



**QG工作室项目报告**

学 院 计算机学院

专 业 计算机类

班 级 23级计算机类（3）班

组 别 后台组

姓 名 冯昊铿

学 号 3123004225

2024年 5月5 日

广东工业大学计算机学院

数据库发生死锁了

目录

[1.项目介绍 3](#_Toc165759038)

[2.项目要求 3](#_Toc165759039)

[3.设计思路 6](#_Toc165759040)

[4.详细设计 11](#_Toc165759041)

[5.技术搭建 18](#_Toc165759042)

[6.程序测试 18](#_Toc165759043)

[7.项目亮点 20](#_Toc165759044)

[8项目难点 29](#_Toc165759045)

[9.心得体会 32](#_Toc165759046)

# 1.项目介绍

**背景** **:**

数字货币成为越来越多人的支付方式选择，一个安全高效，功能丰富的资金管理网站可以更好地满 足用户的需求，可以让用户更好地管理自己的资金，确认资金明细。

本项目旨在构建一个多功能的企业群组管理平台，该平台融合了企业群组管理、用户信息管理、资金分配与监控等功能。以满足用户对流动资金的管理，旨在提供一个安全高效，功能丰富的 资金管理系统。

# 2.项目要求

**1.**主要对象分为三个模块： 系统模块 、用户模块、资金模块

**（1）系统模块**

**1.企业群组**

1. 普通用户可以申请企业群组并成为企业群组负责人

2. 企业群组中有企业的介绍和相关信息，如企业名称、企业人数、企业规模、企业工作方向等。

3. 企业群组可以设置申请群组方式，公开模式下群组可以被搜索到，通过ID等其他方式主动添 加，私密模式下仅可通过管负责人拉取

4. 企业群组内，有权限的人可以收款

**2. 用户中心**

1. 用户可以设置、查看、修改个人信息，包括用户名、头像、手机号等信息。

2. 用户可以查看自己的资金明细，并且分类查看账单流水。

3. 用户可以选择查看所属企业群组，并且进行申请或者退出。

**3. 资金中心**

1. 用户可以在这里支配自己的资金。

2. 用户支出资金可以通过企业群组内分配的资金支付并且选择是否上传报销文件。

3. 用户可以查看在软件的总资金，也可以查看不同群组的资金余额。

4. 企业群组管理员可以充值到企业群组里并且分配资金给企业群组成员。 5. 企业群组管理员可以查看企业群组里所有资金流动明细。

**(2) 用户模块**

**1. 普通用户**

账号注册（录入用户名、昵称、密码、地址等基本信息，可以自行拓展）

实现登录、退出登录功能（未登录为游客权限）

支持查看、修改个人信息（昵称、密码、资金和所属企业群组等。用户注册后会有一个默认头 像）

可以进行搜索公开群组查看群组的基本信息

可以申请加入公开群组也可以被企业群组负责人拉入群组，也可以选择退出群组。

可以查看流水

**2. 游客**

未注册账号的用户是游客，仅可以搜索公开企业

**3. 企业群组负责人**

企业负责人具有普通用户所有功能 企业群组负责人是普通用户向平台申请创办企业群组负责人

申请成功后，企业负责人可设置企业的基本信息，并且设置是否公开 企业负责人可以将资金充值到某个企业群组中，并且任意支配（分配或者收回）资金到人，

企业负责人可以选择注销企业群组

**4. 网站管理员**

可以设置企业、用户是否封禁。

可以通过企业管理员、用户的恢复申请。

可以查看网站内余额总数以及异常交易

**(3) 资金模块**

. **1.**用户使用自己或群组资金支付流程，需注意，群组资金实际仍在群组名义下，因此转账过程中主体为群组

**2. 用户使用个人或者群组资金收款流程：**

**1.支付：**开始支付（发起支付） 授权（输入密码并校验密码是否正确，检验支付环境是否正常） 校验余额，检测余额是否充足。

资金冻结（获取用户资金并冻结对应资金，存储在数据库） 结算

结算：更改支付状态为已结算，并进行入账到第三方 不结算：更改支付状态为退款，将资金从数据库转回用户账户

入账：向第三方发起付款请求 入账成功：成功入账，资金从数据库清除，告知用户并扣除资金 入账失败：入账失败，资金保留数据库，提示用户是否再次进行入账 记录交易信息

**2.收款：** 用户使用个人或者群组资金收款流程： 提供收款接口 接收第三方收款请求，验证请求合法性 授权成功，冻结金额，避免重复支付或者支付失败 授权失败，拒绝收款请求

获取收款用户ID及其他附加信息 生成资金存储在数据库 将收款金额入账到收款人账户，如果入账成功，则进行余额更改；如果入账失败，则进行相应 处理。 更新收款人账户余额，表示收款成功。返回成功请求 记录交易信息

# 3.设计思路

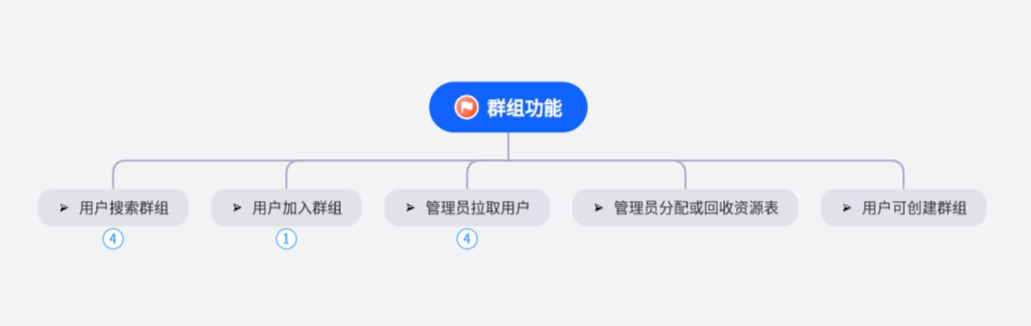
大体上照着这三个模块完成，分别是群组的界面，用户中心界面，资金交易界面，其中群组界面作为主要界面，用户中心用于查看、修改用户有关信息、资金交易界面主要用于支付收款。详情如下

