

本科生毕业设计

设计题目 基于SSM的外卖服务平台

学 院 计算机与信息工程学院

学生姓名 雷东宸 学号 2018122026708

专 业 网络工程 年级 19级

指导教师 陈晨 职称 讲师

内蒙古农业大学教务处制

年 月 日

摘 要

外卖平台和互联网有机结合,使实体餐饮行业有了新的突破,有良好的发展前景。本文基于Spring Boot和Mybatis-Plus，开发了一款面向用户和商家的外卖服务平台。该平台具有在线点餐、订单管理、用户管理、套餐管理、支付配送等基本功能。除此之外，本设计将专注于降低外卖开店成本，让利与商家和用户。为用户提供食品安全的保障，为商家提供便捷的服务。

在系统架构设计中，本文利用MySQL主从复制机制实现了读写分离，提高了系统的读写性能。同时，采用Nginx反向代理和负载均衡技术，将请求均匀地分配到多个服务器上，实现多个服务器协同工作，进一步提高系统的稳定性和可用性。还通过Redis缓存机制和Nginx负载均衡技术实现平台的高可用性、高可靠性和高读写性。此外，平台还具有一些安全、支付和数据统计等特性，够保证用户和商家的信息安全和数据隐私。但鉴于在本设计过程中，并没有商家资质，所以有些功能并没有真正涉及，例如支付接口，短信验证码等等。

最后，本文将利用上述框架构建一款功能丰富、性能优异的外卖服务平台，并采用一些现代软件开发方法来确保平台的可靠性。

关键词：Spring Boot，Mybatis-Plus，Redis，Nginx。

Takeout Service Platform Based on SSM

Abstract

The combination of takeaway platforms and the Internet has brought new breakthroughs to the physical catering industry and has good development prospects. This paper is based on Spring Boot and Mybatis-Plus, and develops a takeaway service platform for users and merchants. The platform has basic functions such as online ordering, order management, user management, package management, payment and delivery. In addition, this design will focus on reducing the cost of opening a takeaway store and benefiting merchants and users. It provides food safety guarantees for users and convenient services for merchants.

In the system architecture design, this article uses the MySQL master-slave replication mechanism to achieve read-write separation and improve the system’s read-write performance. At the same time, Nginx reverse proxy and load balancing technology are used to evenly distribute requests to multiple servers, realize the collaborative work of multiple servers, and further improve the stability and availability of the system. The platform also achieves high availability, reliability and read-write performance through Redis caching mechanism and Nginx load balancing technology. In addition, the platform has some security, payment and data statistics features to ensure user and merchant information security and data privacy. However, since there is no merchant qualification in this design process, some functions have not really been involved, such as payment interface, SMS verification code, etc.

Finally, this paper will use the above framework to build a feature-rich and high-performance takeaway service platform, and adopt some modern software development methods to ensure the reliability of the platform.

Keywords: *Spring Boot, Mybatis-Plus, Redis, Nginx.*

目 录

1 绪论 1

1.1 研究背景和意义 1

1.2 国内外研究现状 1

1.2.1 国外研究现状 1

1.2.2 国内研究现状 2

2 相关知识和开发工具 3

2.1 SSM概述 3

2.2 IntelliJ IDEA 2022.1 3

2.3 VMware Workstation pro 3

3 系统分析 4

3.1 需求分析 4

3.2 过程分析 4

3.3 环境分析 4

3.4 成本效益分析 5

4 系统设计 6

4.1 系统架构设计 6

4.2 数据库设计 6

4.3 接口设计 8

4.4 系统性能设计 9

4.4.1 硬件配置 9

4.4.2 软件配置 9

4.5 用户界面设计 10

4.5.1 后台管理界面 10

4.5.2 用户界面设计 10

5 系统实现 12

5.1 环境配置 12

5.1.1 IntelliJ IDEA 12

5.1.2 服务器配置 12

5.2 模块开发 13

5.2.1 登录模块 13

5.2.2 菜品管理模块 14

5.2.3 订单管理模块 14

5.2.4 购物车管理模块 15

5.2.5 地址管理模块 16

5.3 项目部署 16

5.3.1 部署架构 16

5.3.2 部署项目到服务器 17

6 系统测试 20

6.1 测试目标和策略 20

6.2 执行测试 20

6.2.1 登录功能测试 20

6.2.2 菜品管理功能测试 21

6.2.3 订单功能测试 21

6.2.4 其他功能测试 22

6.3 测试分析与评估 23

致谢 25

参考文献 26

附录 27

# 1 绪论

## 1.1 研究背景和意义

中国的外卖平台自2011年产生以来,呈现不断发展和前进的趋势。校园外卖以其食品选择多样化、配送便捷化的特点,给大学生的就餐带来了一定的便利,提高了大学生的生活质量。然而外卖食品安全问题引发的投诉数量逐年增加，2019年上半年，全国食品药品监管部门共受理食品安全投诉7.2万余件，其中涉及外卖平台的投诉占比超过70%。其次，外卖平台通常会对商家收取高额的佣金，这导致有些商家被迫提高价格或者降低服务质量，而消费者也被迫支付更高的价格。一些外卖平台利用算法来分析用户的行为习惯，从而给用户打上标签，实施差异化定价。这就导致了同样的餐品、同样的地址、同样的时间，不同的用户可能会看到不同的价格，甚至开通会员后反而更贵。这种做法不仅损害了用户的利益，也损害了市场的公平竞争。[1]

而价格是消费者选择的主要因素之一，但是餐饮企业在外卖平台开店时，必须承担外卖平台的索取的各种手续费、佣金，同时还必须保证菜品价格具有竞争力，这使得企业的利润空间很小。同时，餐饮企业为了吸引消费者不得不通过打折、优惠等方式来降低价格，使得企业盈利困难。因此，卖家只有降低菜品质量才能提高利润，侵害了用户的权益。

同时，由技术引发的公民个人隐私泄露问题也不断出现，行业面临严峻的监管合规及安全挑战。包括用户的姓名、电话号码、地址等，以及餐饮企业的营业数据、财务数据等。如果数据不当处理或被黑客攻击，将对用户和企业造成严重的损失和风险。因此，随着互联网技术的不断进步，人们对于在线服务的要求越来越高，对于外卖平台而言，需要提供更加高效、安全、可靠的服务。[2]

因此，本设计选择开发一个基于Spring Boot和Mybatis-Plus框架的外卖服务平台，旨在探索如何利用现代技术构建高效、稳定和可扩展的外卖服务平台，同时保证商家、骑手和用户的利益，提高外卖行业的数字化转型水平，满足人们的在线订餐需求。

## 1.2 国内外研究现状

### 1.2.1 国外研究现状

国内外卖的供给丰富度和配送效率高于国外外卖。国内外卖不仅可以购买餐饮，还可以购买手机、生活用品等。国内外卖平台已经形成了覆盖全国2800市县的即时配送网络，数百万的骑手小哥每天奔波在路上，为消费者提供快捷便利的服务。而国外外卖平台上几乎无法购买餐饮以外的商品，而且餐饮品类也非常少，可供消费者选择的餐品并不多。除此之外，国外的地广人稀，配送距离长，配送时间长、配送费贵，导致消费者根本没有形成长期使用外卖的消费习惯。且国外外卖平台的佣金普遍超过30%，其中Ubereats的自提订单佣金抽成比例就达到15%。这还不包括支付给骑手的配送费。

在俄罗斯点一份外卖，可选的餐厅很少，外卖菜品比饭店的量少，价格还贵，一般只有大餐馆、连锁店才会推出外送服务，而且金额必须要满七八百卢布才能起送。而在日本，点外卖是一件麻烦事。日本人很重视环保，所以即便是盖饭、乌冬面、寿司这样的外卖，也会用瓷碗或者漆碗盛装，为了安全起见，有些商家甚至会用布袋子装外卖食物。这还是其次，最令人难以接受的是，“顾客吃完后还要将餐具洗干净，放在自己家的门口，等餐馆的人来收餐具”。去日本旅游的国人如是说。尽管欧美国家送外卖小哥以不准时著称，但也有例外，比如德国。众所周知，德国人以严谨著称，在送外卖方面他们同样如此。在德国，配送方式比较多样。德国外卖员使用的交通工具大致可分为如下几种：汽车、自行车或者地铁。不过，外卖小哥更喜欢汽车。由于德国电动车的价格非常高，一台可能要数千欧元，且无论是车辆审核还是办理证件，烦琐程度都不亚于汽车。此外，在德国驾驶电动车还需要汽车驾照，因此外卖小哥常常开汽车送餐 [3] 。

因此国外的外卖服务发展水平远不及国内，在此不做过多讨论。

### 1.2.2 国内研究现状

图一为iiMedia Research的一组研究数据，数据显示，在2012-2022年中国在线外卖市场规模增长率中，2018年中国在线外卖市场规模增长率为55.05%，2019年中国在线外卖市场规模增长率为35.97%，2020年中国在线外卖市场规模增长率为15.00%，2021年中国在线外卖市场规模增长率为18.20%，2022年中国在线外卖市场规模增长率为19.80%。可以看出增长率明显放缓，人们对外卖的态度也从“方便快捷”逐渐转变为了“脏”、“不卫生”等等。究其原因，最关键的是平台对骑手和卖家的佣金设置不合适，导致外卖服务平台乱象丛生。

