天津大学

# 《程序设计高级实践》课程报告

# 饿了么外卖平台 Spring Cloud 微服务架构



| | |
|---|---|
| 学　　院 | 智能与计算学部 |
| 专　　业 | 软件工程 |
| 年　　级 | **21**级 |
| 姓　　名 | 易东廷　张中天　张梅梅　于鑫慧　杜伟乐 |
| 学　　号 | **3021244187　3021210045　3021244259　3020244119　6321012105** |


**2024** 年 **5** 月 **8** 日

# 目　　录

# 第一章　团员分工

## 1.1　Spring Cloud 注册中心

- 项目负责团员：张梅梅
- 项目描述：本项目主要负责搭建并配置 Eureka Server 注册中心，实现管理服务的注册和发现，确保服务之间的通信顺畅。另外，本项目也需要集成服务路由和负载均衡功能，提高系统的稳定性和性能。

## 1.2　Spring Cloud 微服务网关

- 项目负责团员：张梅梅
- 项目描述：本项目主要负责配置并管理 Spring Cloud Gateway 等网关组件，设置路由规则，并实现请求转发和过滤功能。

## 1.3　Spring Cloud 集中配置管理中心

- 项目负责团员：张中天
- 项目描述：本项目主要负责搭建 Spring Cloud Config Server，用于集中式管理配置信息。此外，需要负责配置Git或其他版本控制系统，用于存储和管理配置文件，最后集成配置中心到项目中，并实现动态配置刷新功能。

## 1.4　Spring Cloud 新增微服务架构

### 1.4.1　积分系统

- 项目负责团员：张中天
- 项目描述：本项目主要负责设计积分系统的数据库结构，包括用户积分表、积分规则表等。此外，本项目需开发积分交易接口，包括积分查询、积分兑换等功能，并且实现积分计算逻辑，根据订单金额、活动规则等计算用户应得积分。

### 1.4.2　虚拟钱包系统

- 项目负责团员：张中天
- 项目描述：本项目主要负责设计钱包系统的数据库结构，包括用户钱包表、交易记录表等，并实现虚拟钱包功能，包括余额查询、充值、提现等。本项目的主要目的是为了实现外卖平台的支付流程，以让外卖平台的操作更方便且具人性化。

### 1.4.3 身份校验

- 项目负责团员：易东廷
- 项目描述：本项目主要负责设计用户身份验证流程，包括注册、登录、认证等环节。本项目具体需要开发用户注册和登录接口，实现用户身份信息的验证和存储，并集成第三方身份验证服务，增强登录状态的管理和用户权限的控制。

## 1.5 Spring Cloud 基础微服务架构

### 1.5.1 用户服务

- 项目负责团员：易东廷
- 项目描述：本项目主要负责开发用户微服务的功能，包括开发用户注册、登录和个人信息管理功能，并实现用户权限管理功能，包括用户角色、权限分配等。

### 1.5.2 地址服务

- 项目负责团员：易东廷
- 项目描述：本项目主要负责开发用户收货地址微服务的管理功能，包括添加、修改、删除地址信息。

### 1.5.3 商家服务

- 项目负责团员：杜伟乐
- 项目描述：本项目主要负责开发商家微服务的管理功能，包括商家名称、商家信息、商家地址等。本项目主要用于用户在使用平台时可以游览所支持的商家，并选择指定商家进行点餐。

### 1.5.4 食品服务

- 项目负责团员：杜伟乐
- 项目描述：本项目主要负责开发外卖食品微服务的管理功能，实现菜单查询接口，用于用户浏览和点餐。

### 1.5.5 购物车服务

- 项目负责团员：于鑫慧
- 项目描述：本项目主要负责开发购物车服务的管理功能，包括商品添加、删除、数量修改等。此外，本项目也需要实现购物车与订单的关联，确保购物车中的商品可以正确结算。

### 1.5.6 点餐服务

- 项目负责团员：于鑫慧
- 项目描述：本项目主要负责开发点餐微服务的核心功能，包括菜品浏览、加入购物车、下单等。此外，本项目也需要实现订单管理功能，包括订单查询、订单状态更新等。

## 1.6 项目云部署

- 项目负责团员：张中天
- 项目描述：本项目主要负责选择合适的云服务提供商，并将项目部署到云服务器上。另外，本项目需要配置云服务器环境，包括虚拟机、数据库、存储等，并实现持续集成和持续部署流程，确保项目的稳定性和安全性。

## 1.7 项目文档

- 项目负责团员：杜伟乐
- 项目描述：本项目主要负责书写本项目文档的内容，其中内容包括了本软件的团员分工和需求分析。

## 1.8 项目测试

- 项目负责团员：张梅梅、于鑫慧
- 项目描述：本项目主要负责对项目进行单元测试，验证每个模块的功能和逻辑是否符合预期。此外，本项目也需要进行集成测试，测试不同模块之间的交互和整体功能，以及进行系统测试，模拟真实场景，验证系统的性能和稳定性。

## 1.9 项目演示

- 项目负责团员：张梅梅
- 项目描述：本项目主要负责演示项目的核心功能和特点，包括展示外卖平台的用户界面和具体点餐操作流程。

# 第二章　软件需求分析

## 2.1　系统背景

在过去的几十年中，餐饮行业一直是人们生活中重要的组成部分。然而，随着城市化和工作压力的增加，传统的进餐方式面临着许多挑战。人们的时间日益宝贵，越来越多的人选择在家或办公室点外卖，以节省时间和精力。这为外卖平台的崛起提供了机会。随着智能手机的普及，移动互联网的兴起以及在线支付的便捷性，外卖平台逐渐演变成一个综合性的餐饮服务平台，为用户提供了更加便利、多样化的选择。

“饿了么”外卖平台为用户和餐厅提供了一个互相连接的桥梁，具有以下核心功能：

- 用户点餐：用户通过“饿了么”移动应用程序或网站浏览附近的餐厅和菜单，并可以根据自己的口味和需求下单。
- 多样选择：平台上的餐厅涵盖了各种菜系，从传统美食到国际化的料理，满足了用户不同的口味偏好。
- 配送服务：“饿了么”为用户提供送餐服务，用户可以实时跟踪订单状态，知道餐食的准确送达时间。
- 在线支付：用户可以通过平台进行在线支付，不需要现金交易，实现O2O（Online to Offline）外卖模式
- 积分系统：用户支付订单会返回一定数目的积分，其可以用于下一次消费的金额抵扣。此外，用户也可以通过每日签到来赚取积分。
- 虚拟钱包系统：用户可以享有虚拟钱包的服务，其中包括收款、转账、支付订单以及查询余额的功能。
- 身份验证：用户在注册时需要经过验证确认身份，以便平台可以确保用户的安全性和可信度，从而提供更可靠的点餐、配送和支付服务。

## 2.2　系统架构

### 2.2.1　饿了么外卖平台系统架构描述

本软件采用前后端分离的开发方式，为移动端开发一个外卖平台软件。

- 数据库：项目将采用MySQL数据库来存储和管理数据。MySQL是一种强大的关系型数据库管理系统，具有高性能、可扩展性和广泛的社区支持。
- 后端：后端应用程序采用了逐层递进的架构，以确保代码的可维护性、可扩展性和性能。首先是使用 JDBC 构建了一个简易的数据库商家系统和管理员系统，以确定后端和数据库的交互逻辑。接着，整合了 Spring

