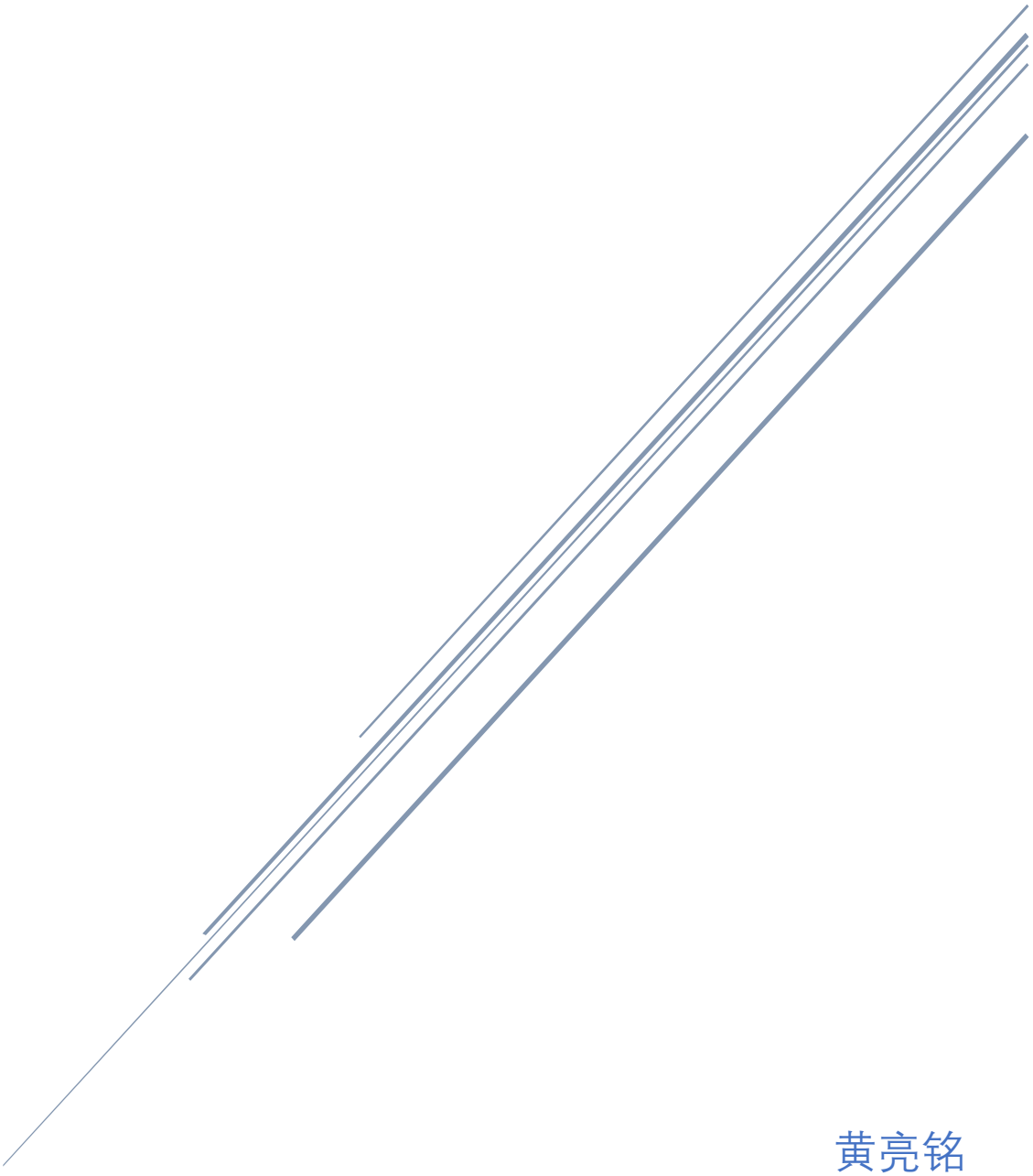
账易通

项目设计报告



黄亮铭

2022155028

目录

1 软件设计

本项目将采用面向对象设计的方法进行软件设计，严格遵循系统设计的核心原则。在设计过程中，我们将依据先前编制的可行性报告、需求分析报告以及系统原型设计报告，从中提炼出系统的核心功能模块。在此基础上，我们将精心设计每个核心功能模块的详细内容，并确保整个设计过程维持清晰的 B/S 结构和 MVC 架构，以实现系统的高效性和可维护性。

