rBook概要设计规约

修订历史：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编写日期 | 版本 | 说明 | SPEG | 修订人 |
| 2018.12.8 | 1 | 初始编写 | 软件工程课程杜庆峰老师 | 俞杨 |
| 2018.12.12 | 2 | 数据库结构改动 | 同上 | 俞杨 |

## 引言

* 1. **概要设计依据**

[rBook需求分析规约](rBook需求分析规约.docx)

* 1. **参考资料**

[Spring Boot Reference Document](https://docs.spring.io/spring-boot/docs/2.1.1.RELEASE/reference/htmlsingle/)

[Spring Data Neo4j Reference Document](https://docs.spring.io/spring-data/neo4j/docs/5.1.3.RELEASE/reference/html/)

* 1. **假定和约束**

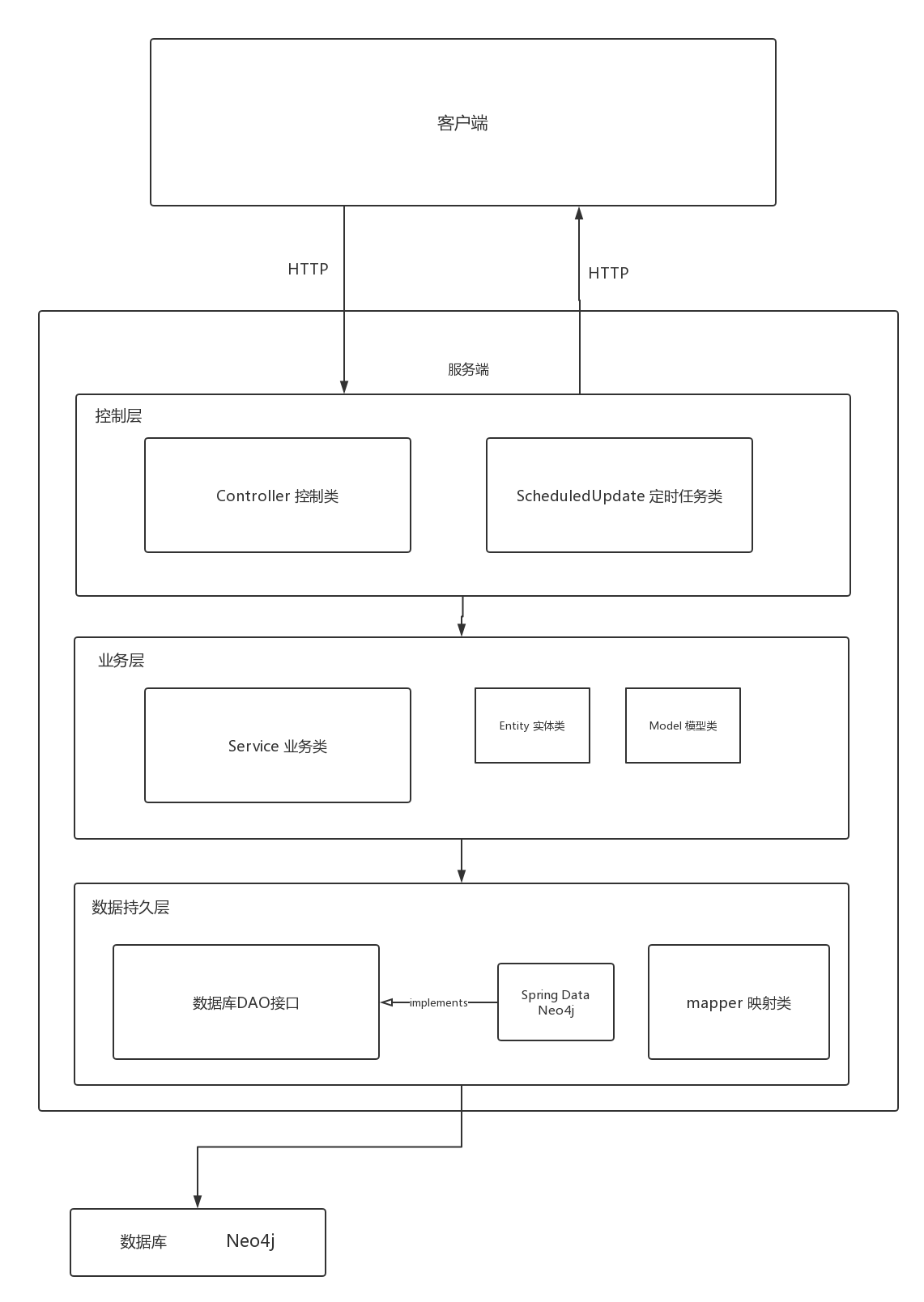
项目需在2019.1.11前交付至少第一个增量

## 概要设计

* 1. **系统总体架构设计**

本系统服务端使用Spring Boot作为框架，数据持久化中使用Neo4j图形数据库作为及解决方案。服务端和用户端之间通过HTTP的POST请求进行交互。设计阶段考虑以上特性，设计系统架构如下：