**1.前端页面**



图表 1

**2.群组界面下对应的功能或者子页面如下**



图表 2

**3.用户中心主页面对应的功能或者其子页面**



图表 3

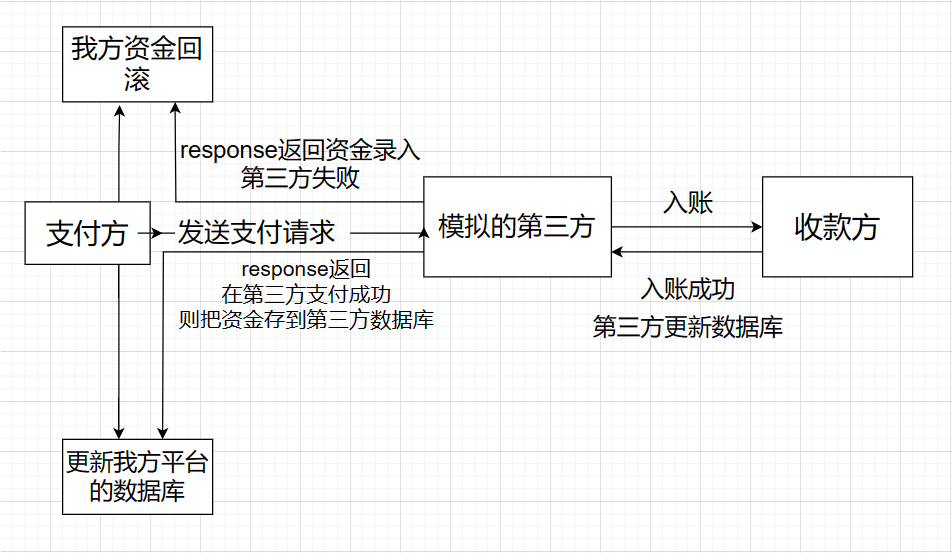
**4.资金中心的页面对应的功能或者子页面**



图表 4

**5.第三方接口的设计思路：**

首先我对于第三方接口的理解是，它不根植与本平台，是与本平台独立的系统。而在现实生活中，调用接口存在延时问题，因此在支付转账交易过程中实现数据的同步是一重点，或者说，如何处理解决在突发的情况下导致两端数据不一致的问题。因此在设计之初，就想到了结合数据库的事务回滚，解决一致性的问题。

****

图表 5

**6.数据库表设计思路：**

**（1）资源分配的权限表的设计：**

用于管理用户、角色以及他们对系统资源的访问权限,用了经典的五张表设计思路：

1.用户表（用户的基本信息）

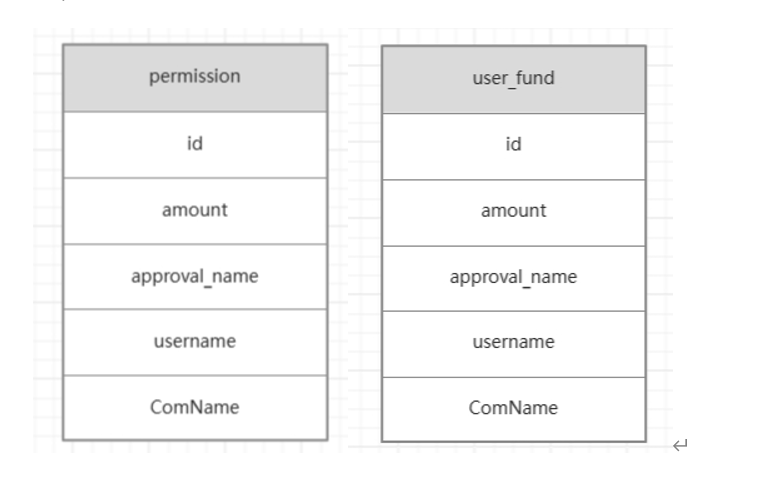
2.用户角色表（用户扮演的角色）

3.资源表（用户被分配的资源）

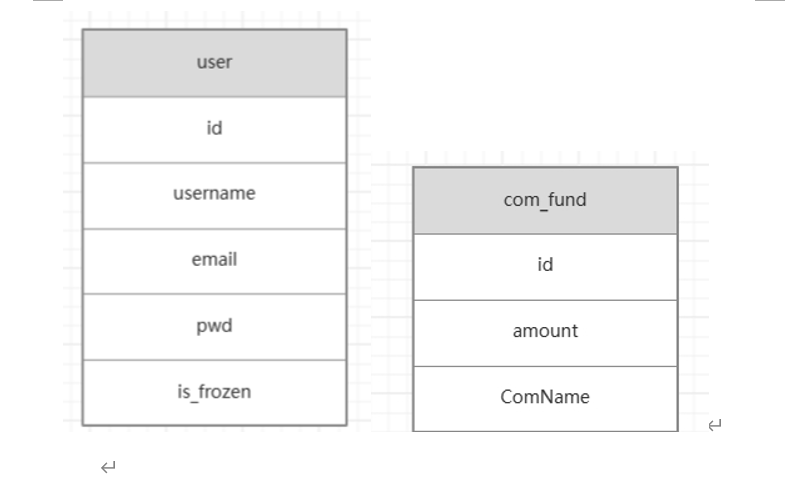
4.角色权限表 （用户能访问哪些资源）

5.角色表（平台中所有的角色）

而在本项目中，考虑到其他限制的因素，我优化了五张表的设计结构。以下就是本项目中资金分配的表的设计。

****

图表 6

****

图表 7

**（2）用户关系表**

本项目中用户有四种角色身份，一是游客、二是普通用户、三是群组管理员，还有网站的管理人员四种角色。其中前三种角色是用户可以切换成为的角色。

为了简化数据库表的数量以及减少服务器对用户身份的查询，以上用户四种角色都由一张表维护。表中由id、用户名、用户密码、用户邮箱、头像地址、用户角色、是否冻结这七个字段组成。