Cloud来进行开发和部署。Spring Cloud是基于Spring框架的一套开发工具，用于构建分布式系统的快速开发和部署。通过Spring Cloud，项目获得了高度的可扩展性，并且易于维护。

- 前端：首先采用 HTML5、 CSS3 和 JavaScript 语言的相关技术开发外卖平台的静态页面。其次采用 Vue3 架构构造一个与后端交互的前端页面。

### 2.2.2 饿了么外卖平台系统架构图

本软件的系统架构图如图 2-1 所示。

## 2.3 业务架构

### 2.3.1 用户点餐业务流程描述

首先，用户在首页选择了相应的商家分类，便会跳转至对应的商家列表页面。在商家页表页面里，用户选择相应的商家并跳转至对应的商家信息页面。在商家信息页面里，用户可以选择往购物车里添加或删除菜品。当用户选择完毕，并会跳转至确认订单页面。

在确认订单页面中，用户可以选择送货地址以及检查自己的订单是否正确，确认无误后就会跳转至在线支付页面。在此页面中，用户可以选择不同的第三方支付方式并确认支付。点餐业务到此结束。

### 2.3.2 用户点餐业务流程图

本软件的用户点餐流程图如图 2-2 所示。

## 2.4 用户需求

本软件由三个用户组成，分别为用户、商家和管理员。用户角色由表 2-1 所示。

## 2.5 功能性需求

### 2.5.1 首页

首页是本软件的主要页面。此页面展示了用户当前的收货地址、商家搜索栏、菜品分类、推荐商家列表以及其他优惠部分。首页的底部有一个菜单，显示外卖平台的【首页】、【钱包】、【订单】以及【我的】按钮。用户可以点击相应的按钮跳转至对应的页面进行操作。