**1）系统层次架构如下：**

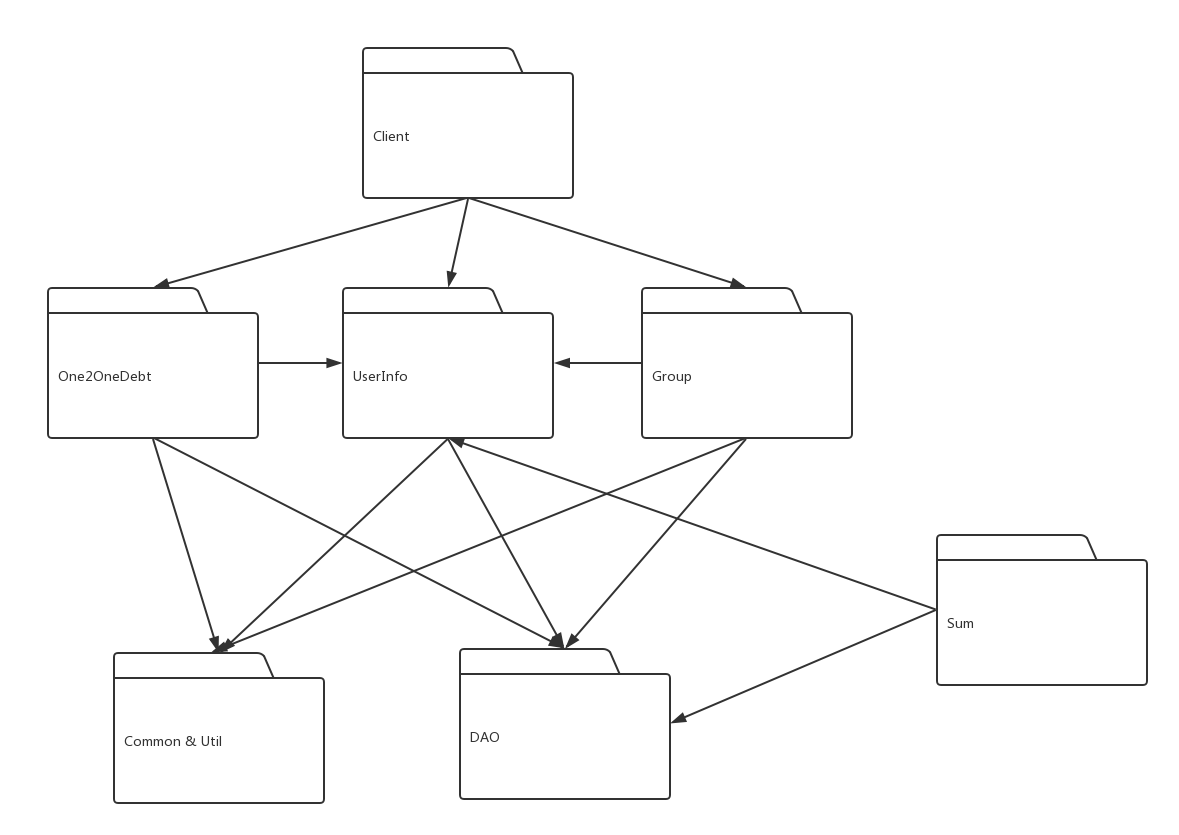


控制层负责控制系统的行为和功能，其中Controller类负责接收来自用户端的HTTP请求，并进行一些初步的请求合法性检查，同时在系统功能执行完毕后将结果返回给用户端。定时任务类负责在设定好的时间开始指定任务。

业务层负责系统业务功能的实现，其中Service类为功能逻辑实现，Entity类为业务实体，Model类数据来源为Entity类，是为满足某些特定功能而整合成的临时数据模型。

数据持久层负责与数据库的交互。在本系统中定义数据库交互DAO接口，并指定Query，由Spring Data Neo4j 来的对其进行实现。Mapper映射类对应Entity实体类，为数据库数据通过OGM直接映射成的数据库中间结果。

**2）系统子系统架构如下：**



说明：

1. Client子系统

负责客户端用户接口，用户交互的实现，向服务端的POST HTTP请求的发出和接受以及显示结果

1. One2OneDebt子系统

负责一对一记账相关功能包括创建结对关系，更新账目，确认账目，查看结对列表，查看详细账目等的控制和业务逻辑实现，包含对应的Controller, Service, Entity

1. UserInfo子系统

负责用户信息相关功能包括注册账号，登陆账号，查看用户统计信息等的控制和业务逻辑实现，包含对应的Controller, Service, Entity

1. Group子系统

负责群组记账相关功能包括创建创建群组，加入群组，查看群组列表，查看群组内情况，更新群组账目，确认群组，查看结清账目结果，确认结清账目等的控制和业务逻辑实现，包含对应的Controller, Service, Entity

