

**2024年春季学期**

**计算学部《软件工程》课程**

**实验报告**

**Lab4 OO分析与设计**

|  |  |  |
| --- | --- | --- |
| **姓名** | **学号** | **联系方式** |
| 程瑞琦 | 2021112268 | 2021112268@stu.hit.edu.cn |
| 秦浩伦 | 2021111747 | 2021111747@stu.hit.edu.cn |
| 唐思琪 | 2021113454 | 2021113454@stu.hit.edu.cn |

**目 录**

[1 本组项目概述 1](#_Toc16225)

[2 用例模型 2](#_Toc14406)

[2.1 角色清单 2](#_Toc2294)

[2.2 用例模型 2](#_Toc30457)

[一.管理用户账户用例模型 2](#_Toc5671)

[三.下载应用用例模型 3](#_Toc8367)

[参与者：普通用户（User） 3](#_Toc7637)

[简要描述： 普通用户浏览和下载平台上的应用。 3](#_Toc4439)

[2.3 用例1管理用户账户模型 4](#_Toc29284)

[2.4 用例2上传应用模型 4](#_Toc26421)

[2.5 用例3下载应用模型 5](#_Toc8396)

[3 类识别 5](#_Toc19479)

[3.1 边界类 5](#_Toc8562)

[3.2 控制类 5](#_Toc15708)

[3.3 实体类 6](#_Toc28830)

[4 领域模型 6](#_Toc28932)

[4.1 分析类图 6](#_Toc21445)

[4.2 领域类图 8](#_Toc10329)

[5 时序模型 8](#_Toc31235)

[5.1 用例1：用户上传模型 9](#_Toc1285)

[5.2 用例2：用户下载模型 10](#_Toc22898)

[5.3 管理账号时序模型 11](#_Toc23424)

[6 部署模型 11](#_Toc30795)

[7 小结 12](#_Toc5690)

[文档全部完成之后，请更新上述区域]

# 本组项目概述

**业务目标与业务价值**

**功能概述**

本组的业务目标为，实现一个APP的管理系统，提供一个平台让用户能够上传自己的APP，并将其展示给更广泛的用户群体。同时还可以确保上传的APP质量，通过管理员的审核流程过滤掉低质量或有害的应用。

通过评论和点赞功能增加用户的互动和参与度，通过有效的分类和搜索功能，使用户能够轻松地找到他们喜欢的应用，并进行下载和使用。

业务价值为：

创建一个用户活动平台，提高平台的粘性和吸引力。通过审核机制提高平台上应用的质量，增强用户对平台的信任。通过用户的评论和点赞数据，提供有价值的市场反馈，帮助开发者改进他们的产品，同时还可以提高平台的粘性和吸引力。在可能的条件下，通过广告、推广费用等方式，平台可以获得商业收益。