1.1 面向对象设计

面向对象设计是将先前分析阶段完成的分析模型转换为软件架构构建的过程。在该阶段，软件设计人员应当在分析模型的基础上补充与软件实现相关的内容。

为了准确表达面向对象设计阶段的系统，软件设计人员可以使用 UML 中的：

1） **类图**建模软件体系架构。

2） **顺序图**建模软件体系架构中对象的顺序交互过程。

在面向对象设计阶段，软件设计人员在完成软件架构设计之外，还应该从软件架构设计质量和软件架构设计效率等方面对软件体系架构进行优化。

1.2 设计原则

在软件设计的过程中，软件设计人员遵循面向对象设计的原则，不仅能够提高系统的稳定性和可扩展性，还能确保开发过程的高效性和设计的可维护性。

在软件设计时，软件设计人员应当遵循以下原则：

- **单一职责原则**：每个模块仅负责完成一个特定的功能，从而降低模块之间的耦合度，提升代码的可读性和维护性。在软件设计过程中，本项目中的各个核心功能模块将按照职责划分，确保每个模块聚焦于单一任务。
- **开闭原则**：系统应对扩展开放、对修改关闭。在本项目的软件设计的过程中将尽可能避免对现有代码的直接修改，而是通过新增模块或方法来实现功能扩展。
- **依赖倒置原则**：本项目的软件设计将广泛使用接口或抽象类来定义模块间的交互规则，确保模块间的解耦，提高系统的灵活性。
- **里氏替换原则**：本项目的软件设计将尽量保证子类能够完整实现父类的功能和行为约定，从而确保系统的健壮性和一致性。
- **接口隔离原则**：接口的设计应尽量精简，避免臃肿。每个接口只需要包含必须的功能，避免不必要的依赖。

通过遵循上述原则，软件在设计阶段为后期的功能维护和未来的功能拓展做好了准备，提升了软件的可维护性和可拓展性。此外，遵循上述原则可以降低软件长期开发成本，提高软件的质量和用户满意度。

