1.项目需求

1.1简介

1.1.1背景

在已经完成的"饿了么"项目中，增加两个新功能模块，即虚拟钱包模块和积分系统模块，新增的两个功能模块与原功能模块融合为一个新的系统。

1.1.2约束

- 无需改造除订单组件外其它的功能组件
- 不需要实现第三方银行系统，充值和提现模拟实现即可

1.1.3参考资料

- 老师给的大作业任务书
- (182条消息) 设计模式之美-11 | 实战一（下）：如何利用基于充血模型的DDD开发一个虚拟钱包系统？_如我般骄傲的博客-CSDN博客
- (182条消息) 设计模式：积分兑换系统的设计与实现*OceanStar的学习笔记的博客-CSDN博客*积分模型设计

1.2目标、涉众分析和范围

1.2.1目标

- 方便客户购物
- 实现钱包支持充值、提现、支付、查询余额、查询交易流水这五个核心的功能
- 增加客户粘性
- 实现积分获取、消费、查询三个核心功能

1.2.2涉众分析

序号	涉众	代表人物	待解决的问题
1	商家	玩家饺子老板	商家的虚拟钱包能够收到客户的转账
2	客户	张帅	能够使用虚拟钱包向商家转账，可使用积分系统获得积分以及享受折扣
…			