# 4.详细设计

**（1）用户登录流程**

用户若未登录则会被后台认定为有游客状态，前端获取接口的资源就会受限，并且前端使用的Vue框架，根据识别到的游客身份，会自动隐藏只有登录后才能获取的资源路径。

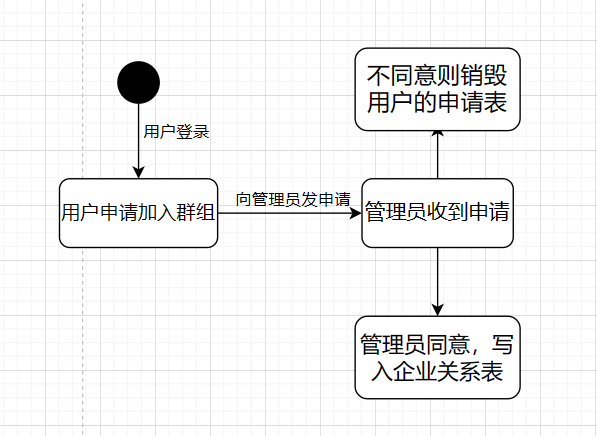
服务器拿到用户登录的数据后会与数据库的数据比较，若一致，则会把用户名保存到session当中去，后期也是根据这个session来判定用户的身份以及获取对应的资源，这样既在服务器保证了资源的获取，也达到了权限限制的目的。

用户忘记密码的功能实现需要通过用户注册时的邮箱找回，我这里实现了qq邮箱的接口，用户输入正确的邮箱信息，管理员就会发送验证码至用户邮箱，用户填写正确的验证码就可以重新设置新密码。

****

图表 8

**（2）用户加入群组的流程**

****

图表 9

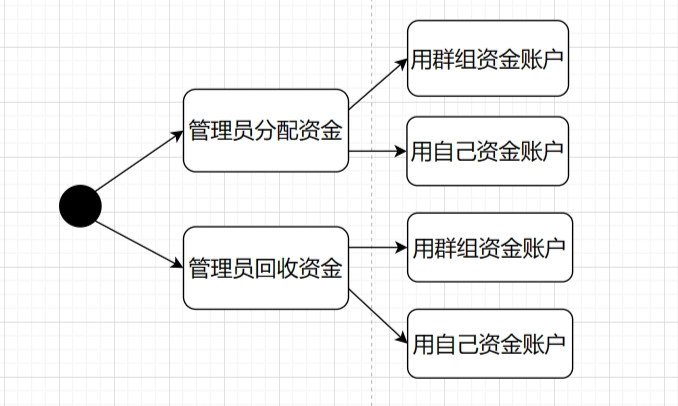
**（3）群组管理员分配或回收资源流程**

管理员用**群组资金**分配或回收用户资金时，群组账户资金不变，但该用户下的资金增添了一个权限表，账户主体是群组，使用者是该用户，一旦该用户动用群组资金，资金的减少发生在群组账户上，管理员也因此能查看是哪位用户动用了资金。但这里会有一个问题：如果群组账户的资金小于用户被分配到的资金时，用户使用群组资金会出现限额现象，即金额不能大于群组账户资金。

同理，回收资金时，只是减少了该用户的权限表上的资金金额，群组账户不变。

管理员使用**自己账户**回收资金时，会导致群组账户的资金发送变化，同时该用户的权限表资金也会发生同步的变化。

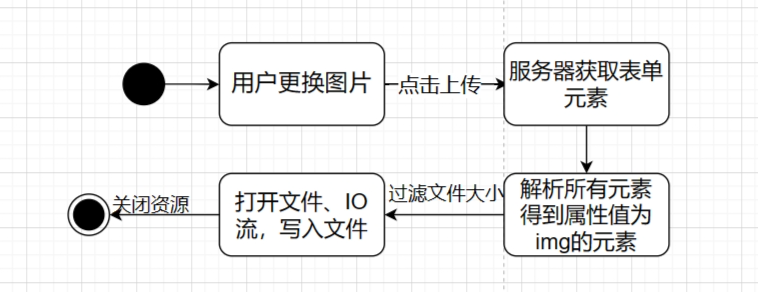
这时候，假设管理员故意给用户分配超额资金时（即分配给用户的资金大于群组账户的资金时），操作虽然能成功（类似现实生活中做假账，开空头支票的行为），但用户是能查看到加入的群组账户的资金总额的，也就是说用户能发现管理员分配超额资金的行为。

****

图表 10

**（4）用户注册的默认头像与头像的上传**

头像上传可以归类为文件的上传，只不过在文件的格式上固定为jpg。头像格式的限制在前端实现，固定表单的接收类型是image。除此还需要注意的是，要限制头像文件上传的大小，要存储每一个用户头像的存放地址，头像图片的名称用时间戳表示，避免出现图片地址名的重复。



图表 11

**（5）第三方接口的模拟**

首先，第三方接口平台有属于自己的数据库，该数据库用于暂存转账的资金。

第三方接口需要支付转账的有关信息，但不需要回调的URL地址，直接返回请求就行,因为此接口的服务对象仅为本平台，仅仅是简单的模拟。

接口返回给支付方的数据被封装为一个对象，并且在接口内部自动把这个对象response给收款方，收款方就可以得到相关的信息。需要注意的是，因为要记录相关的交易信息，以及同步我方平台与第三方接口的数据，必须得开启事务，发生意外时可及时回滚数据。

如果失败入账第三方，则回滚我方被冻结的资金，并于前端提示用户是否再次提交支付，如果入账成功，则删除被冻结资金，并于前端刷新页面，避免出现二次支付问题。

如果第三方失败入账收款方，记录收款方的异常交易行为，在调用接口，对收款方进行第二次入账。如果在现实生活中，收款方可以找到交易异常的地方，并向第三方发起补偿。在本项目中因为只是简单模拟，未实现补偿这一操作。