功能有：

1. 用户注册和登录

2.APP上传和管理：

用户可以上传自己的应用。

上传时提供应用的基本信息。

上传后应用进入待审核状态。

3.审核系统：

管理员可以查看待审核应用。管理员审核通过或拒绝应用。

审核通过的应用展示在平台上，用户可见。

4.应用展示，搜索和下载：

用户可以浏览、搜索、分类筛选应用。

展示应用信息、下载链接、用户评论等。

5.用户互动：

用户可以对应用进行评论和点赞应用。

6.后台管理系统：

管理员可以管理用户、审核应用。

7.安全和隐私保护：

确保用户数据和应用数据的安全。

# 用例模型

## 角色清单

**Actor：**

**普通用户：**

交互：注册、登录、浏览应用、下载应用、评论、点赞、上传应用、查看审核状态。

角色：最终用户，使用平台提供的服务和功能。

管理员：

交互：注册、登录，上传应用，管理用户账户、审核应用、管理应用和评论、角色：系统管理员，负责平台的内容和用户管理。

**外部软件系统**

支付网关：

交互：处理用户付款和交易。

角色：处理财务交易，确保支付安全和便捷。

**外部硬件系统**

服务器：

交互：主机应用存储库系统，处理用户请求和后台管理操作。

角色：核心硬件，运行系统的服务器。

用户设备：

交互：用户使用的设备，如PC、智能手机、平板等，用于访问系统。

角色：客户端设备，用户通过这些设备与系统进行交互。

**系统时钟：**

交互：提供准确的时间戳，用于记录操作日志、用户活动、审核时间等。

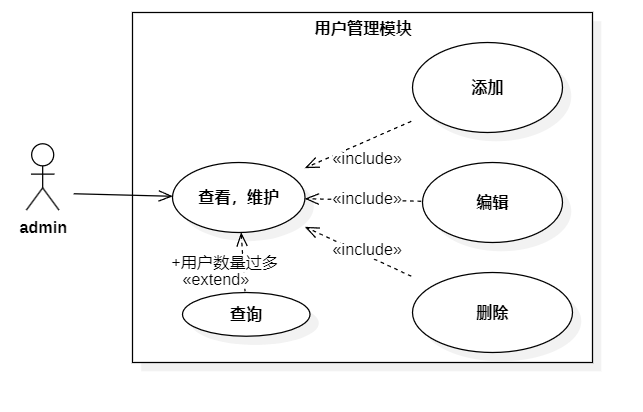
角色：时间管理，确保所有时间相关的操作准确记录。

## 用例模型

### 一.管理用户账户用例模型

参与者：管理员

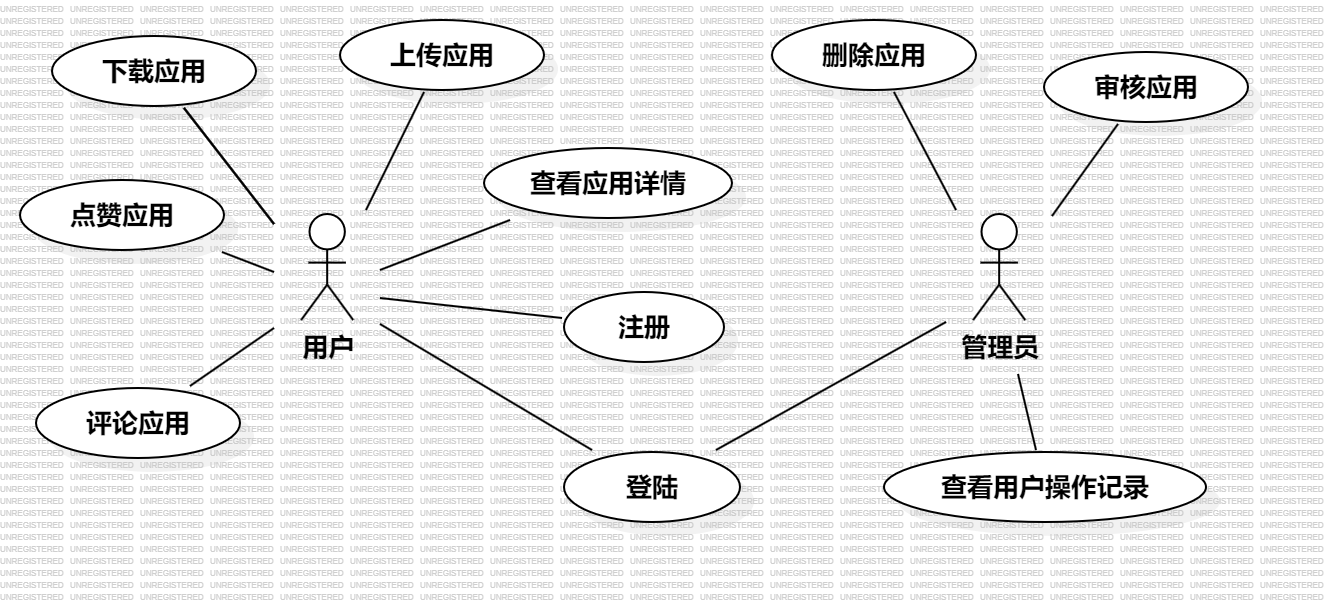
简要描述： 管理员管理平台上用户账户，进行检查和统计。



1. 上传应用模型

参与者：普通用户（User） 管理员（Admin）

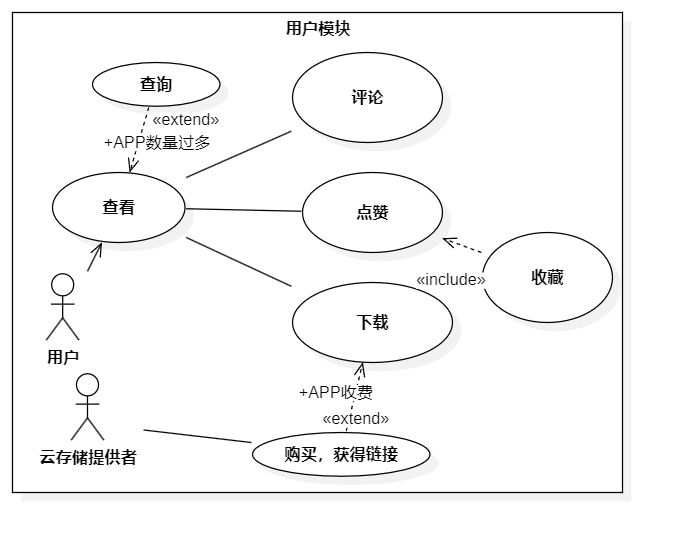
简要描述：用户上传应用，管理员进行审核



### 三.下载应用用例模型

### 参与者：普通用户（User）

### 简要描述： 普通用户浏览和下载平台上的应用。



**从所设计的用例中选取3个最核心、最主要的，详细描述其事件流，如下。**

## 用例1管理用户账户模型

本用例所涉及的actor(s)：admin