1.2.3范围

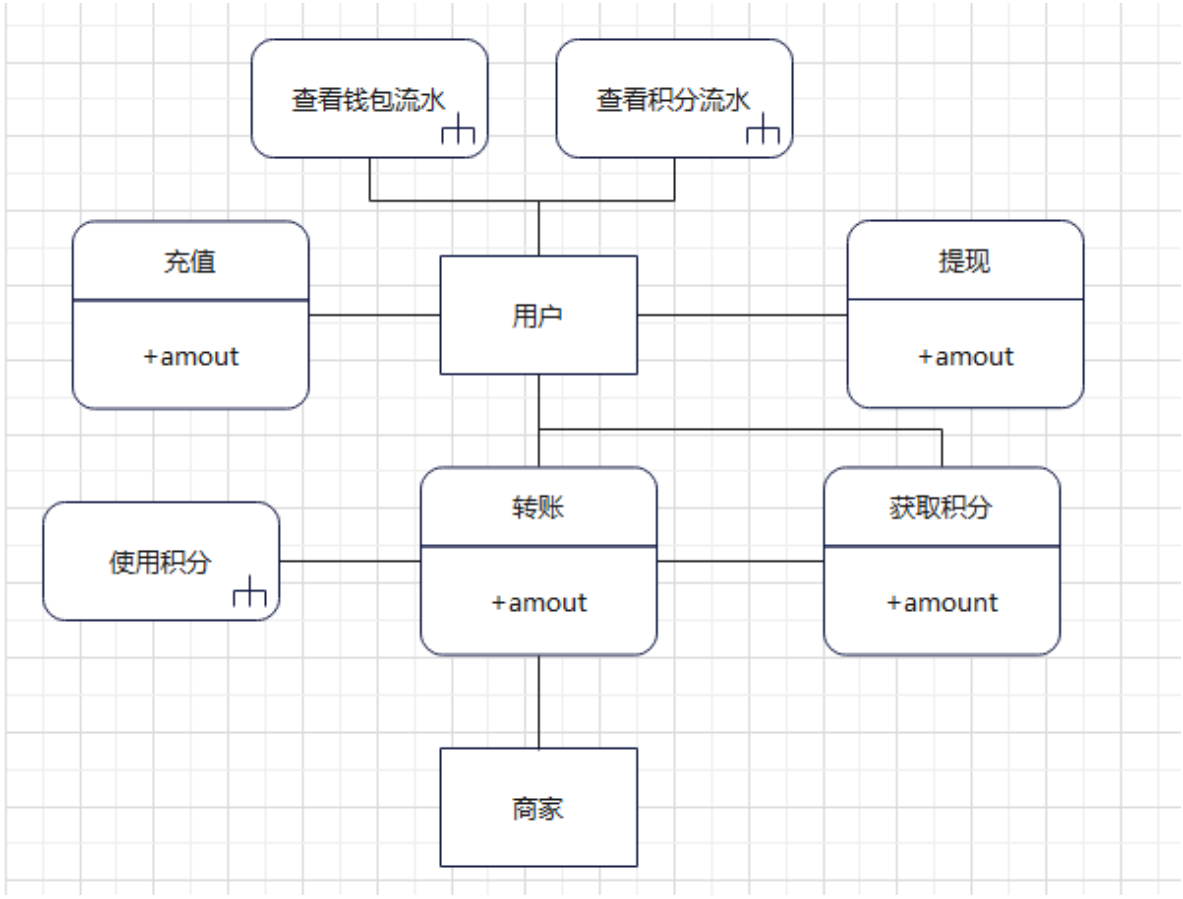
本系统需要与银行系统对接，不能直接处理现金。

1.3业务概念分析

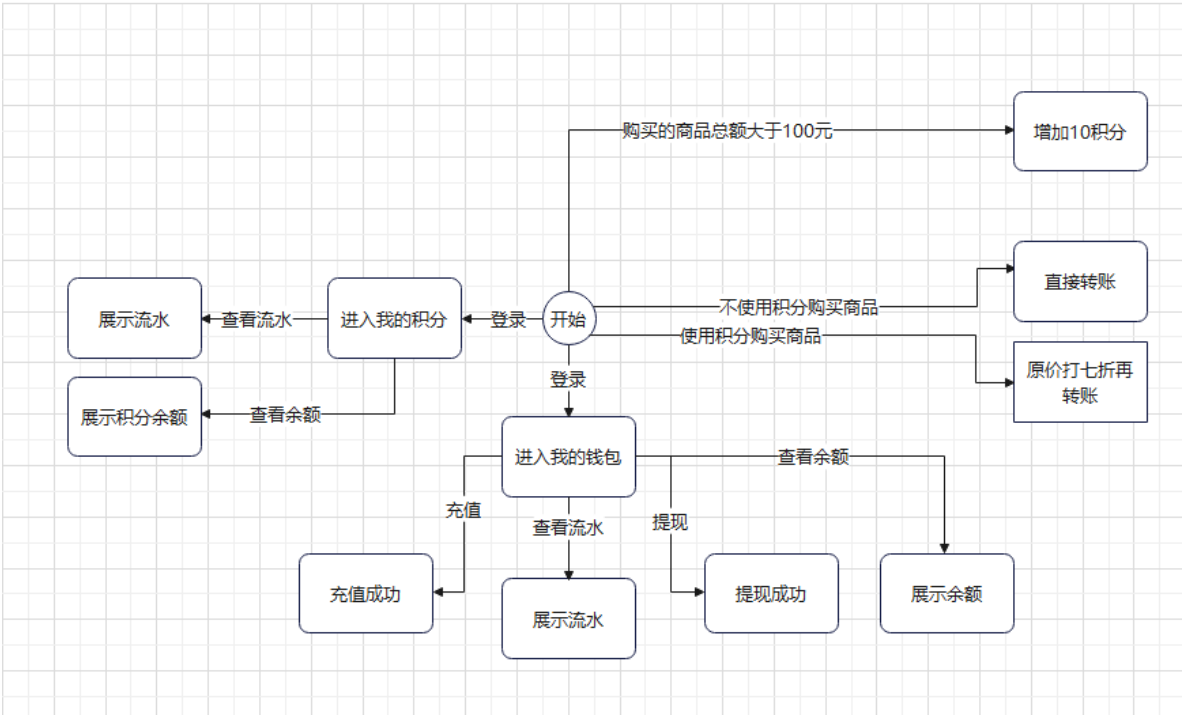
1.3.1概述

本系统主要管理的业务有：虚拟钱包的交易查询，积分系统的获得使用以及查询。

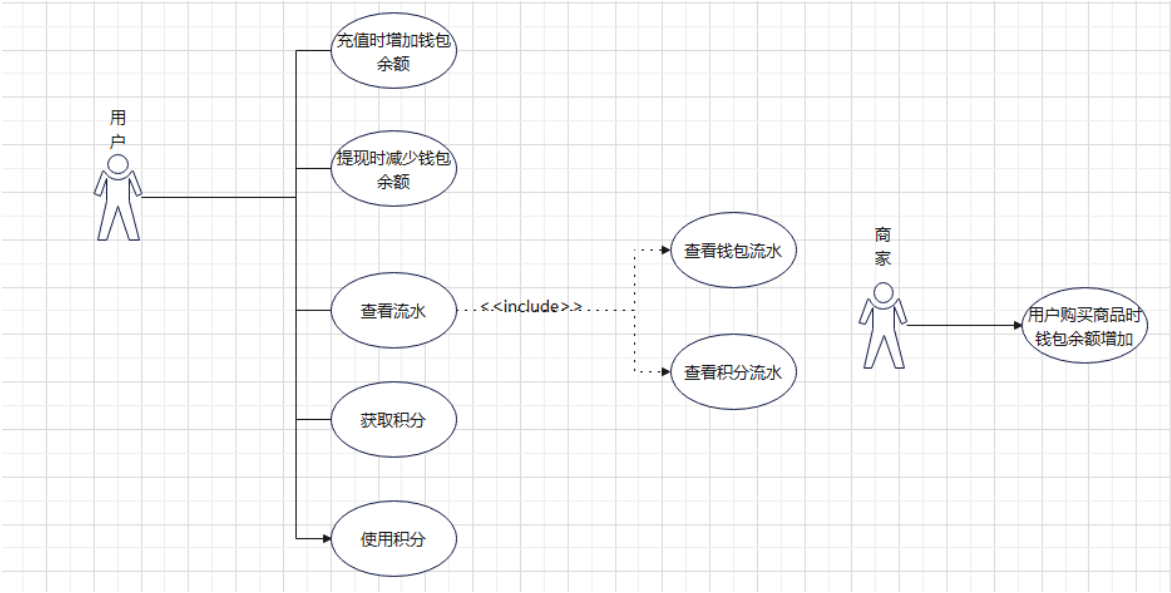
1.3.2业务概念一览



1.4业务流程分析



1.5功能性需求



2.项目设计

2.1软件架构与分层设计说明

使用MVC架构。

在po层、controller层、service层和mapper层添加虚拟钱包系统（VirtualWallet）和积分系统(Bonuspoints)的实现文件。其中，入账功能和出账功能（包括虚拟钱包的出入帐和积分系统的增加、使用积分）在虚拟钱包和积分系统的po层实现（充血模型）。而储存两个系统的流水的逻辑都在它们的service层实现：当账户余额被改变时，service层调用po层的getter和setter方法设置流水的各项数值并通过调用mapper层操作储存进数据库。