1. Common & Util子系统

包含一些通用的工具和类包括ID生成，身份信息格式验证，加密方法，通用的请求格式等

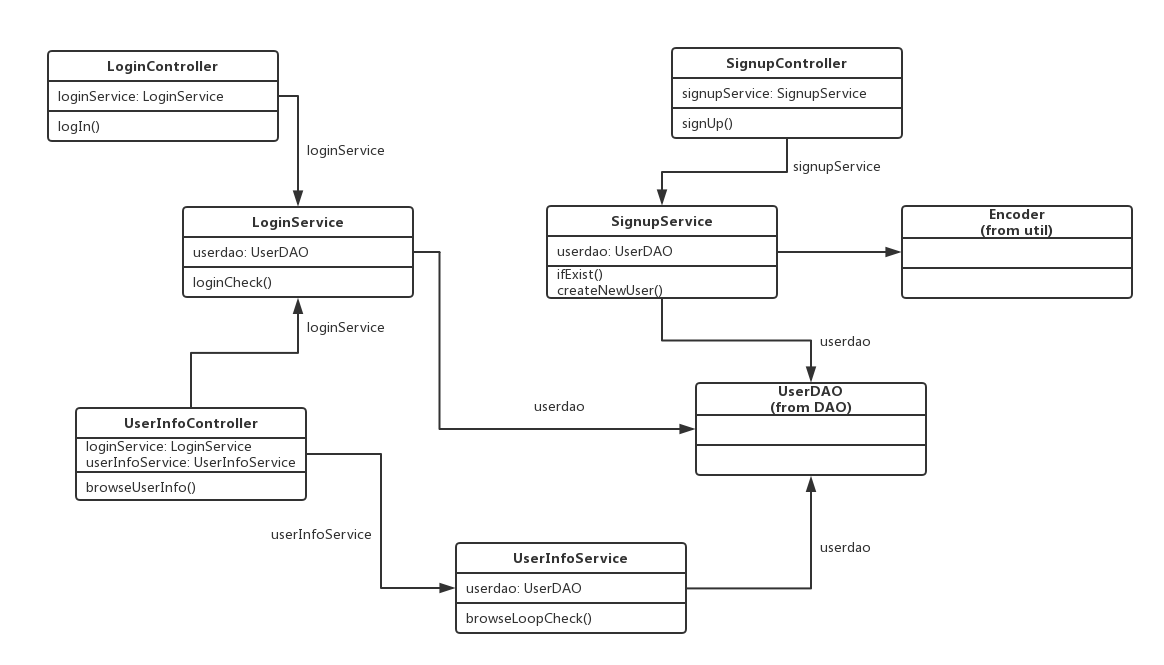
1. DAO子系统

负责与数据库进行交互的子系统，包括各个业务中涉及的数据库读写操作接口，包括数据库Query后OGM得到的映射结果类

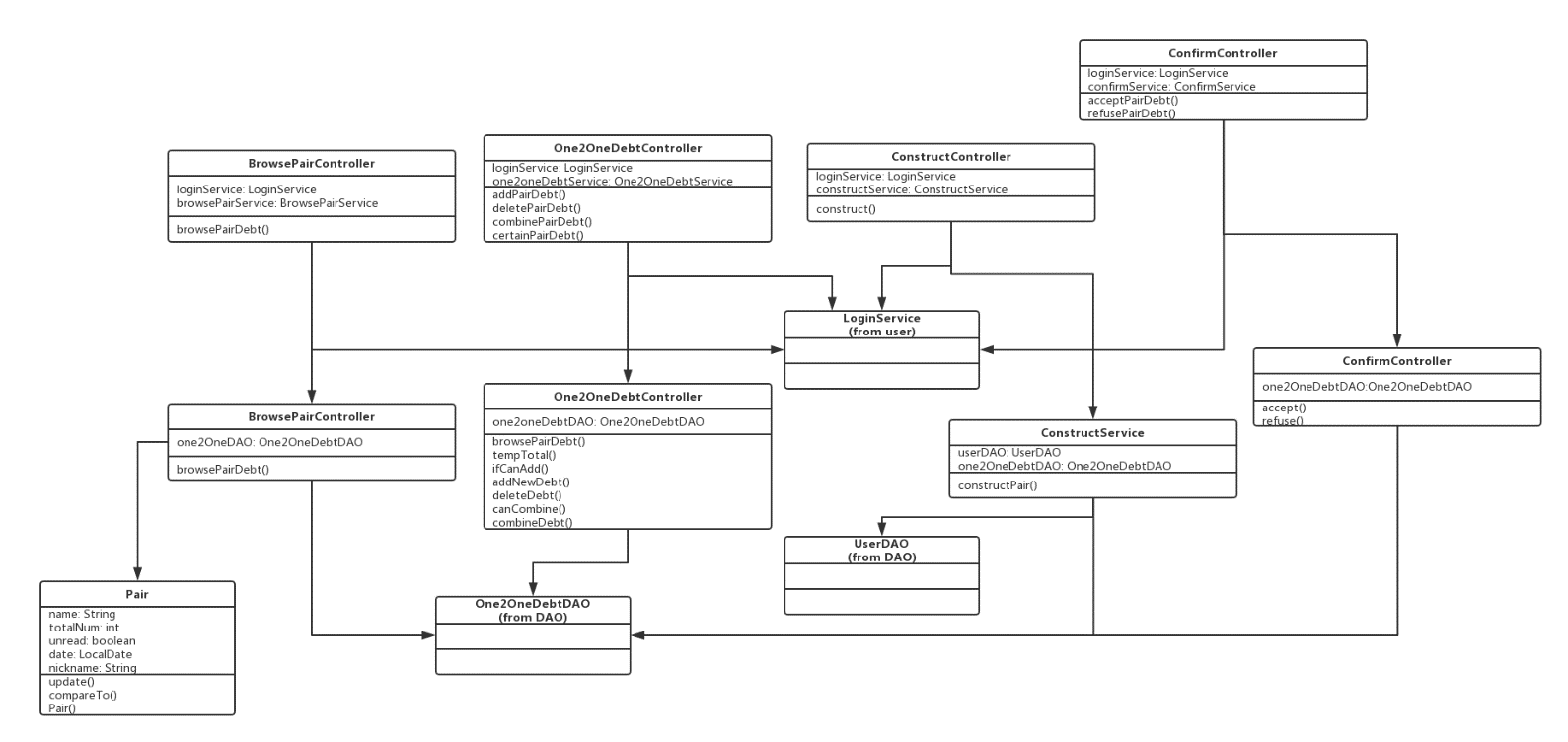
1. Sum子系统

负责定时更新统计信息相关功能包括更新用户总账目信息，更新用户账目倾向，环路检测等的定时任务和业务逻辑实现，包括对应的定时任务类以及Service类

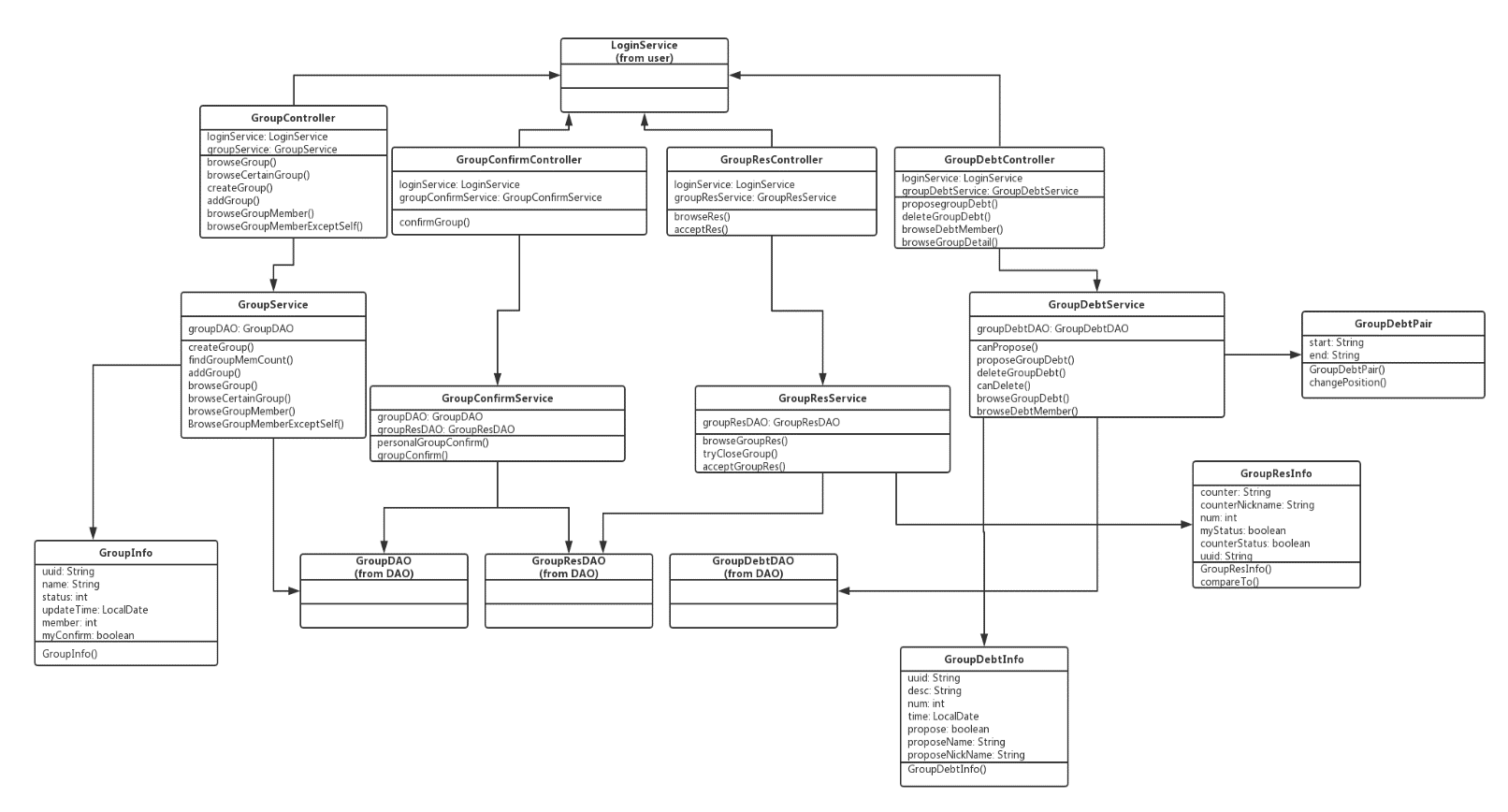
* 1. **系统软件结构设计**
     1. User类图



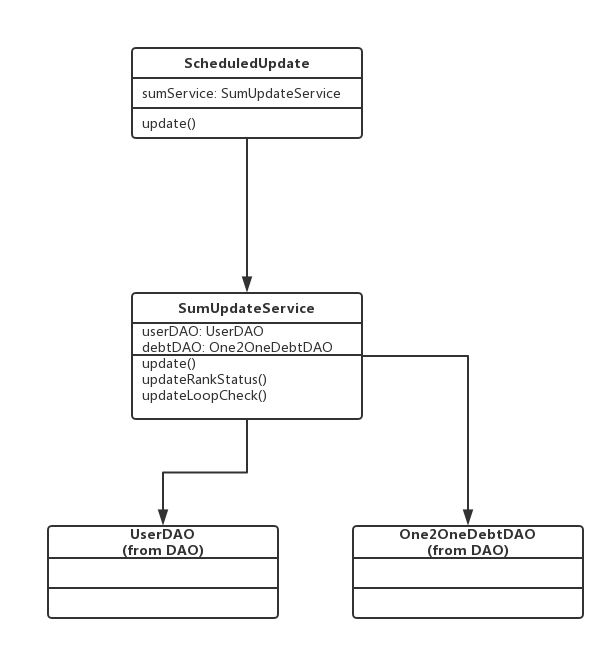