#### 图1 2012-2022年中国在线外卖市场规模增长率

##### Fig.1 2012-2022 China's online food delivery market growth rate.

互联网外卖平台是将平台、商家与消费者的利益连接，这三家处于同一个利益主体中。其中，平台要设计规则，帮助商家与消费者在平台上交易，进行消费、生产活动；商家是生产的主力军，需要向消费者提供各种食物或服务；消费者是最终消费端，消费商家的产品。所以在这个闭环中，平台的规范性成了最重要环节 [4] 。

# 2 相关知识和开发工具

## 2.1 SSM概述

SSM是由Spring、Spring MVC和MyBatis三个开源框架组成的一种Java开发框架的组合，它是目前非常流行的一种Web应用程序开发技术栈。它可以帮助开发人员实现Web应用程序的快速开发和部署。在应用程序开发过程中，SSM框架组合可以提供高效、稳定、可扩展和易维护等优点。而在本设计中将使用Spring和MyBatis的扩展框架来实现。

Spring Boot是一个快速构建Spring应用程序的框架，它使用了基于约定的方式来简化Spring的配置和开发过程。Spring Boot可以自动配置许多Spring的模块，如Spring MVC、Spring Data、Spring Security等，使得开发者能够快速构建出一个可运行的、高度集成的Web应用程序。同时，Spring Boot还提供了一些其他的功能，如内嵌Web服务器、Actuator监控等，使得应用程序的开发、测试和部署都变得更加简单和高效。[5]

Mybatis-Plus是Mybatis框架的增强工具，它可以帮助开发者更快速地编写Mybatis的CRUD（增删改查）操作。Mybatis-Plus提供了许多便捷的方法和注解，如自动填充、条件构造器、分页插件等，使得开发者可以轻松地进行数据库操作。与原生的Mybatis相比，Mybatis-Plus可以极大地提高开发效率和代码质量。

Spring Boot和Mybatis-Plus的联合使用能够进一步简化开发流程，提高开发效率和代码质量。Spring Boot具备自动配置和集成能力，可以轻松集成Mybatis-Plus，而Mybatis-Plus则提供了许多便捷的方法和注解，可帮助开发者更快地完成数据库操作。同时，Spring Boot和Mybatis-Plus还具有很好的可扩展性，支持方便的功能扩展和自定义配置。因此，它们已成为Java开发中常用的框架之一。

## 2.2 IntelliJ IDEA 2022.1

IntelliJ IDEA是一款由JetBrains公司开发的Java集成开发环境（IDE），是目前最受欢迎的Java开发工具之一。它提供了智能代码编辑、自动代码生成、代码重构、调试和测试等丰富的功能，同时还支持多种语言和框架的开发。就Java的web 应用而言，通常需要选择一款框架，这样会给开发带来便捷，同时应用结构也会比较规范。 回到Java的web框架选择上，目前Java的框架比比皆是，而IDEA对主 流的框架都提供了支持，如Struts、JBoss seam、Spring MVC、Spring Boot等。如 果选择的框架不在其列，扩展也非常简单，只需要添加一写plugin即可实现。[6]

## 2.3 VMware Workstation pro

VMware Workstation Pro是一款基于虚拟化技术的桌面虚拟机软件，可以在同一台计算机上模拟多个操作系统，无需对物理计算机进行重启或分区。它支持多种操作系统，如Windows、Linux、Mac OS X等，并且提供了虚拟化网络、虚拟化存储和虚拟化CPU等功能和工具。VMware Workstation Pro具有可视化的虚拟机管理界面，用户可以方便地创建、启动、停止、暂停、克隆和删除虚拟机，并配置虚拟机的硬件和网络设置等。用户可以在虚拟机中安装软件、测试应用程序、访问互联网、进行网络配置等操作，实现多个操作系统的测试、开发和部署等任务，节省成本和时间。[7]

# 3 系统分析

## 3.1 需求分析

设计外卖服务平台，首先要考虑到用户的功能需求。

用户的需求时多方面的，比如登录、注册、个人信息管理、购物车、订单管理、支付等这些需求为一些基本需求，必须实现，否则就不能称之为一个完整的系统。其次一些高级功能也会直接影响用户体验。例如把地址设置为默认地址，可直接再次下单同款菜品等等。

除了用户，还需要考虑商家的需求。商家的需求包括商家登录、商铺管理、商品管理、订单管理等。在商家修改一些数据时，必须能够同步展示在前台用户端。除上述功能外，系统需要与第三方平台集成，例如支付宝、微信等支付平台。这些第三方平台需要提供短信接口、支付接口等，但这些接口必须匹配到商家才能正常使用，因此在本文将不做详细介绍。

其次还要考虑到一些非功能需求。在点外卖的高峰期，平台必须能够处理大量的并发请求并且能够快速响应用户请求，以保证平台的稳定的同时提高用户体验。当然，还要对用户的个人信息进行保护。在用户下单，支付，配送的过程中，需要采用加密技术，以确保数据不会被非法获取、篡改或丢失。

## 3.2 过程分析

在整个外卖平台运行过程中，将涉及用户，商家和骑手等角色。在此期间，用户填写的地址将保存到数据库中被永久保存。商家关联着用户和骑手，当商家接单后，可以选择“配送”召唤骑手前来取餐。但是商家并没有取消订单的权限，当存在订单疑惑时，可以根据用户留下的联系方式与用户沟通协调。具体过程如图2所示。

#### 图2 外卖服务平台的过程分析流程图

##### Fig.2 Process analysis flowchart of takeout service platform.

## 3.3 环境分析

对于外部环境而言，尽管食品监管局的管控力度大，投入资金多，但是商家却并不原意在这方面投入任何成本。因此，政府部门的管控作用就显得尤其重要，确保食品安全能够真正落实到每一个人手中。而对于外卖行业而言，很多时候会出现“见利忘义”的行径。商家之间恶意竞争，未达目的不择手段。甚至会出现未拿到《食品经营许可证》就上网经营的例子。这样的市场对于消费者而言是无利的，也是无力的。因此，需要平台在期间做好管控，加强对商家的考察。[8]

## 3.4 成本效益分析

在传统外卖行业中，部分平台商家设置了“最低起送费”，导致了消费者可能需要加购一些不需要的商品。商家以此来提高利润。究其缘由，是外卖平台对商家的抽成导致，进而对消费者利益造成损害。因此，应该尽可能降低成本以保护商家和用户的权益。[9]

外卖服务平台的运营需要投入的成本主要包括技术开发成本、营销和推广成本以及运营成本。为了保持平台的顺利运营和不断发展，首先就要考虑平台的服务器租用，选择高性价比服务器才能尽可能降低成本。

同时，也要注重前期的推广和宣传。这将有助于吸引更多商家入驻本外卖服务平台，从而增加平台的营业收入。

# 4 系统设计

## 4.1 系统架构设计

首先，客户端是平台的用户接口，提供用户注册、登录、下单、支付等功能。服务器则承担平台业务逻辑处理和数据存储，包括tomcat服务器、Nginx服务器、MySQL服务器和Redis服务器。不过在本设计中为了节约成本，将MySQL的主服务器和从服务器分别整合在tomcat服务器和Nginx服务器中。因此，Nginx服务器除了接收客户端请求以外，还需要对MySQL数据库进行写操作。而tomcat服务器中的MySQL则只需要日常的读操作即可。

外卖服务平台还需要重视系统的安全，其中id部分可以使用MyBatis Plus自带的雪花算法进行加密，密码部分采取md5进行数据加密。这样就能大致保证用户数据的安全性。

## 4.2 数据库设计

在数据库模型设计中包含多个实体，其中包括用户、地址、菜品、菜品分类、菜品口味、订单、订单明细、员工、套餐和购物车信息。而对于每个实体，还需要考虑实体之间的关系，如用户和地址之间是一对多的关系，订单和订单明细之间也是一对多的关系。但菜品和菜品口味表是多对多的关系。在建立关系时，还需要确定关系的类型和级联操作，以保证数据的完整性和一致性。

在外卖服务平台的数据库逻辑设计中，将模型设计中的实体转换为关系表，而在每个关系表中，需要定义该表的属性和约束。例如，在用户表中，可以定义用户的ID、用户名、密码、电话和性别等属性，其中用户ID必须定义为唯一约束且非空。此外，还需要定义表之间的约束和操作，以确保数据的完整性和一致性。例如，在用户和地址之间的关系中，用户实体为父实体，地址实体为子实体，在删除用户时，应该将地址一起删除。具体设计如表1所示。