2.2接口说明

- /VirtualWalletController/getWalletbyuserId:通过用户Id获取虚拟钱包实例（用户id即为钱包id）
- /VirtualWalletController/debit：出账
- /VirtualWalletController/credit：入账
- /VirtualWalletController/transfer：转账
- /VirtualWalletController/listtransactionbyid：根据用户id列出钱包流水

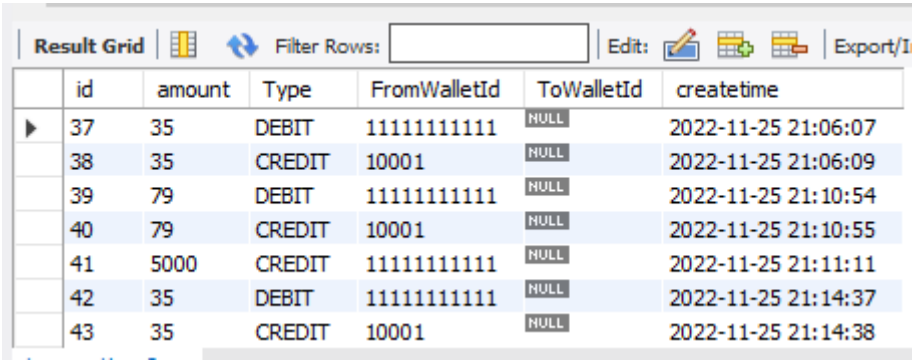
- .../BonuspointsController/getBonuspointsbyuserId：通过用户Id获取积分账户实例
- .../BonuspointsController/debit：减少（使用）积分
- .../BonuspointsController/credit：增加（获得）积分
- .../BonuspointsController/listtransactionbyid：根据用户id列出积分流水

