**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于jQuery的游戏论坛社区管理系统的分析与设计 | | |
| 课 程 | 软件工程 | | |
| 学 院 | 信息工程学院 | | |
| 专 业 | 软件工程 | 年级 | 2023级 |
| 学生姓名 | 周发浪 | 学号 | 112322100006 |

|  |  |  |  |
| --- | --- | --- | --- |
| **评分项** | **评分标准** | **满分** | **得分** |
| 文档整体 | 文档内容详实、规范，美观大方，内容: 可行性分析、需求分析、概要设计、详细设计。 | 20 |  |
| 用例图及规约 | 用例图完整，准确，能够完全体现需求 | 10 |  |
| 顺序图与活动图 | 顺序图能够完全描述用例的设计思路和业务流程，活动图能够描述部分算法的流程。 | 10 |  |
| 类图 | 类图能够完整，准确反映业务的需数据的组织结构 | 10 |  |
| 数据库 | 数据库表的设计来自于类图，合理 | 10 |  |
| 界面设计 | 界面设计美观，清晰，合理，能够完全反映用例图的设计思路 | 10 |  |
| API接口设计 | 为每个界面设计合理的API接口，以便进行前后端分离开发 | 15 |  |
| 内容一致性 | 用例图，类图、数据库、界面相互印证，相互依赖，环环相扣 | 15 |  |
| **得分合计** | | |  |

**2024年12月25号**

# Galgame游戏社区管理系统绪论

随着互联网的飞速发展，游戏产业呈现出爆发式增长，游戏社区作为玩家聚集、交流、分享的重要平台，其规模和复杂性也日益提升。为了更好地运营游戏社区，满足玩家需求，提升管理效率，开发一款功能完善的游戏社区管理系统迫在眉睫。

## 一、系统目标

1. 提升用户体验：为玩家打造一个便捷、友好的交流环境，使玩家能够轻松找到感兴趣的话题、群组，快速获取游戏资讯，如攻略、更新公告等，增强玩家对游戏社区的粘性。

2. 高效管理社区：帮助管理员对社区内容进行精准管控，包括但不限于帖子审核、用户违规处理，确保社区氛围健康、积极，维护社区秩序。

3. 数据驱动决策：系统能够收集并分析用户行为数据、社区活跃度数据等，为游戏运营方提供决策依据，以便针对性地优化游戏、举办活动，促进游戏的长期发展。

4. 增强互动性：支持多种互动形式，如私信、评论、点赞等，鼓励玩家之间、玩家与管理员之间的交流，形成活跃的社区生态。

## 二、系统功能

1 .用户管理：涵盖用户注册、登录、信息修改、账号冻结与解封等功能，同时对用户进行分类管理，如普通玩家、版主、管理员等，不同角色拥有不同权限。

2 .内容管理：包括帖子发布、编辑、删除，评论管理，以及对优质内容的推荐与置顶。支持图片、视频、文字等多种形式的内容展示，满足玩家多样化的分享需求。

1. 社区运营：具备活动发布、通知推送功能，管理员可根据游戏节点、节日等策划线上线下活动，并及时告知玩家。还能对社区板块进行灵活划分与调整，适应不同游戏主题与玩家兴趣方向。
2. 数据统计与分析：统计用户在线时长、发帖频率、热门话题等数据，以可视化图表呈现，让运营团队直观了解社区动态，洞察玩家喜好。
3. 社交互动：私信功能保障玩家间的私密交流，好友系统方便玩家建立社交关系，动态点赞、评论让交流更具实时性与趣味性。

## 三、硬件要求

1. 服务器：选用高性能服务器，处理器建议采用多核架构，如 Intel Xeon 系列，以应对大量用户并发访问时的数据处理需求；内存容量应不低于 16GB，保障系统运行流畅，避免卡顿，随着用户量增长可按需扩展；存储方面，配备大容量高速硬盘，采用 RAID 技术保证数据冗余与读写速度，初始存储容量可规划为 1TB 及以上。

2 网络设备：配备千兆级以上网络交换机，确保内部网络高速稳定传输数据；防火墙设备必不可少，抵御外部网络攻击，保障社区数据安全，选择具备入侵检测、防 DDoS 攻击等功能的专业防火墙产品。