#### 表1 数据库设计

##### Table.1 Database Design

|  |  |  |  |
| --- | --- | --- | --- |
| 实体名称 | 实体属性 | 数据类型 | 约束 |
| 用户 | user\_id | INT | 主键，非空 |
|  | username | VARCHAR(50) | 非空 |
|  | password | VARCHAR(50) | 非空 |
|  | phone | VARCHAR(20) | 非空 |
| 地址 | address\_id | INT | 主键，非空 |
|  | user\_id | INT | 外键，非空 |
|  | province | VARCHAR(20) | 非空 |
|  | city | VARCHAR(20) | 非空 |
|  | district | VARCHAR(20) | 非空 |
|  | detail | VARCHAR(100) | 非空 |
| 菜品分类 | category\_id | INT | 主键，非空 |
|  | name | VARCHAR(20) | 非空 |
| 菜品口味 | taste\_id | INT | 主键，非空 |
|  | name | VARCHAR(20) | 非空 |
| 菜品 | dish\_id | INT | 主键，非空 |
|  | name | VARCHAR(50) | 非空 |
|  | price | DECIMAL(10,2) | 非空 |
|  | picture | VARCHAR(100) | 非空 |
|  | category\_id | INT | 外键，非空 |
|  | taste\_id | INT | 外键，非空 |
| 订单 | order\_id | INT | 主键，非空 |
|  | user\_id | INT | 外键，非空 |
|  | order\_time | DATETIME | 非空 |
|  | total\_price | DECIMAL(10,2) | 非空 |
| 订单明细 | detail\_id | INT | 主键，非空 |
|  | order\_id | INT | 外键，非空 |
|  | dish\_id | INT | 外键，非空 |
| 员工 | employee\_id | INT | 主键，非空 |
|  | name | VARCHAR(20) | 非空 |
|  | gender | VARCHAR(10) | 非空 |
|  | phone | VARCHAR(20) | 非空 |
| 套餐 | package\_id | INT | 主键，非空 |
|  | name | VARCHAR(20) | 非空 |
|  | description | VARCHAR(100) | 非空 |
|  | price | DECIMAL(10,2) | 非空 |
| 套餐详细 | detail\_id | INT | 主键，非空 |
|  | package\_id | INT | 外键，非空 |
|  | dish\_id | INT | 外键，非空 |
| 购物车 | cart\_id | INT | 主键，非空 |
|  | user\_id | INT | 外键，非空 |
|  | dish\_id | INT | 外键，非空 |
|  | quantity | INT | 非空 |

注：此表项中的表属性并不是全部属性

最后，需要考虑到外卖服务平台数据库的物理设计。大致来说，需要使用MySQL主从复制架构和Redis服务器来实现高可用性和性能需求。因此，数据库物理设计需要考虑如下几个方面：

在MySQL主从复制架构中，必须将主库和从库分布在不同的服务器上，以实现分布式的数据存储和读写请求处理。同时设置MySQL主从同步机制，以确保数据同步。

随着互联网web2.0网站的兴起，传统的关系数据库在应付web2.0网站，特别是超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库redis则由于其本身的特点得到了非常迅速的发展。而在本设计中，需要将缓存数据和一些必要的即时信息如手机验证码等存入redis。[11]

为实现高性能的数据库访问和数据处理，需要制定数据库性能优化策略。可以使用MySQL和Redis提供的性能监控和优化工具来实时监测数据库的性能指标，并针对性地进行优化和调整。

## 4.3 接口设计

首先，最重要的时登录注册这一接口设计。这一部分内容主要包含员工登录接口、员工信息展示接口、新增员工接口以及用户登录/注册接口。其中，员工登录接口需要接收HttpServletRequest对象和Employee实体作为请求参数，并返回员工ID、密码和状态作为响应数据；员工信息展示接口需要接收当前页数、每一页大小和员工姓名作为请求参数，并返回符合条件的员工信息；新增员工接口需要接收Employee实体作为请求参数，并返回新增成功的提示信息；用户登录/注册接口需要接收存储用户实体的map集合和session对象作为请求参数，并返回手机号和验证码作为响应数据。

在套餐管理设计接口时，新增套餐可通过发送POST请求到/setmeal，请求参数为套餐实体类，其中包含套餐名称、套餐价格、图片等信息。套餐分页查询可通过发送GET请求到/setmeal/page，请求参数包括当前页数、每一页大小以及套餐名字。删除套餐可通过发送DELETE请求到/setmeal，请求参数为套餐ID。根据条件查询套餐数据可通过发送GET请求到/setmeal/list，请求参数包括当前页数page、每一页大小pageSize以及套餐名字name。启售或停售套餐可通过发送POST请求到/setmeal/0，请求参数为套餐ID，响应数据为套餐信息。若操作成功，则响应数据为“新增/删除套餐成功”等。

订单管理的接口设计如下。当用户下单时，需要向服务器发送POST请求，使用接口为/order/submit，并传递订单实体类作为请求参数，包含订单号和订单状态（待付款、待派送、已派送、已完成、已取消）。如果下单成功，服务器会返回“下单成功”的响应数据。用户可以通过向服务器发送GET请求，使用/orderDetail接口获取订单明细，请求参数为无。服务器将返回外卖名称、订单ID、菜品ID、套餐ID、口味、数量、金额、图片等相关信息作为响应数据。

当然，本设计中涉及大量系统接口，因篇幅有限不能做一一展示，其他接口涉及方法与上面三种大致相同。用户与员工信息管理的相关接口信息如表12所示。

#### 表2 外卖服务平台部分接口设计

##### Table.2 part of the interface design of the takeout service platform

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 请求方法 | 请求路径 | 请求参数 | 响应数据 |
| 员工登录 | POST | /employee/login | HttpServletRequest对象、员工实体 | 员工ID、员工密码、员工状态 |
| 员工信息展示 | GET | /employee/page | 当前页数、每一页大小、员工姓名 | 员工信息 |
| 新增员工 | POST | /employee/save | 员工实体 | “新增员工成功” |
| 用户登录/注册 | POST | /user/login | 存储用户实体的集合、session对象 | 手机号、验证码 |
| 新增套餐 | POST | /setmeal | 套餐实体类 | "新增套餐成功" |
| 套餐分页查询 | GET | /setmeal/page | 当前页数、每一页大小、套餐名称 | 套餐分页查询结果 |
| 删除套餐 | DELETE | /setmeal | 套餐ID | "删除套餐成功" |
| 搜索套餐 | GET | /setmeal/list | 当前页数、每一页大小、套餐名称 | 符合条件的套餐数据 |
| 启售/停售套餐 | POST | /setmeal/0 | 套餐ID | 启售或停售套餐的信息 |

## 4.4 系统性能设计

### 4.4.1 硬件配置

为了支持高并发的写操作和读操作，以及快速响应和处理高并发请求，平台需要配置不同的服务器。

主数据库服务器需要至少8核CPU，32GB以上内存，1TB以上SSD硬盘，双路电源供电，支持RAID，以应对高并发的写操作；从数据库服务器需要至少4核CPU，16GB以上内存，500GB以上SSD硬盘，双路电源供电，支持RAID，以支持高并发的读操作。

负载均衡服务器需要至少4核CPU，16GB以上内存，双路电源供电，支持高速网络接口，支持负载均衡软件（如Nginx、HAProxy等），以处理高流量的请求并快速分发请求。

应用程序服务器需要至少4核CPU，8GB以上内存，500GB以上SSD硬盘，双路电源供电，支持RAID，支持应用程序框架（如Spring、Django、Ruby on Rails等），以保证应用程序的快速响应和高并发处理。这样的配置可以提高系统的性能和可靠性，同时满足平台对服务器处理能力、内存、磁盘速度、网络带宽等方面的需求。

### 4.4.2 软件配置

为了确保平台能够处理高并发请求，可以采用Nginx作为负载均衡器，以快速、准确地分发请求，并保持应用服务器的负载尽可能平均。它是一款性能强大的Web和反向代理服务器，具有多个优势，包括对网络依赖性较小、安装和配置简单、能够承受高负载压力并保持稳定性等。Nginx的最大特点之一是其负载均衡技术，当服务器接收到具体的服务请求时，它可以根据预设的策略将请求转发到不同的服务器，以防止某台服务器过载而导致其他服务器闲置的情况发生。[10]

其次，可以采用数据库分库分表的方式，将数据分散到多个数据库实例上，避免单个数据库出现性能瓶颈。此外，建立合理的索引可以提高查询效率，减少全表扫描的时间，编写高效的SQL语句也可以减少数据库的查询时间，进一步提高查询效率。除此之外，增加数据库内存缓存可以提高查询效率。另外，将读操作和写操作分离，采用读写分离的方式，使读操作的负载均衡到多个从库上，减轻主库压力，提高系统的性能。通过这些优化措施的组合应用，可以提高数据库的性能和可扩展性，应对高并发的情况。

缓存技术是提高系统性能的重要手段之一。其中，缓存热点数据可以有效减轻数据库的访问压力，提高系统的响应速度。将手机验证码等短期有效的数据放入Redis缓存中，减少数据库的访问次数。其次，采用spring cache技术，将套餐数据和菜品数据存入缓存中，用户只需要加载一次数据，后续访问时可以大大提高访问速度。例如加入如下注解：@CacheEvict(value = "setmealCache",allEntries = true)，当新增套餐时，可以删除缓存数据。

## 4.5 用户界面设计

### 4.5.1 后台管理界面

为了提高后台管理人员的体验和操作效率，一个良好的后台界面设计是非常重要的。一个常用的设计方案是采用侧边栏菜单和顶部栏来组织主要功能模块，同时在顶部栏显示当前操作的模块名和登录用户信息，方便用户快速找到所需功能。

