# 虚拟账户系统-软件设计说明书

目录

[一、引言 4](#_Toc516065133)

[1.1 编写目的 4](#_Toc516065134)

[1.2 命名规范 4](#_Toc516065135)

[1.3 术语定义 4](#_Toc516065136)

[1.3.1 虚拟账户 4](#_Toc516065137)

[1.3.2 支付 4](#_Toc516065138)

[1.3.3 支付工具 4](#_Toc516065139)

[1.3.4 电子支付 4](#_Toc516065140)

[1.4 参考资料 5](#_Toc516065141)

[1.5 相关文档 5](#_Toc516065142)

[1.6 版本更新信息 5](#_Toc516065143)

[二、总体设计 5](#_Toc516065144)

[2.1 运行环境 5](#_Toc516065145)

[2.1.1 硬件运行环境 5](#_Toc516065146)

[2.1.2 软件运行环境 5](#_Toc516065147)

[2.2 开发环境 6](#_Toc516065148)

[2.3 功能模块清单 6](#_Toc516065149)

[三、数据库设计 7](#_Toc516065150)

[3.1 数据库技术概述 7](#_Toc516065151)

[3.2 数据库表名列表 8](#_Toc516065152)

[3.3 数据库表间关系 8](#_Toc516065153)

[3.4 数据库表详细清单 9](#_Toc516065154)

[3.4.1 平台信息表 9](#_Toc516065155)

[3.4.2 机构信息表 9](#_Toc516065156)

[3.4.3 用（商）户信息表 9](#_Toc516065157)

[3.4.4 交易信息表 10](#_Toc516065158)

[3.4.5 备注 10](#_Toc516065159)

[四、接口设计 10](#_Toc516065160)

[4.1 接口说明 10](#_Toc516065161)

[4.1.1 agencyLogin方法 10](#_Toc516065162)

[4.1.2 userLogin方法 11](#_Toc516065163)

[4.1.3 userRegister方法 11](#_Toc516065164)

[4.1.4 userPasswdChanging方法 12](#_Toc516065165)

[4.1.5 agencyInformation方法 13](#_Toc516065166)

[4.1.6 agencyAllUser方法 13](#_Toc516065167)

[4.1.7 userInformation方法 13](#_Toc516065168)

[4.1.8 freezeUnfreeze方法 14](#_Toc516065169)

[4.1.9 foundPasswd方法 15](#_Toc516065170)

[4.1.10 agencyTradeInformation方法 15](#_Toc516065171)

[4.1.11 userTradeInformation方法 16](#_Toc516065172)

[4.1.12 transferConsumer方法 16](#_Toc516065173)

[4.1.13 reCharge方法 17](#_Toc516065174)

[4.1.14 drawMoney方法 17](#_Toc516065175)

[4.2 已使用框架 18](#_Toc516065176)

[4.2.1 Mybatis 18](#_Toc516065177)

[4.2.2 Druid 18](#_Toc516065178)

[4.2.3 Redis 18](#_Toc516065179)

[4.2.4 Springframework 19](#_Toc516065180)

[4.2.5 Log4j 20](#_Toc516065181)

[4.2.6 Dubbo 20](#_Toc516065182)

[4.3 异常处理 20](#_Toc516065183)

[五、角色授权设计 21](#_Toc516065184)

[六、系统错误处理 21](#_Toc516065185)

[6.1 错误类型 21](#_Toc516065186)

[6.1.1 事务内部的故障 21](#_Toc516065187)

[6.1.2 系统故障 21](#_Toc516065188)

[6.1.3 介质故障 21](#_Toc516065189)

[6.1.4 计算机病毒故障 22](#_Toc516065190)

[6.2 故障预防及补救 22](#_Toc516065191)

[6.2.1 预期的事务内部故障 22](#_Toc516065192)

[6.2.2 非预期的事务内部故障 22](#_Toc516065193)

[6.2.3 系统故障 22](#_Toc516065194)

[6.2.4 介质故障的软件容错 22](#_Toc516065195)

[6.2.5 介质故障的硬件容错 22](#_Toc516065196)

[6.2.6 计算机病毒故障 22](#_Toc516065197)

[6.3 系统维护 22](#_Toc516065198)

[七、测试计划 23](#_Toc516065199)

## 一、引言

## 1.1 编写目的

在完成虚拟账户系统需求分析的基础上，我们依据虚拟账户系统的需求规格说明书对虚拟账户系统进行设计。为了文档化对系统的设计，编写本文档。本文档主要供团队在具体实现虚拟账户系统时参考使用。