**（6）用户查询群组**

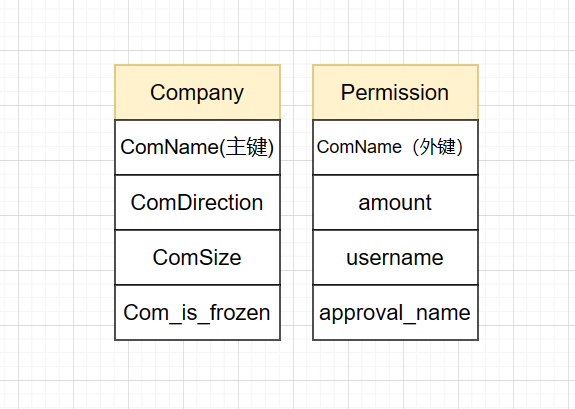
用户可以搜索群组相关信息，通过搜索群组名称的关键词，即可显示与关键词有关的群组，实现了模糊查询。实际上，服务器通过调用Dao层，Dao层的mysql语句LIKE关键字实现了模糊查询的效果。

**（7）数据的同步**

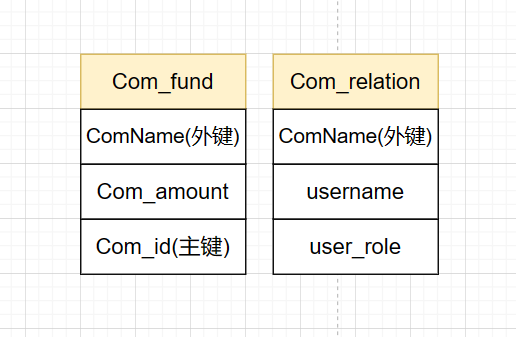
本项目的数据库表一共设计了11张表，其中有些表之间是有关联的，并有主外键之属，例如有企业信息表（记录企业相关信息），企业群组关系表（记录用户与企业的关系），企业资金表（企业的资金账户）、企业群组资源表（记录管理员给用户分配的资金） 其中企业信息表是主表，可以说其余表是从表关系。

用户在申请创建企业群组是，先写入申请表，一旦网站管理员批准申请，则服务器在Service层自动调用了多个写入操作，包括企业关系表、企业资金表。如果某一处写入出错了，则全部回滚，并用统一结果集返回前端相关信息。

管理员注销群组时，因为企业信息表的设计有主外键关联，必须先删除从表在删除主表，否则回滚数据。



图表 12



图表 13

# 5.技术搭建

#### 后端：

综合运用java语言，运用反射、泛型、集合、IO流、时间与日期、单元测试、正则表达式、加密安全、多线程、JDBC编程，servlet等知识，用tomcat搭建服务器。

#### 前端：

前端主要采用Vue框架搭建，以jQuery库为辅，因为用Vue能极大简化了对数据的DOM操作。

除此还深入学习了JavaScript语言，有部分页面用JavaScript语言实现DOM操作。

与后端的交互响应用Ajax和axios库实现。

# 6.程序测试

**一、测试目的：**

本测试旨在验证程序功能的完整性、性能稳定性以及用户体验的流畅性，确保程序能够成功运行

**二、后端测试：**

**1.Controller层**运用专业的测试工具postman测试提供给前端访问资源的接口，后台的测试通常需要数据库的数据，因此测试之前先模拟出数据库的数据，一般直接把数据写到缓存里边,不直接从数据库中CRUD,既保证了数据库里的数据不会被污染，也保证了实现的Dao层与controller层的分离。

在controller层测试中，实现头像上传这一个功能花费的时间是最多的，因为细分的化涉及文件上下传的知识，这方面的了解比较少，前期需要花费时间去重新学习，然后测试过程中也遇到了文件夹的存放路径的问题，像采用绝对路径还是相对路径，本次采用相对路径的做法。

**2.Dao层**测试Mysql语句的正确性，包括像count（\*）、关键字LIKE等含特殊关键词的语句，把数据库的数据能展现出来即可。

虽然Dao层只是简单的写数据库增删改查的语句，但是偶尔会犯一些低级的错误，如select语句、delete语句的混淆。

**3.Service层**就是最后一步的测试，通常我的习惯是部署服务器，打开前端页面进行调试，与controller层和Dao层一并测试。

在逻辑比较复杂的地方，例如支付过程中要对支付密码是否正确的、金额判断是否充足、资金的冻结、判断返回请求的状态等细节，要随时预判可能出错的地方，并写到日志文件上，尽可能写提示为什么会出现异常的错误，这些错误有可能是属于逻辑上的问题，也可能是属于程序固有的异常问题，尽可能分清楚，以便后续的测试。

**4.单元测试**

单元测试测试一些自己编写的工具类，如CRUDutils增删改查的封装，数据库连接池在多线程下能否被正确使用，是否有多线程安全，还有像响应体数据的封装类，数据是否能被JSON格式化，用户头像上传的方法、加密解密数据是否正确等等。

单元测试使测试程序的颗粒度更小更细，非常有助于测试某一组独立的逻辑。在测试过程中，需要注意单元测试的格式，也需要养成良好的测试习惯，好的测试习惯感觉像空气（AIR）一样并不存在，而且不能受到外界环境的影响。下次目标是在核心业务、核心模块的增量代码确保单元测试通过。

**三、前端测试：**

前端测试的方式较为单一，开发工具IDEA并没有能直接测试前端页面的功能，前端的测试一般借助浏览器的开发者工具上进行。在源代码界面上，对可能出现的错误的代码块打断点，一步步调试。

一开始使用的Vue通过CDN引入，不好找出bug而且有时因为网速不稳定的问题使页面加载Vue速度很慢，严重影响了调试的效率和用户的体验。后来使用Node.js来下载Vue.js到本地，很好地解决了这些问题。