常规事件流：

1. 管理员登录系统..
2. 管理员浏览系统的用户管理页面.
3. 管理员查看用户数量，名称等信息

备选事件流：

1. 如果用户信息涉及违规内容或不符合平台规定，管理员可以选择删除或编辑相关用户信息。
2. 如果涉及非常规的用户注册或升级，管理员可以选择添加或编辑相关用户信息
3. 如果用户数量过多，可以使用查询功能来快速找到目标用户

## 用例2上传应用模型

本用例所涉及的actor(s)：用户

常规事件流：

1. 用户点击应用服务界面（APP上架）
2. 用户点击查询，搜索想要上传的应用名称，查看是否已在应用列表中
3. 用户点击上架，在弹出的界面里填写要上架的应用信息，包括名称、作者、大小和类型等
4. 用户点击应用列表旁的上传APP按钮，上传相应应用文件

备选事件流：在（2）中，若应用查询到已在列表中，则跳过（3）直接来到（4）

## 用例3下载应用模型

本用例所涉及的actor(s)：用户，存储提供方

常规事件流：

1. 用户浏览平台上的应用列表找到感兴趣的应用..
2. 用户查看应用信息、评论等.
3. 用户点击下载按钮
4. 用户从存储提供方得到下载链接

备选事件流：

1. 如果应用需要付费，系统进行收费。
2. 如果APP数量过多，可以使用查询功能来快速找到目标APP
3. 如果用户想要表达对APP的喜爱或先收藏APP，可以使用点赞功能

# 类识别

## 边界类

|  |  |  |
| --- | --- | --- |
| 类名（中文） | 类名（英文） | 类的作用概述 |
| 用户界面类 | UserUI | 提供用户与系统的交互界面，处理用户的登录、注册、查看应用详情、点赞、评论和上传应用等操作请求。 |
| 管理员界面类 | AdminUI | 提供管理员与系统的交互界面，处理管理员的登录、审核应用、删除应用和查看用户操作记录等操作请求。 |

## 控制类

|  |  |  |
| --- | --- | --- |
| 类名（中文） | 类名（英文） | 类的作用概述 |
| 用户控制类 | UserController | 处理用户界面类的请求，协调用户与系统之间的交互，执行用户的登录、注册、查看应用详情、点赞、评论和上传应用等操作。 |
| 管理员控制类 | AdminController | 处理管理员界面类的请求，协调管理员与系统之间的交互，执行管理员的登录、审核应用、删除应用和查看用户操作记录等操作。 |