## 1.2 命名规范

①类名采用大驼峰命名法

②变量数据库列名方法名等均采用小驼峰命名法。

## 1.3 术语定义

## 1.3.1 虚拟账户

虚拟账户是有用户在网站注册时自行设置，进行交易活动时的一个中介，是根据会计科目设置，具有一定结构，对各种经济业务进行分类和系统、连续的记录，反映资产、负债和所有者权益增减变动的记账实体。支付服务商为客户提供的实现交易资金的收付和暂存管理的支付账户，一般可以通过银行账户进行资金的充值或提取。

## 1.3.2 支付

指货币债券从付款人向收付人的转移，是为清偿商品交换和劳务活动所引起的债权债务关系。

## 1.3.3 支付工具

使货币债券发生转移的发起工具与支付流程。

支付工具的演变过程：实物货币（黄金）—信用货币（纸币）—电子货币。

非现金支付工具包括：支票、转账支付、银行卡、网银等。

## 1.3.4 电子支付

电子支付是指消费者、商家和金融机构之间使用安全电子手段把支付信息通过信息网络安全地传送到银行或相应的处理机构，用来实现货币支付或资金流转的行为。电子支付过程中，货币债券以数字信息的方式被持有、处理、接收，由电子支付工具发起实现货币债券的转移。

## 1.4 参考资料

[1]吕云翔，软件工程实用教程，清华大学出版社，2015

[2]吕云翔，刘瀚诚，刘天毅，软件工程项目实训教程，清华大学出版社，2016

[3]Roger S. Pressman，Bruce R. Maxim 著，软件工程：实践者的研究方法（原书第8版），郑人杰等译，机械工业出版社，2016

[4]Ian Sommerville著，软件工程（原书第10版），彭鑫等译，机械工业出版社，2018

[5]张海藩，吕云翔，软件工程（第4版），人民邮电出版社，2013

## 1.5 相关文档

“虚拟账户系统”的《需求规格说明书》；软件设计说明书编写指南。

## 1.6 版本更新信息

版本更新记录如表1.1所示：

**表1.1 版本更新记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本号** | **创建者** | **创建日期** | **维护者** | **维护日期** |
| V1.0 | 索宇澄等 | 2018/5/1 | 李航 | 2018/5/1 |
| V2.0 | 林未等 | 2018/5/22 | 江峻 | 2018/5/22 |
| V3.0 | 林未 | 2018/5/29 | 江峻等 | 2018/5/29 |