**四、性能测试**

1.本项目的性能测试良好。无论在本地页面刷新了多少次，用edge和谷歌浏览器加上影子浏览器打开了三个界面，几乎同一时间进行多次刷新页面，页面也能很快地展现出数据。

原因分析：使用了自定义数据库连接池，极大提升了CRUD的速度。

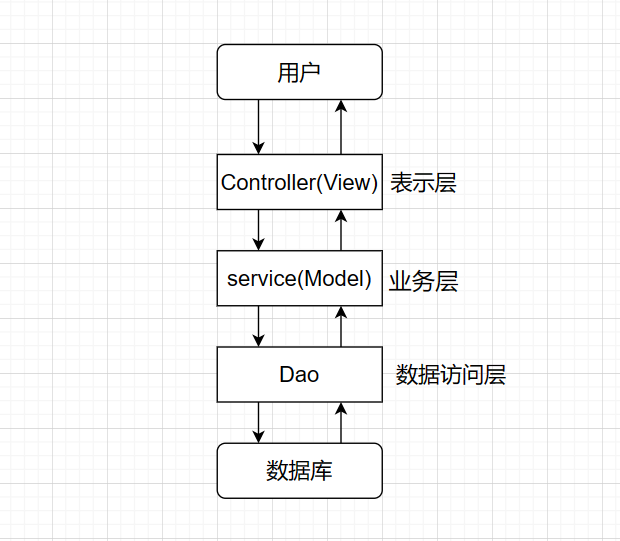
# 7.项目亮点

**1.采用MVC三层设计模式**

**Controller层**负责用于显示数据和接收用户输入的数据，为用户提供一种交互式操作的界面。主要作用是界面展示，接收请求，分发请求。

**Dao层主要是负责对数据库的访问,实现对数据的增删改查功能**。

**Service层**实现业务的主要逻辑。

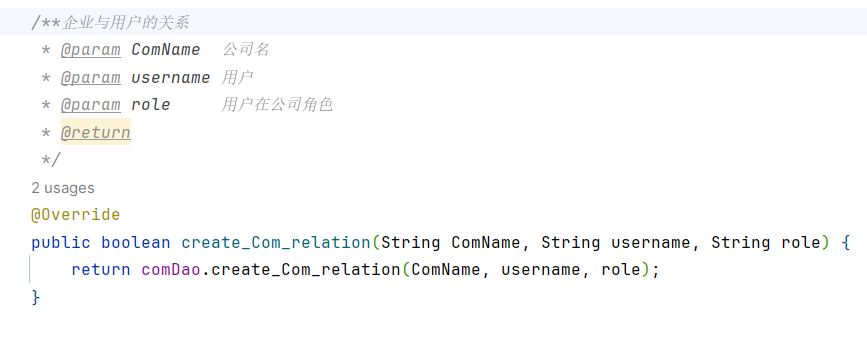


图表 14

**2.异常日志和文档注释**

使用日志框架slf4j的api，使用门面模式的日志框架，有利于维护类的日志处理方式，有意识地预判出错地地方并写上日志，分辨出是逻辑上的错误，还是程序上的出错。

使用文档注释，标注一些难以通过变量名理解的函数。



图表 15

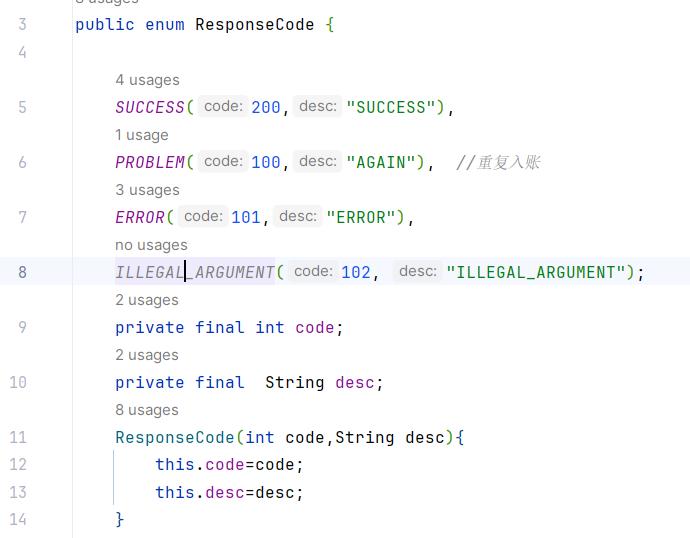
**3.命名规范**

对于常量的定义，统一使用静态变量，变量名直接用英文翻译。

**4.统一结果集**

封装了响应体数据，从后端返回前端的数据统一封装成一个含有状态码、消息内容、数据的泛型对象。

状态码是自定义的枚举类，枚举类能十分清晰地表示了哪些状态码代表什么信息。



图表 16

**5.自定义CRUD封装工具类**

该项目需要多次对数据库CRUD，如果不封装增删改查的操作的话会导致Dao层代码高度冗杂，代码重复性很高，也不利于后期的维护，因此结合小组的作业实现了CRUD封装的工具。



图表 17

**6.自定义数据库连接池**

多次的使用JDBC注册驱动创建数据库连接，得到数据后再销毁连接，这样重复性的创建和销毁连接会给服务器带来很大的性能开销，因此结合小组作业实现了自定义数据库连接池，极大的提高了CRUD的性能。

