

****

**软件工程中级实践报告**

**学院名称 智能与计算学部**

**专 业 软件工程**

**学生姓名 梁益铭 郑志嘉 周圣喻**

**学号 3020244108 3020244109 3020244111**

**年 级 2020级**

**班 级 5班**

**时 间 2022年 11月 29日**

**目录**

[一． 项目需求 3](#_Toc32028)

[1.1. 虚拟钱包 3](#_Toc605)

[1.1.1. 简介 3](#_Toc4515)

[1.1.2. 目标和涉众分析 3](#_Toc21986)

[1.1.3. 用例分析 4](#_Toc1375)

[1.2. 积分系统 5](#_Toc19176)

[1.2.1. 简介 5](#_Toc25659)

[1.2.2. 目标和涉众分析 5](#_Toc26794)

[1.2.3. 用例分析 6](#_Toc32664)

[二． 项目设计 6](#_Toc19916)

[2.1. 项目结构 6](#_Toc27437)

[2.2. 功能模块设计 7](#_Toc30783)

[2.3. 接口设计 10](#_Toc27091)

[三． 项目测试 17](#_Toc12649)

[3.1 黑盒测试 17](#_Toc4513)

[3.2 集成测试 18](#_Toc26687)

[四． 面向对象设计原则 19](#_Toc27145)

[五． 开发过程 20](#_Toc24583)

[六． 小组分工 21](#_Toc11847)

[七． 项目中遇到的问题 21](#_Toc15933)

[八． 结果反思 22](#_Toc5206)

1. **项目需求**
   1. **虚拟钱包**
      1. **简介**
2. **背景**

在软件工程综合实践项目的基础上，基于“饿了么”项目三的前端（VUE）

和“饿了么”项目四的后端（Spring Boot）继续开发，完成新功能虚拟钱包模块

的添加。

1. **约束**

1) 要求使用充血模型实现 Service 层

2) 将虚拟钱包功能融合到已经实现的“饿了么”系统中

* + 1. **目标和涉众分析**

1. **目标**

在“饿了么”系统中添加虚拟钱包功能，钱包支持充值、提现、支付、查询余额、 查询交易流水这五个核心的功能。 顾客用户可以通过将银行卡中的钱存入虚拟钱包，或将虚拟钱包中的余额提现到银行卡中，并且可以使用虚拟钱包中的余额进行支付。其他比如冻结、透支、转赠等不常用的功能，暂不考虑。

1. **涉众分析**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **涉众** | **代表人物** | **待解决的问题/对系统的期待** |
| 1 | 顾客 | 张三 | 1. 能够利用虚拟钱包进行充值提现。 2. 能够查询虚拟钱包中的余额。 3. 能够通过查询虚拟钱包查询交易流水。 4. 能够通过虚拟钱包中的余额进行支付。 |
| 2 | 商家 | xx餐厅 | 1. 能够利用虚拟钱包进行收款、提现。 2. 能够查询虚拟钱包中的余额。 3. 能够通过查询虚拟钱包查询交易流水。 4. 能够通过虚拟钱包中的余额进行收款。 |

* + 1. **用例分析**

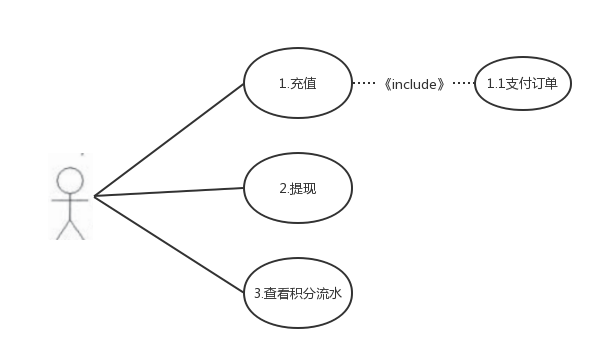
****

图1 虚拟钱包模例图

* 1. **积分系统**
     1. **简介**

1. **背景**

为提高用户粘性，电商平台一般都会有积分系统，积分系统可以让用户在消费时获取积分，并且可以用积分兑换礼品或抵扣金额。基于阶段三的前端（Vue）和项目四的后端（Springboot）的基础上继续开发，完成积分系统的开发。