## 二、总体设计

## 2.1 运行环境

## 2.1.1 硬件运行环境

2.4 GHz Intel Xeon E5-2680 Broadwell（v4）双核

DDR4 2G

50G 机械硬盘

## 2.1.2 软件运行环境

Ubuntu 16.04 TLS 64位

MySQL 5.7

apache-tomcat 9.0.7

dubbo-monitor-simple 2.0.0

redis 4.0.9

zookeeper-3.4.10

jre 1.8.0\_162

## 2.2 开发环境

IntelliJ IDEA 2017.3 及之后版本

jdk 1.8.0及以上

com.alibab.druid 1.1.9

org.springframework.spring-context 5.0.5.RELEASE

org.springframework.spring-jdbc 5.0.5.RELEASE

com.alibaba.dubbo 2.6.1

org.apache.curator.curator-framework 4.0.1

org.apache.zookeeper.zookeeper 3.4.10

mysql.mysql-connector-java 6.0.6

org.mybatis.mybatis 3.4.6

org.mybatis.mybatis-spring 1.3.2

org.mybatis.caches.mybatis-redis 1.0.0-beta2

javax.servlet.servlet-api 2.5

## 2.3 功能模块清单

功能模块清单如表2.1所示：

**表2.1 功能模块清单**

|  |  |  |
| --- | --- | --- |
| **编号** | **名称** | **功能描述** |
| 1 | 机构登录 | 机构在金融服务平台上通过机构名与密码进行登录操作。 |
| 2 | 用（商）户登录 | 用（商）户在金融服务平台上通过用（商）户名与密码进行登录操作。 |
| 3 | 用（商）户注册 | 用（商）户根据提示输入姓名、手机号、邮箱等信息进行注册开户，验证信息正确后传回后端，进行账户创建，注册成功后自动跳转到用（商）户信息页面。 |
| 4 | 用（商）户密码修改 | 用（商）户输入原密码及新密码进行密码修改。 |
| 5 | 机构信息查询 | 机构进入机构信息页面后自动显示基本机构信息。 |
| 6 | 机构用（商）户管理 | 机构进入管理页面后自动显示机构用（商）户管理信息。 |
| 7 | 用（商）户信息查询 | 用（商）户进行基本信息和账户信息查询。 |
| 8 | 用（商）户账户冻结与解冻 | 对用（商）户账户进行冻结与解冻操作。 |
| 9 | 用（商）户密码找回 | 对用（商）户账户进行密码找回操作。 |
| 10 | 机构交易查询 | 对机构交易信息进行约束查询，约束涉及交易时间和交易类型。 |
| 11 | 用（商）户交易查询 | 对用（商）户交易信息进行约束查询，约束涉及交易时间和交易类型。 |
| 12 | 转账与消费 | 用（商）户进行转账或消费操作。 |
| 13 | 充值 | 用（商）户进行充值操作。 |
| 14 | 提现 | 用（商）户进行提现操作。 |

## 三、数据库设计

## 3.1 数据库技术概述

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择MySQL作为网站数据库。

由于其社区版的性能卓越，搭配 PHP 和 Apache 可组成良好的开发环境。

## 3.2 数据库表名列表

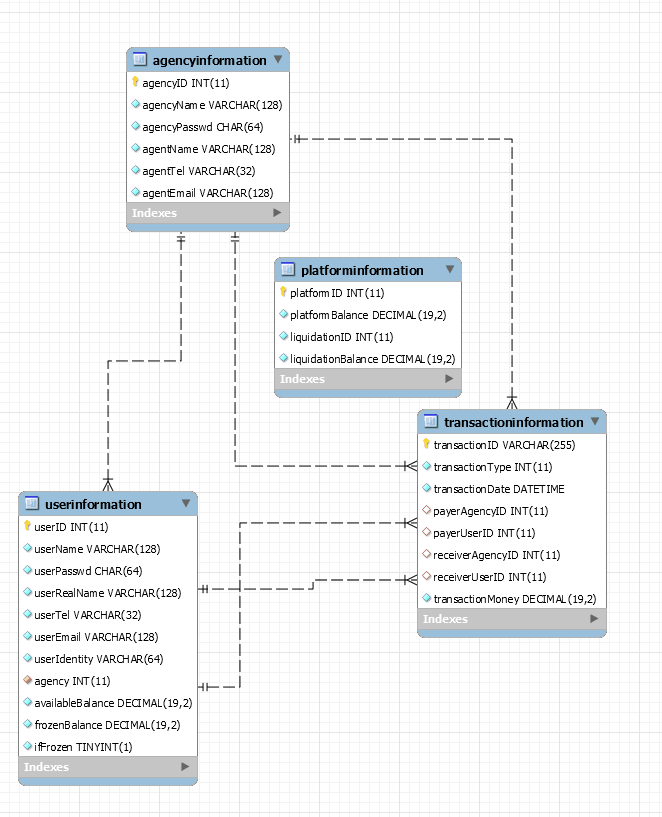
数据库表名列表如表3.1所示：

**表3.1 数据库表名列表**

|  |  |  |
| --- | --- | --- |
| **编号** | **表名** | **表功能说明** |
| 1 | PlatformInformation | 平台信息表 |
| 2 | AgencyInformation | 机构信息表 |
| 3 | UserInformation | 用（商）户信息表 |
| 4 | TransactionInformation | 交易信息表 |

## 3.3 数据库表间关系

数据库中的表之间的关系如图3.1所示：



**图3.1 数据库表间关系**

## 3.4 数据库表详细清单

## 3.4.1 平台信息表

如表3.2所示，表名为PlatformInformation。

**表3.2 平台信息表**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| 平台账户ID：platformID | int | NOT NULL, Default 1 |
| 平台账户余额：platformBalance | decimal(19,2) | NOT NULL, Default 0.0 |
| 待清算账户ID：liquidationID | int | NOT NULL, Default 2 |
| 待清算账户余额：liquidationBalance | decimal(19,2) | NOT NULL, Default 0.0 |