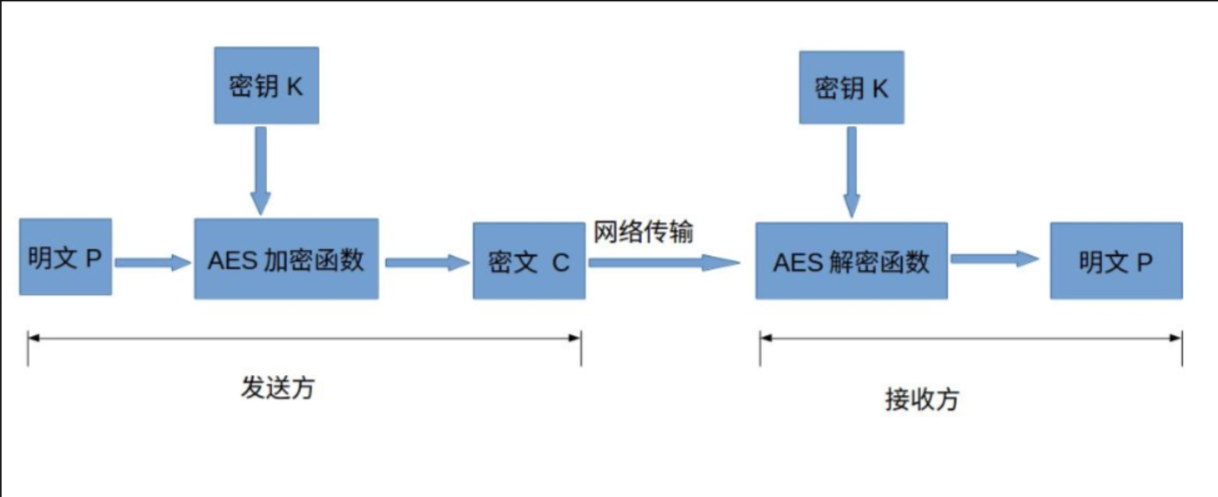
**（1）**.因为Servlet是单例多线程的，所有可访问的接口资源都继承了BaseServlet，因此在BaseServlet的init()方法初始化数据库连接池，使连接池部署在服务器全局范围内，在destory()方法销毁数据库连接池，释放资源。

**(2)** .数据库连接池分为空闲池和使用池，从连接池获取连接时会判断使用池里的连接是否超时，定义连接超过2秒则自动返回空闲连接池，初始化5个连接，最大可连接数10个，对获得释放连接加了锁，在多线程下使用连接池也是没有问题的。

**7.加密与安全**

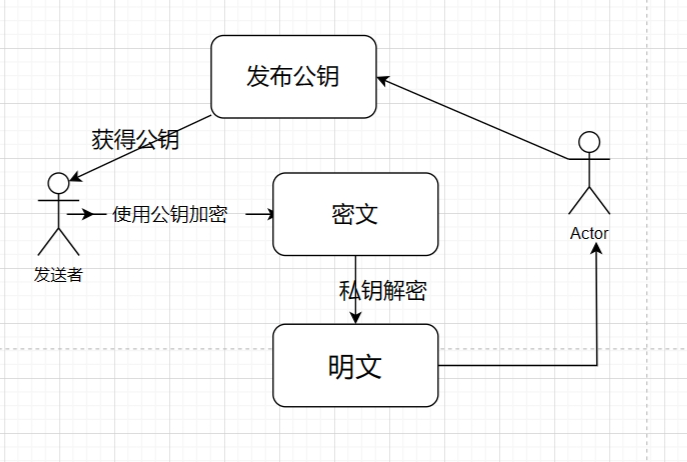
本项目涉及资金信息，并模拟借助第三方平台支付，因此涉及到交易等敏感信息的传输，需保证前端敏感信息的加密安全，本项目使用了java内置的security 库，使用AES+RSA组合实现加密。

AES加密是一种**对称加密算法**，对称加密算法也就是加密和解密用相同的密钥，这种加密方式加密速度非常快，适合经常发送数据的场合。缺点是密钥的传输比较麻烦。，AES具体的加密流程如下图：



图表 18

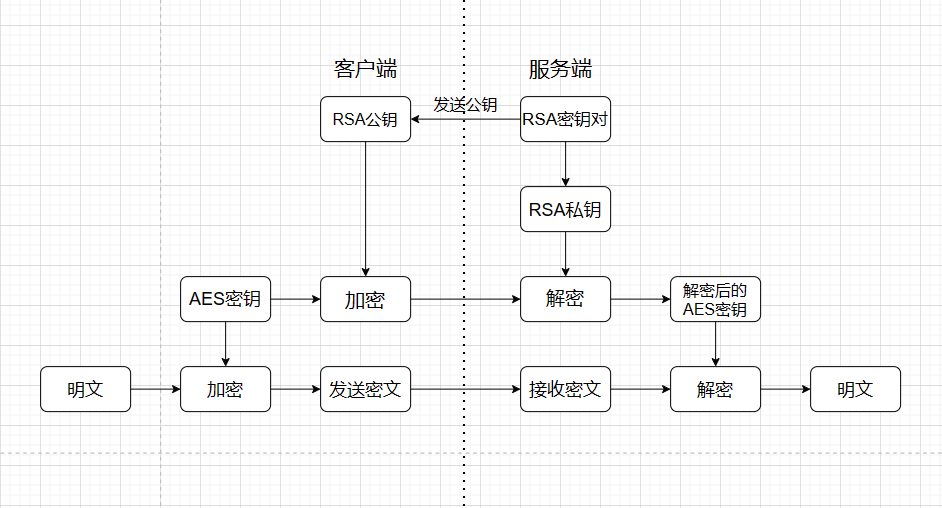
RSA加密是一种**非对称加密算法**，加密和解密用的密钥是不同的，这种加密方式是用数学上的难解问题构造的，通常加密解密的速度比较慢，适合偶尔发送数据的场合。优点是密钥传输方便。RSA加密流程图如下：



图表 19

本次项目结合AES和RSA加密的各自优点，使用AES密钥加密数据，RSA公钥加密AES密钥，接收者用RSA密钥解密AES密钥，后用AES密钥解密数据，

这样既很好的保护了数据，安全性拉满，也提高了在解密大数据方面的速度,优雅地结合了AES和RSA各自优点。以下就是RSA和AES组合的具体流程：



图表 20

**流程：**

1.服务方生成一对RSA密钥，自己保留私钥，将公钥通过response到互联网给到前端，

2.前端使用AES密钥对数据Data加密，生成密文EncryData;