2.3数据库设计

虚拟钱包账户（积分账户类似）：

	userid	balance
▶	11111111111	530.00
*	NULL	NULL

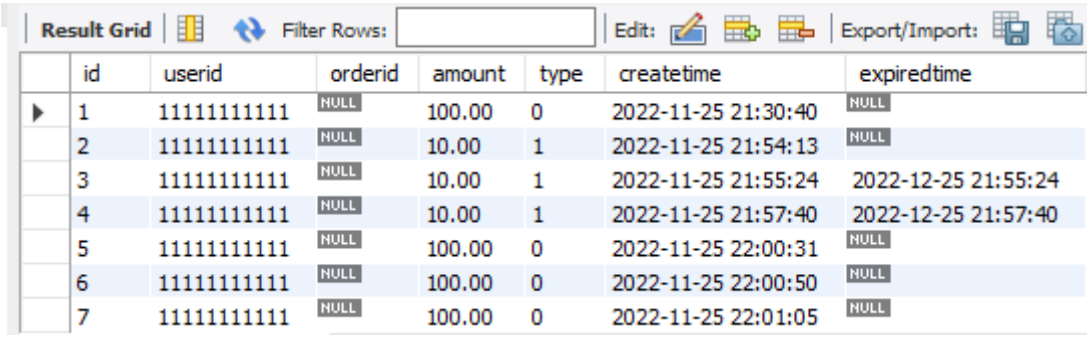
虚拟钱包流水：



Result Grid | Filter Rows: | Edit: | Export/I

id	amount	Type	FromWalletId	ToWalletId	createtime
37	35	DEBIT	11111111111	NULL	2022-11-25 21:06:07
38	35	CREDIT	10001	NULL	2022-11-25 21:06:09
39	79	DEBIT	11111111111	NULL	2022-11-25 21:10:54
40	79	CREDIT	10001	NULL	2022-11-25 21:10:55
41	5000	CREDIT	11111111111	NULL	2022-11-25 21:11:11
42	35	DEBIT	11111111111	NULL	2022-11-25 21:14:37
43	35	CREDIT	10001	NULL	2022-11-25 21:14:38

积分账户流水：

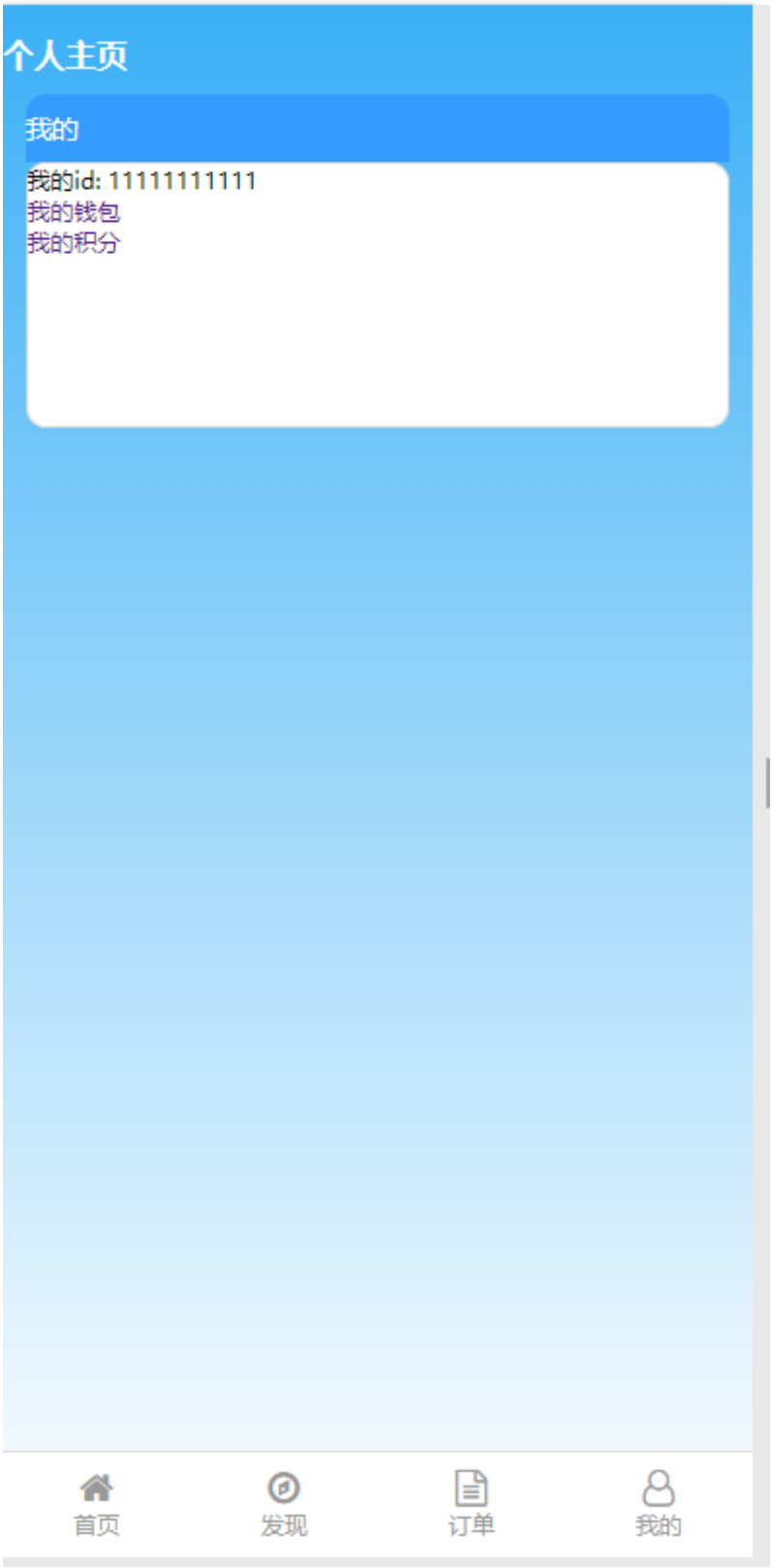


Result Grid | Filter Rows: | Edit: | Export/Import:

id	userid	orderid	amount	type	createtime	expiredtime
1	11111111111	NULL	100.00	0	2022-11-25 21:30:40	NULL
2	11111111111	NULL	10.00	1	2022-11-25 21:54:13	NULL
3	11111111111	NULL	10.00	1	2022-11-25 21:55:24	2022-12-25 21:55:24
4	11111111111	NULL	10.00	1	2022-11-25 21:57:40	2022-12-25 21:57:40
5	11111111111	NULL	100.00	0	2022-11-25 22:00:31	NULL
6	11111111111	NULL	100.00	0	2022-11-25 22:00:50	NULL
7	11111111111	NULL	100.00	0	2022-11-25 22:01:05	NULL

3.项目测试

“我的”页面

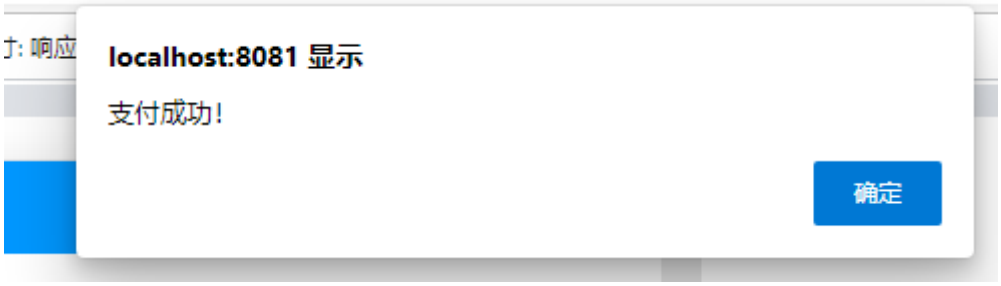


3.1虚拟钱包系统

支付（转账）


在线支付
订单信息:
大于100元的订单可获得10积分噢!
万家饺子（软件园E18店）
¥35
使用100点积分后的价格（打七折）：¥24.5
支付宝
微信支付
确认支付
使用积分
首页
发现
订单
我的

点击“确认支付”