3 客户端：对于玩家使用的终端设备，要求具备基本的网络连接能力，如智能手机需支持 4G 及以上网络，屏幕分辨率能适配主流游戏社区 APP 布局，电脑端则要求处理器性能不低于 Intel Core i3 系列，内存 4GB 以上，显卡能正常渲染网页图形元素，以保证良好的浏览与交互体验。

综上所述，游戏社区管理系统旨在通过明确的目标设定、丰富实用的功能模块以及适配的硬件支持，为游戏社区的繁荣发展提供坚实保障，满足游戏产业日益增长的社区运营需求。

## 四、技术框架

1 后端框架：采用 Spring Boot 框架，它提供了便捷的开发体验，能够快速搭建稳定、高效的后端服务。通过自动配置、起步依赖等特性，大大减少了繁琐的初始配置工作，让开发人员专注于业务逻辑的实现。其内置的 Tomcat 服务器，方便项目的部署与运行，轻松应对高并发请求，为游戏社区管理系统的后端数据处理与接口服务提供坚实支撑。

2 前端框架：选用 Vue.js，它是一款流行的渐进式 JavaScript 框架，能够构建用户界面，实现高效的组件化开发。利用 Vue 的双向数据绑定机制，实时更新页面数据，给玩家带来流畅的交互体验。配合 Vue Router 进行前端路由管理，实现页面的无刷新跳转，优化用户操作流程，适配游戏社区多样化的页面需求。

3 数据库：使用 MySQL 关系型数据库，它具有性能稳定、可靠性高的特点，能满足大规模数据的存储与查询需求。支持事务处理，确保数据的完整性和一致性，在用户管理、内容管理等模块的数据持久化操作中表现出色，为系统提供可靠的数据存储保障。

## 五、技术分析

1 前后端分离优势：采用前后端分离的架构模式，前端专注于用户界面展示与交互逻辑，后端负责数据处理与接口提供，两者通过 API 进行通信。这种模式使得开发团队分工明确，提高开发效率；同时便于后期维护与扩展，无论是前端界面的更新换代，还是后端业务逻辑的调整优化，都不会对另一方造成过大影响，降低系统迭代成本。

2 数据缓存优化：考虑到游戏社区频繁的数据读取需求，引入 Redis 缓存数据库。将热点数据，如热门帖子、常用游戏资讯等缓存到 Redis 中，减少对 MySQL 数据库的直接查询，降低数据库压力，提高数据读取速度，进而提升系统整体性能，确保玩家能够快速获取所需信息。

3 安全技术保障：在安全方面，除了配备硬件防火墙外，在软件层面采用数据加密技术，对用户密码、敏感信息等进行加密存储与传输，防止数据泄露；同时，利用身份验证与授权机制，严格控制不同用户角色的访问权限，确保只有合法用户在授权范围内操作系统资源，全方位保障游戏社区管理系统的信息安全。