## 3.4.2 机构信息表

如表3.3所示，表名为AgencyInformation。

**表3.3 机构信息表**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| 机构ID：agencyID | int | PRIMARY KEY, NOT NULL,AUTO\_INCREMENT |
| 机构名：agencyName | varchar(128) | UNIQUE INDEX agencyName, NOT NULL |
| 机构密码：agencyPasswd | char(64) | NOT NULL |
| 负责人姓名: agentName | varchar(128) | NOT NULL |
| 负责人手机：agentTel | varchar(32) | NOT NULL |
| 负责人邮箱：agentEmail | varchar(128) | NOT NULL |

## 3.4.3 用（商）户信息表

如表3.4所示，表名为UserInformation。

**表3.4 用（商）户信息表**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| 用（商）户ID：userID | int | PRIMARY KEY, NOT NULL,AUTO\_INCREMENT |
| 用（商）户名：userName | varchar(128) | UNIQUE INDEX userName, NOT NULL |
| 用（商）户密码：userPasswd | char(64) | NOT NULL |
| 用（商）户姓名：userRealName | varchar(128) | NOT NULL |
| 用（商）户手机号：userTel | varchar(32) | NOT NULL |
| 用（商）户邮箱：userEmail | varchar(128) | NOT NULL |
| 用（商）户身份凭证：userIdentify | varchar(64) | NOT NULL |
| 所属机构：agency | int | FOREIGN KEY agencyID, NOT NULL，NORMAL INDEX agency,ON DELETE/UPDATE CASCADE |
| 可用余额：availableBalance | decimal(19,2) | NOT NULL,Default0.0 |
| 冻结余额：frozenBalance | decimal(19,2) | NOT NULL,Default0.0 |
| 状态： ifFrozen | tinyint | NOT NULL,Default 0 |

## 3.4.4 交易信息表

如表3.5所示，表名为TransactionInformation。

**表3.5 交易信息表**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| 交易ID：transactionID | varchar(255) | PRIMARY KEY, NOT NULL,AUTO\_INCREMENT |
| 交易类型：transactionType | int | NOT NULL |
| 交易时间：transactionDate | datetime | NOT NULL |
| 付款方机构ID：payerAgencyID | int | FOREIGN KEY agencyID |
| 付款方用（商）户ID：payerUserID | int | FOREIGN KEY userID |
| 收款方机构ID：receiverAgencyID | int | FOREIGN KEY agencyID |
| 收款方用（商）户ID：receiverUserID | int | FOREIGN KEY userID |
| 交易金额：transactionMoney | decimal(19,2) | NOT NULL |

## 3.4.5 备注

①身份凭证中，前缀Y表示用户、S表示商户。

②UNIQUE INDEX (userIdentify, agency) Account。

③所有索引均是BTREE索引，不存在HASH索引。

④int默认长度为11位。

⑤用（商）户ID从3开始递增。

## 四、接口设计

## 4.1 接口说明

虚拟账户系统服务接口信息如表4.1所示：

**表4.1 虚拟账户系统服务接口**

|  |  |
| --- | --- |
| **接口名称** | buaa.jj.accountservice.api.AccountService |
| **接口地址** | dubbo://140.143.4.171:20880/buaa.jj.accountservice.api.AccountService |
| **接口协议** | dubbo |
| **服务注册中心** | <dubbo:registry address="zookeeper://140.143.4.171:2181"/> |

## 4.1.1 agencyLogin方法

①方法描述：机构登录方法。对登录机构名密码进行验证，成功登录返回机构ID，失败返回-1。

②请求参数：如表4.2所示。

**表4.2 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| String | agency\_name | 机构名 | 无 |
| String | agency\_passwd | 机构密码 | SHA256后传入 |

③响应参数：如表4.3所示。

**表4.3 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| int | 成功登录返回对应机构ID>=0，否则返回-1 |

## 4.1.2 userLogin方法

①方法描述：用户登录方法。对登录用户名密码进行验证，成功登录返回用户ID，失败返回-1。（注：这里用户泛指用（商）户，下同。）

②请求参数：如表4.4所示。

**表4.4 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| String | user\_name | 用户名 | 无 |
| String | user\_passwd | 用户密码 | SHA256后传入 |

③响应参数：如表4.5所示。

**表4.5 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| int | 成功登录返回对应用户ID>=0，否则返回-1 |

④异常描述：如表4.6所示。

**表4.6 异常描述**

|  |  |
| --- | --- |
| **异常类型** | **异常描述** |
| UserNotExistException | 用户不存在，无法登录 |
| UserFrozenException | 用户被冻结，无法登录 |