1.3 体系架构设计

软件架构设计是对整体的规划与设计，包括系统的层次结构、模块划分和组件关系等方面。

本项目的软件架构采用 MVC 架构，将软件划分为视图层、控制器层和模型层三部分，达到简化设计、增强可维护性和提高扩展性的目的。

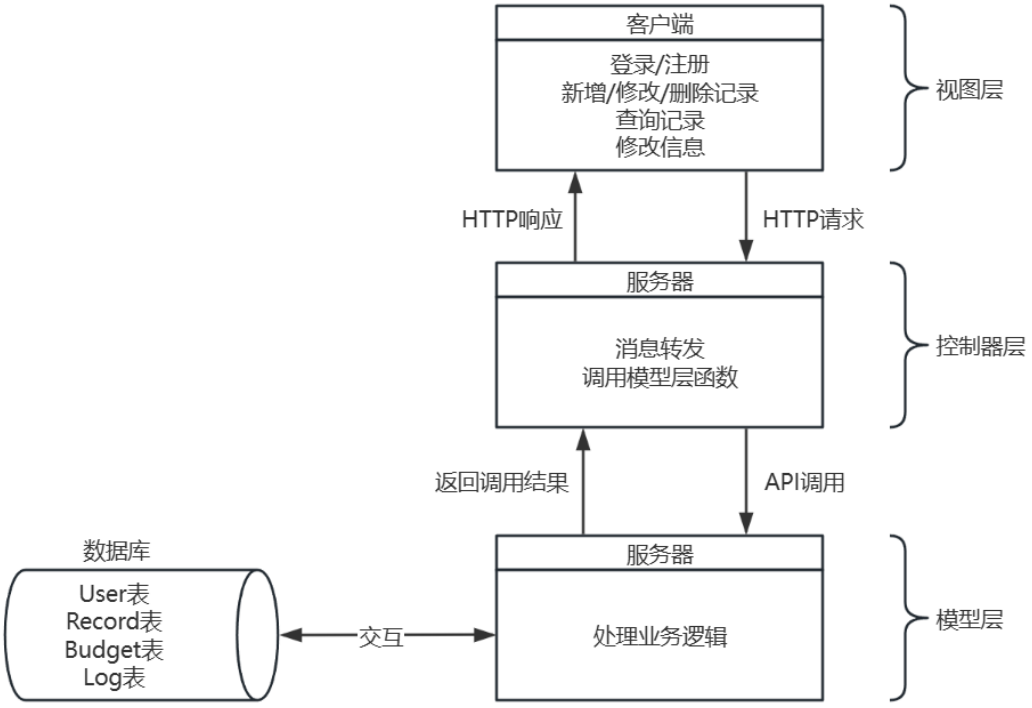


图 1 软件 MVC 架构示意图

- **视图层**：负责软件的用户界面显示，基于 Uniapp 开发，提供用户友好的交互体验。
- **控制器层**：作为视图层和模型层之间的桥梁，负责接收用户输入，处理用户交互，调用模型层进行具体的业务逻辑处理，然后更新视图层的显示内容。
- **模型层**：负责具体的业务逻辑的实现，包括但不限于数据的读取、数据的存储以及用户的请求。

MVC 架构的优点：

1) **模块化清晰**：模型、视图和控制器三部分分工明确，便于独立开发和调试。
2) **高可维护性**：模块之间的低耦合设计使得每一层的修改不会直接影响其他层。
3) **易扩展性**：新增功能时，只需更新相关的模块，无需对整个系统进行大规模改动。

此外，本项目遵循 B/S 结构（一种分布式系统架构，强调系统的部署方式），将功能分布在客户端和服务端。这种客户端请求服务、服务端响应服务的处理方式是一种新型的计算机应用模式。

- **客户端**：完成数据表示、数据处理以及各种接口功能。
- **服务端**：完成业务逻辑处理和 DBMS 的核心功能，以及为客户端提供接口。

B/S 结构的优点：

1) **安全性高**：用户权限和访问控制由服务器统一管理，更容易实现安全策略。同时借助 HTTPS、数据加密等技术，提升数据传输和存储的安全性。
2) **拓展性强**：服务器可以根据需求动态扩展，通过负载均衡支持更多用户访问，无需对客户端进行调整。
3) **维护简单**：所有功能和逻辑均运行在服务器端，更新或修复时只需对服务器进行操作，客户端无需更新，维护效率更高。
4) **用户体验好**：B/S 架构能够快速引入新特性，提升用户体验。

1.4 功能架构设计

功能架构设计旨在明确系统的各项功能，并将其划分为不同的模块，以便更好地实现系统的功能特性。本项目将软件功能划分为如下模块：

- **用户管理模块**：该模块主要负责处理与用户信息相关的业务，包括用户的登录/注册、个人信息编辑以及密码修改等功能，确保用户身份的管理与安全性。
- **记账管理模块**：负责处理用户的记账数据，包括新增、编辑、删除记账记录等操作，帮助用户高效管理日常收支。同时依据用户需求，提供符合要求的记账数据统计与分析服务，并以直观的形式呈现收支结构和消费趋势。
- **预算提醒模块**：该模块负责帮助用户设定月度或年度支出预算，并通过实时监控用户支出进度，提供直观的预算状态展示和超支提醒。当用户的支出接近或超过预算时，系统会自动发送通知，确保预算规划的有效执行。此外，该模块还支持历史预算记录查询，便于用户回顾和分析预算执行情况。