## 系统功能需求

针对游戏论坛社区管理项目设计，我们需对各个板块进行全面复习，如登录注册，社区管理和用户相关权益(如注销)等等，分析如下：

**1登录注册**：

支持多种注册方式，如常规的用户名 + 密码注册，同时提供手机号验证码注册、邮箱验证注册，以及主流社交平台（微信、QQ 等）一键授权登录，方便用户快速接入社区。

注册过程中，实时验证用户名、密码格式合法性，确保信息符合安全规范；对手机号、邮箱进行唯一性校验，避免重复注册。

登录界面友好简洁，提供找回密码功能，通过邮箱或手机验证码重置密码，保障用户账户安全。

**2社区管理员**：

用户管理：查看、编辑所有用户信息，对违规用户标记处理记录，可批量操作封禁、解封账号，设置不同用户等级（普通、高级玩家、版主等）及对应权限。

内容审核：对新发布的帖子、评论进行预审核，支持图片、视频人工复查，根据社区规则对违规内容进行删除、屏蔽操作，并向发布者发送通知告知处理原因。

社区设置：灵活调整社区板块分类、名称、排序，创建特色主题板块；制定社区规则，发布公告通知，确保玩家及时知晓社区动态与规范要求。

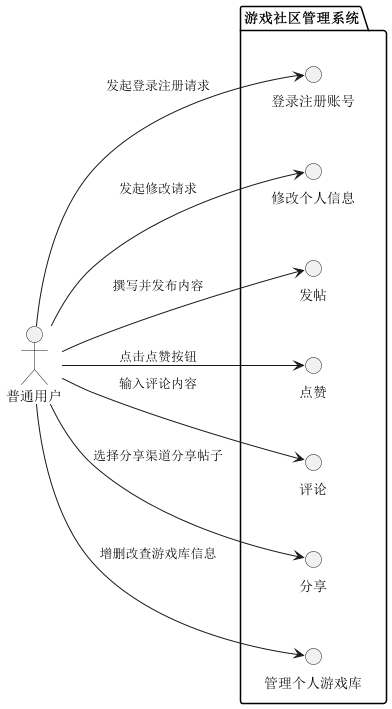
**3用户注销**：

用户自主申请注销账号时，系统需进行二次确认，提示注销后数据不可恢复，包括个人资料、发布内容、社交关系等将全部清除。

注销流程符合隐私法规要求，彻底删除用户相关数据，在数据库中抹去痕迹，防止信息泄露风险。

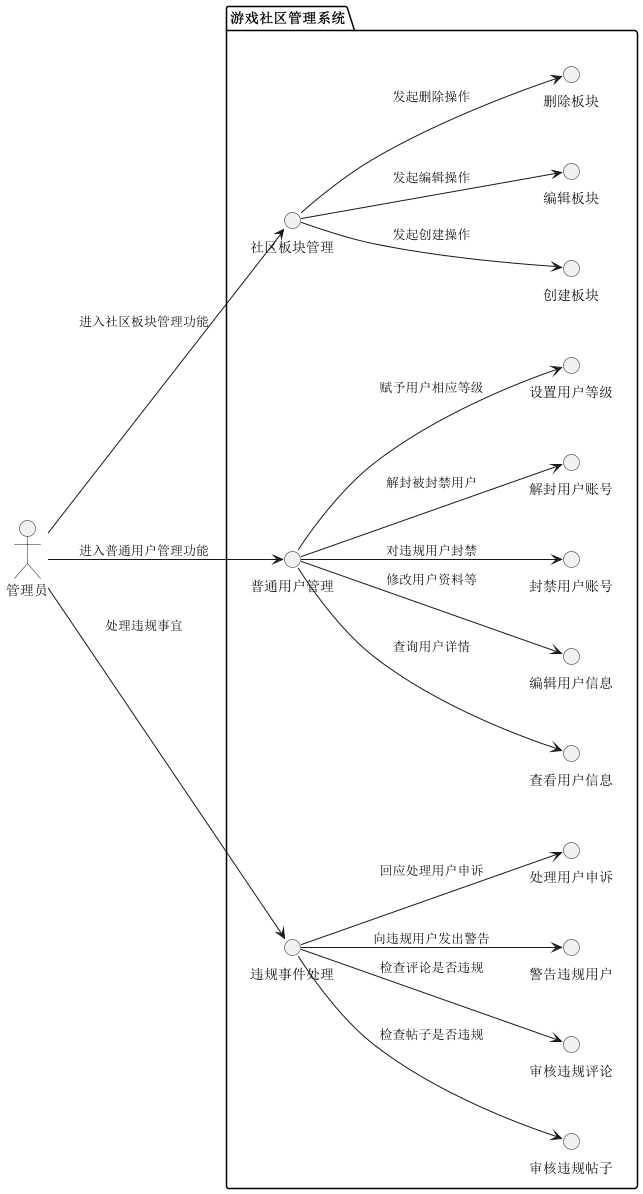
注销后，用户无法再使用原账号登录，同时在社区内相关展示（如评论、点赞记录）同步更新，消除已注销用户的残留信息影响。