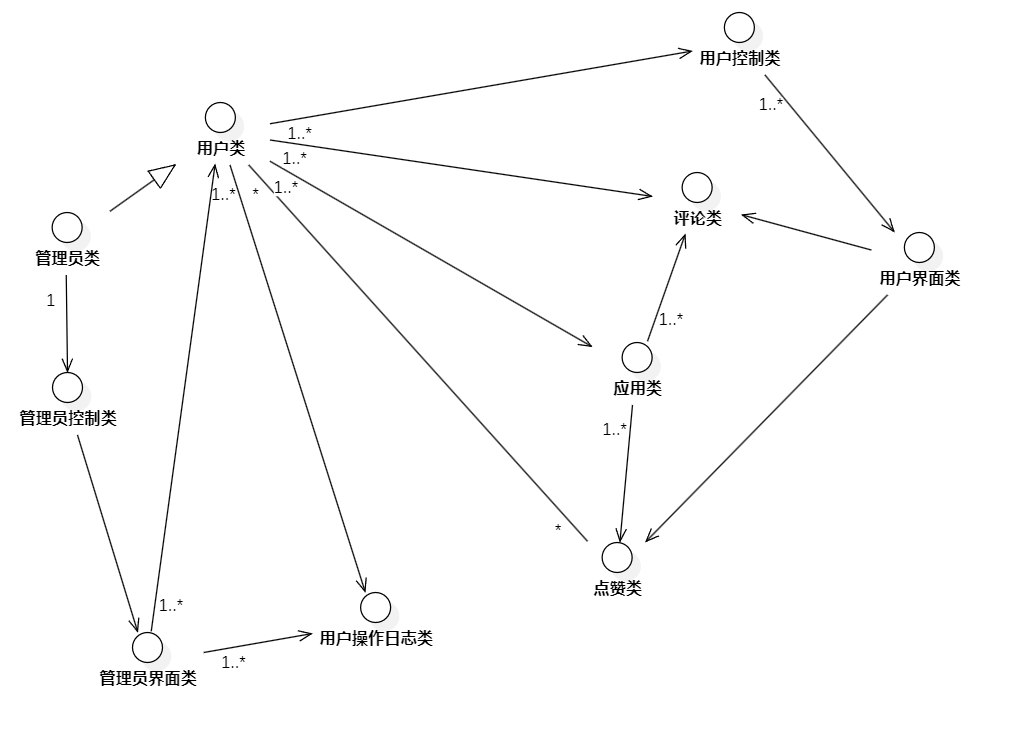
## 实体类

|  |  |  |
| --- | --- | --- |
| 类名（中文） | 类名（英文） | 类的作用概述 |
| 用户类 | User | 表示系统中的用户实体，包括用户的id、密码、权限、应用库以及电子邮件等。 |
| 应用类 | App | 表示系统中的应用，包含应用的名称、大小、类型、图标以及下载链接等。 |
| 评论类 | Comment | 表示对应用的评论。 |
| 点赞类 | Like | 代表用户对应用的点赞记录，包含点赞用户、应用及时间戳等信息。 |
| 用户操作日志类 | UserOpreationLog | 记录用户在系统中的操作，如评论、点赞、上传应用等，帮助管理员进行监控和审计。 |
| 管理员类 | Admin | 继承自用户类，代表系统管理员，包含额外的操作权限如审核和删除应用，以及查看用户操作记录等。 |