#### 2.5.1.1 界面设计

本软件的首页界面设计如图 2-3 所示。

#### 2.5.1.2 功能按钮

本软件首页功能按钮如表 2-2 所示。

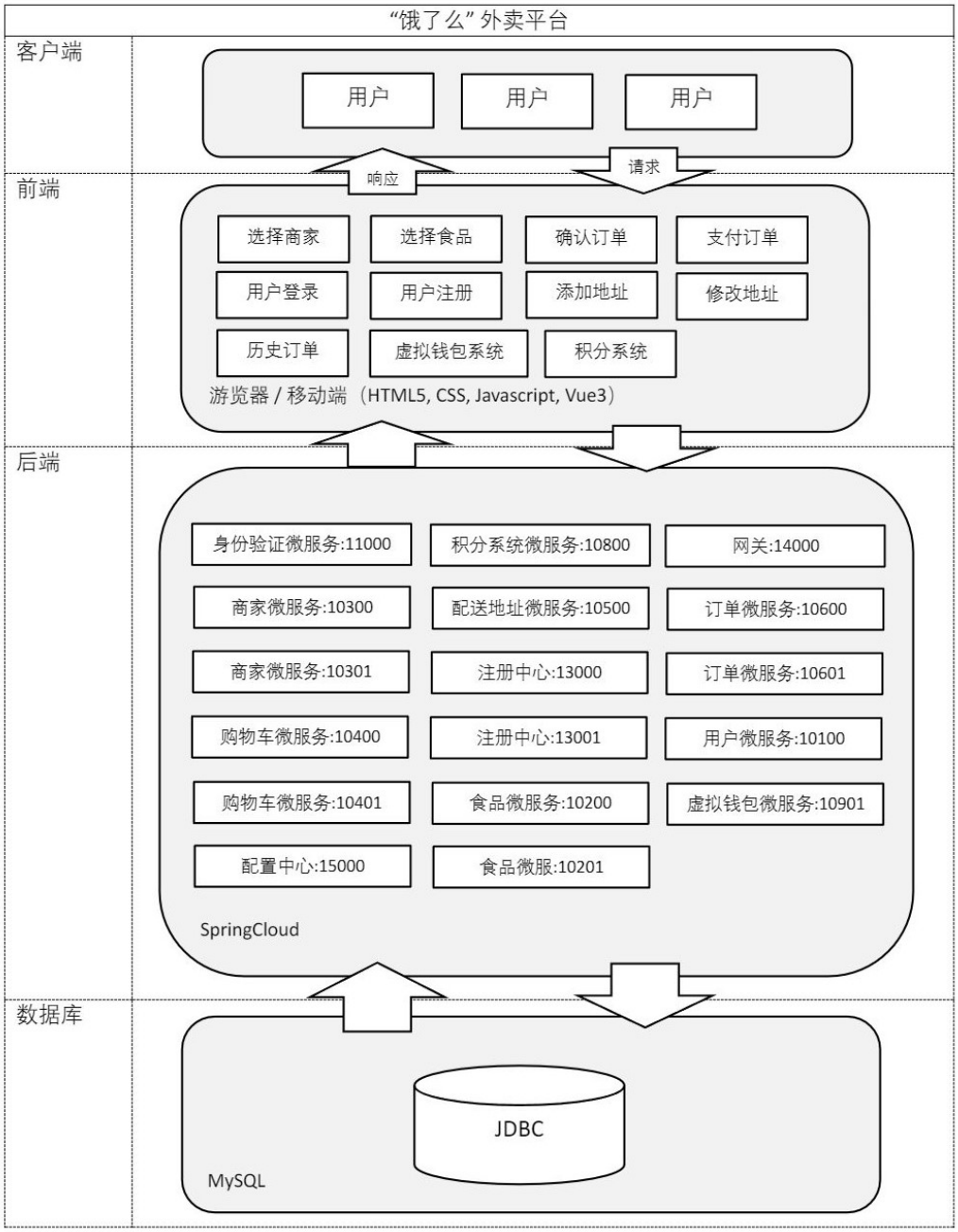


图 2-1　　饿了么外卖平台系统架构图

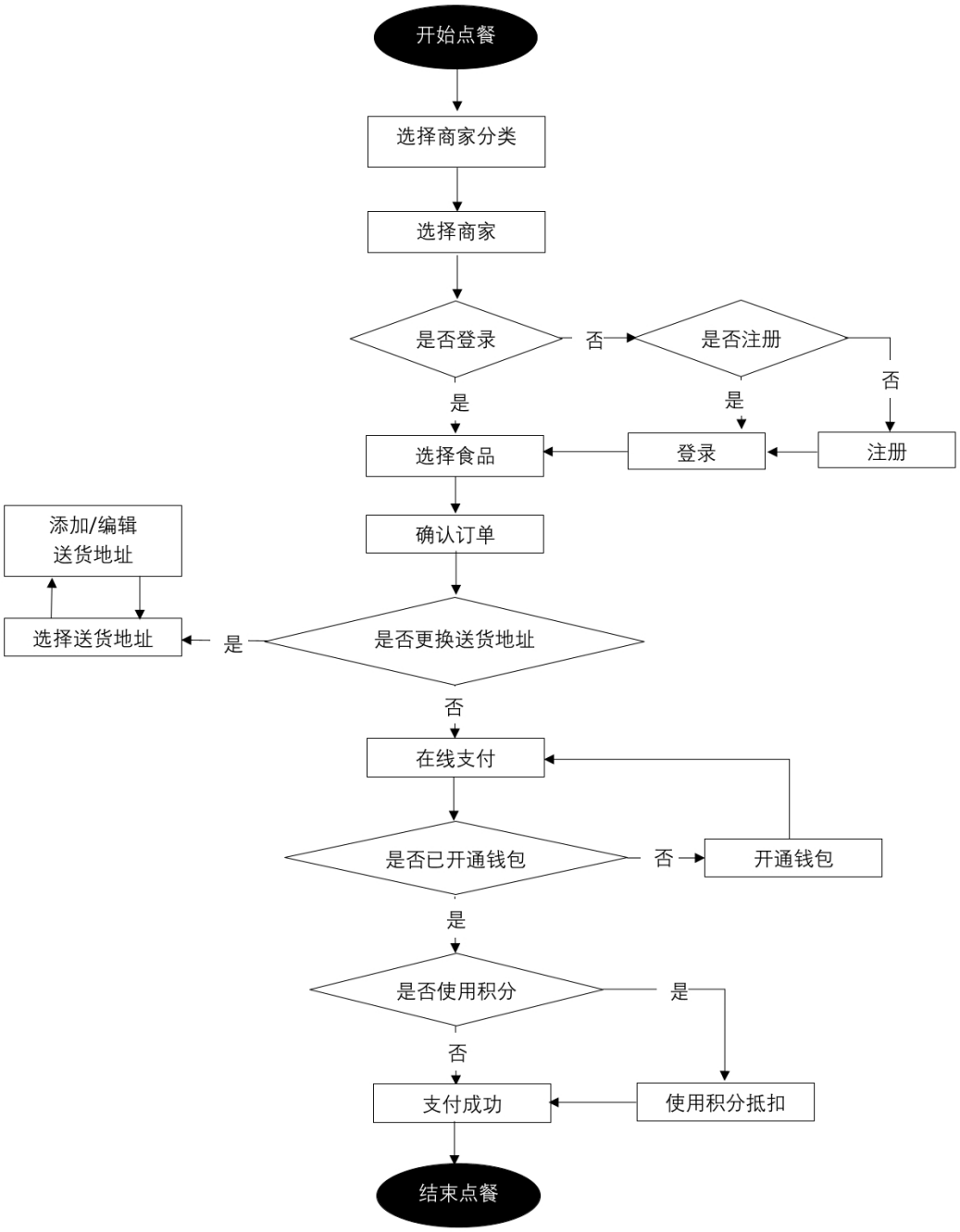


图 2-2　　饿了么外卖平台用户点餐业务流程图

表 2-1　饿了么外卖平台不同用户的功能描述

| 序号 | 用户 | 功能描述 |
| --- | --- | --- |
| 1 | 用户 | • **游览菜品**：消费者可以在首页浏览附近的餐厅列表。他们可以按照菜系、评分、特别优惠等条件筛选餐厅，了解每家餐厅的菜单和菜品详情。<br>• **选择菜品**：消费者可以根据菜品的图片、描述以及价格做出最合适的选择。<br>• **在线支付**：消费者将所选的菜品添加到购物车，确认订单后选择第三方的支付方式进行支付，完成订单。<br>• **管理地址信息**：消费者可以自行添加、删除和查看收获地址的信息。<br>• **管理订单信息**：消费者可以在历史订单页面查看已生成的订单状态，其中包含已选购的商家和菜品明细。<br>• **登陆注册**：消费者可以通过手机号注册一个饿了么外卖平台的账号。 |
| 2 | 商家 | • **注册和上架**：商家需要先在平台上注册自己的餐厅。一旦注册成功，他们可以填写餐厅的基本信息、菜单、菜品描述和价格等。<br>• **菜单管理**：商家可以通过平台管理菜单，包括添加新菜品、修改菜品信息、调整价格等。<br>• **接受订单**：商家在平台上收到订单通知后，可以查看订单详情，包括所点菜品、数量、送达地址以及支付信息。<br>• **准备食物**：商家接受订单后，就可以开始准备食物。 |
| 3 | 管理员 | • **平台监督和管理**：管理员负责监督外卖平台的整体运营情况。他们使用管理后台工具，监控平台的性能、稳定性和安全性，确保用户能够顺利访问平台。<br>• **监控订单和投诉**：管理员跟踪订单处理流程，确保订单按时送达。<br>• **技术支持和升级**：管理员负责监督技术团队，确保平台的技术架构和功能持续适应市场需求，并在必要时进行系统更新。<br>• **风险管理和安全性**：管理员制定隐私政策、安全措施，确保用户的个人信息和支付数据得到妥善保护。 |

表 2-2　饿了么外卖平台首页功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 当前送货地址 | 用户选择送货地址 | 点击跳转至地址管理页面。 |
| 2 | 搜索框 | 用户搜索商家名称 | 输入商家名称可根据商家名称查找商家。 |
| 3 | 商家分类 | 用户选择商家分类 | 点击跳转指定商家分类。 |
| 4 | 首页 | 刷新页面 | 点击跳转至首页。 |
| 5 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 6 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 7 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |

### 2.5.2　商家列表

当用户在首页点击指定的商家分类后，会跳转至指定分类的商家列表页面。此页面展示了一系列商家，其中包括他们的商家图片、商家名称、商家的起送费、配送费和菜品。

#### 2.5.2.1　界面设计

本软件的商家列表界面设计如图 2-4 所示。

#### 2.5.2.2　功能按钮

本软件商家列表功能按钮如表 2-3 所示。

表 2-3　饿了么外卖平台商家列表功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 商家信息 | 用户选择指定商家 | 点击跳转至指定商家信息页面。 |
| 2 | 首页 | 进入首页 | 点击跳转至首页。 |
| 3 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 4 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 5 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |

### 2.5.3　商家信息

当用户在商家列表页面中点击指定的商家后，会跳转至该商家的信息页面。此页面展示了该商家的图片、名称、起送费、配送费以及该商家一系列的菜品。此页面的下方显示了当前购物车的菜品数量、总价格以及【去结算】的按钮。


图 2-3　首页界面设计


图 2-4　商家列表界面设计

#### 2.5.3.1　界面设计

本软件的商家信息界面设计如图 2-5 所示。

#### 2.5.3.2　功能按钮

本软件商家信息功能按钮如表 2-4 所示。

表 2-4　饿了么外卖平台商家信息功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 添加菜品 | 用户添加该菜品数量 | 点击将该菜品在购物车里的数量加1。 |
| 2 | 减少菜品 | 用户减少该菜品数量 | 点击将该菜品在购物车里的数量减1。 |
| 3 | 去结算 | 进入确认订单页面 | 点击会生成订单，跳转至确认订单页面。 |

### 2.5.4　确认订单

当用户在商家信息页面中点击【去结算】按钮后，会跳转至确认订单页面。此页面展示了当前用户的送货地址、用户名称和手机号。在用户信息的下方显示了商家名称、购物车中所选择的菜品图片、菜品名称、菜品数量和配送费。在此页面的底部显示了总价格和【去支付】的按钮。

#### 2.5.4.1　界面设计

本软件的确认订单界面设计如图 2-6 所示。

#### 2.5.4.2 功能按钮

本软件确认订单功能按钮如表 2-5 所示。

表 2-5 饿了么外卖平台确认订单功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 当前送货地址 | 用户选择送货地址 | 点击跳转至地址管理页面。 |
| 2 | 去支付 | 进入在线支付页面 | 点击跳转至在线支付页面。 |