* + 1. **目标和涉众分析**

1. **目标**

积分系统可以在用户购买订单成功时增加积分，并且累计的积分可以用来在以后的订单中抵扣金额或者在积分商城中兑换礼品，积分数为整数，消费时消费一元可以积一分，每次积分的有效时长为一个月，在积累了积分以后，以后的订单结算时可以选择用积分抵扣金额，100积分可以抵扣1元钱，并且在积分商城中可以用积分兑换礼品。

1. **涉众分析**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **涉众** | **代表人物** | **待解决的问题/对系统的期待** |
| 1 | 顾客 | 张三 | 1. 能够通过购买订单进行积分。 2. 能够利用积分抵扣订单金额。 3. 能够利用积分在积分商城兑换礼品。 4. 能够查询积分流水情况。 |

* + 1. **用例分析**

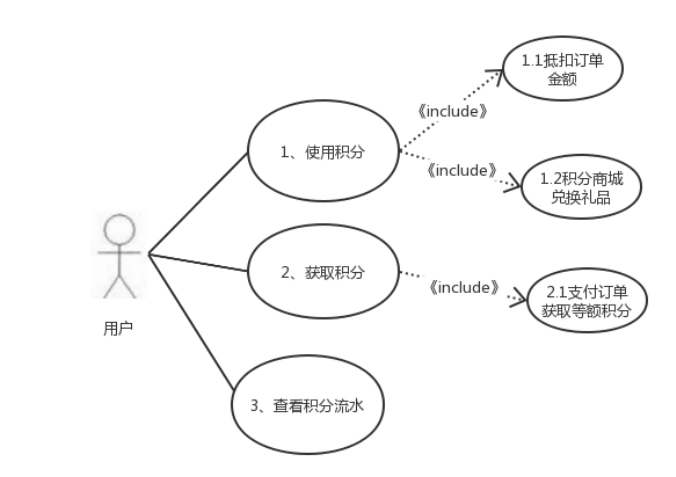
****

图2 积分系统用例图

1. **项目设计**
   1. **项目结构**

本项目主要结构由上到下分为Controller层、Service层、Mapper层三层架构：Controller层主要负责与客户端进行交互， 接收并响应前端请求，并将请求返回到Service层，Service层主要负责处理需要实现的业务逻辑，最后Service层调用Mapper层，Mapper层主要负责编写Mysql语句对数据库进行操作。

在Service层的实现上，本项目采用充血模型，包括 VirtualWalletService 类、VirtualWallet 类和 VirtualWalletTransaction 类，分别代表服务类、 钱包类和交易流水类。在充血模型中，Service 层包含 Service 类和 Domain 类两部分。Domain就相当于血模型中的 BO。不过，Domain 与 BO 的区别在于它是基于充血模型开发的， 既包含数据， 也包含业务逻辑。 在这种开发模式下， 我们把虚拟钱包 VirtualWallet 类设计成一个充血的 Domain 领域模型， 并且将原来在Service 类中的部分业务逻辑移动到 VirtualWallet 类中，让Service 类的实现依赖VirtualWallet 类。