## 4.1.3 userRegister方法

①方法描述：用户注册方法，成功注册返回对应用户ID。

②请求参数：如表4.7所示。

**表4.7 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| String | user\_name | 用户（商户）名 | 无 |
| String | user\_passwd | 用户（商户）密码 | SHA256后传入 |
| String | user\_realname | 用户（商户）真实姓名（商户名称） | 无 |
| String | user\_tel | 用户（法人代表）电话 | 无 |
| String | user\_email | 用户（商户）Email | 无 |
| String | user\_identity | 用户身份认证号（商户许可证号） | 无 |
| int | under\_agency\_id | 用户（商户）所属机构 | 存在String类型的重载方法，under\_agency\_name，即用户（商户）所属机构名 |

③响应参数：如表4.8所示。

**表4.8 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| int | 成功返回用户ID，失败抛出异常 |

④异常描述：如表4.9所示。

**表4.9 异常描述**

|  |  |
| --- | --- |
| **异常类型** | **异常描述** |
| NameDuplicateException | 注册用户的用户名重复，无法注册 |
| AgencyNotExistException | 注册用户的所属机构不存在，无法注册 |
| UserAgencyDuplicateException | 注册用户的身份证号在该机构下已注册过帐号，无法注册 |

## 4.1.4 userPasswdChanging方法

①方法描述：用户（商户）修改密码，成功返回true。

②请求参数：如表4.10所示。

**表4.10 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | user\_id | 用户（商户）ID | 无 |
| String | old\_passwd | 用户（商户）原密码 | SHA256后传入 |
| String | new\_passwd | 用户（商户）新密码 | SHA256后传入 |

③响应参数：如表4.11所示。

**表4.11 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| Boolean | 成功返回true，失败抛出异常 |

④异常描述：如表4.12所示。

**表4.12 异常描述**

|  |  |
| --- | --- |
| **异常类型** | **异常描述** |
| UserNotExistException | 用户不存在，无法更改密码 |
| UserFrozenException | 用户被冻结，无法更改密码 |

## 4.1.5 agencyInformation方法

①方法描述：查询机构信息。

②请求参数：如表4.13所示。

**表4.13 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | agency\_id | 机构ID | 无 |

③响应参数：如表4.14所示。

**表4.14 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| Map<String,Object> | 成功返回机构信息，失败返回空Map |

④Map字段描述：如表4.15所示，注意键大小写敏感。

**表4.15 Map字段描述**

|  |  |
| --- | --- |
| **键** | **值描述** |
| agencyID | 机构ID |
| agencyName | 机构名 |
| agentName | 机构代理姓名 |
| agentTel | 机构代理电话 |
| agentEmail | 机构代理邮箱 |

## 4.1.6 agencyAllUser方法

①方法描述：获取机构下所有用户ID。

②请求参数：如表4.16所示。

**表4.16 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | agency\_id | 机构ID | 无 |

③响应参数：如表4.17所示。

**表4.17 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| List<Integer> | 成功返回机构用户ID列表，失败返回空表 |

## 4.1.7 userInformation方法

①方法描述：查询用户信息。

②请求参数：如表4.18所示。

**表4.18 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | user\_id | 用户ID | 无 |

③响应参数：如表4.19所示。

**表4.19 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| Map<String,Object> | 成功返回用户信息，失败返回空Map |

④Map字段描述：如表4.20所示，注意键大小写敏感。

**表4.20 Map字段描述**

|  |  |
| --- | --- |
| **键** | **值描述** |
| userID | 用户ID |
| userName | 用户名 |
| userTel | 用户电话 |
| userEmail | 用户邮箱 |
| userIdentity | 用户身份证明 |
| agency | 所属机构 |
| availableBalance | 可用余额 |
| frozenBalance | 冻结余额 |
| isFrozen | 是否被冻结（冻结为true） |

## 4.1.8 freezeUnfreeze方法

①方法描述：冻结、解冻账户。

②请求参数：如表4.21所示。

**表4.21 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | user\_id | 用户ID | 无 |
| boolean | is\_frozen | 操作类型 | true代表冻结，false代表解冻 |

③响应参数：如表4.22所示。注意该方法不判断操作前用户的账户状态，例：若账户原本已经冻结，is\_frozen传入true，账户状态不变，但仍返回1，即认为冻结成功。