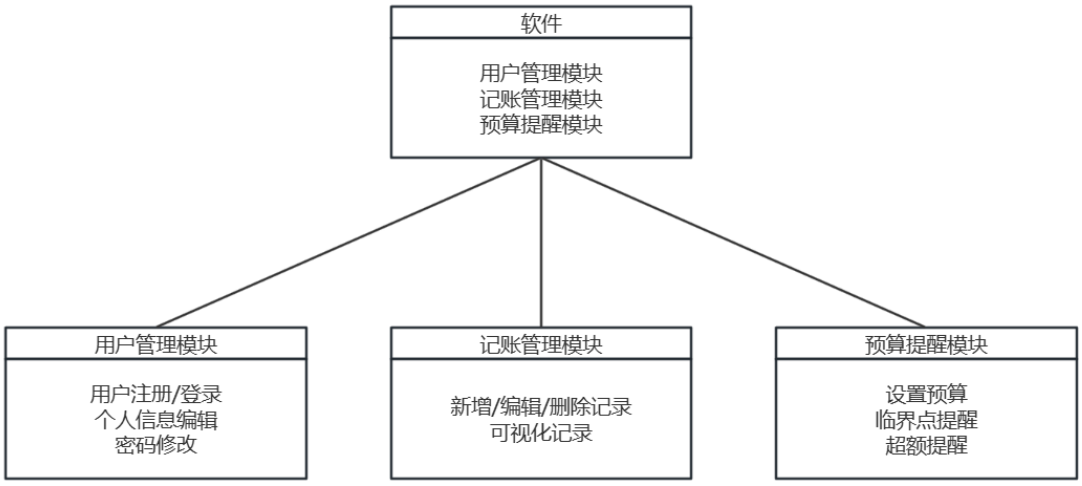


图 2 软件功能架构示意图

2 功能模块设计

2.1 用户管理模块

2.1.1 功能描述

用户管理模块是软件的核心模块之一，旨在为用户提供独立且安全的账户，提供良好的用户体验。

以下是用户管理模块的功能描述：

- **用户注册**：实现用户的注册操作，用户输入账号和密码进行注册。
- **用户登录**：实现用户的登录操作，用户输入账号和密码进行登录。
- **修改密码**：实现用户的修改密码操作，用户在登录之后可以通过输入新密码和确认新密码来修改密码。
- **信息编辑**：实现用户对个人信息的编辑操作，用户在登录之后可以在个人主页上修改自己的个人信息。

2.1.2 类图设计

用户管理模块的类图包括以下类：

- **User**：包含用户的基本信息，如用户名、密码等。
- **UserInfo**：存储用户的其他个人信息。
- **UserMgr**：与用户相关的消息的发送者和接收者，与用户关联。
- **Notification**：存储系统向用户发送的消息。
- **NotificationMgr**：用于存储系统向用户发送消息。
- **Log**：记录日志，如用户操作。
- **LogMgr**：日志发送者，与日志关联。
- **SystemMgrModule**：用于前端和数据库之间的消息转发和简单逻辑处理的类，位于控制器层。