* 1. **功能模块设计**

新增功能后整体项目的功能模块如下图所示。

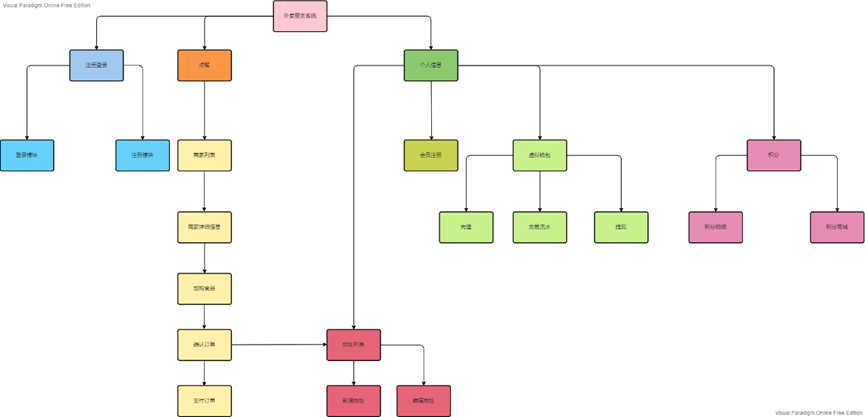


图3 外卖服务系统模块图

|  |  |
| --- | --- |
| **模块名** | **功能描述** |
| 首页 | 1. 点击点餐分类小图片，跳转到商家列表页面。 2. 点击下方菜单栏中的“订单”，跳转到历史订单页面。 3. 点击下方菜单栏中的“我的”，跳转到个人信息页面。 |
| 登录 | 1. 核实用户输入的账号、密码的正确性。 2. 点击“登陆”按钮，进行登陆业务处理。 如果登陆成功，跳转到上一个页面。 3. 点击“去注册”按钮，跳转到注册页面。 |
| 注册 | 1. 依次输入用户名、账号、密码、性别等信息，检查格式。 2. 点击“注册”按钮，进行新用户注册。注册成功后，跳转到登陆页面。 |
| 商家列表 | 1. .根据点餐分类显示商家列表信息。 2. 如果处于登录状态，那些需要查询购物车中是否有此商家的食品。如果有，在页面上显示食品数量。 3. 点击某个商家，跳转到此商家的详细信息页面。 |
| 商家详细信息 | 1. 显示商家详细信息及所属食品信息。 2. 在商家详细信息中，点击某食品后的+按钮，实现食品数量加1。 3. 点击某食品后的-按钮，实现食品数量减1。 4. 点击“去结算”按钮，跳转到确认订单页面。 5. 自动计算购物车总价。 6. 如果没有登录，那么上述三个动作会自动跳转到登录页面。 |
| 确认订单 | 1. 点击送货地址，跳转到送货地址列表页面。 2. 显示订单明细、总价。 3. 点击“去支付”按钮，跳转到支付页面。 |
| 历史订单 | 1. 显示历史订单信息及订单明细信息。 2. 点击某个历史订单，可以对订单明细信息进行显示和隐藏。 |
| 送货地址列表 | 1. 显示当前用户的送货地址信息。 2. 点击某个送货地址，跳转回确认订单页面。 3. 点击“新增”按钮，跳转到新增送货地址页面。 4. 点击“编辑”按钮，跳转到送货地址编辑页面。 5. 点击“删除”按钮，删除此送货地址。 |
| 新增送货地址 | 1. 添加新的送货地址，点击“保存”按钮，添加新的送货地址，并跳转到送货地址列表页面。 |
| 编辑送货地址 | 1. 编辑送货地址，点击“更新”按钮，保存编辑后的送货地址，并跳转到送货地址列表页面。 |
| 个人信息 | 1. 显示个人信息。 2. 修改个人信息。 3. 点击“我的地址”按钮，跳转到送货地址列表页面。 4. 点击“注册会员”按钮，跳转到会员注册页面。 5. 点击“我的钱包”按钮，跳转到虚拟钱包页面。 6. 点击“我的积分”按钮，跳转到用户积分页面。 |
| 注册会员 | 1. 点击“白银”、“黄金”、“钻石”按钮选择会员等级。 2. 点击“去支付”按钮进行支付。 |
| 虚拟钱包 | 1. 显示虚拟钱包余额。 2. 点击“充值”按钮进行钱包充值。 3. 点击“提现”按钮进行钱包提现。 4. 点击“交易流水”按钮，查询用户交易流水。 |
| 用户积分 | 1. 显示用户现有积分。 2. 点击“积分商城”按钮，跳转到积分商城页面兑换商品。 3. 点击“积分明细”按钮，查询积分变化明细。 |