3.使用得到的RSA公钥对AES密钥加密，生成EncryData

4. 将加密后的 AES 秘钥 EncryKey 和加密后的报文 EncryData 通过网络传输给服务器端；

5. 服务端用 RSA 私钥对 EncryKey （加密的 AES 秘钥）进行解密操作，得到 AesKey

6用 AesKey 解密传入过来的加密报文 EncryData，得到报文数据（明文） Data ，流程结束。

**8.防SQL注入**

Mysql的编译语句采用prepareStatement语句注入，尽量包含预编译符。

**9.权限表的设计**

因为群组管理员可以分配资金或回收资金给用户，但主体却并不在用户身上，故基于此设计了权限分配表 ，参考了经典的5张表，很好的实现了底层数据库的权限分离。

1用户表

2用户角色表

3资源表

4角色权限表

5角色表

**10.数据库敏感信息加密**

后台服务器对接收用户注册时的密码进行**加盐不可逆加密**，加盐是为了防止用户输入简单的密码，出现彩虹表碰撞的可能，算法采用的是哈希加密算法，虽然是不可逆加密，换句话说就是无法还原为用户输入的原密码。所以即使数据库的信息被盗窃了，那也无法得到用户的密码来登录盗窃用户的信息。

用户登录时，把密码进行同样的加盐加密，再与数据库里密码对应，如果一致，则通过。因为哈希碰撞是有概率发生的，但加了盐之后，发生的概率几乎为零。

**11.前后端正则表达式校验**

**（1）前端：**

用户在注册输入邮箱时，用正则表达式限制用户输入正确的邮箱格式，在输入密码时限制用户必须输入大于8位的密码。

用户在转账支付过程中前端校验金额小数位必须在两位以下。

**（2）后端：**

用户在转账支付时，校验是否有负号等恶意输入。

**12.避免连表查询**

本项目的表设计之初有主外键之分，但并未使用连表查询，在数据量多的情况下使用连表查询会降低查询的效率，这也是阿里巴巴规范手册里面禁止的，但带来的缺点就是表的数量增多。

**13.使用postman测试工具调试接口**

一开始是先做好资源接口后再把前端页面写上，才能测试得上资源接口，这种测试方式时间周期长，往往会出现很多问题，而且过于麻烦。后面就学习了postman测试工具，后端接口的测试就不需要等前端的页面的完成了，提高了测试的效率。

# 8项目难点

**1.**该系统涉及到**多用户多级权限**的身份验证，逻辑较为复杂，在设计之初数据库表的设计尤为关键，好的表的设计可以减轻CRUD的频繁操作，减少实体类的数量，更可以避免后期不得不删表，重新设计表的关系的可能。

**2.**不能使用mybaits,spring等框架,意味着要求开发者必须用原生代码去反射获得一些数据，增加了编写项目的难度，也不利于后期的维护，在两个星期就要实现交付的情况下给开发者带来一点小压力。

**3.涉及前后端的交互**，前端的数据显示必须与后台相关，数据以什么形式、什么的体量传输，这就要求前后端有一个良好的沟通、约定。

**4.**本项目涉及数据库的**CRUD功能较多**，如果只是多次建立连接、销毁链接，会导致服务器性能的下降，导致前端页面数据慢加载，给用户带来不愉快的体验。

**5.对于一名后台开发人员**，编写一个前端页面的比较困难的（这几个星期速通JavaScript），特别是前端页面的Dom操作,需要较为扎实基础的JavaScript知识。

**6.前后端的权限分离**。本项目涉及游客、普通用户、群组管理员、网站管理员四种身份的权限分配问题，如果前端页面的权限分配只是通过页面分流实现的话，无疑增加了前端编写的冗杂，因此使用Vue框架来解决这个问题。

**7.第三方接口的实现**。模拟第三方支付平台来进行交易过程，而非简单的在本地中对数据库进行数据的增减操作，因此要求开发者需要对接口理解到位，以及如何模拟使用第三方接口会出现的问题。

**8.加密与安全。**本项目涉及资金的交易，涉及到保护用户隐私安全的问题。在业界中，如微信支付、支付宝支付等都会采用加密手段来实现对交易数据的保护。而加密解密并不好的测试，特别是AES和RSA这套加密方案，涉及前后端加密传输。后端生成的RSA公钥Base64编码传给前端，前端的数据也要经过Base64编码，后端拿到数据后又要对数据Base64解码，期间过程的细节非常之多，加之所有的数据都是密文，但凡有一步出错了解码出来的数据也是乱码字符串，所以不利于调试，测试的所花费的时间非常多。

**解决：**

前后端打上断点，并把前后端各自的原密钥和密文数据记录下来，进行比对，检查前后端传输数据过程中是否有数据损坏，前后端密钥的填充方式是否一致。后端解码数据要注意字节与字符串的转换。

**9.本地缓存**。利用本地缓存存消息，或统计数据，以减轻数据库压力。业界中的本地缓存如Ehcache、Guava Cache或Caffeine 等都具有非常良好的存储性能。当时比较理想主义化，想模仿实现业界的本地缓存，达到这个本地缓存能自我优化，在一定时间内能够释放内存，或者缓存量到一定程度上会自我优化的功效。服务器接收到前端的数据，先把所有数据放到缓存里边，然后再对缓存里边的数据写入数据库。这样以来就可以避免在高并发下的CRUD访问数据库导致数据库挂掉。而难点就在于怎样把对应的数据写到对应的数据库表中，缓存如何自我进行优化。

当时的话有考虑过**多开一个线程**（类似监听器）去优化本地缓存，本地缓存是全局范围使用，又要考虑到多线程的问题。因为过于仓促，时间不足，所有最后并未实现本地缓存。