为了体现类的属性和方法，简化后的类图如下所示：

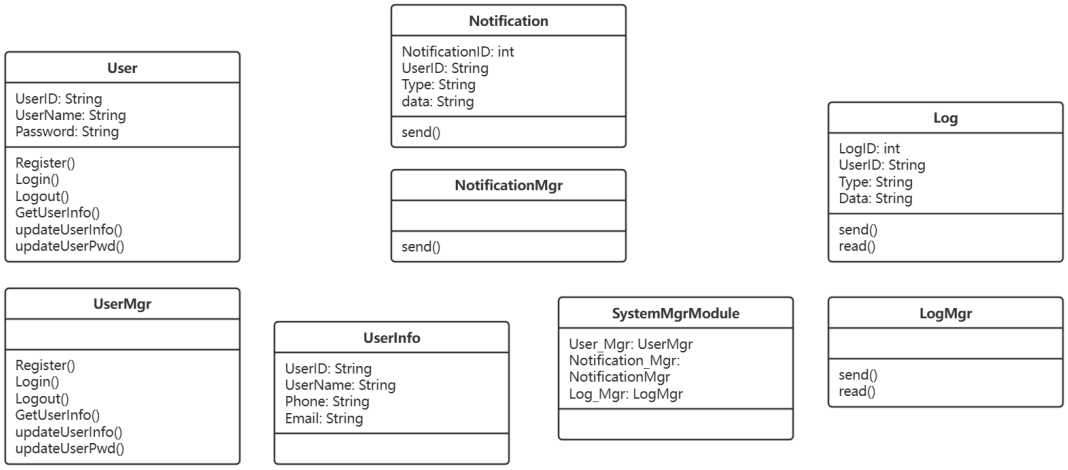


图 3 用户管理模块类图

2.1.3 顺序图设计

用户管理模块的顺序图描述了新用户从注册阶段到登录阶段，不同功能之间的交互过程。以下是用户管理模块的简化顺序图：

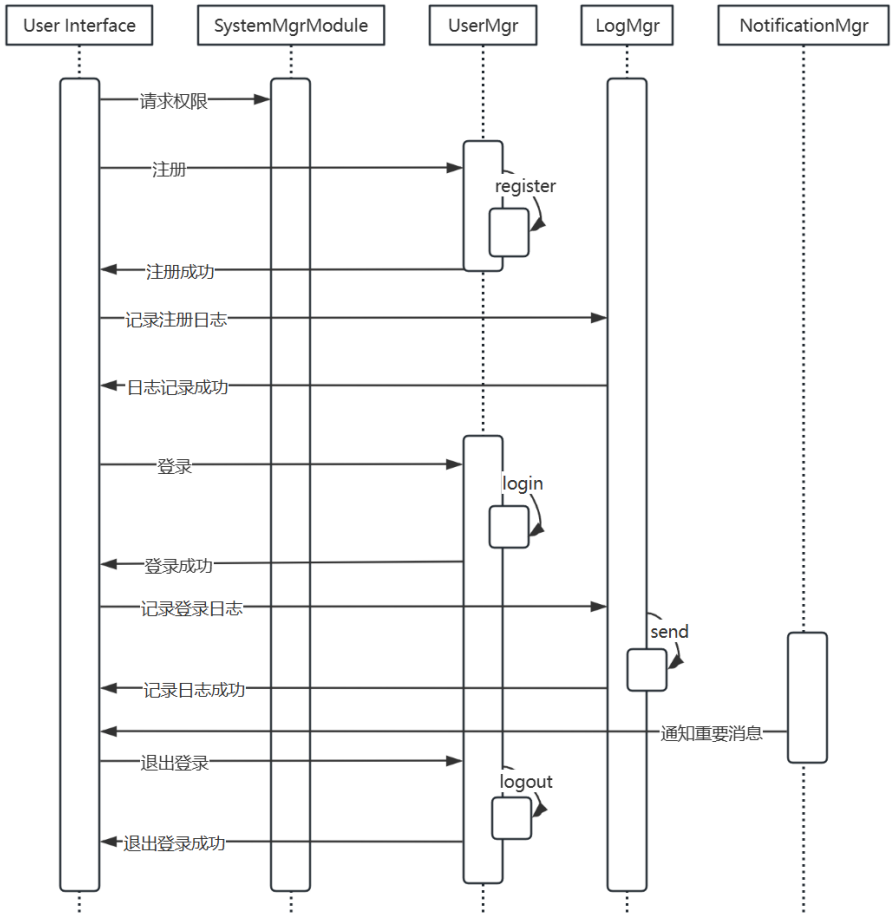


图 4 用户管理模块顺序图

2.2 记账管理模块

2.2.1 功能描述

记账管理模块是软件最为关键的模块，旨在以直观的形式向客户呈现收支结构和消费趋势，同时处理用户的记账数据，包括新增、编辑、删除记账记录等操作，帮助用户高效管理日常收支。

以下是记账管理模块的功能描述（前置条件：用户已经登录）：

- **记录管理**：提供新增、编辑、删除记账记录等操作。
- **可视化显示**：提供列表、饼状图等多种可视化收支记录的方式供用户选择。

2.2.2 类图设计

记账管理模块的类图主要包括以下核心类（为简化设计，体现核心功能，在 2.1 中提到的 **Notification、Log** 等类在此处仍有作用，但不再重复提及）：

- **Record**：包含记账记录的基本信息，如金额、类别和时间。
- **RecordMgr**：记账记录的数据发送者，与记录关联。
- **Graph**：包含图表的信息，如类别、比例和总金额等。
- **GraphMgr**：图标信息的接收者，与图表关联。