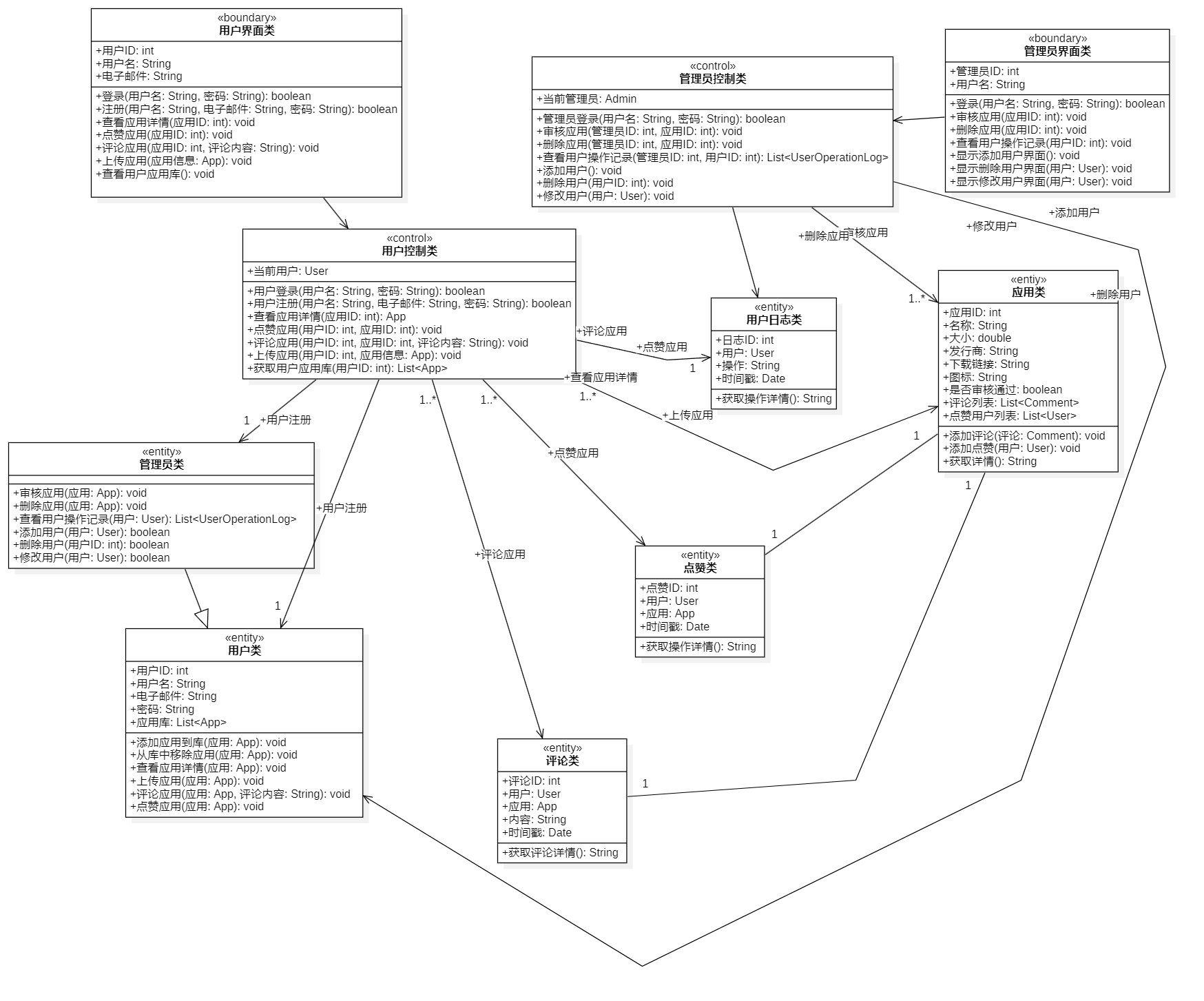
# 领域模型

## 分析类图

建立边界类、控制类、实体类之间的关联关系，使用UML类图形式描述。



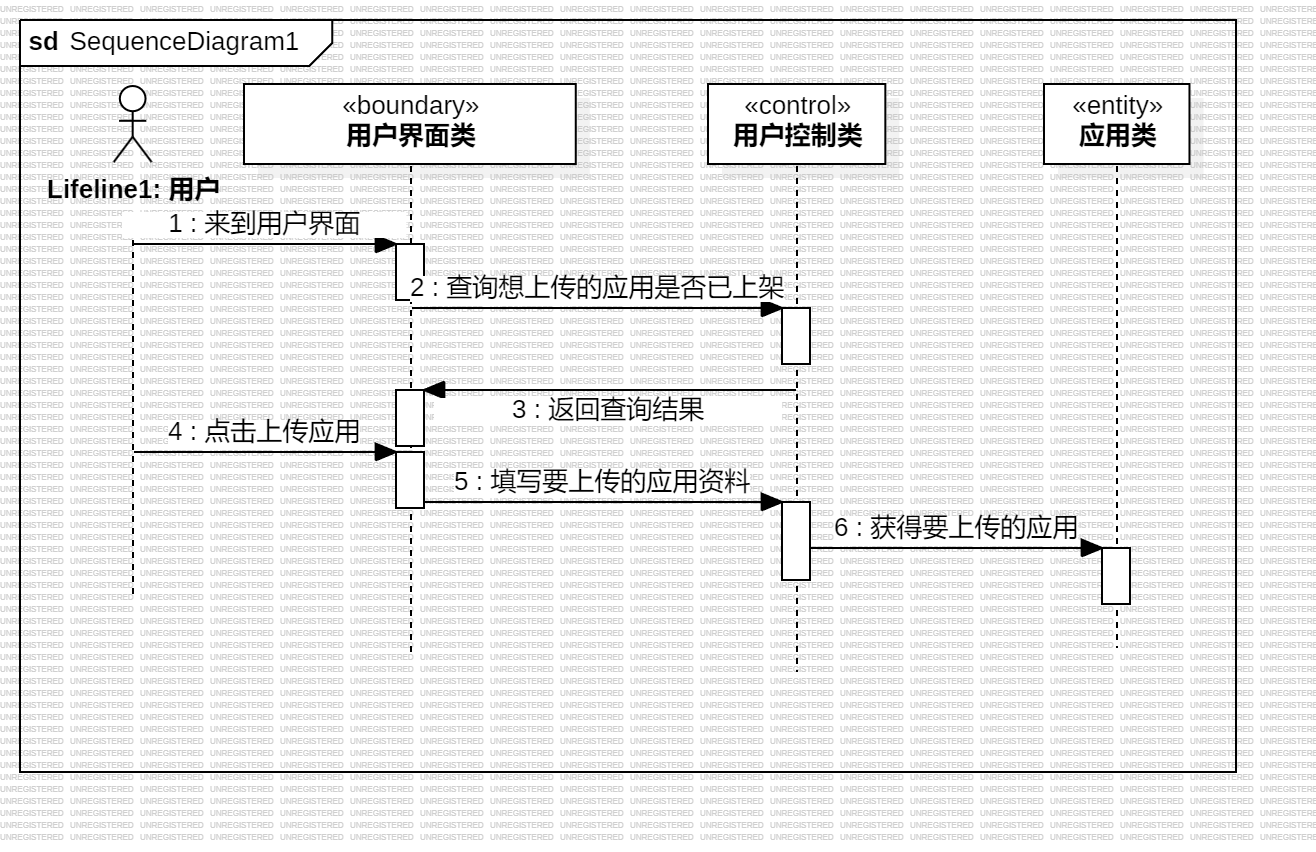
## 领域类图



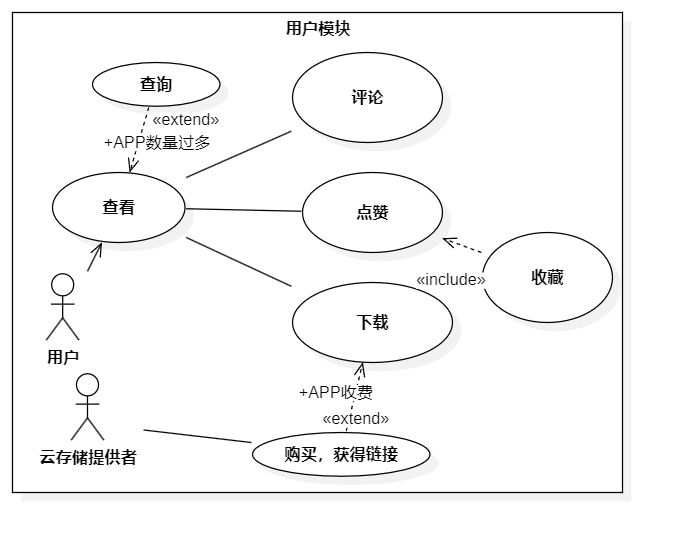
# 时序模型

针对第2部分里识别出的3个用例，分别建立其时序模型。该模型中包含的所有对象必须在第3部分中被识别出来，各类包含的操作需在4.2节的类图中体现。

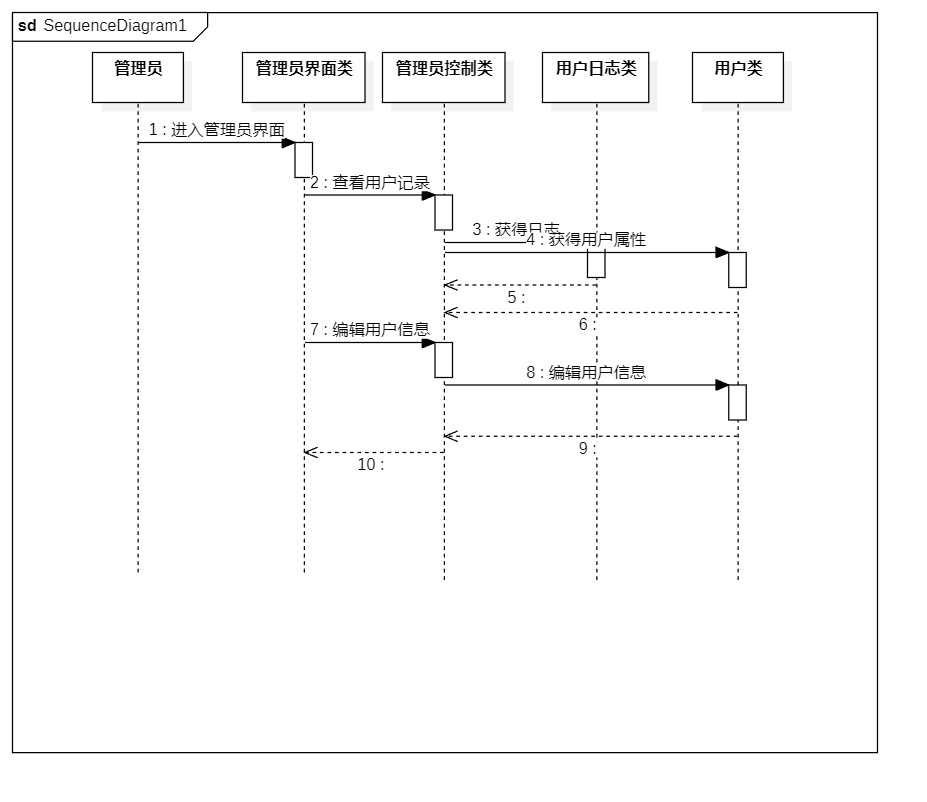