**表4.22 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| int | 0操作无效，1冻结，2解冻 |

④异常描述：如表4.23所示。

**表4.23 异常描述**

|  |  |
| --- | --- |
| **异常类型** | **异常描述** |
| UserNotExistException | 不存在该ID的用户 |

## 4.1.9 foundPasswd方法

①方法描述：用身份认证直接替换现有密码。

②请求参数：如表4.24所示。

**表4.24 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| String | user\_name | 用户名 | 无 |
| String | user\_identity | 身份证号/商户许可证号 | 根据用户身份增加相应前缀，用户为Y，商户为S |
| String | user\_passwd | 用户新密码 | SHA256 |

③响应参数：如表4.25所示。

**表4.25 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| boolean | true 成功，false失败 |

④异常描述：如表4.26所示。

**表4.26 异常描述**

|  |  |
| --- | --- |
| **异常类型** | **异常描述** |
| UserNotExistException | 用户不存在，无法找回密码 |
| UserFrozenException | 用户被冻结，无法找回密码 |

## 4.1.10 agencyTradeInformation方法

①方法描述：获取机构交易记录。

②请求参数：如表4.27所示。

**表4.27 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | agency\_id | 机构ID | 无 |
| String | start\_date | 记录开始日期 | 无 |
| String | end\_date | 记录结束日期 | 无 |
| int | trade\_type | 交易类型 | 0充值，1提现，2转账与消费 |

③响应参数：如表4.28所示。其中map的key为交易ID，map的value为交易信息。

**表4.28 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| List<Map<Integer,Object>> | 由交易编号、交易明细组成的键值对的列表 |

## 4.1.11 userTradeInformation方法

①方法描述：获取用户交易记录。

②请求参数：如表4.29所示。

**表4.29 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | user\_id | 用户ID | 无 |
| String | start\_date | 记录开始日期 | 无 |
| String | end\_date | 记录结束日期 | 无 |
| int | trade\_type | 交易类型 | 0充值，1提现，2转账与消费 |

③响应参数：如表4.30所示。其中map的key为交易ID，map的value为交易信息。

**表4.30 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| List<Map<Integer,Object>> | 由交易编号、交易明细组成的键值对的列表 |

## 4.1.12 transferConsumer方法

①方法描述：转账与消费。

②请求参数：如表4.31所示。

**表4.31 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | pay\_user\_id | 付款方用户ID | 无 |
| int | get\_user\_id | 收款方用户ID | 无 |
| double | amount | 转账额 | 无 |
| boolean | trade\_type | 交易类型 | false转账，true消费 |

③响应参数：如表4.32所示。

**表4.32 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| boolean | 成功返回true，失败返回false |

④异常描述：如表4.33所示。

**表4.33 异常描述**

|  |  |
| --- | --- |
| **异常类型** | **异常描述** |
| UserNotExistException | 用户不存在，无法转帐或消费 |
| UserFrozenException | 用户被冻结，无法转账或消费 |

## 4.1.13 reCharge方法

①方法描述：充值。

②请求参数：如表4.34所示。

**表4.34 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | user\_id | 用户ID | 无 |
| double | amount | 充值金额 | 无 |
| boolean | recharge\_platform | 使用平台 | false微信，true支付宝 |

③响应参数：如表4.35所示。

**表4.35 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| boolean | 成功返回true，失败返回false |

④异常描述：如表4.36所示。

**表4.36 异常描述**

|  |  |
| --- | --- |
| **异常类型** | **异常描述** |
| UserNotExistException | 用户不存在，无法充值 |

## 4.1.14 drawMoney方法

①方法描述：提现。

②请求参数：如表4.37所示。

**表4.37 请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数类型** | **参数名** | **参数描述** | **备注** |
| int | user\_id | 用户ID | 无 |
| double | amount | 取款金额 | 无 |
| boolean | recharge\_platform | 使用平台 | false微信，true支付宝 |

③响应参数：如表4.38所示。

**表4.38 响应参数**

|  |  |
| --- | --- |
| **响应类型** | **响应描述** |
| boolean | 成功返回true，失败返回false |

④异常描述：如表4.39所示。

**表4.39 异常描述**

|  |  |
| --- | --- |
| **异常类型** | **异常描述** |
| UserNotExistException | 用户不存在，无法提现 |
| UserFrozenException | 用户被冻结，无法提现 |

