建人ID
* 输出参数：
* OUT p\_club\_id BIGINT：新社团ID
* 返回值/记录集：无
* 依赖数据表：club, club\_member
* 依赖存储过程：无
* 特殊要求：需启用事务，保证社团和成员表数据一致。

3.2 删除社团成员存储过程

* 功能：将指定用户从社团成员表中移除，并可级联删除其在该社团下的活动报名、发帖等。
* 输入参数：
* IN p\_club\_id BIGINT
* IN p\_user\_id BIGINT
* 输出参数：无
* 返回值/记录集：无
* 依赖数据表：club\_member, activity\_participant, forum\_post
* 依赖存储过程：可调用proc\_delete\_activity\_participant、proc\_delete\_forum\_post\_by\_user
* 特殊要求：需启用事务，保证级联删除一致性。

4. 论坛相关存储过程/触发器

4.1 发布帖子存储过程

* 功能：插入新帖子记录。
* 输入参数：
* IN p\_title VARCHAR(255)：帖子标题
* IN p\_content TEXT：帖子内容
* IN p\_user\_id BIGINT：发帖人ID
* IN p\_club\_id BIGINT：所属社团ID
* 输出参数：
* OUT p\_post\_id BIGINT：新帖子ID
* 返回值/记录集：无
* 依赖数据表：post
* 依赖存储过程：无
* 特殊要求：无

4.2 删除帖子触发器

* 功能：删除帖子时，自动删除其下所有评论和点赞记录。
* 触发时机：AFTER DELETE on post
* 输入输出参数：无
* 返回值/记录集：无
* 依赖数据表：comment, like
* 依赖存储过程：无
* 特殊要求：需保证级联删除。

二、外部依赖性描述

1. 与业务逻辑层（Service）的依赖

* 存储过程/触发器的调用通常由后端Service层发起。Service层负责业务参数校验、权限校验、事务控制等，调用存储过程实现批量写入、复杂删除等操作。
* 例如，活动报名、社团成员删除等操作，Service层会根据业务逻辑决定是否调用相关存储过程。

1. 与前端模块的依赖

* 前端通过API请求触发后端的写入、删除操作，间接影响存储过程的执行。例如，用户在前端点击“报名活动”按钮，最终会触发后端调用报名存储过程。
* 删除操作（如删除公告、帖子）也由前端发起，后端通过API接口调用相关存储过程或触发器。

1. 与权限/认证模块的依赖

* 存储过程的执行前，通常需要业务层校验用户权限（如是否为管理员、是否为社团成员等），以防止非法操作。
* 例如，只有社团管理员才能调用“删除活动”相关存储过程。

三、内部依赖性描述

1. 存储过程之间的依赖

某些存储过程会调用其他存储过程以实现复杂的业务逻辑。例如，删除社团成员的存储过程可能会调用“删除活动报名”、“删除论坛发帖”等子过程，实现级联删除。

1. 存储过程与数据表/视图的依赖

存储过程直接操作相关数据表，实现数据的插入、更新、删除。例如，活动报名过程操作activity\_participant表，公告删除过程操作announcement表。

触发器则在表的写入、删除等事件发生时自动执行，维护数据一致性。例如，删除帖子时自动删除comment、like表中相关数据。

1. 存储过程与视图的依赖

某些存储过程可能会读取视图（View）以获取统计信息或联合查询结果。例如，统计某社团成员报名活动数时，可能会用到v\_club\_activity\_participation视图。

# 原型或界面设计