* 1. **接口设计**

Controller层：

1. VirtualWallet

1.1 WalletController/recharge

参数：walletId、amount

返回值：int（影响的行数）

功能：用对虚拟钱包进行充值

1.2 WalletController/withdrawal

参数：walletId、amount

返回值：int（影响的行数）

功能：用对虚拟钱包进行提现

1.3 WalletController/getBalanceById

参数：walletId

返回值：double

功能：查询虚拟钱包余额

1.4 WalletController/listCheckFlowByUserId

参数：userId

返回值：VirtualWalletTransaction数组

功能：查询用户交易流水

2.User

2.1 UserController/getWalletIdByUserId

参数：userId

返回值：int

功能：通过用户id查询对应的虚拟钱包id

3.Integral

3.1 IntegralController/getPointsById

参数：user

返回值：是否读取成功

功能：根据用户查询其积分值

3.2 IntegralController/payPoints

参数：userId，point

返回值：是否支付成功

功能：使用积分在商城兑换礼品

3.3 IntegralController/listPointFlowByUserId

参数：userId

返回值：积分流水列表

功能：根据用户Id查询其所有的积分流水

4.Orders

4.1 OrdersController/usePointPayOrders

参数：orderId

返回值：是否支付成功

功能：使用积分抵扣订单金额

Service层：

1.VirtualWallet

1.1 VirtualWalletService/recharge

参数：walletId、amount

返回值：int（影响的行数）

功能：用对虚拟钱包进行充值

1.2 VirtualWalletService/withdrawal

参数：walletId、amount

返回值：int（影响的行数）

功能：用对虚拟钱包进行提现

1.3 VirtualWalletService/getBalanceById

参数：walletId

返回值：double

功能：查询虚拟钱包余额

1.4 VirtualWalletService/transfer

参数：fromWalletId、toWalletId、amount

返回值：void

功能：用户和商家之间进行转账

1.5 VirtualWalletService/listCheckFlowByUserId

参数：userId

返回值：VirtualWalletTransaction数组

功能：查询用户交易流水

2.User

2.1 UserService/getWalletIdByUserId

参数：userId

返回值：int

功能：通过用户id查询对应的虚拟钱包id

3.Orders

3.1 OrdersService/payOrders

参数：orderId

返回值：int（影响的行数）

功能：利用虚拟钱包中的余额支付订单，并记录交易明细

3.2 OrdersService/usePointPayOrders

参数：orderId

返回值：是否支付成功

功能：使用积分抵扣订单金额

4.Integral

4.1 IntegralService/getPointsById

参数：user

返回值：是否读取成功

功能：根据用户查询其积分值

4.2 IntegralService/payPoints

参数：userId，point

返回值：是否支付成功

功能：使用积分在商城兑换礼品

4.3 IntegralService/listPointFlowByUserId

参数：userId

返回值：积分流水列表

功能：根据用户Id查询其所有的积分流水

Mapper层：

1.VirtualWallet

1.1 VirtualWalletMapper/updateBalance

参数：walletId、newBalance

返回值：int（影响的行数）

功能：更新虚拟钱包余额

1.2 VirtualWalletMapper/getBalanceById

参数：walletId

返回值：double

功能：查询虚拟钱包余额

1.3 VirtualWalletMapper/getUserIdByWalletId

参数：walletId

返回值：String

功能：利用用户id查询虚拟钱包id

2.VirtualWalletTransaction

2.1 VirtualWalletTransactionMapper/saveCheckFlow

参数：virtualWalletTransaction

返回值：int（影响的行数）

功能：存储交易流水

2.2 VirtualWalletTransactionMapper/listCheckFlowByUserId

参数：userId

返回值：VirtualWalletTransaction数组

功能：查询交易流水

3.User

3.1 UserMapper/getWalletIdByUserId

参数：userId

返回值：int

功能：通过用户id查询对应的虚拟钱包id

4.Business

4.1 BusinessMapper/getWalletIdByBusinessId

参数：businessId

返回值：int

功能：通过商家id查询对应的虚拟钱包id

5.Integral

5.1 IntegralMapper/getPointsById

参数：user

返回值：是否读取成功

功能：根据用户查询其积分值

5.2 IntegralMapper/payPoints

参数：userId，point

返回值：是否支付成功

功能：使用积分在商城兑换礼品

5.3 IntegralMapper/listPointFlowByUserId

参数：userId

返回值：积分流水列表

功能：根据用户Id查询其所有的积分流水

6.Orders

6.1 OrdersMapper/usePointPayOrders

参数：orderId

返回值：是否支付成功

功能：使用积分抵扣订单金额

* 1. **业务对象设计**

对于贫血模型来说，业务对象包含数据，不包含业务逻辑；对于充血模型来说，业务对象将数据和对应的业务逻辑都封装起来。 本项目在虚拟钱包功能的实现上，对于业务对象的设计采用充血模型。