为了体现类的属性和方法，简化后的类图如下所示：

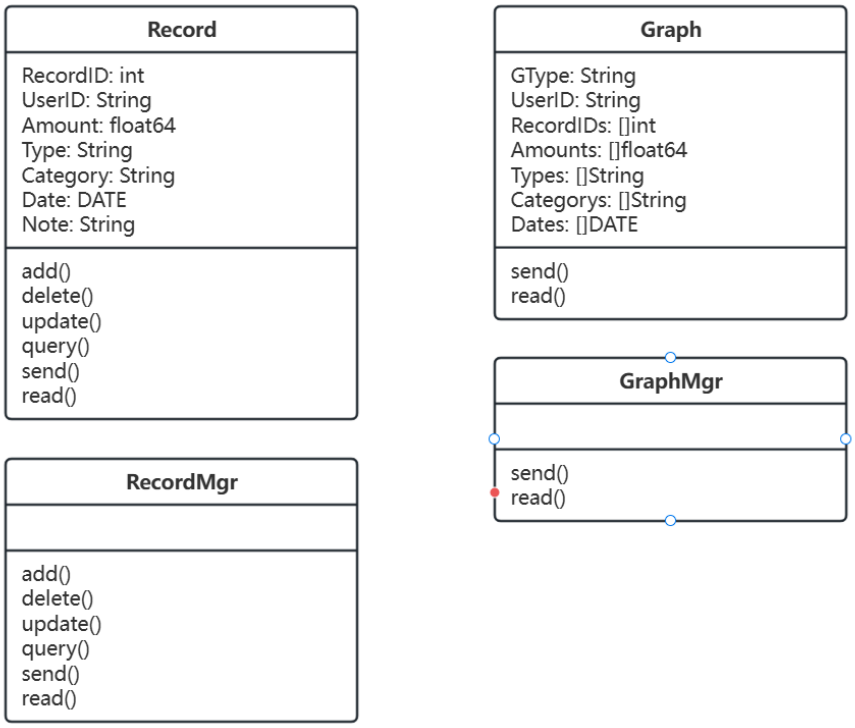


图 5 记账管理模块类图

2.2.3 顺序图设计

记账管理模块的顺序图描述了用户在登录之后进行新增、编辑、删除记账记录和查阅可视化图表等操作时，不同功能的交互的过程。

以下是记账管理模块的简化顺序图：

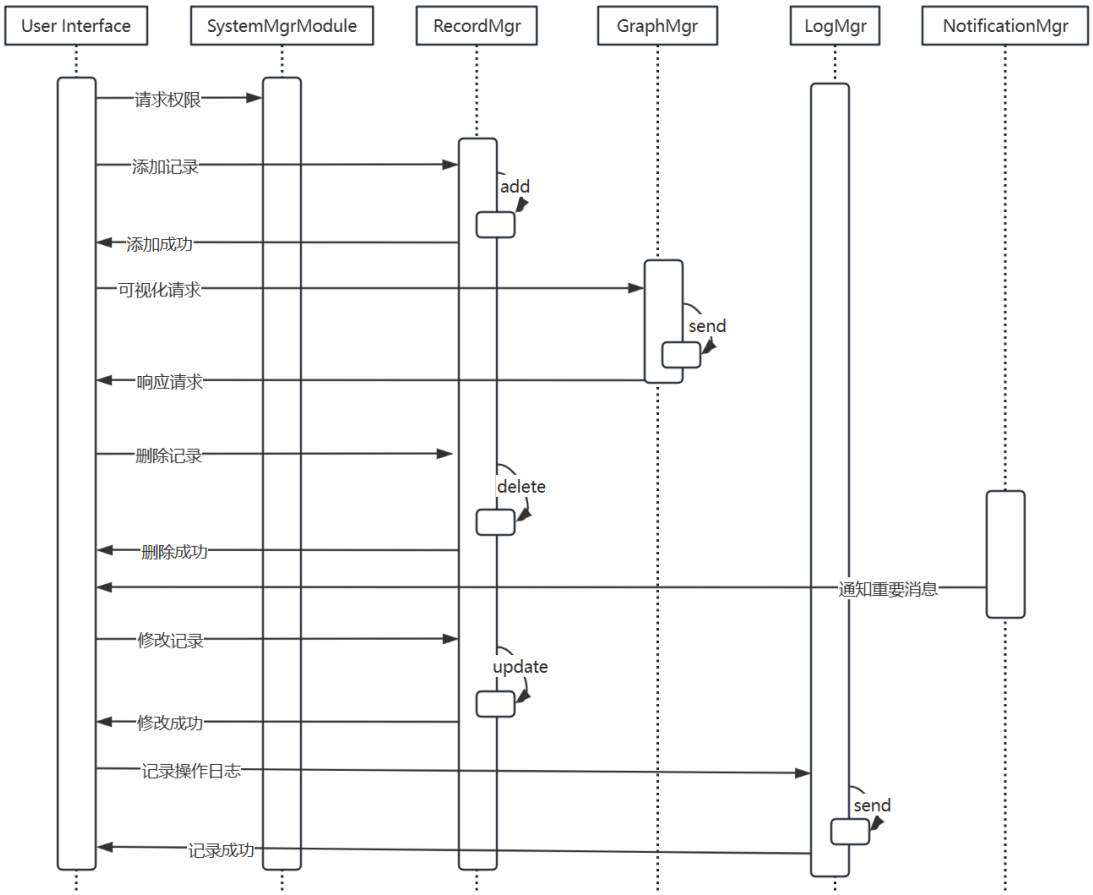


图 6 记账管理模块顺序图

2.3 预算提醒模块

2.3.1 功能描述

预算提醒模块是软件的核心模块，旨在确保用户预算规划的有效执行。

以下是预算提醒模块的功能描述（前置条件：用户已经登录）：

- **预算管理**：提供设置、修改预算操作。
- **消息管理**：当用户支出接近预算时系统自动发送消息提醒用户。

2.3.2 类图设计

预算提醒模块的类图主要包括以下核心类（为简化设计，体现核心功能，在 2.1 中提到的 **Notification**、**Log** 等类在此处仍有作用，但不再重复提及）：