具体来说，后台功能模块分为用户管理、菜品管理、套餐管理、订单管理等。每个功能模块大致相似，有展示信息、添加、删除、搜索功能，这些功能为一些后台管理界面的基本功能。例如菜品管理界面，在打开时直接进入菜品展示列表，然后可以根据列表信息进行修改菜品，删除菜品和停售菜品，在右上角可以选择添加菜品。在删除菜品时，必须先停售菜品。修改菜品页面应该获取到菜品原来信息并自动填入，然后进行选择性修改。添加菜品设有必填项和选填项，在选择口味信息时设有复选框和下拉菜单。

在订单管理页面上方设有搜索栏，包含一个输入框和一个日期选择器，用于搜索订单。在搜索栏下面展示订单信息，包含订单号、订单状态、用户、手机号、地址、下单时间、实收金额、操作等列，其中操作列包含查看、派送、完成等按钮。当点击查看时，可以展示订单详细信息。最后，也可以利用MyBatis的分页管理达到翻页的效果，提供更好的信息浏览和操作体验。

其他几个功能模块与上述模块大差不差，这里不做过多赘述。当然，除了以上设计方案，根据实际需求还可以在管理界面添加商家logo，在后台管理界面的登录窗口添加特色菜品图片。

### 4.5.2 用户界面设计

设计一个完整的用户界面需要考虑到用户的整个购物流程，包括商家详情页设计、购物车设计和个人中心设计。

商家详情页设计包括商家的一些基本信息、商品列表、侧边栏、商家公告和底部栏。商家基本信息包括商家图片、名称、销售量、距离、配送费、预计时长等信息，这些信息应该放在整个页面的最上方。侧边栏用于展示菜品分类，菜品列表应该根据菜品分类展示各类菜品的图片、名称、价格、月售量等信息，并支持选择规格并添加到购物车。底部栏主要是购物车功能，将选择的菜品进行结算，最终提交到订单页面。

购物车设计内嵌在首页中，在未点击时，在下方只展示已添加到购物车的菜品数量。当点击购物车时，购物车界面弹出，展示刚才已选的商品。在购物车可以进行添加或减少某一菜品数量，也可以直接全部清除。底部栏设有“去支付”接口，当点击支付时，会跳转到提交订单页面。当订单提交过来时，若未设置过地址信息或默认地址信息，在地址栏将不会显示地址。滑动到页面底部，将有“备注”栏方便用户与商家对菜品达成协商。

在首页点击个人头像，可以进入个人中心。个人中心包括个人信息、收货地址信息、订单信息。订单信息展示用户的最新订单、历史订单等。个人信息展示用户的头像、昵称、联系方式等信息，并支持修改个人信息。收货地址展示用户的收货地址列表，支持添加、修改和删除地址。底部栏设有退出登录功能。

以上几个部分是外卖服务平台的重要组成部分，细致的设计能够提高用户体验，提高系统的可用性和稳定性。

# 5 系统实现

## 5.1 环境配置

### 5.1.1 IntelliJ IDEA

创建一个名为“takeout\_service”的项目，使用maven构建。为了满足项目需求，选择JDK1.8.0\_291版本和Spring Boot版本3.0.5。在此基础上，需要添加Spring Web和jdbc API工具。

在项目创建完成后，需要设置pom.xml文件使得项目具备相应的功能和特性。首先，添加mybatis-plus依赖和spring boot核心依赖以保证框架的正常使用。为了简化代码并提高开发效率，引入Lombok库。最后，添加一些数据库的必要依赖，比如JDBC和Druid数据库连接池等。

为了实现这些功能，需要在application.yml文档中添加相应的配置信息。例如，可以使用spring.datasource配置节来指定数据库连接信息，使用server.port配置节来设置服务器端口号，使用redis配置节来配置Redis连接信息，使用mybatis-plus配置节来配置MyBatis使得应用程序能够与MySQL数据库进行交互等。配置完成后，保存并关闭文件。

最后，在编辑SpringBoot启动类时，需要添加一些注解以支持项目的正常运行。首先，需要添加@ComponentScan注解，以便扫描驱动。其次，需要添加@EnableTransactionManagement注解，以便在代码中使用事务管理。最后，添加@Log4j2注解，以便使用Lombok库提供的日志功能。这些注解的添加可以在启动类上方进行，以确保项目可以正常启动。

### 5.1.2 服务器配置

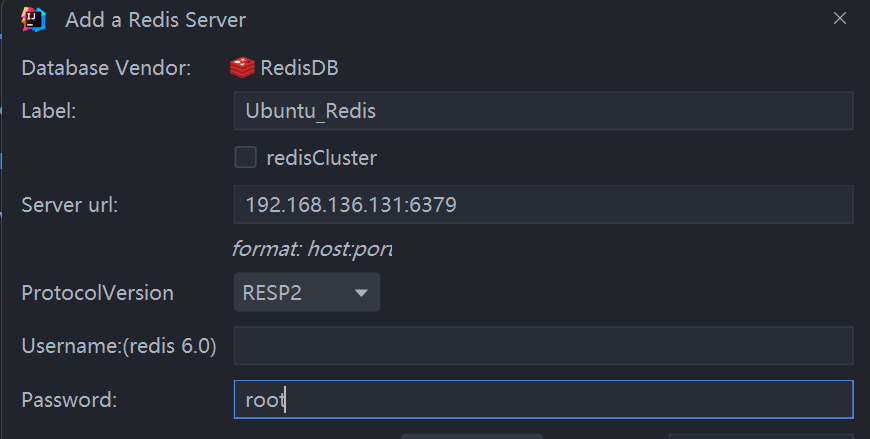
首先，需要在服务器安装JDK。从官网下载JDK1.8.0\_371的安装包，然后使用命令将其解压到/usr/local/目录下。接着需要配置环境变量，可以通过编辑/etc/profile文件并在文件末尾添加Java相关的配置信息来实现。最后，通过输入Java -version命令来检查是否成功安装JDK。

第二，需要在服务器安装Tomcat。先从官网下载Tomcat 8.5.88版本，然后使用命令行解压到指定目录。接着，在命令行中输入"./bin/startup.sh" 启动Tomcat，在启动成功的信息中可以看到Tomcat的基本信息，例如CATALINA\_BASE、CATALINA\_HOME等。

第三，需要在服务器上配置mysql。首先，使用命令行工具安装mysql-server。其次，登录mysql，并进行必要的配置。最后，可以在mysql中配置主从复制功能，以实现数据备份和故障转移。具体的配置过程将在后续的文本中详细介绍。

第四，在服务器中配置Nginx。直接使用命令"sudo apt-get install nginx"进行安装。安装完成后，可以使用"sudo systemctl start nginx"来启动Nginx。最后，在服务器上开放80端口，可以使用"sudo ufw allow 80/tcp"命令来添加。

第五，安装redis。使用命令sudo apt-get install -y redis-server，安装完成后需要进行密码设置和远程登录设置，打开/etc/redis/redis.conf文件进行修改。可以设置requirepass选项来设置密码，设置bind选项为0.0.0.0可以允许远程登录，同时关闭protected-mode选项以允许远程登录。最后，使用redis-server /etc/redis/redis.conf命令重启服务并使用配置文件。也可以用IntelliJ IDEA连接到Ubuntu的redis服务器，可以在IDEA中安装redis插件，并在连接配置中设置相应的IP地址和密码。如图2所示。



#### 图2 IDEA中Redis插件的配置

##### Fig.2 Configuring Redis Plugin in IDEA

最后，安装git。通过命令“sudo apt-get install git”进行安装。接着，需要生成SSH密钥，可以通过命令“ssh-keygen -t rsa -b 4096 -C "248107148@qq.com"”生成。生成后，通过命令“cat ~/.ssh/id\_rsa.pub”查看公钥，然后将公钥添加到GitHub账户中。具体操作是复制公钥，然后在GitHub网站上打开账户设置页面。选择“SSH and GPG keys”，然后点击“New SSH Key”。在“Key”字段中粘贴复制的公钥，并为该密钥提供一个描述性名称。最后，点击“Add SSH Key”按钮即可。最后还需要配置Git全局变量，包括用户名和邮箱，可以通过命令“git config --global user.name "Ray"”和“git config --global user.email "248107148@qq.com"”进行配置。