localhost:8081 显示
支付成功!
确定

充值

点击“我的”—>“我的钱包”里的"充值"按钮

点击确定即可完成充值

提现

和充值过程类似

查看流水

点击“我的钱包”里的“查看流水”按钮可看到刚才支付和充值的流水

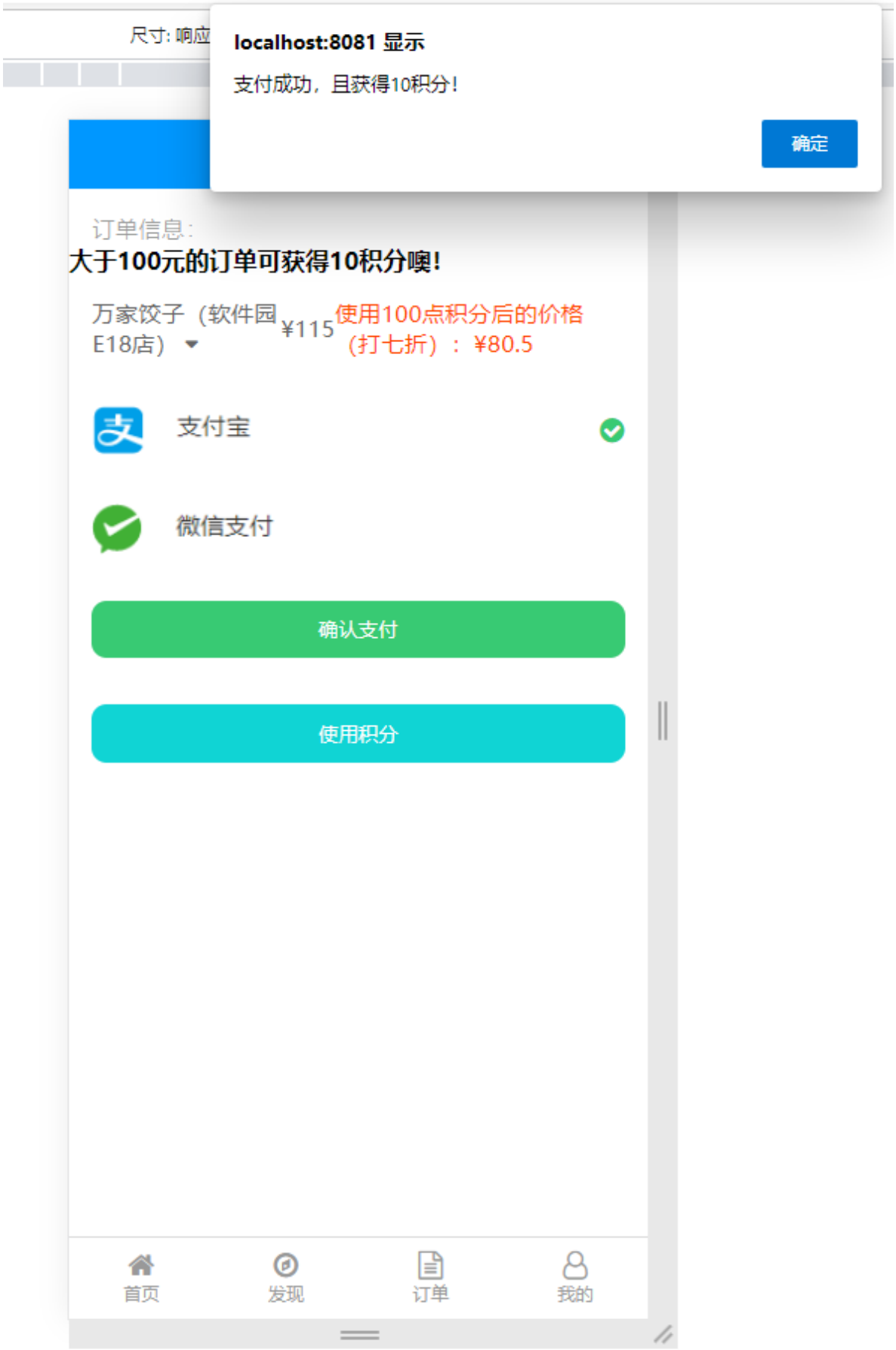
3.2积分系统

使用积分支付

在上文提到的支付页面选择“使用积分”按钮即可使用积分支付。积分规则为原价打7折。

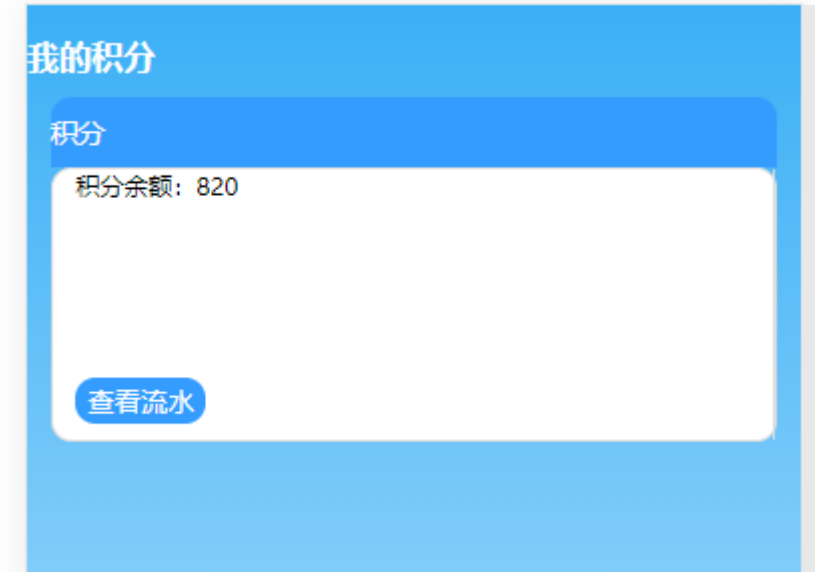
获取积分

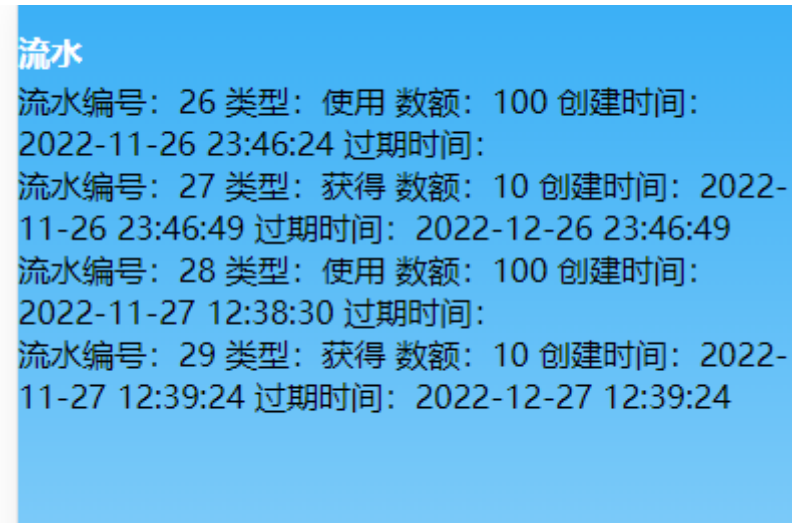
不使用积分并且一个订单的价格大于100元，则可获得10积分。



查看流水

点击“我的积分”里的查看流水可查看刚才操作的流水





4.项目部署

将后端项目部署在云服务器上。（前端程序部署时出现了一些问题，只部署了后端）

项目文件结构

```
[lighthouse@VM-8-3-centos ~]$ ls
cacert.pem                          ngix
caextract.html                      node-v14.17.5-linux-x64
dist                                node-v14.17.5-linux-x64.tar.xz
dist.tar                            nohup.out
downloads                           phpMyAdmin-5.2.0-english
elmboot-0.0.1-SNAPSHOT.jar          phpMyAdmin-5.2.0-english.tar.gz
elmboot-0.0.1-SNAPSHOT.war          projectName.log
```

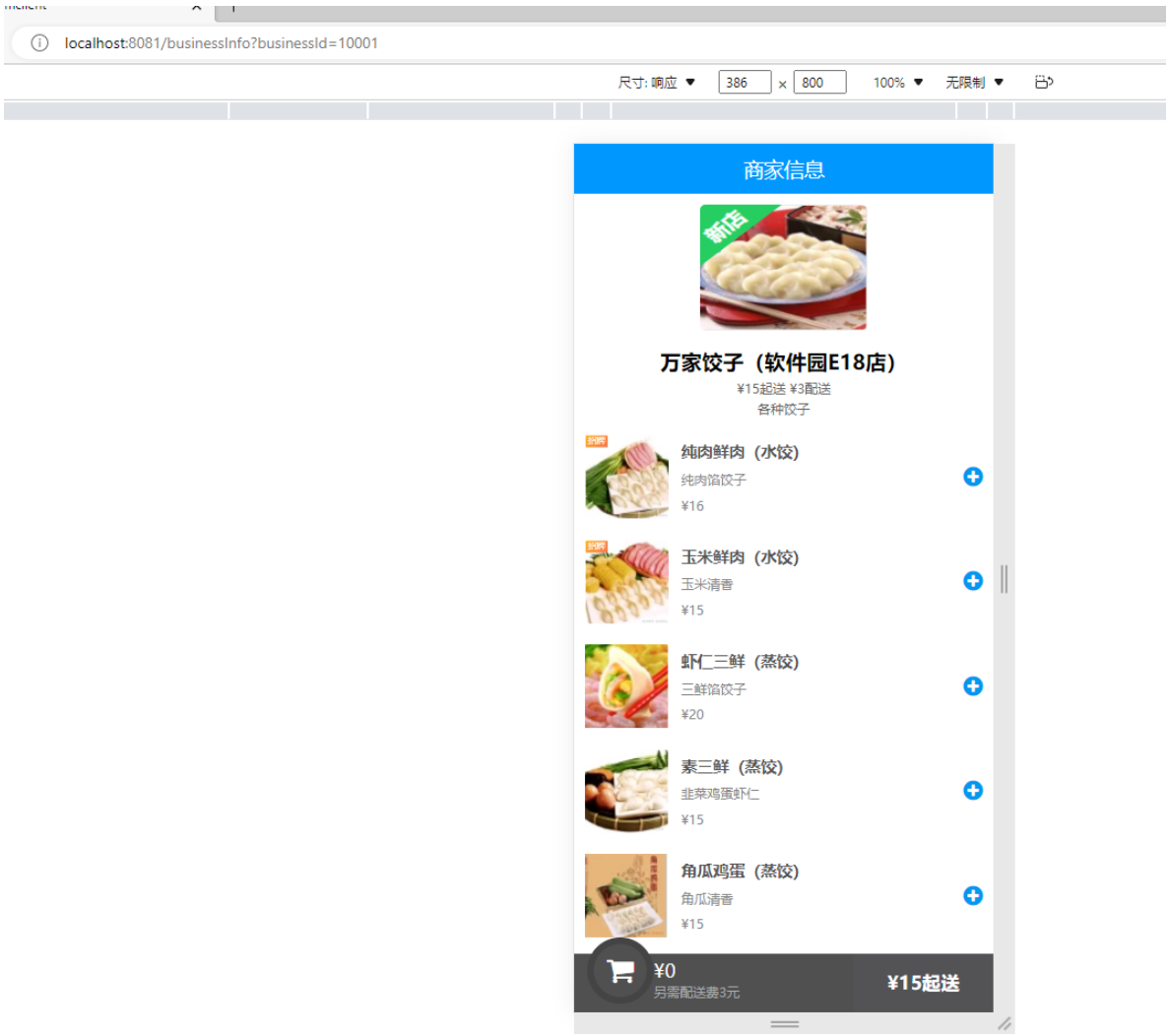
运行

```
56.server.deploy.WebSocketServerContainerInitializer' (OnClassCondition)

   WebSocketServletAutoConfiguration.UndertowWebSocketConfiguration:
      Did not match:
         - @ConditionalOnClass did not find required class 'io.undertow.websockets.jsr.Boots
trap' (OnClassCondition)

   XADataSourceAutoConfiguration:
      Did not match:
         - @ConditionalOnClass did not find required class 'javax.transaction.TransactionMan
ager' (OnClassCondition)


Exclusions:
-----------

    None


Unconditional classes:
----------------------

    org.springframework.boot.autoconfigure.context.ConfigurationPropertiesAutoConfiguration

    org.springframework.boot.autoconfigure.context.LifecycleAutoConfiguration

    org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration

    org.springframework.boot.autoconfigure.availability.ApplicationAvailabilityAutoConfigura
tion

    org.springframework.boot.autoconfigure.info.ProjectInfoAutoConfiguration



2022-11-28 19:09:50.935  INFO 23397 --- [           main] com.neusoft.elmboot.ElmbootApplica
tion    : Started ElmbootApplication in 5.243 seconds (JVM running for 8.098)
```