## 用例1：用户上传模型



## 用例2：用户下载模型



## 管理账号时序模型



# 部署模型

主要组件

客户端：

1.移动端（Mobile Client）：用户通过手机APP访问系统。

2.PC端（PC Client）：用户通过浏览器访问系统。

服务器端：

1.Web服务器（Web Server）：处理HTTP请求并提供静态内容和前端资源。

2.应用服务器（Application Server）：处理业务逻辑和后端API。

3.数据库服务器（Database Server）：存储用户数据、应用数据等。

4.文件存储服务器（File Storage Server）：存储应用的文件和图标等。

网络连接

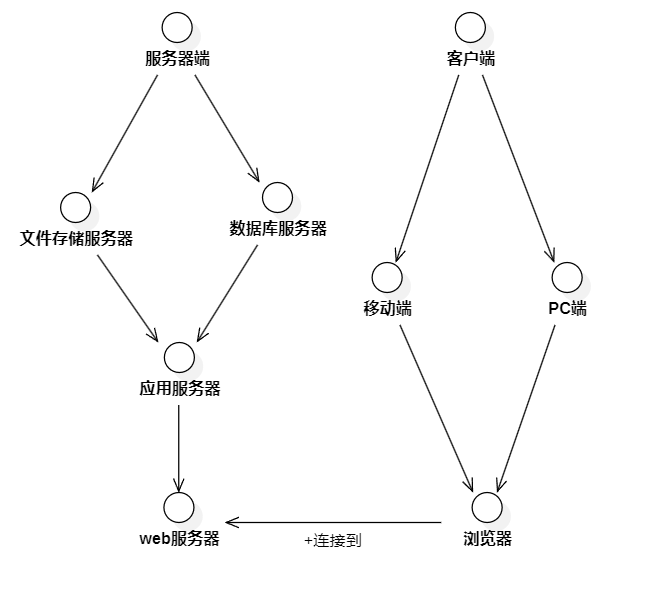
1.客户端与服务器端的连接：

移动端和PC端通过互联网与Web服务器和应用服务器进行通信。

2.服务器端内部连接：

Web服务器与应用服务器之间通过内网连接。

应用服务器与数据库服务器、文件存储服务器之间通过内网连接。



# 小结

1. 建模是否有必要？直接去写代码呗！边写代码边思考，岂不是省了建模的大量时间？

建模在软件开发中是非常重要且必要的。建模有助于开发人员理清需求和系统结构，从而减少开发过程中的错误和后期修改的成本。它能帮助团队在开发过程中保持一致性和效率，提高系统的可扩展性和维护性。虽然直接编码看似更快，但缺少建模可能会导致重复工作和逻辑混乱，最终需要重新编写或花费大量时间整理代码，增加时间和资源浪费，还可能大大提高出现错误的概率