## 5.2 模块开发

### 5.2.1 登录模块

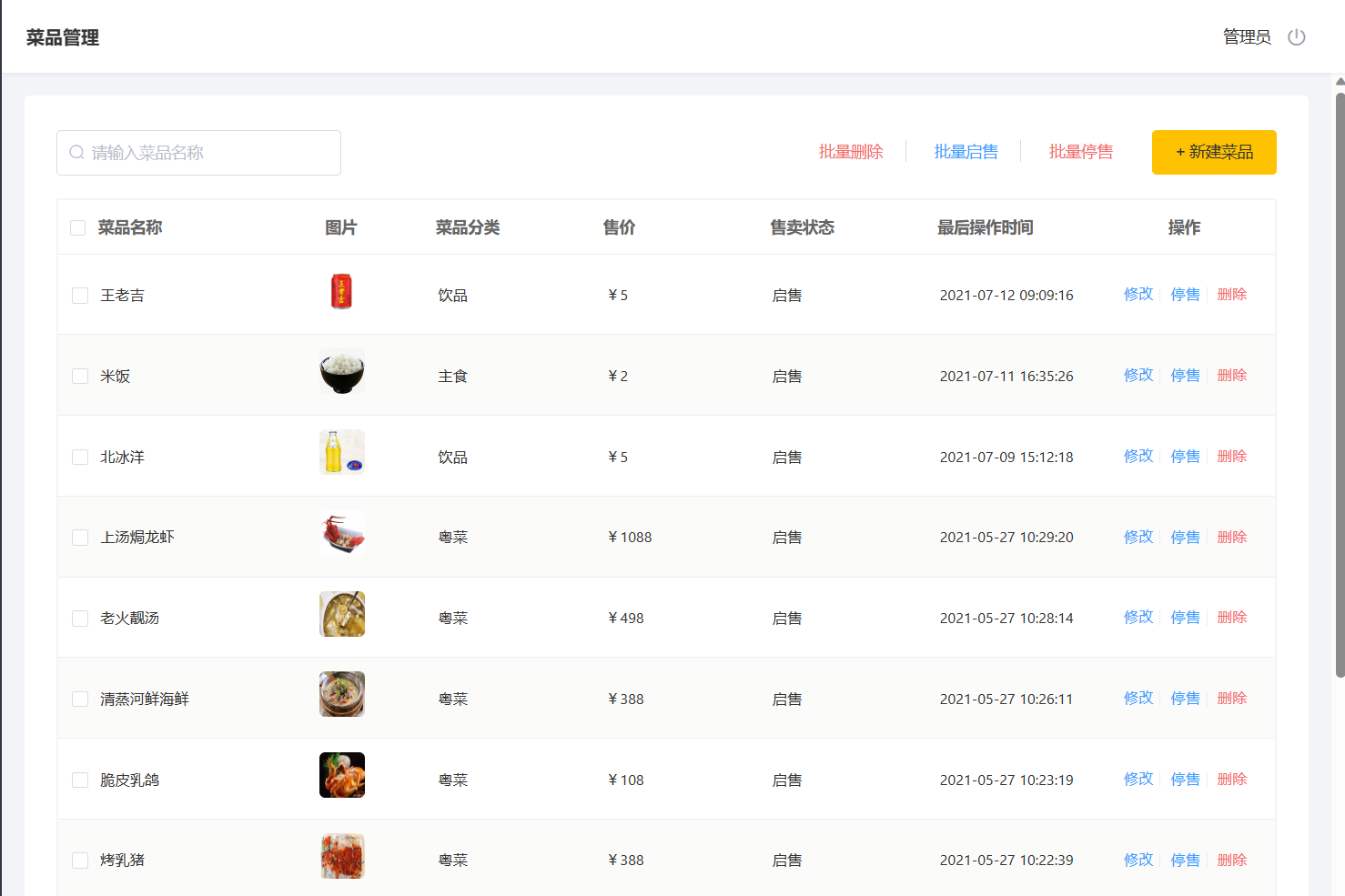
在创建登录页面时，使用HTML、CSS和JavaScript等前端技术。首先创建一个HTML页面，在该页面中添加用户名和密码的输入框，并且在底部放置一个登录按钮。然后使用CSS样式美化页面，并使用JavaScript编写交互逻辑，比如验证用户名和密码是否符合要求，或者对登录按钮添加点击事件等。登录按钮绑定一个函数，通过AJAX或者其他方式向后端发送请求，进行身份验证，根据身份验证的结果，进行页面跳转或者提示用户登录失败等信息。

然后编辑后端代码逻辑。创建一个Controller，用于处理登录请求和返回登录页面。代码要首先验证用户名和密码是否正确，并生成登录token。再将生成的登录token存储在Redis缓存中。然后从Redis缓存中获取存储在该手机号码下的验证码，将其与请求中的验证码进行比较。如果验证码匹配，则从数据库中查询该手机号对应的用户信息，如果不存在，则创建一个新的用户。最后将该用户的ID存储在HttpSession中，并将其返回给客户端。当然，如果验证码不匹配，则返回一个包含"登录失败"的错误响应消息。

再创建一个DAO类，用于操作数据库，验证用户名和密码是否匹配。接着需要在项目中配置数据源，以便连接到数据库。然后需要配置MyBatis-Plus，以便更方便地进行数据库操作。最后需要在Spring MVC中配置登录拦截器，在请求到达时进行拦截和处理。首先，将ServletRequest和ServletResponse转换为HttpServletRequest和HttpServletResponse。然后，获取当前请求的URI，检查是否需要处理这个请求。然后定义一些请求不需要处理，将这些请求的URI存储在一个数组中。接着，如果需要处理这个请求，检查当前会话是否已经存在“employee”或“user”属性。如果存在，将属性值转换为Long类型，并将其设置为BaseContext的当前ID。最后，如果没有登录，将返回一个JSON格式的错误响应。最后，如果用户注销，则在用户注销时，要删除存储在Redis中的登录token。

### 5.2.2 菜品管理模块

前端展示菜品列表页面，包含搜索、排序、筛选、分页等功能，同时提供新增、修改、删除、批量操作等按钮，通过与后端接口交互，实现对菜品数据的增删改查。页面使用Element UI组件库实现样式和交互效果，并引入了axios库实现与后端接口的数据交互。在添加菜品页面，以在该表单中输入菜品名称、菜品分类、菜品价格和口味做法配置等信息。其中，菜品分类和口味做法配置使用了下拉列表和多选框的形式进行输入。实现效果图如图3所示。



#### 图3 菜品管理页面

##### Fig.3 Dish management page.

商家可以在菜品管理模块中添加、修改和删除菜品。这些数据库查询方法全部通过DishMapper继承Mybatis-plus中的BaseMapper获得。BaseMapper是MyBatis-Plus提供的一个通用Mapper接口，它封装了一些常用的数据库操作方法，如插入、查询、更新和删除等。要使用它时，先用注解@Mapper来标注这个接口是MyBatis框架需要的Mapper接口。然后指定它需要泛型类型参数为接口所对应的实体类Dish。

### 5.2.3 订单管理模块

在前端页面中使用Element UI的组件库，通过导入CSS文件实现样式的渲染。整个页面的交互逻辑是基于Vue.js框架实现的，通过调用相关的Vue组件和方法实现页面的响应式更新和交互操作。

在订单管理模块中的后端部分，首先在数据库中创建订单表，包括订单编号、订单状态、下单时间、支付时间、支付金额、配送地址、收货人等字段。在后端代码中创建Order实体类与数据库表对应，定义相关属性和方法。在Service层创建OrderService接口并继承MyBatis中的IService类，并在其实现方法ServiceImpl中实现业务逻辑方法。

在实现方法前，要使用@Transactional注解来实现事务处理。在订单分页查询时，首先获取当前用户id，然后根据该id查询用户购物车数据。如果购物车数据为空，则抛出CustomException异常。接着查询用户数据和地址数据，生成一个唯一的订单号。生成订单号后，要遍历购物车数据并把这个数据写入到订单数据中。最后，将订单详情数据批量保存到订单明细表中，并清空购物车数据。这个方法的整个过程都被包含在一个事务中，因此如果在任何一个步骤中出现错误，整个操作都将回滚。

最后，在Controller层创建OrderController类来处理前端请求，然后调用Service层方法进行业务逻辑处理。

### 5.2.4 购物车管理模块

购物车管理模块分为添加购物车，查看购物车，清空购物车等方法。外卖服务平台的示例图如图4所示。



#### 图4 外卖服务平台购物车界面

##### Fig.4 The shopping cart interface of the takeout service

添加购物车时，通过调用BaseContext类的getCurrentId方法来获取当前用户的ID，并将其设置为ShoppingCart对象的用户ID属性。接下来，使用LambdaQueryWrapper类创建一个查询对象，用于查询购物车中是否已经存在特定的菜品或套餐。如果购物车中已经存在相同的菜品或套餐，则将现有的购物车数据的数量加1。否则，将新的购物车数据插入到购物车表中，并将其数量设置为1。

查看购物车的代码逻辑比较简单。通过LambdaQueryWrapper构建查询条件，只查询当前用户的购物车数据，并按照创建时间进行升序排序。最后将查询结果返回即可。

### 5.2.5 地址管理模块

在地址管理模块中，用户可以通过添加、修改和删除地址来管理其地址信息。这些方法实现与购物车的增删改查相似，不做过多赘述。

在购物途中，把地址设置为默认地址可以极大的改善用户体验。当用户修改地址时，首先根据用户ID将该用户所有地址的默认标记设为0，然后再将指定的地址的默认标记设为1，从而确保用户只有一个默认地址。然后通过Lambda表达式构建一个UpdateWrapper来设置地址的默认标记，并使用AddressBookService中的update和updateById方法将数据更新到数据库中。