具体实现把虚拟钱包 VirtualWallet 类设计成一个充血的 Domain 领域模型，并且将原来在 Service 类中的获取余额、 提现、 充值逻辑移动到 VirtualWallet 类中。 这样的好处是保持了对象的封装性，在需要完善添加虚拟钱包的功能时，

系统需要支持更复杂的业务逻辑，利用充血模型会更加方便简洁。

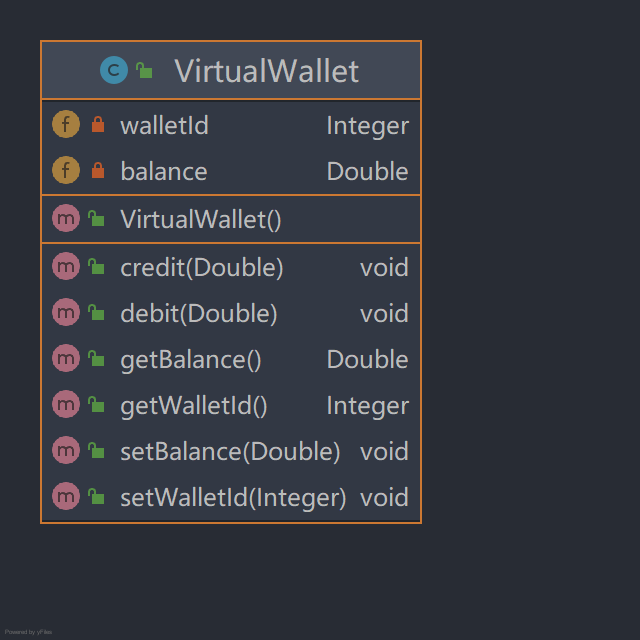


图 4 业务对象图

* 1. **数据库设计**

本项目使用MySQL数据库，选择DataGrip作为开发工具。 在数据表的设计上，需要新增虚拟钱包表、交易流水表和积分表。

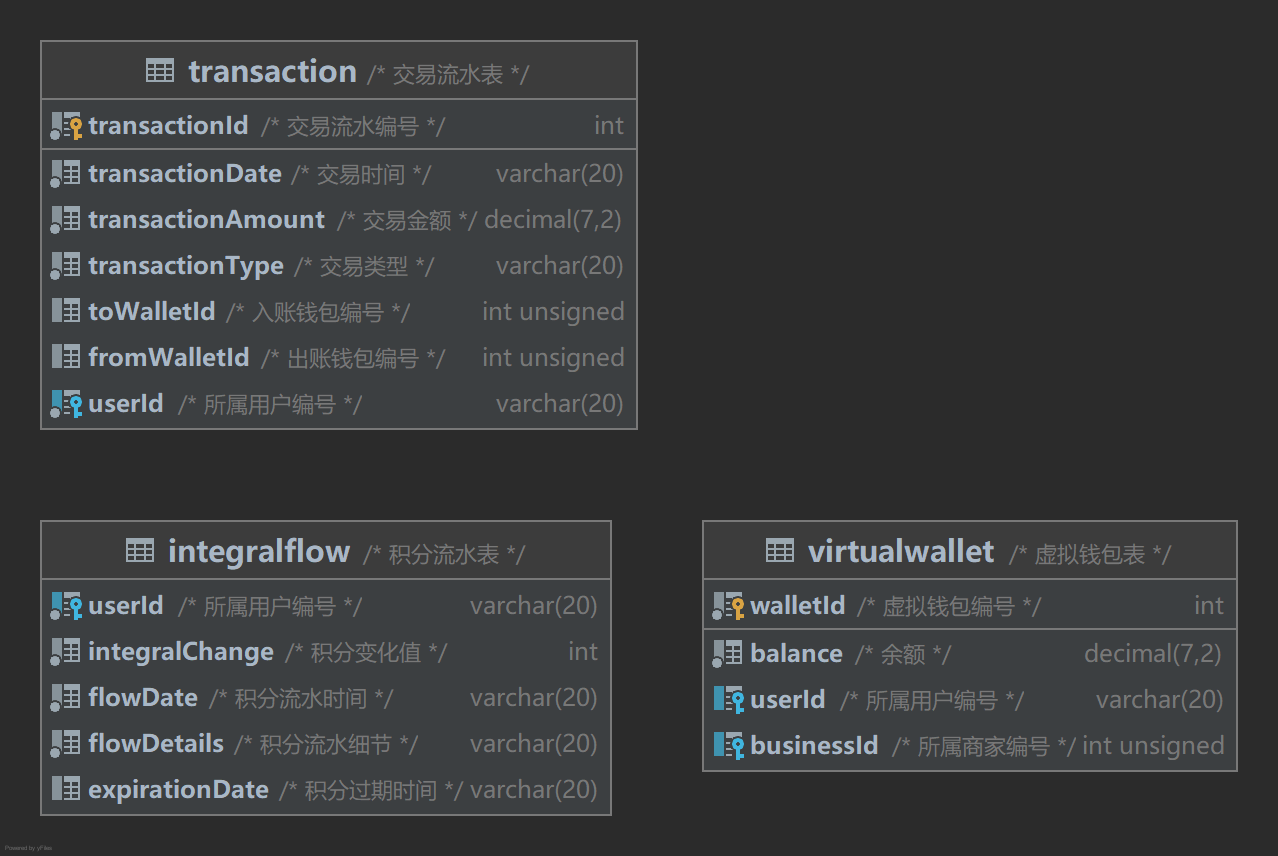


图 5 新增数据表图

* 1. **实现效果**

虚拟钱包和积分系统功能实现效果如下图所示。

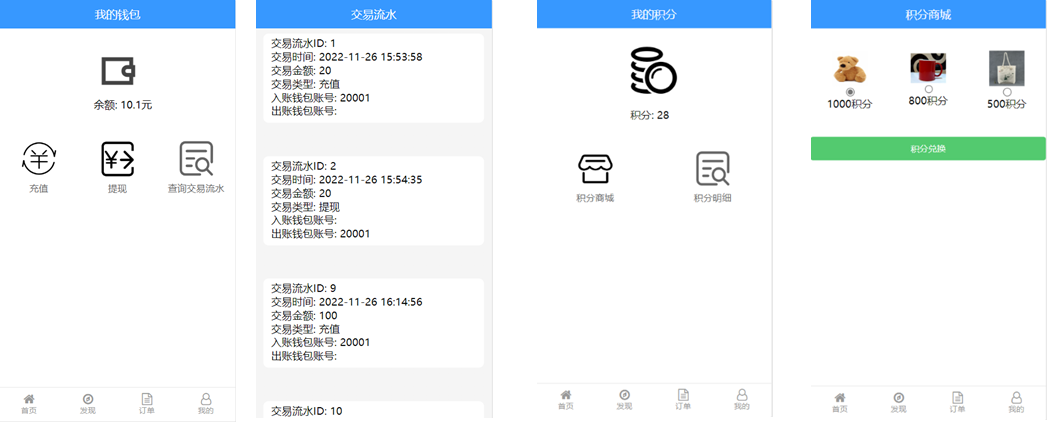


图6 实现效果图

1. **项目测试**
   1. **黑盒测试**

在黑盒测试中主要关注被测软件的功能实现，而不是内部逻辑。在黑盒测试中，被测对象的内部结构，运作情况对测试人员是不可见的，测试人员对被测产品的验证主要是根据其规格，验证其与规格的一致性。在绝大多数没有用户参与的黑盒测试中，最常见的测试有：功能性测试、容量测试、安全性测试、负载测试、恢复性测试、标杆测试、稳定性测试、可靠性测试等。

测试支付模块：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 钱包余额 | 订单金额 | 支付状态 | 订单状态 | 结果 |
| 1 | 80.00 | 27.00 | 已支付 | 已生成 | 订购成功 |
| 2 | 20.00 | 27.00 | 支付失败 | 未生成 | 余额不足 |
| 3 | 80.00 | 0.00 | 未支付 | 未生成 | 订单生成错误 |