。

1. 建模时如果能思考得很细节，其实就相当于写代码时的思考工作。如何能逼着自己在建模时想得更细节，你们是否总结出什么方法？

1. 分解问题：将复杂问题拆解成更小的部分，逐个进行建模和分析。

2. 深入需求分析：结合实际经验，进行资料查阅和用户调研，深入理解需求。

3. 考虑特殊情况：预测可能出现的异常情况，如输入错误和操作冲突，设计合适的异常处理策略以确保系统稳定性和可靠性。

4. 利用学习到的建模方法：认真设计用户故事等。

1. 你对“面向对象”这套软件分析与设计的方法，怎么看？它与传统的以“算法+数据结构”为单位的结构化分析方法相比有什么好的地方和不好的地方？

面向对象的方法将系统视为一组相互作用的对象，每个对象都有自身的状态和行为。与传统的“算法+数据结构”方法相比，面向对象的方法有以下优点：

1.抽象和模块化：通过对象的抽象能力，系统更易理解和管理，对象将数据和操作封装在一起，提供模块化设计方式，有助于降低复杂性和增加可重用性。

2.继承和多态：继承机制允许重用现有代码和行为，促进代码扩展和维护。多态性提高了代码灵活性和可扩展性，使系统更好适应变化和需求变更。

3.封装和信息隐藏：封装将对象的内部状态和行为隐藏，仅通过有限接口暴露必要功能，提高系统安全性和稳定性。

4.问题域直接映射：面向对象方法更自然地映射真实世界中的问题域，有助于分析和理解需求，减少沟通和理解障碍。

缺点包括：

1.设计复杂性：大型系统中，面向对象方法可能涉及大量对象和类，需要良好的设计和结构，否则可能导致复杂性增加和设计混乱。

2.过度设计风险：可能导致过度使用继承、复杂关系和抽象层次，从而增加系统复杂性和维护成本。

1. 你对UML这种建模语言怎么看？它给出了这么多的diagram，各diagram是否的确起到了帮助开发者的作用？你们对各个diagram的作用怎么看？

UML 作为标准化的建模语言，对于软件开发团队非常重要。它能够帮助开发者在不同阶段和层次上进行系统抽象和设计，提高沟通效率、降低开发成本，并提升软件质量和可维护性。各个图表的作用和评价：

1. **用例图（Use Case Diagram）**：描述系统功能和角色间的交互，对需求分析和系统功能理解非常重要，帮助团队捕捉和定义系统功能需求。
2. **类图（Class Diagram）**：展示系统中的类、属性和方法之间的静态结构关系，是设计和实现过程中的重要参考，明确对象间关系和行为。
3. **时序图（Sequence Diagram）**：描述对象间的动态交互，特别适合展示消息传递的时间顺序和对象间的协作。
4. **状态图（State Diagram）**：描述对象的状态变化及其触发条件，对于建模复杂状态机或流程控制非常有帮助。
5. **活动图（Activity Diagram）**：描述系统中的活动流程和控制流，展示系统操作流程和业务逻辑非常实用。
6. **组件图（Component Diagram）**：展示系统各组件及其依赖关系，对系统物理架构设计有帮助。
7. **部署图（Deployment Diagram）**：描述系统如何部署在硬件节点上，包括节点、连接和通信路径等，对系统部署和扩展规划至关重要。
8. 在以后的开发中，你们是否愿意先建模、开发团队再针对模型进行讨论修改、然后再进入到开发环节？

我们团队愿意在开发前先进行建模，讨论和修改模型后再进入开发阶段。这样做的好处显而易见。首先，通过建模和讨论，我们可以在编写代码前理清系统的整体设计和架构，避免后期的重构和设计错误。其次，团队成员可以在模型指导下更有效地分工合作，提高工作效率和协作效果。最后，讨论和修改模型阶段是团队沟通和协商的重要时机，有助于发现和解决技术挑战，确保项目按时交付并满足用户需求。因此，我们团队愿意采用这种先建模后开发的方法，以确保项目顺利进行和成功交付。