* + 1. One2OneDebt类图



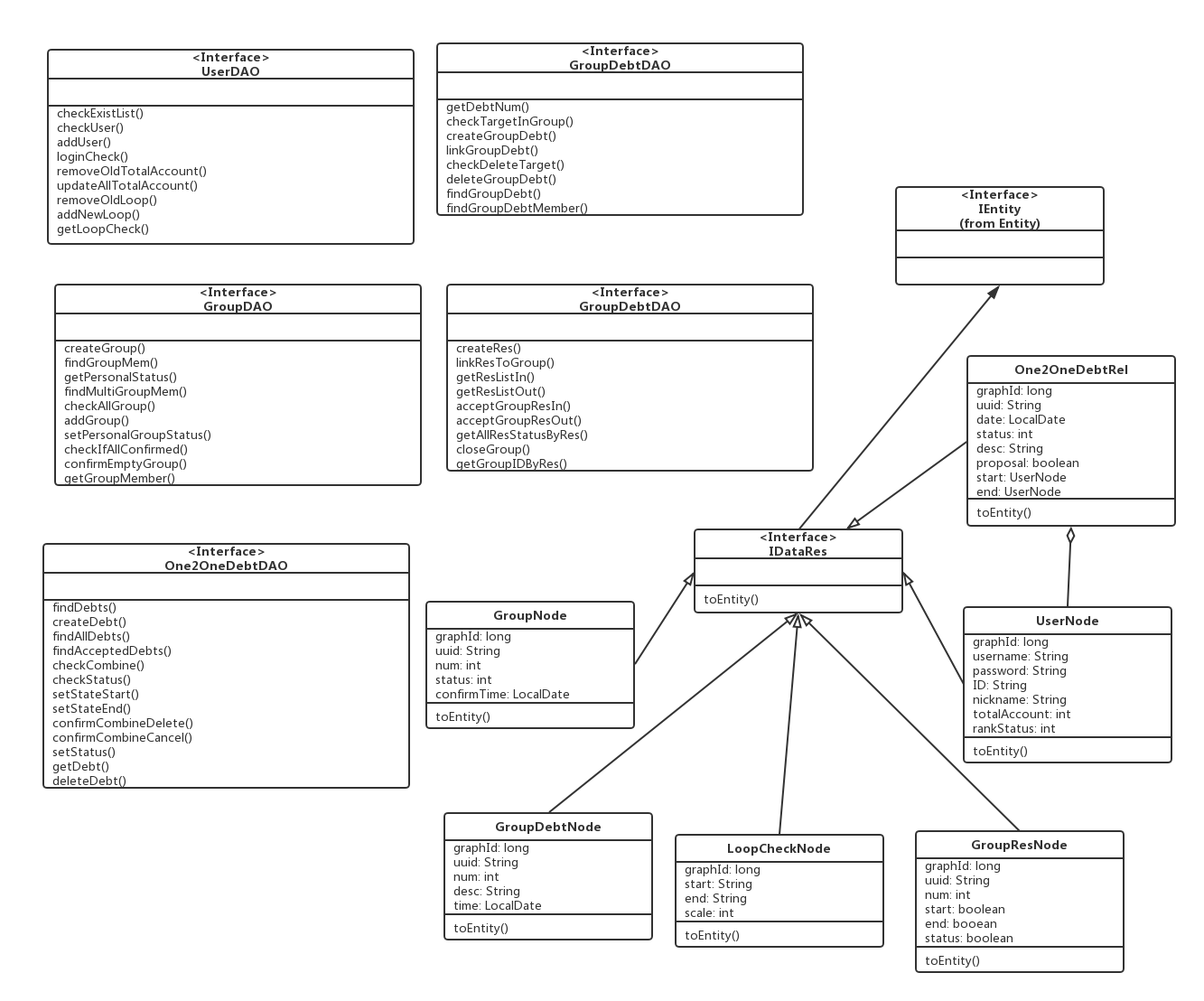
* + 1. Group类图



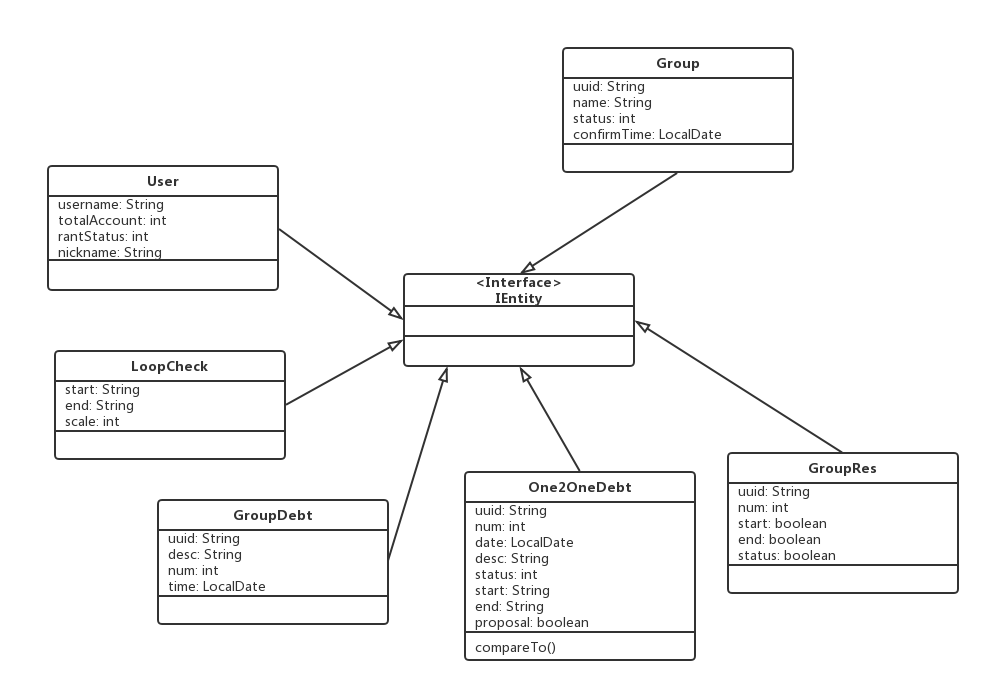
* + 1. Sum类图



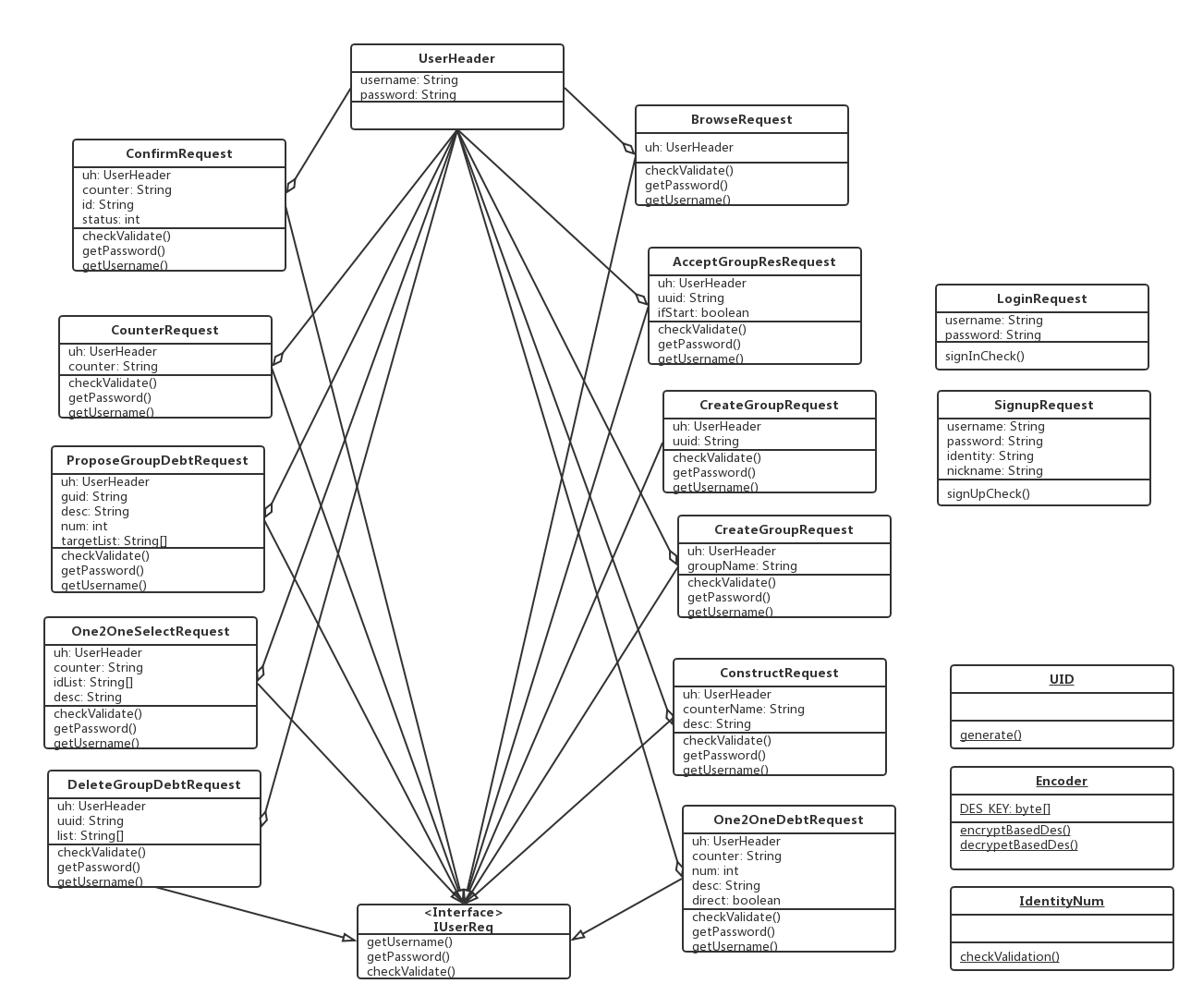
* + 1. DAO类图



* + 1. Entity类图



* + 1. Common & Util相关类图



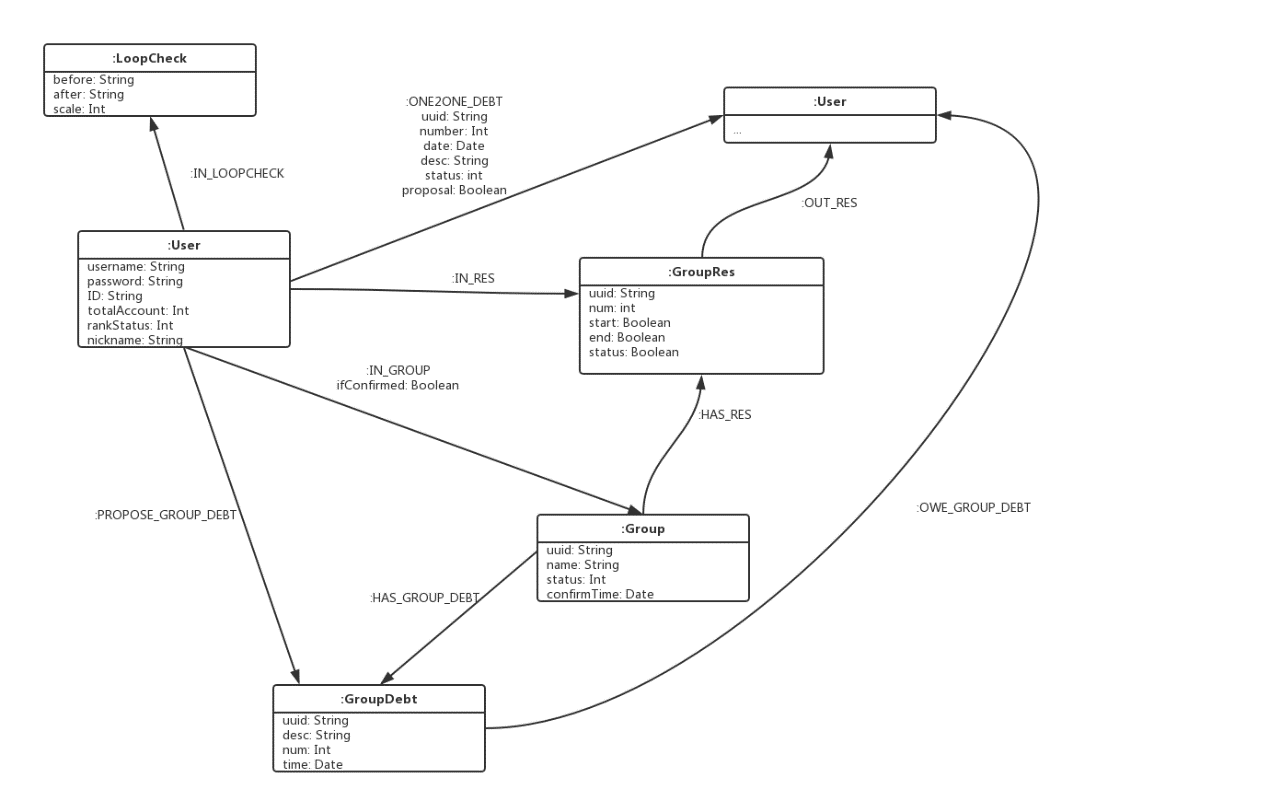
* + 1. Client类图
  1. **接口设计**

本次迭代设计中暂无与其他系统的接口

* 1. 界面设计
  2. **数据库设计**

本系统使用图数据库Neo4j作为数据库

数据库内存储结构如下：



图数据库中的数据按“节点”与“边”存储

**节点：**

类型：User