测试积分模块：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 积分数量 | 需扣积分 | 积分状态 | 订单状态 | 结果 |
| 1 | 800 | 500 | 已扣除 | 已生成 | 积分兑换成功 |
| 2 | 200 | 500 | 扣除失败 | 已生成 | 积分不足 |
| 3 | 200 | 0 | 未扣除 | 未生成 | 未选择积分抵扣 |

* 1. **集成测试**

在前后端集成之后，再进行软件流程测试。

积分系统页面功能：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试步骤 | 场景 | 测试步骤 | 预期结果 | 实际结果 |
| 1 | 进入积分商城 | 点击“积分商城”按钮 | 跳转到积分商城 | 与预期结果一致 |
| 2 | 查看积分明细 | 点击“积分明细”按钮 | 跳转到积分明细查看页面 | 与预期结果一致 |
| 3 | 积分兑换礼品 | 在积分商城中点击对应物品并点“兑换”按钮 | 积分兑换成功 | 与预期结果一致 |

虚拟钱包页面功能：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试步骤 | 场景 | 测试步骤 | 预期结果 | 实际结果 |
| 1 | 进行充值 | 点击“充值”按钮 | 跳转到充值页面 | 与预期结果一致 |
| 2 | 钱包提现 | 点击“提现”按钮 | 跳转到提现页面 | 与预期结果一致 |
| 3 | 查看交易明细 | 点击“交易流水”按钮 | 跳转到交易流水查看页面 | 与预期结果一致 |

1. **面向对象设计原则**
2. 开闭原则

在新功能的实现中应用了开闭原则，例如在Service层中创建了VirtualWalletService接口，用具体的实现类进行实现。当在添加有特殊功能的虚拟钱包时，只需要实现VirtualWalletService接口，对扩展开放；也不需要修改原有方法，对修改关闭。

1. 面向接口而非实现编程原则

在Service层的设计中，首先设计Service接口，用ServiceImpl去实现接口进行编程。

1. 多用组合加委托少用继承原则

在项目设计中，通过Autowired自动注入将Mapper层对象组合进来，之后用委托的方式将方法实现。同样Controller层也是用这种方式来设计，避免了使用继承。

1. 单一职责原则

例如在虚拟钱包功能的设计中，设计VirtualWallet和VirtualWalletTransaction两个部分，一个负责虚拟钱包的余额查询、提现和充值，一个负责虚拟钱包交易流水的存储和查询，实现了单一职责原则，降低了类的复杂度、提高类的可读性，提高系统的可维护性、降低变更引起的风险。

1. 高内聚低耦合原则

以虚拟钱包功能的实现为例，虚拟钱包每个部分VirtualWallet和VirtualWalletTransaction都只完成一类事，体现了高内聚；在各层的实现类中，并没有出现一个类内部一个成员方法调用另一个成员方法的情况，体现了低耦合。

1. 接口隔离原则

以虚拟钱包的接口为例，Controller层接口的设计只向外暴露出了前端用的到的方法，没有将不使用的方法塞到接口中。

1. 依赖倒转原则

在新功能的实现中，Controller层作为高层模块需要依赖低层模块Service层，而本项目将Service层增加一个抽象接口层，让Service中的类去实现这些接口，这样Controller层就不直接依赖Service层，而是依赖抽象接口层，抽象接口也不依赖Service层的实现细节，而是Service层实现接口。