1. **系统模型**
2. **普通用户用例图**

**普通用户是我社区的主流人群，主要以对社区的基础权限和个人信息的基础权限如图所示：**

1. **管理员用例图**

**社区的和谐离不开管理员，其拥有较高权限，包括但不限于修改删除屏蔽封禁等等，以下是用例图：**



**四.系统总体结构**



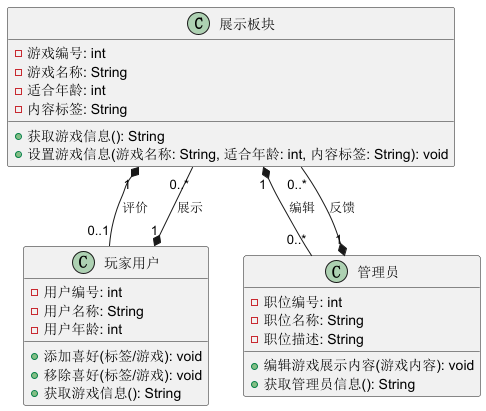
登录界面：



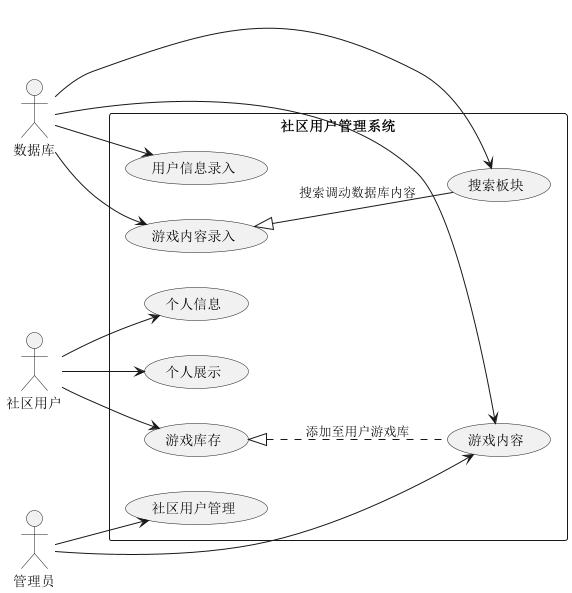
注册界面：



类图



用例图



数据库设计

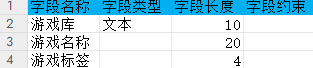
用户设计



社区设计



游戏库设计



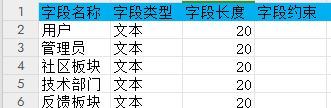
技术管理



管理员



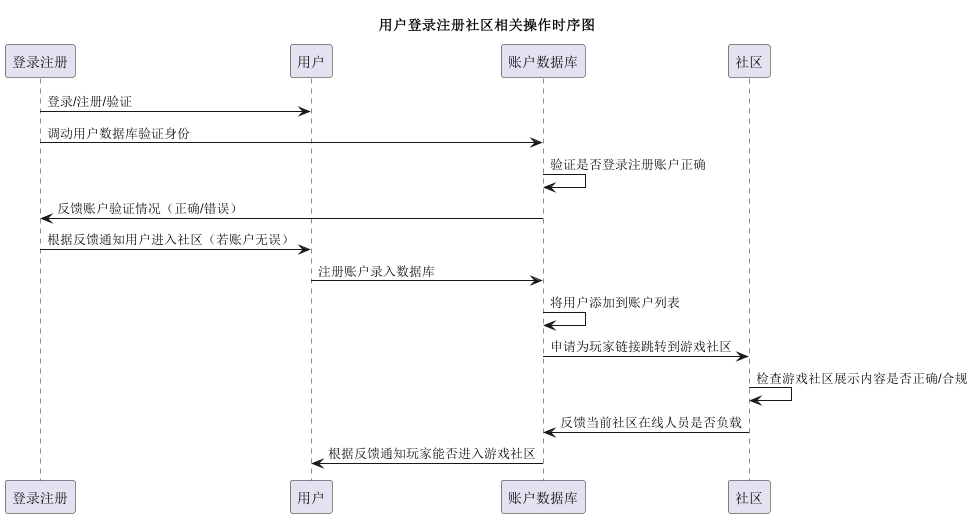
各板块表



1. 详细设计

登录注册流程及时序图

| **用例名称** | **用户登录注册** |
| --- | --- |
| 功能 | 登录/注册及进入社区 |
| 参与者 | 社区用户 |
| 条件 | 申请加入社区 |



用户页面设计

接口设计

**获取用户信息接口**：

**接口地址**：/api/user/info/{user\_id}

**请求方法**：GET

**请求参数**：user\_id（用户唯一标识，必填）

**响应数据**：

{

"user\_id": "001",

"user\_name": "高性能的萝卜子",

"gender": "男",

"age": 25,

"preference\_type": "剧情向",

"current\_role": "用户"