## 4.2 已使用框架

## 4.2.1 Mybatis

MyBatis是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。MyBatis框架具有如下优点：

①简单易学：本身就很小且简单。没有任何第三方依赖，最简单安装只要两个jar文件+配置几个sql映射文件易于学习，易于使用，通过文档和源代码，可以比较完全的掌握它的设计思路和实现。

②灵活：mybatis不会对应用程序或者数据库的现有设计强加任何影响。sql写在xml里，便于统一管理和优化。通过sql基本上可以实现我们不使用数据访问框架可以实现的所有功能，或许更多。

③解除sql与程序代码的耦合：通过提供DAL层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元测试。sql和代码的分离，提高了可维护性。

④提供映射标签，支持对象与数据库的orm字段关系映射

⑤提供对象关系映射标签，支持对象关系组建维护

⑥提供xml标签，支持编写动态sql。

## 4.2.2 Druid

Druid是java下面的数据库连接池，有非常强大的监控功能，并且安全性较高，可以防御多种数据库注入并且监控每个语句的执行者以及执行时间等。

## 4.2.3 Redis

Redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，Redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是Redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis 是一个高性能的key-value数据库。redis的出现，很大程度补偿了memcached这类key/value存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

通过Redis可以实现数据库访问加速。

## 4.2.4 Springframework

Springframework框架具有如下特性：

轻量——从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的：典型地，Spring应用中的对象不依赖于Spring的特定类。

控制反转——Spring通过一种称作控制反转（IoC）的技术促进了低耦合。当应用了IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。你可以认为IoC与JNDI相反——不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

面向切面——Spring提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（例如审计（auditing）和事务（transaction）管理）进行内聚性的开发。应用对象只实现它们应该做的——完成业务逻辑——仅此而已。它们并不负责（甚至是意识）其它的系统级关注点，例如日志或事务支持。

容器——Spring包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，你可以配置你的每个bean如何被创建——基于一个可配置原型（prototype），你的bean可以创建一个单独的实例或者每次需要时都生成一个新的实例——以及它们是如何相互关联的。然而，Spring不应该被混同于传统的重量级的EJB容器，它们经常是庞大与笨重的，难以使用。

框架——Spring可以将简单的组件配置、组合成为复杂的应用。在Spring中，应用对象被声明式地组合，典型地是在一个XML文件里。Spring也提供了很多基础功能（事务管理、持久化框架集成等等），将应用逻辑的开发留给了你。

MVC——Spring的作用是整合，但不仅仅限于整合，Spring 框架可以被看做是一个企业解决方案级别的框架。客户端发送请求，服务器控制器（由DispatcherServlet实现的)完成请求的转发，控制器调用一个用于映射的类HandlerMapping，该类用于将请求映射到对应的处理器来处理请求。HandlerMapping将请求映射到对应的处理器Controller（相当于Action）在Spring 当中如果写一些处理器组件，一般实现Controller 接口，在Controller 中就可以调用一些Service 或DAO 来进行数据操作ModelAndView用于存放从DAO 中取出的数据，还可以存放响应视图的一些数据。如果想将处理结果返回给用户，那么在Spring 框架中还提供一个视图组件ViewResolver，该组件根据Controller 返回的标示，找到对应的视图，将响应response 返回给用户。

## 4.2.5 Log4j

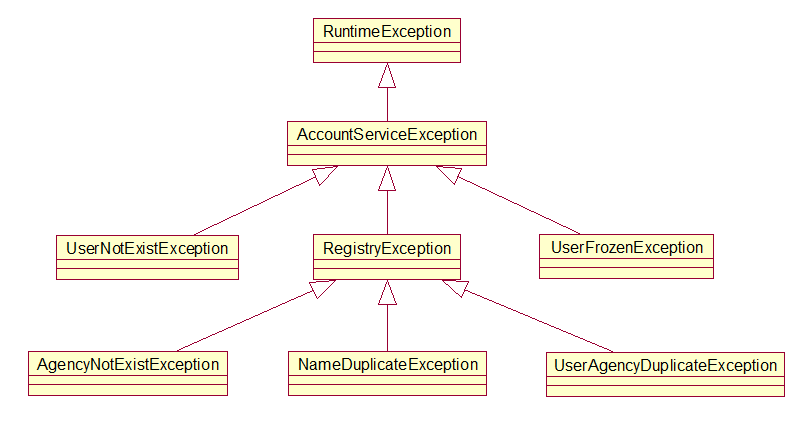
Log4j是Apache的一个开源项目，通过使用Log4j，我们可以控制日志信息输送的目的地是控制台、文件、GUI组件，甚至是套接口服务器、NT的事件记录器、UNIX Syslog守护进程等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。最令人感兴趣的就是，这些可以通过一个配置文件来灵活地进行配置，而不需要修改应用的代码。