说明：存储用户实体信息

属性：

username: String 用户名

password: String 密码（加密后）

ID: String 身份认证信息

totalAccount: Int 用户总账目数字，正为出，负为入

rankStatus: Int 用户账目倾向等级，范围(0,100)

nickname: String 用户的昵称

类型：Group

说明：存储群组实体信息

属性：

uuid: String全局唯一标识，亦作为群组码用作加入群组

name: String 群组名称

status: Int 取值为 [0,1,2]，0代表群组未确认，1代表群组已确认，2代表群组关闭

confirmTime: Date 若群组未确认则为群组创建时间，否则为群组确认时间

类型：GroupDebt

说明：存储群组内账目实体信息

属性：

uuid: String全局唯一标识

desc: String 账目的说明信息

num: Int 账目金额数字，由于方向通过边表示，故必为正

time: Date 账目发起的时间

类型：GroupRes

说明：存储群组确认后产生的结算账目实体信息

属性：

uuid: String全局唯一标识

num: Int 账目金额数字，由于方向通过边表示，故必为正

start: Boolean 垫付者确认状态，true为已确认

end: Boolean 目标者确认状态，true为已确认

status: Boolean 该账目确认状态，true为已确认

类型：LoopCheck

说明：存储检测环路结果实体信息

属性：

before: String 环路中用户相邻的一个其他用户，为更新检测信息效率，此处不用边来标识此关系

after: String 环路中用户相邻的另一个其他用户

scale: Int 环路规模，取值范围 [3,4,5]

**边：**

类型：ONE2ONE\_DEBT