}

**功能描述**：根据传入的用户 ID，从数据库中查询并返回该用户的详细信息，供前端展示用户页面使用。

**更新用户信息接口**：

**接口地址**：/api/user/update

**请求方法**：PUT

**请求参数**：

{

"user\_id": "001",

"user\_name": "新昵称",

"gender": "男",

"age": 26,

"preference\_type": "剧情向",

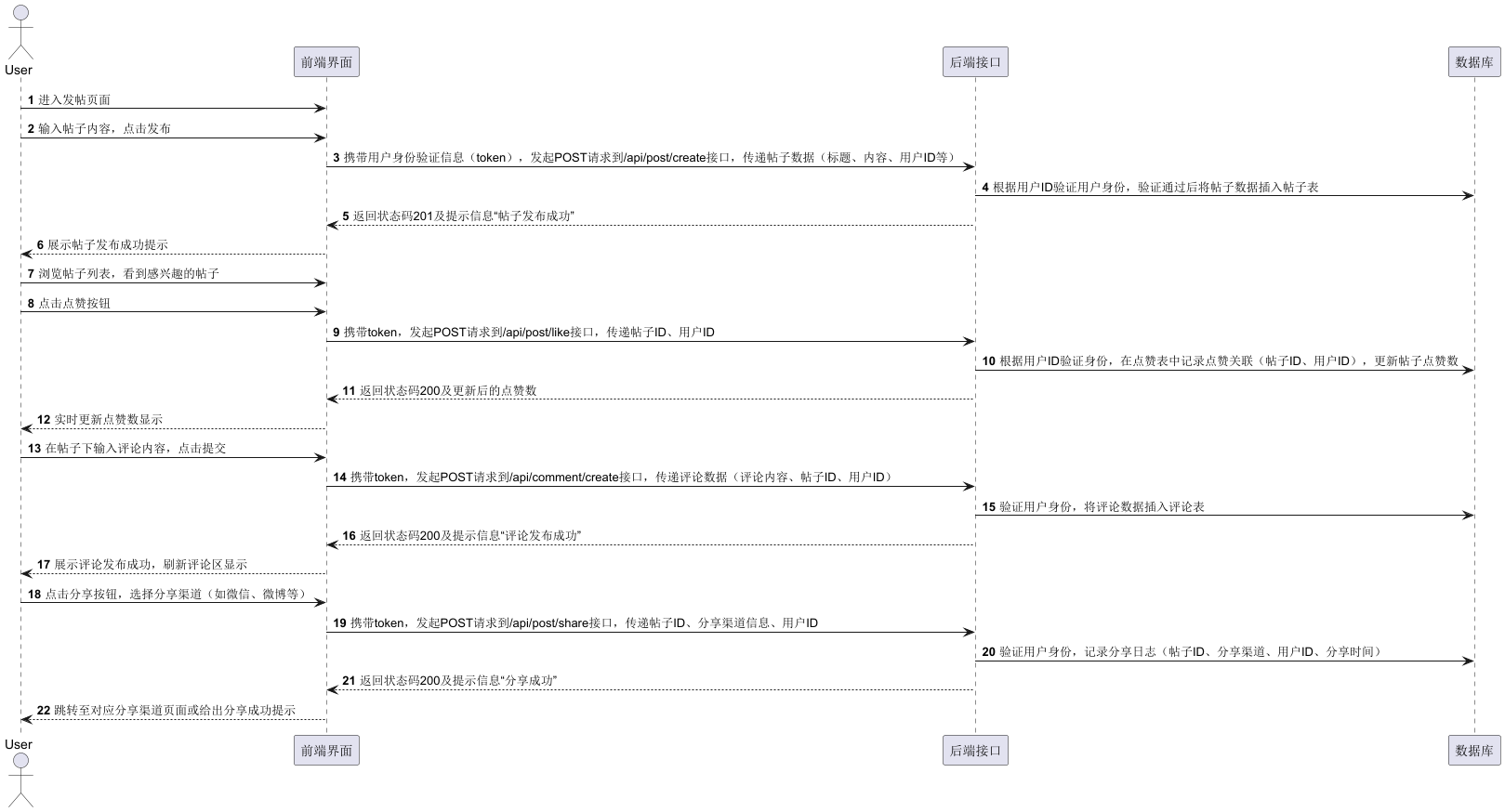
"current\_role": "用户"