1. KISS原则

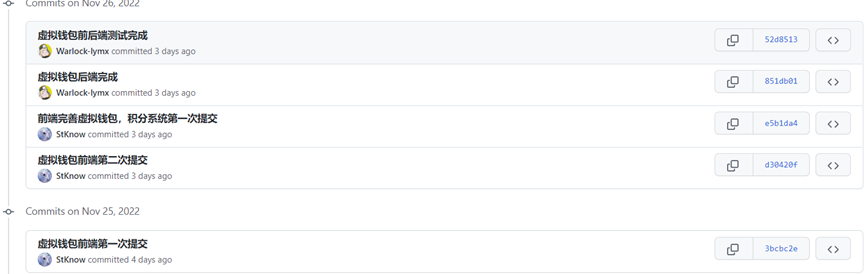
在新功能的实现中，尽量使用简洁的代码编写方式，代码规范、可读性强。

1. dry原则

在新功能的实现中，并没有重复编写逻辑重复、功能语义重复的代码，在遇到需重复使用的功能时都是通过封装为一个方法来实现的。

1. **开发过程**

小组使用github进行远程仓库管理，再结合git实现项目的合作编写，每个人完成自己的部分后，首先将远程仓库的代码拉取到本地仓库，解决冲突后再将修改后的代码推送到github上，完成一整个阶段后再统一做集成测试。仓库分为两个分支，main分支存储的是课程项目的部分，advance分支存储的是新功能开放的过程，整个项目小组总共提交了上百次，下面是github上的一些统计数据。



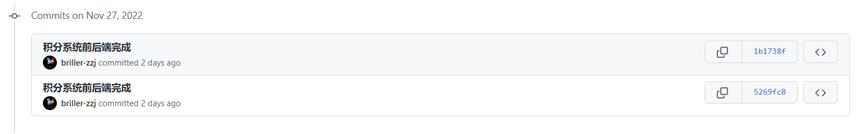


图7 新功能提交记录图

1. **小组分工**

**郑志嘉**

主要负责积分系统模块的后端代码实现以及数据库的建立管理和测试环节，并与前端工程师一同进行集成测试。

**梁益铭**

主要负责虚拟钱包模块的后端代码实现以及数据库的建立管理和测试环节，并与前端工程师一同进行集成测试。

**周圣喻**

主要负责虚拟钱包模块和积分系统模块的前端代码实现，并与后端工程师一同进行集成测试。

1. **项目中遇到的问题**

问题：积分系统的后端实现中，并不知道如何处理积分过期这一机制。

解决方案：积分的有效期为一个月，经过思考我决定在积分流水中记录获得积分的日期和过期的日期。每次在获取积分数的时候，提取该用户近一个月的流水，计算正负值即为当前拥有的有效积分数。

问题：前端查询交易流水时出错，无法展示后端传到前端的数据。

解决方法：与后端人员核对，发现前端与后端对同一元素的命名不一样。前端修改名称解决了此问题。

问题：在测试我的钱包页面时，发现钱包余额无法正常显示，经检查发现时前端created时walletId出错。

解决方法：经过调试，发现在created时，请求并不是串行的。这就导致第一个请求获得了walletId，但是在第二个请求获取余额时并没有更新walletId，从而导致余额出错。将获得walletId的请求写在前一个页面，在从前一个页面跳转到我的钱包页面时发送walletId，这样就可以使我的钱包页面能直接获取到walletId，余额可以正常显示。

1. **结果反思**

**郑志嘉**

本次项目中我负责的是积分系统模块的后端代码以及数据库代码的编写，并负责相关内容的测试和调试。这再一次加深了我对面向对象设计的认识，我对前后端分离式开发也有了比较全面的认识。在积分系统实现的过程中，遇到了一些困难，这对我的算法编写能力也是一个较大的考验，通过思考和查询资料解决这些问题对我的java编写能力也有所提升。此外，在Springboot框架的开发上也有更深刻的认识。总而言之，这是一次很有意义的实践经历。

**周圣喻**

在本次实践中，我主要负责的是前端部分的编写。由于之前进行过软件工程综合实践项目，我拥有了基础的前端代码编写能力。我在实践过程中遇到不会的地方就上网查阅，边学边做使我收获巨大，现在已经基本可以独立编写一个网页的前端部分。除此之外，我们团队在之前的基础上进一步合作。在团队一起完成项目时，小组成员之间相互讨论，相互进步，我也更加体会到团队合作的重要性。总的来说，我在本次实践中收获巨大，进步许多，代码能力和团队合作能力有了显著提升。

**梁益铭**

在本次项目中，我主要负责虚拟钱包功能模块的开发和文档的撰写。在之前的实践中我初步掌握了使用SpringBoot框架进行后端开发，在本次实践中，进一步熟练了框架的使用。更重要的是，本次开发中我更侧重于面向对象设计原则的运用，在代码的编写时更加注重效率和性能，很大程度地提高了开发能力。在文档撰写方面，我也学会写出更加专业地项目文档，为今后的学习工作夯实了基础。