图 2-5 商家信息界面设计


图 2-6 确认订单界面设计

### 2.5.5 在线支付

当用户在确认订单页面中点击【去结算】按钮后，会跳转至确认订单页面。

#### 2.5.5.1 界面设计

本软件的在线支付界面设计如图 2-7 所示。

#### 2.5.5.2 功能按钮

本软件在线支付功能按钮如表 2-6 所示。

### 2.5.6 在线支付成功

当用户在在线支付页面中点击【确认支付】按钮后，会跳转至支付成功页面。

表 2-6　饿了么外卖平台在线支付功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 箭头 | 用户展开订单明细 | 点击展开已生成的订单明细，包括菜品的数量和价格。 |
| 2 | 选择是否使用积分 | 用户选择是否使用积分支付订单 | 点击选择是否花费积分来支付订单。 |
| 3 | 确认支付 | 用户支付订单 | 点击完成点餐流程。 |
| 4 | 首页 | 进入首页 | 点击跳转至首页。 |
| 5 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 6 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 7 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |

#### 2.5.6.1　界面设计

本软件的在线支付界面设计如图 2-8 所示。

#### 2.5.6.2　功能按钮

本软件支付成功功能按钮如表 2-7 所示。

表 2-7　饿了么外卖平台在线支付功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 返回首页 | 跳转至首页 | 点击表示该订单已支付成功，并跳转至首页。 |

### 2.5.7　虚拟钱包

此页面为虚拟钱包页面。用户可以在此页面首次创建钱包，并且可以向钱包充值或提现。此外，用户也可以在此页面查看当前积分以及积分明细。

#### 2.5.7.1　界面设计

本软件的虚拟钱包界面设计如图 2-9 所示。

#### 2.5.7.2　功能按钮

本软件虚拟钱包功能按钮如表 2-8 所示。


图 2-7　在线支付界面设计


图 2-8　支付成功界面设计

表 2-8　饿了么外卖平台虚拟钱包功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 充值 | 用户充值 | 点击向钱包余额增加所充值的数额。 |
| 2 | 提现 | 用户提现 | 点击向钱包余额减少所提现的数额。 |
| 3 | 积分明细 | 进入积分明细 | 点击跳转至积分明细页面。 |
| 4 | 今天签到可以拿 10 积分 | 用户签到 | 点击向积分增加 10。 |
| 5 | 首页 | 进入首页 | 点击跳转至首页。 |
| 6 | 钱包 | 刷新页面 | 点击跳转至虚拟钱包页面。 |
| 7 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 8 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |

### 2.5.8 积分明细

此页面为积分明细页面。在此页面中，用户可以查询当前积分的流水记录，包括已消费的积分和已获取的积分。

#### 2.5.8.1 界面设计

本软件的积分明细界面设计如图 2-10 所示。

#### 2.5.8.2 功能按钮

本软件积分明细功能按钮如表 2-9 所示。

表 2-9　饿了么外卖平台积分明细功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 箭头 | 用户展开积分记录明细 | 点击展开获取积分或消费积分的记录明细。 |
| 2 | 首页 | 进入首页 | 点击跳转至首页。 |
| 3 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 4 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 5 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |


图 2-9　虚拟钱包界面设计


图 2-10　积分明细界面设计

### 2.5.9 用户登录

此页面为用户登录的页面。用户需要输入手机号和密码，若两者正确无误则登陆成功并跳转至上一个页面，否则服务器将会提示信息错误。

#### 2.5.9.1 界面设计

本软件的用户登录界面设计如图 2-11 所示。

#### 2.5.9.2 功能按钮

本软件用户登录功能按钮如表 2-10 所示。

表 2-10 饿了么外卖平台用户登录功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 登录 | 用户登录 | 若手登陆成功则跳转至上一个页面，否则服务器将会显示错误提示。 |
| 2 | 去注册 | 用户注册 | 点击跳转至用户注册页面。 |
| 3 | 首页 | 进入首页 | 点击跳转至首页。 |
| 4 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 5 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 6 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |

### 2.5.10 用户注册

此页面为用户注册的页面。用户需要输入手机号、两次相同的密码、用户姓名以及选择性别。若手机号、密码和用户名都符合要求，则注册成功并跳转至登陆页面，否则服务器将会提示信息错误。

#### 2.5.10.1 界面设计

本软件的用户注册界面设计如图 2-12 所示。

#### 2.5.10.2 功能按钮

本软件用户注册功能按钮如表 2-11 所示。

### 2.5.11 历史订单

历史订单页面展示了用户之前所生成的全部订单，未支付的订单会显示在页面的上方，已支付的订单则会显示在页面的下方。

#### 2.5.11.1 界面设计

本软件的历史订单界面设计如图 2-13 所示。

表 2-11　饿了么外卖平台用户注册功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 性别 | 用户选择性别 | 点击选择任一性别。 |
| 2 | 注册 | 用户注册 | 若注册成功则跳转至登陆页面，否则服务器将显示错误提示。 |
| 3 | 首页 | 进入首页 | 点击跳转至首页。 |
| 4 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 5 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 6 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |


图 2-11　用户登录界面设计


图 2-12　用户注册界面设计

#### 2.5.11.2 功能按钮

本软件历史订单功能按钮如表 2-12 所示。

表 2-12 饿了么外卖平台历史订单功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 箭头 | 用户展开订单明细 | 点击展开已生成的订单明细，包括菜品的数量和价格。 |
| 2 | 去支付 | 用户支付未支付的订单 | 点击跳转至未支付的订单页面。 |
| 3 | 首页 | 进入首页 | 点击跳转至首页。 |
| 4 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 5 | 订单 | 刷新页面 | 点击跳转至历史订单页面。 |
| 6 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |

### 2.5.12 地址管理

当用户在首页或在确认订单页面点击送货地址按钮时，会跳转到此页面。此页面显示了用户的所有送货地址。

#### 2.5.12.1 界面设计

本软件的地址管理界面设计如图 2-14 所示。

#### 2.5.12.2 功能按钮

本软件地址管理功能按钮如表 2-13 所示。

表 2-13 饿了么外卖平台地址管理功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 送货地址 | 用户选择该送货地址 | 点击选择该送货地址，并跳转至上一个页面。 |
| 2 | 编辑图标 | 用户编辑该送货地址 | 点击跳转至编辑送货地址页面。 |
| 3 | 打叉图标 | 用户删除该送货地址 | 点击删除该送货地址，地址管理页面减少该送货地址。 |
| 4 | 新增收货地址 | 用户添加送货地址 | 点击跳转至新增送货地址页面。 |
| 5 | 首页 | 进入首页 | 点击跳转至首页。 |
| 6 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 7 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 8 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |


图 2-13　历史订单界面设计

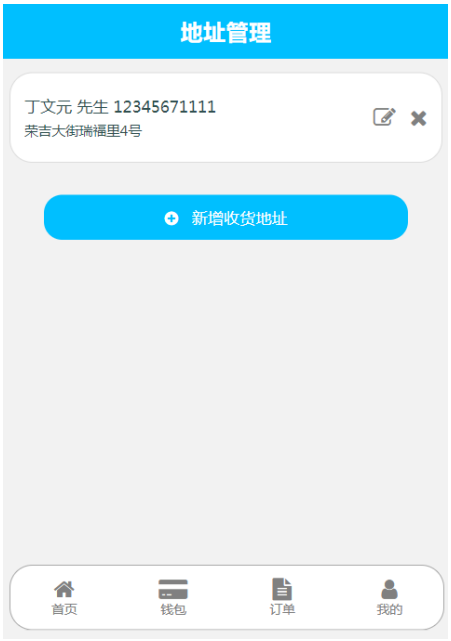


图 2-14　地址管理界面设计

### 2.5.13　新增送货地址

用户在此页面添加新的送货地址。填写完成后，点击保存按钮跳转至地址管理页面。

#### 2.5.13.1　界面设计

本软件的新增送货地址界面设计如图 2-15 所示。

#### 2.5.13.2　功能按钮

本软件新增送货地址功能按钮如表 2-14 所示。

表 2-14　饿了么外卖平台新增送货地址功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 性别 | 用户选择性别 | 点击选择任一性别。 |
| 2 | 保存 | 用户保存当前送货地址 | 点击跳转至地址管理页面。 |
| 3 | 首页 | 进入首页 | 点击跳转至首页。 |
| 4 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 5 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 6 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |

### 2.5.14 编辑地址

用户在此页面编辑已选择的送货地址。填写完成后，点击保存按钮跳转至地址管理页面。

#### 2.5.14.1 界面设计

本软件的编辑送货地址界面设计如图 2-16 所示。

#### 2.5.14.2 功能按钮

本软件编辑送货地址功能按钮如表 2-15 所示。

表 2-15 饿了么外卖平台编辑送货地址功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 性别 | 用户选择性别 | 点击选择任一性别。 |
| 2 | 保存 | 用户保存当前送货地址 | 点击跳转至地址管理页面。 |
| 3 | 首页 | 进入首页 | 点击跳转至首页。 |
| 4 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 5 | 订单 | 进入历史订单 | 点击跳转至历史订单页面。 |
| 6 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |


图 2-15 新增送货地址界面设计


图 2-16 编辑送货地址界面设计

### 2.5.15 个人信息

个人信息页面显示当前用户的图片和名称。用户可以在此页面修改用户名和密码、跳转至地址管理页面以及退出登录。

#### 2.5.15.1 界面设计

本软件的个人信息界面设计如图 2-17 所示。

#### 2.5.15.2 功能按钮

本软件编辑送货地址功能按钮如表 2-16 所示。

表 2-16 饿了么外卖平台编辑送货地址功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 编辑图标 | 进入更新用户名页面 | 点击跳转至更新用户名页面。 |
| 2 | 修改密码 | 进入更新密码页面 | 点击跳转至更新密码页面。 |
| 3 | 收货地址 | 进入地址管理页面 | 点击跳转至地址管理页面。 |
| 4 | 退出登录 | 进入首页 | 点击退出登录，跳转至首页。 |
| 5 | 首页 | 进入首页 | 点击跳转至首页。 |
| 6 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 7 | 订单 | 进入历史订单页面 | 点击跳转至历史订单页面。 |
| 8 | 我的 | 刷新页面 | 点击跳转至用户信息页面。 |

### 2.5.16 更新用户名

用户在此页面更新用户名。填写完成后，点击保存按钮跳转至个人信息页面。

#### 2.5.16.1 界面设计

本软件的更新用户名界面设计如图 2-18 所示。

#### 2.5.16.2 功能按钮

本软件更新用户名功能按钮如表 2-17 所示。

### 2.5.17 更新密码

用户在此页面更新密码。填写完成后，点击保存按钮跳转至个人信息页面。

#### 2.5.17.1 界面设计

本软件的更新密码界面设计如图 2-19 所示。

#### 2.5.17.2 功能按钮

本软件更新密码功能按钮如表 2-18 所示。

表 2-17　饿了么外卖平台更新用户名功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 保存 | 用户保存当前用户名 | 点击跳转至个人信息页面。 |
| 2 | 首页 | 进入首页 | 点击跳转至首页。 |
| 3 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 4 | 订单 | 进入历史订单页面 | 点击跳转至历史订单页面。 |
| 5 | 我的 | 刷新页面 | 点击跳转至用户信息页面。 |

表 2-18　饿了么外卖平台更新密码功能按钮

| 序号 | 按钮名称 | 功能 | 功能规则 |
|---|---|---|---|
| 1 | 保存 | 用户保存当前密码 | 点击跳转至个人信息页面。 |
| 2 | 首页 | 进入首页 | 点击跳转至首页。 |
| 3 | 钱包 | 进入虚拟钱包 | 点击跳转至虚拟钱包页面。 |
| 4 | 订单 | 进入历史订单页面 | 点击跳转至历史订单页面。 |
| 5 | 我的 | 进入用户信息 | 点击跳转至用户信息页面。 |


图 2-17　个人信息界面设计


图 2-18　更新用户名界面设计


图 2-19　　更新密码界面设计

# 第三章　项目特色

## 3.1　Token 身份验证

Token身份校验是现代计算机系统中常用的一种安全措施，主要用于验证用户或设备的身份，确保系统的安全性。具体来说，token身份校验的作用包括：用户身份验证，防止伪装攻击等。

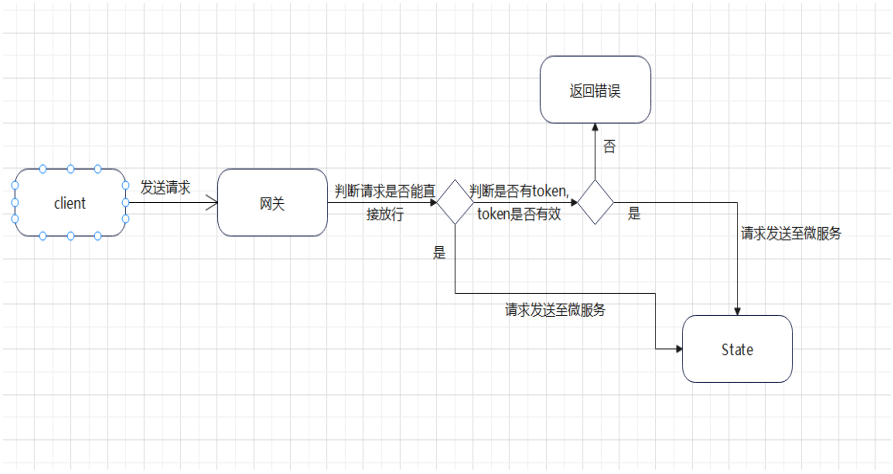


图 3-1　token认证流程

### 3.1.1　认证流程

### 3.1.2　Token分发

主要在用户登录接口中实现了token分发，在用户成功登录之后，会将token存入redis缓存中，并将用户信息以及token返回给用户，实现如下：

```
1        @Override
2        public LoginVo login(String username,String password) {
3                User user = userMapper.getUserInfoByname(username);
4                if (user == null){
5                        return null;
6                }
7
8                // 判断用户名密码是否正确
9                if (!password.equals(user.getPassword())){
10                       return null;
11               }
```

```
12                  // 生成token
13                  String token = jwtProvider.generateToken(user.
                        getUserName());
14
15                  // 将存入tokenredis
16                  redisManager.set(UserConstant.USER_TOKEN_KEY_REDIS +
                        user.getUserName(),token,604800);
17                  LoginVo lg = new LoginVo();
18                  lg.setToken(prefix + " " +token);
19                  lg.setUserId(user.getUserId());
20                  lg.setUserName(username);
21                  return lg;
22          }
```