}

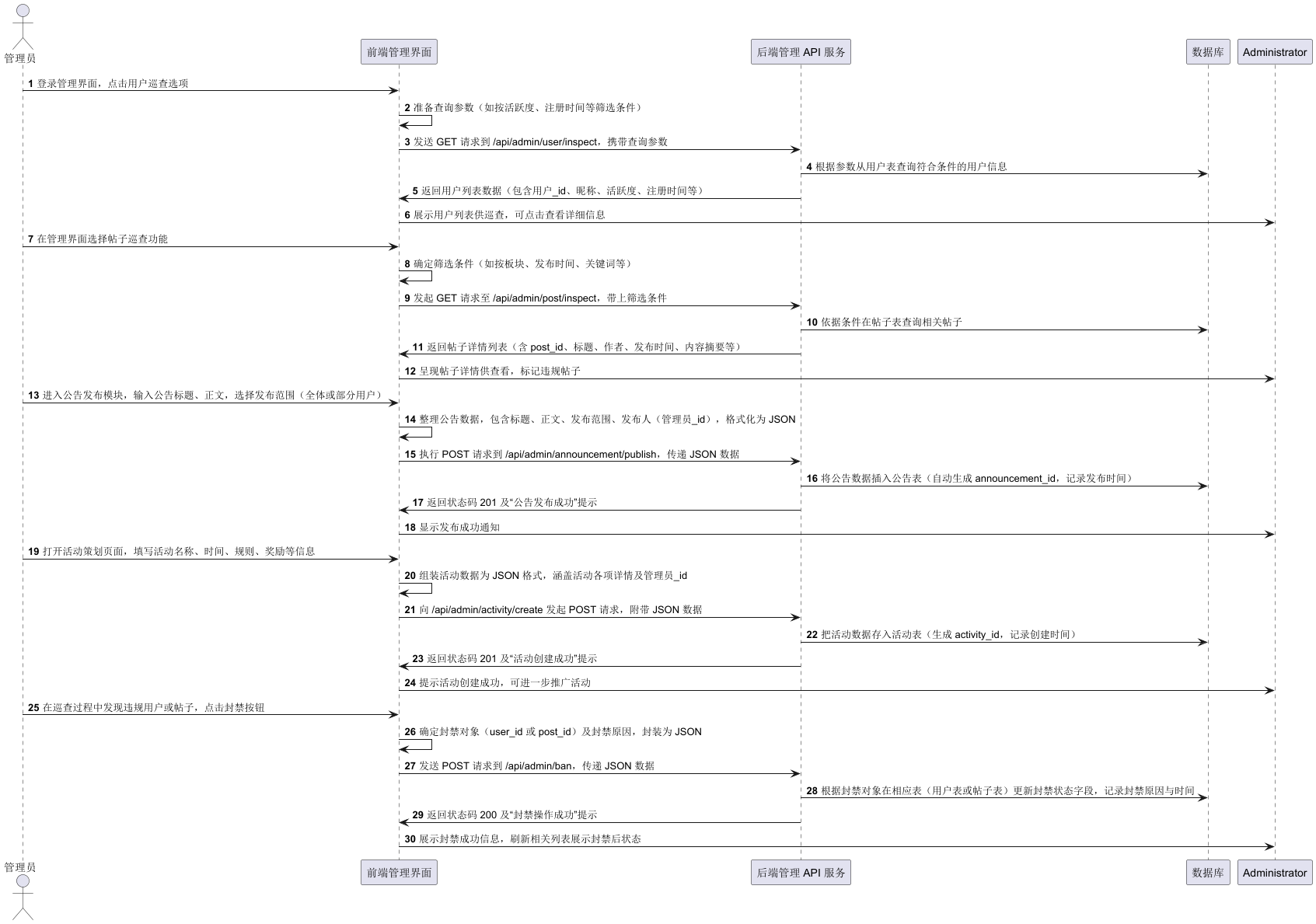
**响应数据**：成功更新返回状态码 200 及提示信息 “用户信息更新成功”，失败则返回相应错误码及错误描述。

**功能描述**：允许用户修改除用户 ID 以外的自身信息，后端接收前端传入的更新数据，对数据库中的用户记录进行更新操作。

用户发帖点赞评论时序图：



管理员管理时序图：



所有人对社区游戏数据库的时序图：



技术员对社区系统的维护：