- **Budget**：包含预算的基本信息，如金额。
- **BudgetMgr**：与预算相关的消息的发送者和接收者。

为了体现类的属性和方法，简化后的类图如下所示：

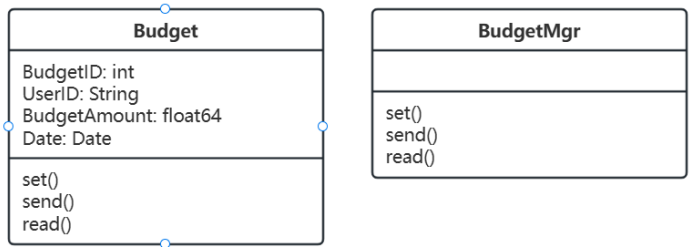


图 7 预算提醒模块类图

2.3.3 顺序图设计

预算提醒模块的顺序图描述了整个预算提醒模块流程中不同功能之间的交互。

以下是预算提醒模块的简化时序图：

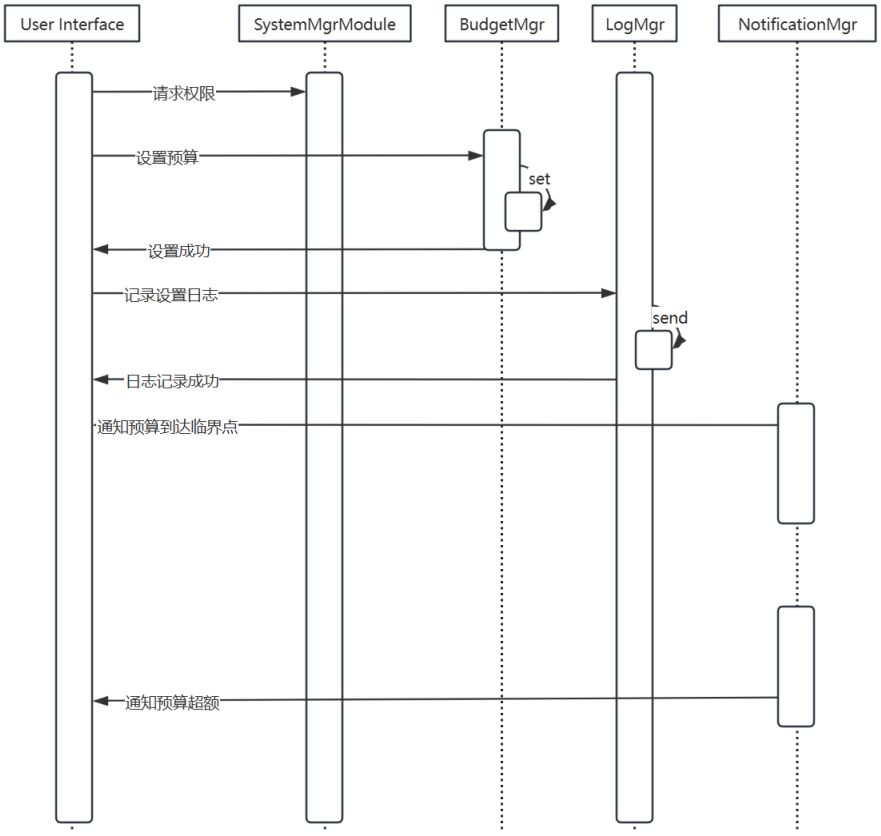


图 8 预算提醒模块顺序图

3 数据库设计

包含所有实体的 E-R 图如下所示：

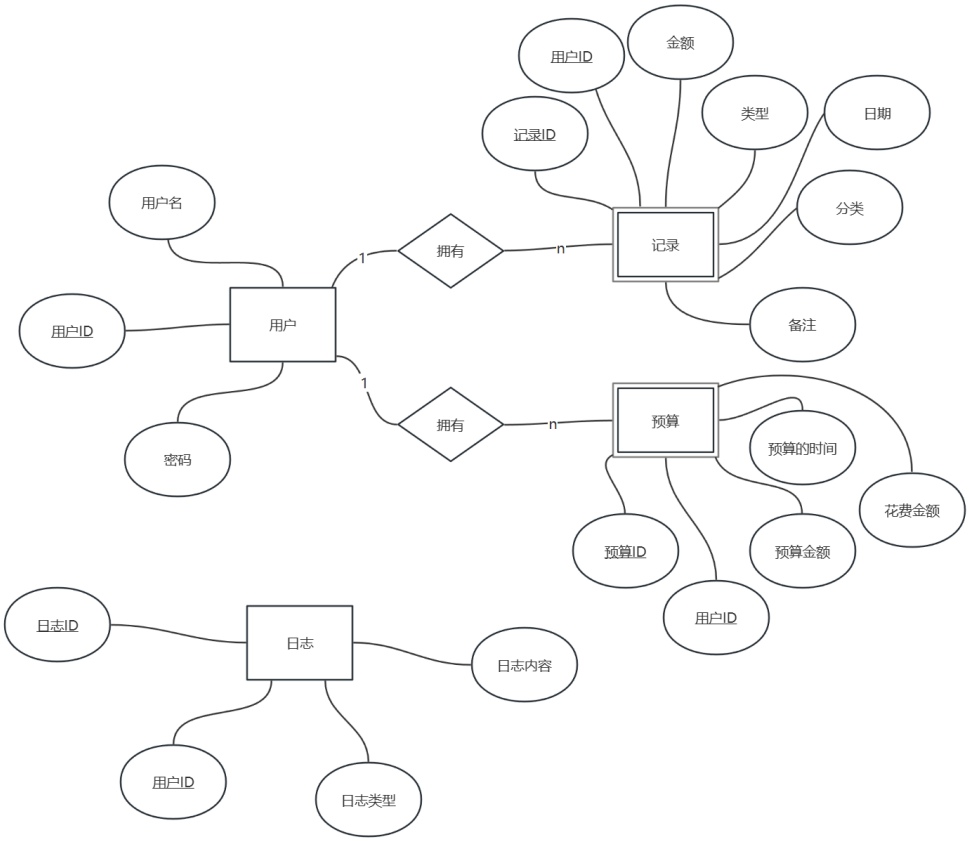


图 9 数据库 E-R 图

通过上述 E-R 图，我们可以设计出数据的逻辑结构，包括实体规范化和关系模型创建。

3.1 实体规范化

规范化是数据库设计中的一个重要步骤，旨在组织数据库中的数据以减少冗余、消除不一致性并提高维护效率。通过关系规范化，数据被划分为多个表，遵循一定的范式（如下），确保数据结构清晰合理。

1) **1NF**：每列必须是原子值，不能包含集合或重复的字段。
2) **2NF**：满足 1NF 的基础上，非主键字段必须完全依赖于整个主键，而不是依赖主键的一部分。
3) **3NF**：满足 2NF 的基础上，非主键字段不能依赖于其他非主键字段。
4) **BCNF**：在满足 3NF 的基础上，每个决定因素都必须是候选码。