### 3.1.3 Token认证

token认证的功能主要实现在gateway的过滤器上。过滤器首先验证是否为放行路径，如果是直接将请求发送到微服务，如果不是会验证token是否存在，已经token是否有效，实现如下：

```
1           @Override
2           public Mono<Void> filter(ServerWebExchange exchange,
                GatewayFilterChain chain) {
3
4                   ServerHttpRequest request = exchange.getRequest();
5                   String requestPath = request.getPath().toString();
6                   System.out.println(paths);
7                   System.out.println(requestPath);
8                   boolean allowedPath = false;
9                   if (paths != null && !paths.equals("")){
10                          allowedPath = requestPath.contains(paths);
11                  }
12                  if (allowedPath || StringUtils.isEmpty(requestPath)){
13                          return chain.filter(exchange);
14                  }
15
16                  // 验证token
```

```
17                  String authHeader = exchange.getRequest().getHeaders
                        ().getFirst(tokenHeader);
18                  if (authHeader != null && authHeader.startsWith(
                        prefix)){
19                      String authToken = authHeader.substring(
                            prefix.length());
20                      String userName = jwtProvider.
                            getUserNameFromToken(authToken);
21
22                      // 查询redis
23                      Object token = redisManager.get(UserConstant.
                            USER_TOKEN_KEY_REDIS + userName);
24                      if (token == null){
25
26                          return writeResponse(exchange.
                                getResponse(),401,"已过期token请重新
                                登录...");
27                      }
28
29                      String trimAuthToken = authToken.trim();
30                      if (! trimAuthToken.equals(token.toString()))
                            {
31                          return writeResponse(exchange.
                                getResponse(),401,"验证失败或已过
                                期token请重新登录...");
32                      }
33                  }else {
34                      return writeResponse(exchange.getResponse()
                            ,500,"不存
                            在token");
35                  }
36                  return chain.filter(exchange);
37          }
```

## 3.2 积分系统

积分系统的设计参考了美团APP的米粒功能

### 3.2.1 规则设计

#### 3.2.1.1 积分获取

1. 签到获得积分，每日通过签到获得的积分上限为1次
2. 消费时获得积分，订单金额按比例兑换为积分数量
3. 充值时获得积分，将电子钱包的充值按比例转化为积分

#### 3.2.1.2 积分消耗

1. 积分存在有效期，积分随着时间的增加自然减少
2. 下单时可以通过消耗积分，来减少支付所需钱的数量

#### 3.2.1.3 特殊的规则

1. 选择优先过期的积分进行消费
2. 连续签到，节假日签到，特殊节日消费特殊商品等，有特殊的积分规则

### 3.2.2 数据库设计

这里展示了积分规则的数据库图表，具体如下：

表 3-1　积分记录表 (creditrecord)

| 字段名 | 数据类型 | 说明 |
|---|---|---|
| id | int | 主键id |
| userId | varchar | 用户id |
| ruleCode | varchar | 规则码 |
| eventId | int | 订单/充值id |
| credit | int | 积分值 |
| createTime | datetime | 创建时间 |
| expiredTime | datetime | 过期时间 |

### 3.2.3 接口设计

查询当前可用积分

CreditController/queryAvailableCredit

参数:userId

返回值:int

功能:查找userId用户的所有可用积分

查询积分流水

CreditController/queryAllCredit

参数:userId

表 3-2　可用积分表 (usablecredit)

| 字段名 | 数据类型 | 说明 |
|---|---|---|
| id | int | 主键id |
| userId | varchar | 用户id |
| recordId | int | 流水表记录id |
| credit | int | 积分值 |
| createTime | datetime | 创建时间 |
| expiredTime | datetime | 过期时间 |
| deleted | int | 过期标记 |

表 3-3　积分扣除表 (reducecredit)

| 字段名 | 数据类型 | 说明 |
|---|---|---|
| id | int | 主键id |
| userId | varchar | 用户id |
| recordId | int | 流水表中消费积分记录id |
| usableId | int | 可用积分表中对应的记录id |
| credit | int | 积分值 |
| createTime | datetime | 创建时间 |
| expiredTime | datetime | 过期时间 |

表 3-4　积分规则表 (creditrule)

| 字段名 | 数据类型 | 说明 |
|---|---|---|
| id | int | 主键id |
| ruleCode | varchar | 积分规则码 |
| type | int | 获取/消费 |
| priority | int | 优先级 |
| credit | int | 积分值 |
| formula | int | 公式 |
| dalyCap | int | 日上限 |
| totCap | int | 总上限 |
| startTime | datetime | 开始时间 |
| endTime | datetime | 结束时间 |
| lifespan | int | 有限期 |
| state | int | 启用/停用 |

返回值:List(记录积分的数组)

功能:查找userId用户的所有专区或者消费的积分明细

签到获得积分

CreditController/earnCreditBySign

参数:userId,creditNum,ruleCode

返回值:int

功能:签到获得积分，每日通过签到获得的积分上限为1次

查询签到可以获得的积分总数

CreditController/queryEarningCreditBySign

参数:userId,ruleCode

返回值:int

功能:查找userId用户，在规则码为ruleCode的情况下，能够获取的积分总数

查询支付可以获得的积分总数

CreditController/queryEaringCreditByPaying

参数:userId,money,ruleCode

返回值:int

功能:给所有userId的用户查询支付money的钱可以获得多少积分

支付后获得积分

CreditController/earnCreditByPaying

参数:userId,orderId,creditNum,ruleCode

返回值:int

功能:让用户支付之后获得积分

支付时抵扣积分

CreditController/queryConsumingCreditByPaying

参数:userId,money,creditNum,ruleCode

返回值:creditNum,deductionMoney

功能:让用户在支付时使用积分做抵扣

## 3.3 虚拟钱包系统

参考美团APP的天天神券功能，制作了一个虚拟支付钱包。

### 3.3.1 规则设计

#### 3.3.1.1 电子钱包的创建

1. 用户同意时，主动创建一个钱包
2. 商家需要主动与平台完成签约创建钱包

#### 3.3.1.2 支付逻辑

1. 优先消耗积分支付
2. 必须保证钱包扣款的线程安全
3. 保障支付本身的事务性

### 3.3.2 实现细节

#### 3.3.2.1 架构分层设计

在定义上，我们不仅采用了 pojo 层，还采用了 VO 层，其中， pojo 层又叫 PO，是面向数据库的，与其协同的对象分别为 Service 层和 Dao 层，放从数据库中直接拿出来的数据，没有任何逻辑实现，只有 getter 域 setter 方法，数据库的每一个字段都对应着 POJO 的一个属性。 VO 是面向 Service 层与 Controller(Web) 层的， VO 的数据是 POJO 中的数据加工后得到的，用于在 Web 页面上展示。

```
1  package com.tju.elmcloud.po;
2
3  public class VirtualWalletPo {
4          private String userId;
5          private Integer walletId;
6          private double balance;
7
8          public VirtualWalletPo(){
9                  this.balance=0.00;
10         }
11         public VirtualWalletPo(Integer walletId,double balance){
12                 this.balance=balance;
13                 this.walletId=walletId;
14         }
15         public Integer getWalletId() {
16                 return walletId;
17         }
18         public double getBalance() {
19                 return balance;
20         }
21         public String getUserId() {
22                 return userId;
23         }
```

```
24          public void setUserId(String userId) {
25                  this.userId = userId;
26          }
27  }
```