说明：连接两个用户，存储用户间一对一记账账目实体属性的边，边的方向代表资金方向（从起始节点流出）

属性：

uuid: 全局唯一标识

number: Int 账目金额数字，必为正

date: Date 账目时间

desc: String 账目描述，长度小于等于140

status: Int 账目状态，取值范围 [0,1,2,3,4]，0代表确认状态，1代表待添加状态，2代表待删除状态，3代表待合并（合并后会消失）状态，4代表合并后新增（即合并后的结果但尚未确认）状态

proposal: Boolean 当账目不处于确认状态时有意义，为true表示上一次更改由起始节点的用户发起，反之则是结束节点用户发起

类型：IN\_GROUP

说明：连接一个用户和一个群组，代表用户在群组中的边

属性：

IfConfirmed: Boolean 用户对于群组是否确认，true则为已确认

类型：PROPOSE\_GROUP\_DEBT

说明：连接一个用户和一个群组内账目节点，代表用户发起了该笔垫付账目

属性：无

类型：OWE\_GROUP\_DEBT

说明：连接一个群组内账目节点和一个用户节点，代表群组内账目包括用户为目标

属性：无

类型：HAS\_GROUP\_DEBT

说明：连接一个群组和一个群组内账目节点，代表账目属于该群组

属性：无

类型：HAS\_RES

说明：连接一个群组和一个群组内结算账目，代表该结算账目属于该群组

属性：无

类型：IN\_RES

说明：连接一个用户和一个群组内结算账目，代表该用户为该笔结算的资金流出方（资金从该用户到另一边，结算账目另一边的人应当还款）

属性：无

类型：OUT\_RES

说明：连接一个用户和一个群组内结算账目，代表该用户为该笔结算的资金流入方（资金从另一边到该用户，该用户应当还款）

属性：无

类型：IN\_LOOPCHECK

说明：连接一个用户和一个环路检测结果，代表该用户处于该环路

属性：无

* 1. 系统出错处理设计