**10.数据的同步更新**。因为本项目数据库表的关联性较大，在写入数据库操作时往往要同步写入多条数据，特别是executeUpdate操作较多。在测试时发现一大问题：**表发生了锁死**。当时IDEA的报错是失去了与数据库的连接，也就是连接超时。

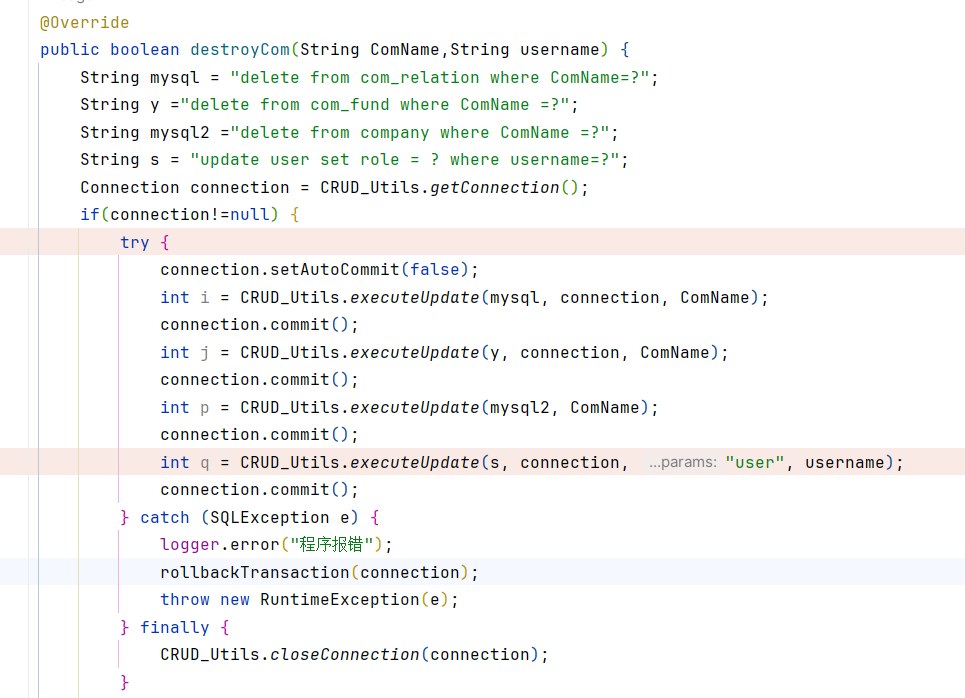
发生死锁的几种原因：

（1）**锁的顺序**：如果两个或多个事务以不同的顺序请求相同的资源锁，那么可能会出现循环等待条件，从而导致死锁。

**（2）索引的使用**：没有为ComName列创建索引，或者索引不是唯一的，那么数据库可能需要锁定更多的行或表来执行操作。这可能会增加死锁的风险。

**解决：**后来经过优化：尽量减少事务中的操作，并在每次查询后立即提交事务，减少锁定时间，使其变得更小且更快完成。我猜是并发操作导致的死锁，

加上第二点的情况导致了行锁升级为表锁，同时有主外键的约束，数据库会检查外键约束以确保数据的引用完整性。检查时锁定了其他表的资源。这才导致了死锁的发生。



图表 21

# 9.心得体会

从开始参加训练营，到最终的考核，不知不觉已经过去了一个多月，我从中学到了很多技术、东西，我受益匪浅。

**在编程能力方面**，我有了量变的提升，在为期四周的训练营中，我掌握了并实现了C语言中多种数据结构和排序算法，如队列、栈、链表、二叉树、并归排序等排序算法，提高了我对C指针这种底层概念的认识，使我更容易理解一些高层抽象的技术，同时我也跟紧小组训练营，学习了跟后台相关的技术，从对后台技术一无所知到现在能看到、看懂后台相关技术，深入学习后台开发基础知识、框架和工具，同时也进行了大量的实战手写demo，完成了四次小组作业对应的四次小demo，期间手写了java其中的一些封装好的库，使我对java语言的理解更上了一个台阶。

更重要的是，一个多月里边学边实践写代码，让我重新拾起了奋斗、继续学习新东西的激情。我可以说，在这短短的几周时间里，我在QG学到的东西学到的技术是我以前不敢想象的，加入QG训练营是我这学期以来做过最正确的一次决定，我有这么大的提升得益于QG，感谢QG工作室给与了我一个学习的契机，让我提升了技术能力，我也希望能留在QG继续学习下去。

但是，在最终的考核中，因为这是我第一次独立实现一个前后端交互的项目，在本次项目中，我对项目的搭建、对代码的调试明显缺少相关的经验，导致开发过程中遇到了较多的bug，花费了较多时间去debug找出问题，并修正出错的地方。边做项目的同时，边学习前端技术，像前端的JavaScript语言是弱相关语言，跟java强语言不一样，需要花点时间学习。虽然在调试测试项目花费了较多时间，但同时我也了解到并运用了相关的调试工具，运用postman测试工具来验证后台接口是否能正常响应数据，极大提高了后来的调试速度。

此外，除了后台，我还学习了前端Html语言、JavaScript语言，jQuery库，Vue框架来搭建我的项目，通过前端发送Ajax请求后台接口数据，现在的我，独立写写前端页面还是很轻松的（有点遗憾是Vue路由还未运用到）。

**在人际交流方面，**我很高兴认识到QG工作室的各位师兄，在学习之路上师兄给了我们许多学习的建议，师兄们也热情回答我们的问题，解决了我们许多疑问，师兄还主动分享一些学习还有其他（与技术无关）的建议和经验，例如如何权衡自学新技术与课内学习（像高数课）的时间，师兄们的经验也提供给我们一个参考和取舍。在一些难题上，我们需要经常互相询问讨论解决问题的方案，师兄们也会在群里与大家一起讨论，提供解决问题的思路。我体会到沟通于团队的重要性，只有相互理解、支持、分享才能更好的解决问题。