#### 3.3.2.2 保障事务性

在本项目中，为了保证钱包支付的事务性，我们采取了 synchronized 和保障线程安全的 cocurrentHashMap 来实现，具体为在操作线程钱进行加锁，来保证关于交易操作的 ACID 事务性，在保障线程安全的同时也保证了并发效率。这里以 queryEarningCreditBySign 方法举例。

```
1   @Override
2   public Integer queryEarningCreditBySign(String userId) {
3           Integer ruleId = 1;
4           String time = CommonUtil.getCurrentDate();
5           String today = time.substring(0, time.indexOf(' ')).trim();
6           int count = creditRecordMapper.todaySignRecord(userId, ruleId
                , today);
7           SignCreditRule signCreditRule = null;
8           synchronized (creditRuleMap) {
9                   signCreditRule = (SignCreditRule) creditRuleMap.
                        getRule(ruleId);
10                  if (signCreditRule == null) {
11                          CreditRulePo creditRulePo = creditRuleMapper.
                                getRule(ruleId);
12                          int credit = creditRulePo.getCredit();
13                          int lifeSpan = creditRulePo.getLifespan();
14                          int totCap = creditRulePo.getDailyCap();
15                          signCreditRule = new SignCreditRule(lifeSpan,
                                 credit, totCap);
16                          creditRuleMap.writeMap(ruleId, signCreditRule
                                );
17                  }
18          }
19          CreditSystem creditSystem = new CreditSystemImpl();
20          return creditSystem.queryEarningCreditBySign(count,
                signCreditRule);
```

```
}
```

#### 3.3.2.3 设计模式

在本项目中，我们采用了单例模式、策略模式、代理模式等设计模式.其中，单例模式使用了双检锁机制，保证了线程安全和高性能;策略模式用于积分规则的设计，使代码耦合度变低;代理模式用于虚拟钱包的设计，使代码更加简洁，易于维护。以单例模式为例，我们在 CreditRuleMap 类中实现了单例模式。

```
package com.tju.elmcloud.creditRuleMap;
import com.tju.elmcloud.domain.Rule;
import com.tju.elmcloud.mapper.CreditRuleMapper;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

public class CreditRuleMap {
        private Map<Integer, Rule> ruleMap;

        private static volatile CreditRuleMap creditRuleMap; // 单例实例，使用 volatile 确保多线程环境下的可见性
        private CreditRuleMapper creditRuleMapper;

        private CreditRuleMap() {
                ruleMap = new ConcurrentHashMap<>();
        }

        public static CreditRuleMap getRuleMap() {
                if (creditRuleMap == null) {
                        synchronized (CreditRuleMap.class) {
                                if (creditRuleMap == null) {
                                        creditRuleMap = new
                                                CreditRuleMap();
                                }
                        }
                }
                return creditRuleMap;
        }

```

```
28          public Rule getRule(Integer ruleId) {
29                  if (ruleMap == null)
30                          ruleMap = new ConcurrentHashMap<>();
31                  if (ruleMap.containsKey(ruleId)) {
32                          return ruleMap.get(ruleId);
33                  } else {
34                          return null;
35                  }
36          }
37
38          public void writeMap(Integer ruleId, Rule creditRule) {
39                  ruleMap.put(ruleId, creditRule); // 将规则与 ID 关联
40          }
41  }
```

# 第四章　总结

## 4.1　项目中遇到的问题及解决方法

### 4.1.1　Feign报错

在项目开发过程中，经常会抛出 IllegalStateException 异常，经过思考和查阅资料，我们发现这是因为在 Feign 中，我们使用了 @RequestParam 注解，而 Feign 不支持解析 @RequestParam 注解，所以我们需要使用要用 value 标明对应的参数。

### 4.1.2　ScanMapper报错

在开发时，由于有时项目不能识别已有的XML资源，我们最开始使用了 @MapperScan 注解，而在 MyBatis 中， @MapperScan 注解需要指定对应的包路径，这导致我们的项目在负载均衡场景下出现了问题，经常由于报错而自动刷新，随后又恢复。后来我们发现，可以在 application.yml 文件中配置 mybatis.mapper-locations 属性，指定对应的XML资源路径，这样就可以解决这个问题。而最终呈现的代码版本去掉了 @MapperScan 注解，改用配置文件的方式指定XML资源路径，解决了这个问题。

### 4.1.3　请求绕过网关直接访问微服务

在测试时，发现请求可以直接绕过网关访问微服务，这样gateway的负载均衡，熔断，以及token认证功能都会直接失效，为了解决这个问题，我们让网关每次发送请求时，给请求添加一个from请求头，微服务端收到请求时会先查看是否有该请求头以判断请求是否是通过网关转发，如果不是会返回错误。

## 4.2　项目开发过程

### 4.2.1　项目共享仓库

我们对于本实践项目使用 GITHUB 进行仓库共享，让每一名组员可以在该仓库上进行代码的修改以及同步，大幅度增加实践效率。通过以下网址，可以游览到本项目的共享仓库页面：https://github.com/stainsatin/elm

本次实践项目的共享仓库于 4 月 4 日创建，仓库里的成员包括我们每一名组员。如上图 4-1 所示，仓库里的内容包括本次实践项目的前端、后端与实践报告的目录，并且我们每一名组员都在此共享仓库上进行了多次代码的复制和提交。经过接近一个月的项目开发，以下是我们的共享仓库数据视图，如图 4-2 至图所示：