## 4.2.6 Dubbo

Dubbo是阿里巴巴公司开源的一个高性能优秀的服务框架，使得应用可通过高性能的 RPC 实现服务的输出和输入功能，可以和 Spring框架无缝集成。

## 4.3 异常处理

各接口对应异常在“接口说明”部分已作详细描述。异常间的继承关系如图4.1所示：



**图4.1 异常继承图**

## 五、角色授权设计

角色授权设计如表5.1所示：

**表5.1 角色授权设计**

|  |  |  |
| --- | --- | --- |
| **功能** | **用（商）户** | **机构** |
| 登录 | ● | ● |
| 注册 | ● |  |
| 密码修改 | ● |  |
| 信息查询 | ● | ● |
| 用（商）户管理 |  | ● |
| 用（商）户冻结解冻 | ● |  |
| 用（商）户密码找回 | ● |  |
| 交易查询 | ● | ● |
| 转账与消费 | ● |  |
| 充值 | ● |  |
| 提现 | ● |  |

## 六、系统错误处理

## 6.1 错误类型

## 6.1.1 事务内部的故障

事务内部故障可分为预期的和非预期的，其中大部分的故障都是非预期的。预期的事务内部故障是指可以通过事务程序本身发现的事务内部故障；非预期的事务内部故障是不能由事务程序处理的，如运算溢出故障、并发事务死锁故障、违反了某些完整性限制而导致的故障等。

## 6.1.2 系统故障

系统故障也称为软故障，是指数据库在运行过程中，由于硬件故障、数据库软件及操作系统的漏洞、突然停电灯情况，导致系统停止运转，所有正在运行的事务以非正常方式终止，需要系统重新启动的一类故障。这类事务不破坏数据库，但是影响正在运行的所有事务。

## 6.1.3 介质故障

介质故障也称为硬故障，主要指数据库在运行过程中，由于磁头碰撞、磁盘损坏、强磁干扰、天灾人祸等情况，使得数据库中的数据部分或全部丢失的一类故障。

## 6.1.4 计算机病毒故障

计算机病毒故障是一种恶意的计算机程序，它可以像病毒一样繁殖和传播，在对计算机系统造成破坏的同时也可能对数据库系统造成破坏（破坏方式以数据库文件为主）。

## 6.2 故障预防及补救

## 6.2.1 预期的事务内部故障

将事务回滚，撤销对数据库的修改。

## 6.2.2 非预期的事务内部故障

强制回滚事务，在保证该事务对其他事务没有影响的条件下，利用日志文件撤销其对数据库的修改。

## 6.2.3 系统故障

待计算机重新启动之后，对于未完成的事务可能写入数据库的内容，回滚所有未完成的事务写的结果；对于已完成的事务可能部分或全部留在缓冲区的结果，需要重做所有已提交的事务（即撤销所有未提交的事务，重做所有已提交的事务）。

## 6.2.4 介质故障的软件容错

使用数据库备份及事务日志文件，通过恢复技术，恢复数据库到备份结束时的状态。

## 6.2.5 介质故障的硬件容错

采用双物理存储设备，使两个硬盘存储内容相同，当其中一个硬盘出现故障时，及时使用另一个备份硬盘。

## 6.2.6 计算机病毒故障

使用防火墙软件防止病毒侵入，对于已感染病毒的数据库文件，使用杀毒软件进行查杀，如果杀毒软件杀毒失败，此时只能用数据库备份文件，以软件容错的方式恢复数据库文件。

## 6.3 系统维护

我们使用数据库的维护功能机制，比如，定期备份数据库，定期检测数据库的一致性，定期查看操作日志等；文件方面：对于下架商品等，定期删除相关文件，减少数据量。编码实现时应采用模块化和分层的思想，提高模块内部的内聚，减少模块间的耦合。使系统逻辑结构清晰，从而增强可读性和可维护性。在编码过程中注意标识符命名的意义，添加适量注释。

## 七、测试计划

测试计划作为本系统的测试指导，测试成员按计划和规定进行测试。本系统的测试将从单元测试、集成测试、系统测试三个方面来设计。