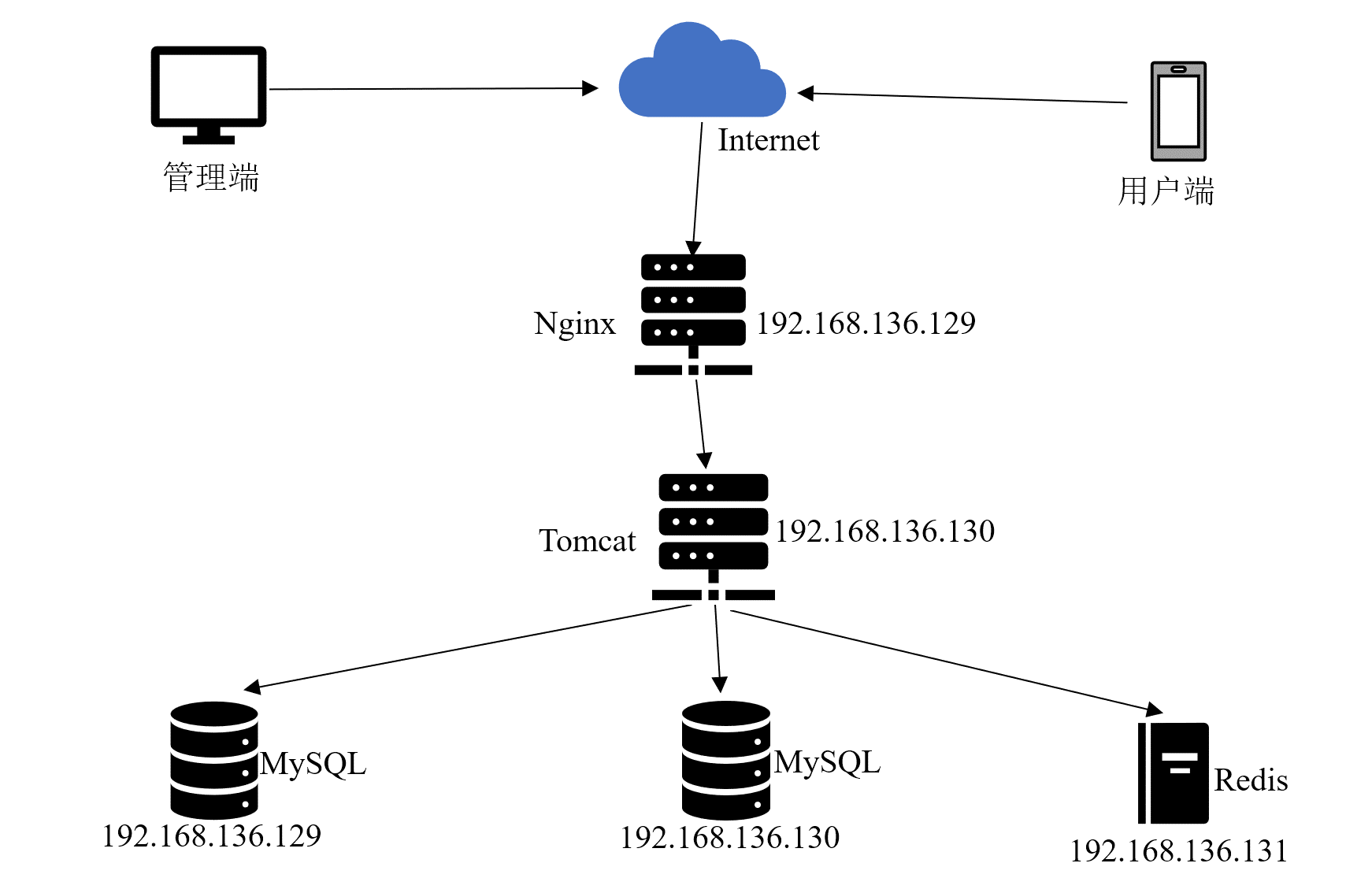
## 5.3 项目部署

### 5.3.1 部署架构

在管理端管理后台菜品数据的过程中，管理员通过后台管理系统来添加、删除、修改、查询菜品数据。后台管理系统由前端页面和后端API组成。前端页面主要负责菜品数据的展示和操作界面的呈现，而后端API则负责与数据库进行交互，实现增删改查等操作。

用户可以通过互联网在线点餐。在用户打开网页时，前端代码会通过Nginx服务器向后端服务器发送请求，请求数据。Nginx服务器主要负责反向代理、负载均衡和静态资源缓存等功能，将用户请求转发给Tomcat服务器。Tomcat服务器接收到用户的请求后，会根据请求路径调用相应的后端API接口，访问主从数据库和Redis获取数据，进行业务处理后返回结果给前端。

因此，整个在线点餐系统的架构通常包括前端页面、Nginx服务器、Tomcat服务器、后端API接口、主从数据库和Redis缓存等组件。如图5所示。



#### 图5 服务器架构图

##### Fig.5 Server architecture diagram

### 5.3.2 部署项目到服务器

首先，部署前端代码到Nginx服务器。将本地编写好的代码上传到GitHub，然后服务器使用git clone命令克隆项目。克隆完成后，修改nginx的配置文件，以便能够访问到该项目的静态资源。这样做的话，之后只需在本地浏览器中输入服务器的IP地址，即可访问到该项目。然后运行setup 命令，在每次系统启动时都会自动启动 Nginx 服务。[12]

其次，部署Tomcat服务器与后端项目。同样先克隆已上传到GitHub的项目到服务器，然后编写脚本文件。停止任何正在运行的应用程序实例，使用ps命令查找进程ID，并使用kill命令杀死它。然后更改到/root/takeout\_service目录，准备从Git仓库拉取最新的代码。接着执行git pull命令，从Git仓库拉取最新的代码。再执行mvn clean package命令打包代码。最后启动应用程序，并将输出追加到nohup.out文件。脚本运行图如图6所示。

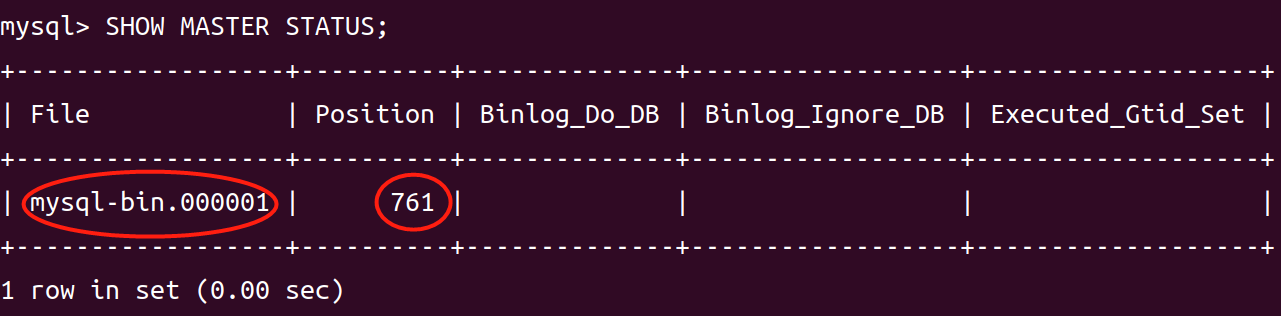


#### 图6 项目运行脚本示例

##### Fig.6 Example of Project Running Script

当部署MySQL数据库的主从复制时，首先在application.yml中定义两个数据源的信息，分别为主数据源和从数据源。主数据源和从数据源分别指向不同的MySQL数据库，并使用相同的用户名和密码进行连接。其中主数据源配置了类型、驱动、URL、用户名和密码等信息；从数据源同样也配置了这些信息。另外，该配置文件还要定义一个读写分离的配置，指定了使用轮询算法来负载均衡数据源，同时也指定最终数据源的名称、主库数据源的名称和从库数据源的名称列表。总体来说，复制分为三步：

第一步，Master主节点改变记录到二进制日志中。首先打开mysqld.cnf文件，并在文末添加log-bin=mysql-bin以启用二进制日志功能，并设置server-id。接着重启MySQL服务。在MySQL中创建一个用于复制的账户'leidongchen'@'%'，并分配replication slave权限。最后，使用SHOW MASTER STATUS命令获取主服务器Master的二进制信息。如图7所示。



#### 图7 主服务器数据库的二进制信息

##### Fig.7 The binary information of the database on the master server.

第二步，Slave将Master的二进制日志事件拷贝到它的中继日志。具体来说，使用CHANGE MASTER TO命令设置主服务器的IP地址、复制账户信息以及二进制日志的文件名和位置。

第三步，Slave重新执行中继日志中的事件，将改变完全应用到它自己的数据库中。使用START SLAVE命令启动从服务器的复制进程。最后输入ip地址和资源目录，查看是否可以正常展示数据库中的数据。[13]

最后，部署redis数据库。首先在application.yml中需要在redis块中添加如下配置：host: 192.168.136.131，port: 6379，password: root，database: 0。然后到redis服务器中安装redis并启动。打开配置文件redis.conf，修改“bind 127.0.0.1”为“bind 0.0.0.0”。以允许从任何IP地址访问Redis服务器。