图 4-1　　GITHUB 共享仓库页面

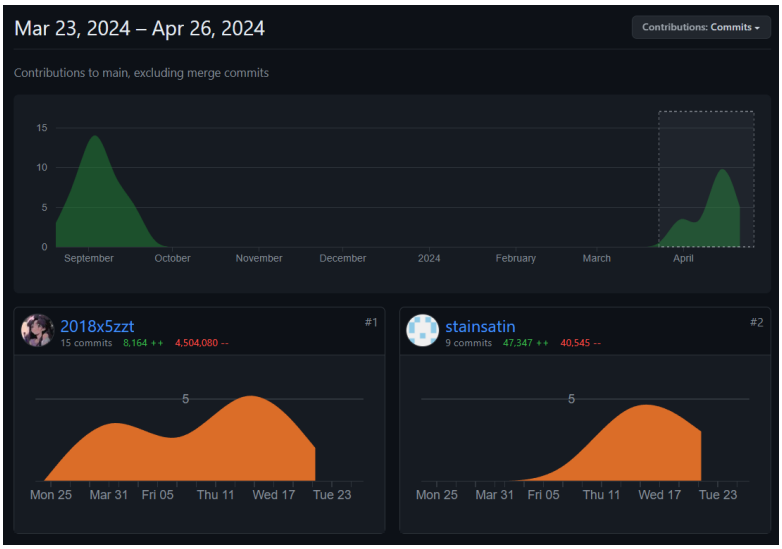


图 4-2　　GITHUB 共享仓库提交视图

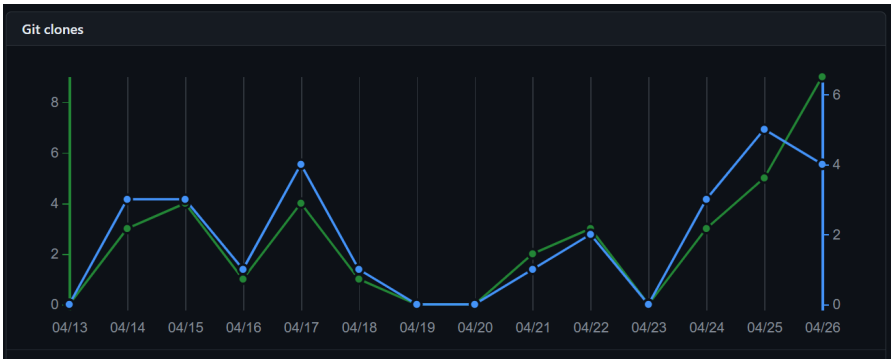


图 4-3　　GITHUB 共享仓库复制视图

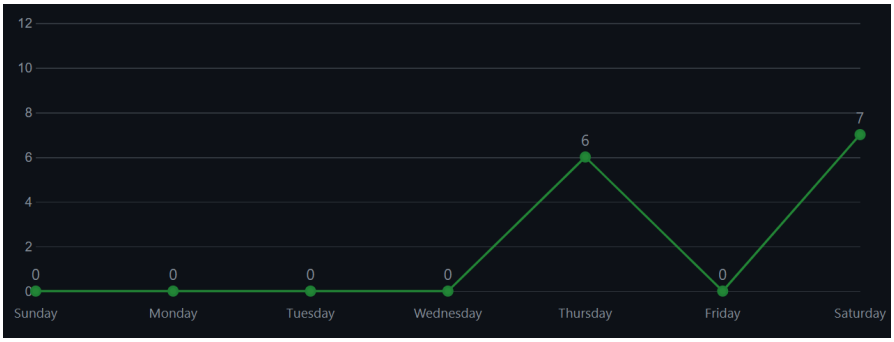


图 4-4　　GITHUB 共享仓库第一周提交次数视图

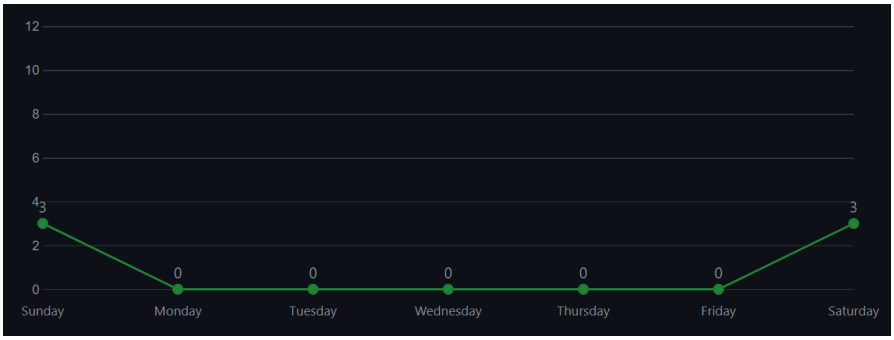


图 4-5　　GITHUB 共享仓库第二周提交次数视图

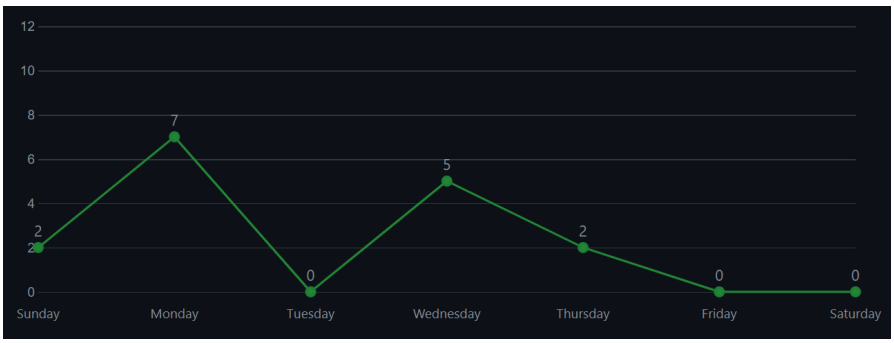


图 4-6　　GITHUB 共享仓库第三周提交次数视图

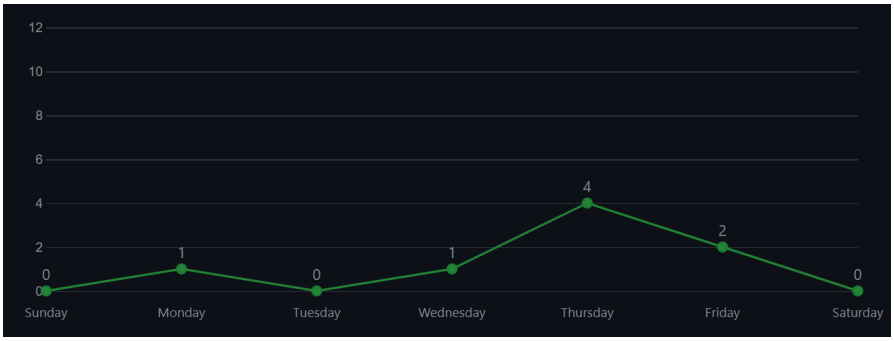


图 4-7　　GITHUB 共享仓库第四周提交次数视图

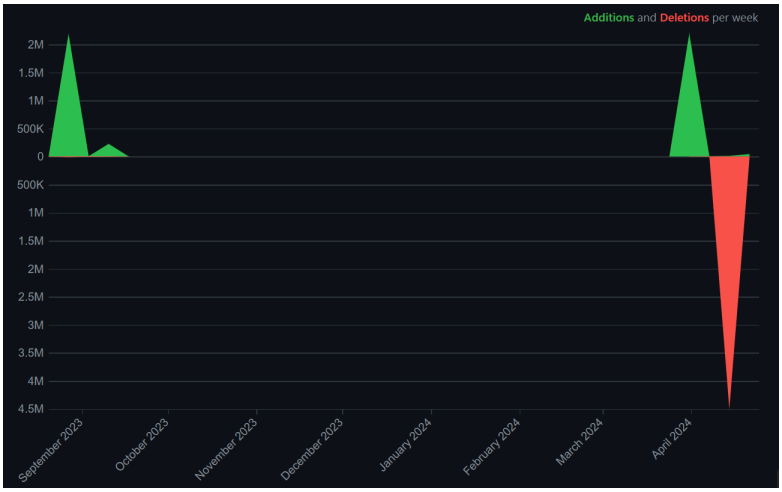


图 4-8　　GITHUB 共享仓库代码频率视图

## 4.3 开发总结

总的来说，这是一个比较复杂的软件项目，其中涉及到了 Spring Cloud 框架、Vue3 等一系列网页开发的技术。在这样的情况下，我们四位小组成员分工明确，可以较为顺利且相互配合地完成自身的开发部分。在交互，测试的过程中，我们遇到了各种各样的问题，这些问题我们也能够准确找到问题出现的地方，知道这部分问题是由谁负责完成。大家积极配合彼此工作，才能让项目比较顺利的进行下去。

在整个项目开发过程中，每个成员都做出了重要的贡献。大家分工明确，对自己所做的部分认真负责，同时在某些阶段，当部分组员任务较重时，大家也会主动去分担任务缓解彼此压力。我们每个人都发挥自己所长，为项目做出贡献。

总的来说，这是一次收获颇丰的实践经历。通过这次对程序设计高级实践的学习，我们学会的很多网页开发的技术，项目管理的经验，也锻炼了在面对复杂问题，出现意外情况时钻研思考，独立解决问题的能力。另外我们也培养了团队成员间积极沟通，配合，团结协作的能力。相信这次宝贵的经验能让我们运用在后续在软件工程专业的学习过程中。