前端请求ip地址改为服务器的ip地址

```
= 'http://43.143.140.126:8080/elm/';
, 使用时就可以 this.$axios 这样使用了
axios;
```

接下来的操作和之前一样。先运行前端，进入本地的8081端口，即可进行同样的页面操作。



5.体现了哪些面向对象设计原则

1. 高内聚、松耦合。使用MVC架构分层开发，各层各司其职，松耦合，而各个功能高内聚。以以购买商品时的转账功能实现为例：购买商品转账需要同时调用多个方法，这些方法之间的耦合松散，修改互不影响，而功能高度内聚。
2. 面向接口而非实现编程。以购买商品时的转账功能实现为例：前端并不直接实现这个功能，而是调用服务器的/VirtualWalletController/debit和.../BonuspointsController/credit接口对用户和商家分别作出账、入账处理，mapper层再通过接口操作数据库的余额属性使其增加或减少，整个功能就这样得以实现。
3. 多用组合加委托少用继承。从代码实现来看，虚拟钱包和积分系统的controller层都是先创造一个service层实例，再调用改实例的方法。service层同样先创造mapper层实例，再调用其方法。这提现了多用组合加委托少用继承的设计原则。
4. 开闭原则。对扩展开放，对修改关闭。在MVC三层架构模型里，我们在原有的elm项目的基础上增加功能非常简便，只需要在各层添加相应的组件即可，不需要担心代码会跑不起来，但不能通过修改原有代码来增加功能。

6.小组成员及分工

我们的仓库：[elm++: 本仓库为elm项目及其扩展 (gitee.com)]

方景亿3020244070：构建数据库，编写后端代码，负责前端的html和javaScript部分，编写报告

朱培根3020244169：负责前端的css部分

周风毅3020244171：负责前端的css部分

孟家贤3020244256：参与积分系统的数据库构建，负责汇报PPT的编写

7.项目开发过程

迭代周期大概一周

第一次迭代：虚拟钱包数据库——>虚拟钱包springboot服务器——>虚拟钱包前端

第二次迭代：积分系统数据库——>积分系统springboot服务器——>积分系统前端

具体过程可参考gitee上的commit记录。

8.项目进展过程中的问题点与反思

转账时需要同时操作入账和出账两个业务，甚至还要考虑积分的增加减少。这肯定需要用上事务控制，也就是把这些步骤作为一个事务实现或撤销。因为功能逻辑比较简单，这次作业的前端调用入账和出账的接口时没有想到要用事务控制。以后应当注意。