在实体规范化中，本项目优先满足高级级别的范式要求，除非满足条件极其苛刻。

3.2 关系模型创建

以下是对数据库每个表的简要描述。

字段名	数据类型	是否为主键	说明
UserID	VARCHAR(11)	Yes	用户的唯一标识
UserName	VARCHAR(20)	No	用户名
Password	VARCHAR(20)	No	密码

表 1User 表

字段名	数据类型	是否为主键	说明
RecordID	INT	Yes	记录的唯一标识
UserID	VARCHAR(11)	No	关联用户表的 UserID
Amount	DECIMAL(10,2)	No	金额
Type	ENUM('收入','支出')	No	记录类型
Category	VARCHAR(20)	No	分类
Date	DATE	No	记录日期
Note	TEXT	No	备注

表 2Record 表

字段名	数据类型	是否为主键	说明
BudgetID	INT	Yes	预算的唯一标识
UserID	VARCHAR(11)	No	关联用户表的 UserID
Budget	DECIMAL(10,2)	No	预算金额
Amount	DECIMAL(10,2)	No	总花费金额
Date	DATE	No	预算的年月

表 3Budget 表

字段名	数据类型	是否为主键	说明
LogID	INT	Yes	日志的唯一标识
UserID	VARCHAR(11)	No	关联用户表的 UserID
Type	VARCHAR(20)	No	日志类型
Data	TEXT	No	日志具体内容

表 4 日志表