最后找到前台的登录界面，输入手机号码，再点击获取验证码，那么在redis服务器中找到验证码信息并填入。登录成功后可以看到已经成功进入到前台点餐页面。

# 6 系统测试

## 6.1 测试目标和策略

软件测试是软件生命周期中的一个重要阶段，是软件质量保证的关键步骤。[10]

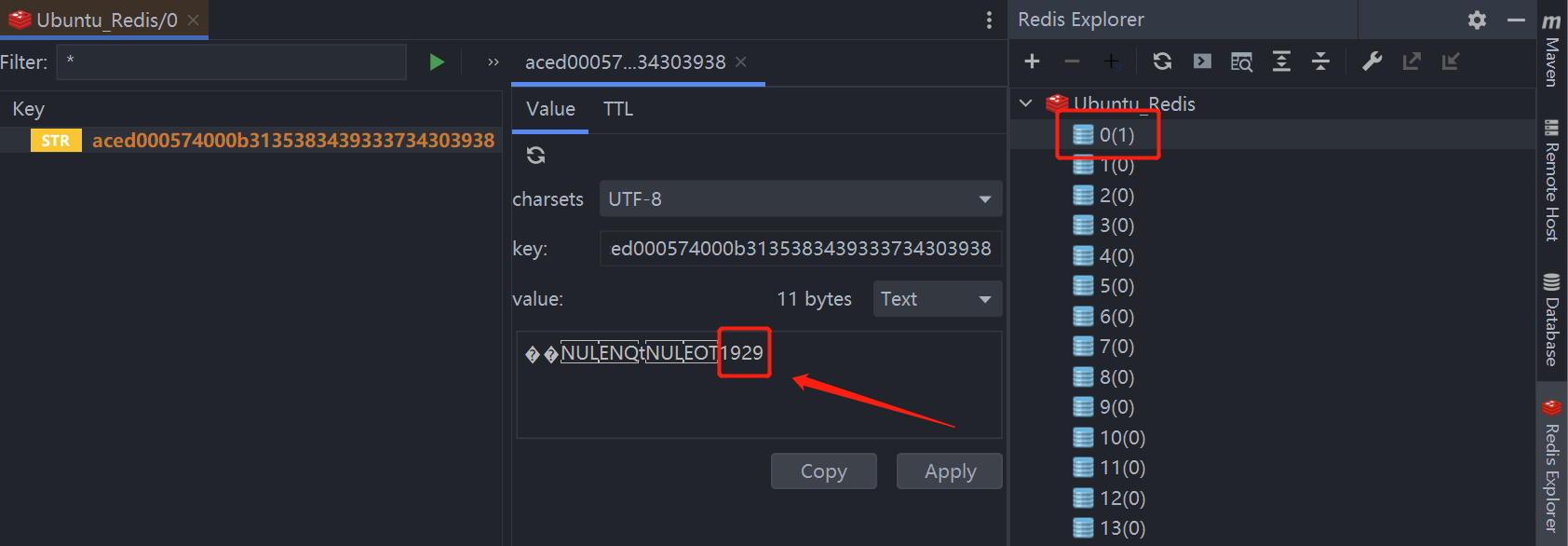
为了保证系统的质量和稳定性，采用黑盒测试方法，从用户的角度出发对系统进行测试，关注系统对用户的响应和行为是否正确，而不关心系统内部的实现细节。针对每个功能点设计测试用例，包括正常情况和异常情况，例如输入格式不正确、网络异常等情况，以覆盖所有可能的路径和情况，尽可能发现潜在的缺陷。为提高测试效率和准确性，采用自动化测试工具和手动测试相结合的方法。利用 JUnit、Selenium 等自动化测试工具进行功能测试和性能测试，利用人工测试进行可靠性和安全性测试。在测试过程中，重视数据的准备和清理工作，确保测试数据的完整性和一致性，避免测试数据对测试结果的影响。还需要关注系统的日志和监控信息，在测试过程中及时发现和记录系统的异常和错误信息，对问题进行定位和跟踪，及时进行处理。通过上述测试方法和策略，能够全面发现和解决系统的潜在问题，确保外卖服务平台的稳定和可靠运行。[11][12]

## 6.2 执行测试

本测试于2023年5月2日执行，测试环境为Windows 11、Ubuntu22.04 和Microsoft Edge浏览器，设计覆盖系统所有功能和各种情况的测试用例，包括正常情况和异常情况。其中重点测试内容为登录功能，菜品管理功能和下单功能。

### 6.2.1 登录功能测试

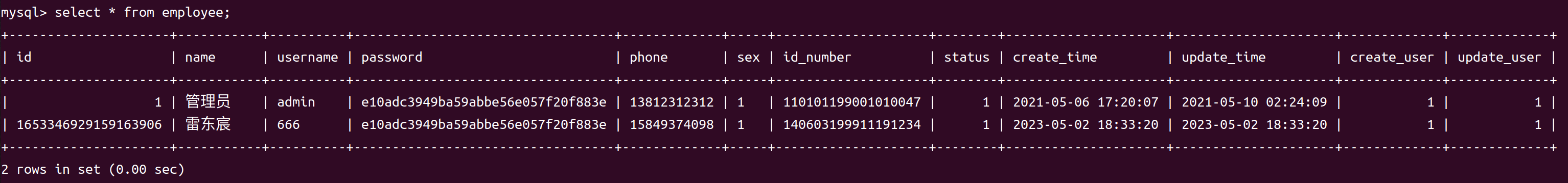
首先测试用户登录功能，用户登录成会直接返回点餐页面。在Edge浏览器地址栏输入192.168.136.129/front/page/login.html，输入手机号码，再点击“获取验证码”。查看redis服务器中是否已经存储了验证码。然后点击登录，查看是否可以正常登录。Redis服务器的存储信息如图8所示。



#### 图8 redis服务器存储的手机验证码

##### Fig.8 The mobile verification code stored on the Redis server.

在测试后台登录功能，登录成功后会跳转后台管理平台的页面。在Edge浏览器地址栏输入192.168.136.129/front/page/login/login.html，输入管理人员账号（数据库中默认已有admin）和密码，再点击“登录”。在员工管理选项卡中点击“添加员工”，输入账号、员工姓名、手机号、性别和身份证号完成添加。再次跳转到员工管理界面，可以看到员工信息已经成功添加。查看数据库，发现已经成功添加了一条数据，如图9所示。密码雪花算法加密也没有问题。退出登录，用新添加的员工账号和密码登录发现可以登录成功。



#### 图9 数据库员工表信息

##### Fig.9 Employee table information in the database.

### 6.2.2 菜品管理功能测试

该测试包含功能测试，可靠性测试，安全性测试等。当后台管理人员对菜品进行增加，删除或者停售时，查看前台用户端是否能即使看到更改内容。具体测试结果如表3所示。

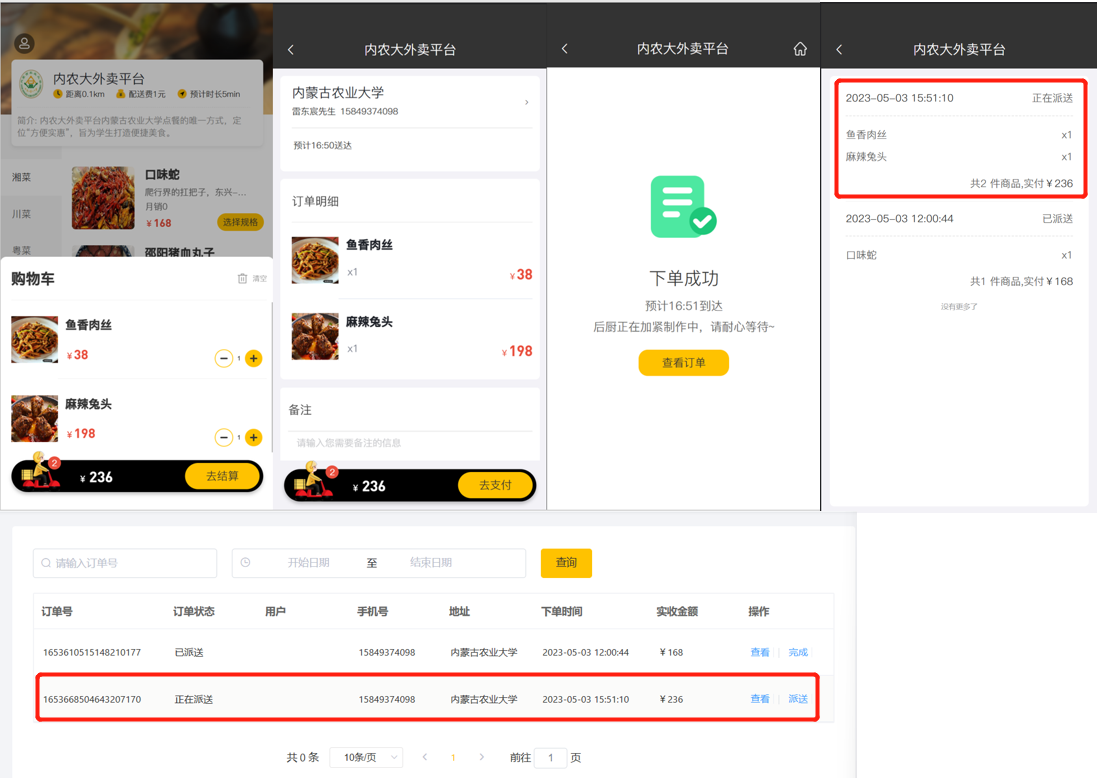
#### 表3 菜品管理功能黑盒测试

##### Table.3 Black-box testing of dish management function.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试类型 | 测试目标 | 测试内容 | 预期结果 | 实际结果 | 是否通过 |
| 功能测试 | 添加菜品 | 在添加菜品页面输入正确信息并提交 | 菜品成功添加到系统中 | 菜品成功添加到系统中 | 是 |
| 功能测试 | 添加菜品 | 在添加菜品页面输入重复菜品信息并提交 | 提示“该菜品已存在” | 提示“该菜品已存在” | 是 |
| 功能测试 | 删除菜品 | 选择一道已存在的菜品进行删除操作 | 菜品成功从系统中删除 | 菜品成功从系统中删除 | 是 |
| 功能测试 | 修改菜品 | 选择一道已存在的菜品进行修改操作 | 菜品成功修改并更新到系统中 | 菜品成功修改并更新到系统中 | 是 |
| 可靠性测试 | 容错能力 | 在添加菜品页面输入格式不正确的信息并提交 | 提示“请填写\*\*信息” | 提示“请填写\*\*信息” | 是 |
| 可靠性测试 | 恢复能力 | 在删除菜品页面删除一道菜品后进行撤销操作 | 被删除的菜品成功恢复到系统中 | 无该功能 | 否 |
| 安全性测试 | 访问控制 | 使用非管理员账号尝试访问菜品管理页面 | 被拒绝访问并提示“无权限访问” | 被拒绝访问并提示“无权限访问” | 是 |

### 6.2.3 订单功能测试

首先测试下单功能。打开点餐页面，选择一款菜品加入购物车，然后点击“去结算”按钮，选择收获地址，然后点击“去支付”。具体流程如图10所示。查看后台管理界面是否已有该订单信息。



#### 图10 用户下单与订单管理

##### Fig.10 User ordering and order management.

从图中不难发现，用户下单到配送到家的整套流程并没有问题。

### 6.2.4 其他功能测试

在主要功能测试完毕后，还有其他很多细小的功能将直接影响用户体验，如地址管理功能。具体测试项目和测试内容如表4所示。

#### 表4 地址管理功能黑盒测试

##### Table.4 Black-box testing of address management function.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试项目 | 测试步骤 | 预期结果 | 实际结果 | 测试结论 |
| 添加地址 | 1. 进入地址管理界面  2. 点击添加地址按钮  3. 填写地址信息并提交 | 地址成功添加，地址信息正确 | 地址成功添加，地址信息正确 | 测试通过 |
| 地址编辑 | 1. 进入地址管理界面  2. 选择一个已有的地址  3. 点击编辑按钮  4. 修改地址信息并提交 | 地址成功编辑，地址信息正确 | 地址成功编辑，地址信息正确 | 测试通过 |
| 地址删除 | 1. 进入地址管理界面  2. 选择一个已有的地址  3. 点击删除按钮 | 出现删除确认提示框，点击确认后地址被删除 | 出现删除确认提示框，点击确认后出现“400”异常 | 测试未通过 |

在上述表格中，地址删除功能未能实现，查看报错信息：Required request parameter 'id' for method parameter type Long is not present。再查看服务器请求的url为：<http://192.168.136.129/addressBook?ids=1653676620267712513>。再查看后端代码时发现，在删除方法中传入的参数为：@RequestParam("id") Long id。这时，前端请求的参数为ids，而后端请求的参数为id，参数名不同导致发生了400异常错误。因此将@RequestParam("id")改为@RequestParam("ids")，重新测试程序后发现，地址可以顺利删除。

其次还有购物车功能影响着用户体验。在选择菜品时，必须先“选择规格”然后才能加入购物车；而在选择套餐时，可以直接加入购物车，无需选择规格等。当加入购物车后，点击已选菜品或套餐的“-”或者“+”可以直接增加或减少，也可以选择购物车中的“清空”直接清空购物车。具体测试项目和内容如表5所示。

#### 表5 购物车管理功能黑盒测试

##### Table.5 Black-box testing of shopping cart management function.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试项目 | 测试步骤 | 预期结果 | 实际结果 | 测试结论 |
| 添加菜品到购物车 | 1. 进入商品列表页面  2. 点击商品详情按钮  3. 在商品详情页面点击添加到购物车按钮 | 商品成功添加到购物车，购物车中显示该商品 | 商品成功添加到购物车，购物车中显示该商品 | 测试通过 |
| 修改购物车菜品数量 | 1. 进入购物车页面  2. 修改某个商品的数量  3. 点击更新购物车按钮 | 购物车中该商品的数量更新，购物车总价更新 | 购物车中该商品的数量更新，购物车总价更新 | 测试通过 |
| 删除购物车菜品 | 1. 进入购物车页面  2. 删除某个商品  3. 点击更新购物车按钮 | 购物车中该商品被删除，购物车总价更新 | 购物车中该商品被删除，购物车总价更新 | 测试通过 |
| 清空购物车功能 | 1. 进入购物车页面  2. 点击清空购物车按钮 | 购物车中所有商品清空，购物车总价为 0 | 购物车中所有商品清空，购物车总价为 0 | 测试通过 |

## 6.3 测试分析与评估

在对外卖服务平台进行测试分析评估时，首先对测试覆盖率进行了分析。针对平台的不同功能点，设计了多组测试用例，并记录了每组测试用例对应的代码覆盖率和功能覆盖率。通过分析测试结果，发现有些功能点的覆盖率仍不足，需要进一步完善测试用例，以提高测试覆盖率和发现更多潜在缺陷。

除此之外，还对外卖服务平台的性能进行了测试，并记录了响应时间、吞吐量、并发性、负载能力和稳定性等性能指标。通过分析测试结果，发现在高负载情况下平台响应时间较长，存在一定的并发性问题。需要进一步优化平台性能，提高平台的吞吐量和稳定性，以满足大量用户同时使用的需求。

# 致 谢

在本论文完成的过程中，我受到了许多人的支持和帮助，在此我要向他们表达我的感激之情。

首先，我要感谢我的导师陈晨老师，她的悉心指导使我在论文研究中受益匪浅。她严谨的治学精神和一丝不苟的工作态度给我留下了深刻的印象，并使我对科研工作充满了热情。

其次，我要感谢我的父母和家人，他们的鼓励和支持一直伴随着我完成学业的道路。没有他们的支持和理解，我是无法完成这篇论文的。

在完成论文的过程中，我遇到了许多困难和挑战，但是在大家的帮助下我克服了它们。这篇论文的完成离不开各位老师、同学、家人和朋友的支持和鼓励，我由衷地感谢你们。

最后，我希望这篇论文能够为相关领域的研究者提供一些参考，同时也为我的未来发展打下坚实的基础。

# 参 考 文 献

* 1. 安乐,杨景.关于校园外卖现状及问题的研究和对策[J].食品安全导刊,2021(24):176+178.DOI:10.16043/j.cnki.cfs.2021.24.094.
  2. 刘菁，李杰，闫丰雅.在外卖服务场景下的消费者个人隐私保护的实践与探索[J].互联网天地，2022，44-48
  3. 高荣伟.外卖小哥在国外[J].检查风云，2021，28(4)：17-19
  4. 潘振武.平台服务费下降对外卖平台盈利模式的影响研究[J].财会通讯，2023，44(4):146-150
  5. 杨懂懂，李明春.一种接口自动化测试系统设计与实现[J].自动化应用与技术，2021，40（10）：142-145
  6. 陈立兵.IntelliJ IDEA——开发人员利器[J].程序员,2009(01):118-120.
  7. 庄绪强.VMWare Workstation在高职计算机教学中的应用.西安航空技术高等专科学校学报，2009，(3)：73-75
  8. 张欣.“互联网+”背景下餐饮外卖卫生监管研究[J].经营与管理,2023(03):163-168.DOI:10.16517/j.cnki.cn12-1034/f.2023.03.013.
  9. 关喜如意,杜玮淦. 同餐不同价 外卖平台佣金谁“买单”？[N]. 南方日报,2023-03-15(A07).DOI:10.28597/n.cnki.nnfrb.2023.001394.
  10. 王欢,李民,邓秀辉等.基于Redis缓存数据库和Nginx负载均衡技术的购物网站性能优化[J].软件导刊,2022,21(08):114-119.
  11. 涩即是Null.Redis入门教程〔OL〕.CSDN-专业开发者社区，2022
  12. 盛铨.基于Centos的Nginx Web服务器的构建[J].福建电脑,2012,28(03):145-146.
  13. 韦一鸣. 基于MySQL复制技术的数据库集群研究[D].杭州电子科技大学,2014.
  14. 肖路.Web应用系统功能测试研究与应用[D].重庆大学，2007，6-9
  15. 向润.黑盒测试方法探讨[J].软件导刊，2009，7(1)：33-35
  16. 张玮.浅析白盒测试和黑盒测试[J].无线互联科技,2013(06):136.

# 附 录