

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

外卖服务平台作为互联网经济的重要组成部分，不仅改变了消费者的消费习惯，也带动了餐饮行业的快速发展。本文以Spring Boot和Mybatis-Plus为基础，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、支付配送等功能，并能够为用户和商家提供一个高效、安全、可靠的服务环境。

本设计的目的是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。在需求分析和系统设计阶段，本文采用了软件开发生命周期模型、用例分析和类图设计等方法，以确保平台具有较好的可维护性和可扩展性。在开发和测试阶段，本文采用了JUnit、Mockito和Postman等工具来测试和调试平台，以确保其性能和稳定性。

在系统架构设计中，本文还利用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，实现了多个服务器协同工作，进一步提高了系统的稳定性和可用性。

通过设计发现，所开发的外卖服务平台在性能、稳定性和用户体验方面表现优异。通过Redis缓存机制和Nginx负载均衡技术的应用，平台的可用性、可靠性和读写性能得到了显著提高。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。

总之，本文利用Spring Boot和Mybatis-Plus框架构建了一款功能丰富、性能优异的外卖服务平台，并采用了现代软件开发方法来确保平台的可维护性和可扩展性。这些工作为外卖服务业的数字化转型提供了有益的借鉴和指导。

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

Takeout Service Platform Based on SSM

Abstract

As an important part of the Internet economy, the online food delivery platform has not only changed consumers' consumption habits, but also driven the rapid development of the catering industry. This paper develops a food delivery platform for users and merchants based on Spring Boot and Mybatis-Plus. The platform has functions such as online ordering, order management, user management, package management, and payment and delivery, and provides users and merchants with an efficient, secure, and reliable service environment.

The purpose of this design is to explore how to use modern technology to build an efficient, stable, and scalable food delivery platform. In the requirements analysis and system design stage, this paper uses methods such as software development life cycle model, use case analysis, and class diagram design to ensure that the platform has good maintainability and scalability. In the development and testing stage, this paper uses tools such as JUnit, Mockito, and Postman to test and debug the platform to ensure its performance and stability.

In the system architecture design, this paper also uses the MySQL master-slave replication mechanism to achieve read-write separation, which improves the read-write performance of the system. At the same time, using Nginx reverse proxy and load balancing technology, the requests are evenly distributed to multiple servers, which realizes the collaboration of multiple servers, further improving the stability and availability of the system.

Through the design, it is found that the developed food delivery platform performs well in terms of performance, stability, and user experience. The application of Redis caching mechanism and Nginx load balancing technology significantly improves the availability, reliability, and read-write performance of the platform. In addition, the platform also has some security, payment, and data statistics features, which ensure the information security and data privacy of users and merchants.

In summary, this paper uses the Spring Boot and Mybatis-Plus frameworks to build a feature-rich and high-performance food delivery platform, and adopts modern software development methods to ensure the maintainability and scalability of the platform. These works provide useful reference and guidance for the digital transformation of the food delivery industry.

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

目 录

1 绪论 1

1.1 SSM概述 1

1.2 研究背景和意义 2

1.3 国内外研究现状 3

1.3.1 国外研究现状 3

1.3.2 国内研究现状 3

2 开发工具介绍 4

2.1 IntelliJ IDEA 2022.1 4

2.2 SQLyog 4

2.3 VMware Workstation pro 4

2.4 Redis Desktop Manager 5

3 系统分析 6

3.1 需求分析 6

3.1.1 需求概述 6

3.1.2 功能需求 6

3.1.3 非功能需求 6

3.1.4 系统接口需求分析 6

3.1.5 数据需求分析 7

3.1.6 报表和分析需求 7

3.1.7 安全性需求分析 7

3.2 数据分析 8

3.2.1 用户数据分析 8

3.2.2 订单数据分析 8

3.2.3 菜品数据分析 8

3.3 设计约束分析 8

3.4 过程分析 8

3.4.1 用户下单流程 8

3.4.2 平台运营流程 9

3.4.3 决策流程 9

3.5 环境分析 9

3.5.1 外部环境分析 9

3.5.2 内部环境分析 9

3.6 风险分析 9

3.7 成本效益分析 10

3.7.1 成本方面 10

3.7.2 效益方面 10

3.8 总结 10

4 系统设计 11

4.1 系统架构设计 11

4.2 数据库设计 11

4.2.1 数据库需求分析 11

4.2.2 数据库模型设计 11

4.2.3 数据库逻辑设计 11

4.2.4 数据库物理设计 13

4.3 接口设计 13

4.3.1 用户与员工信息管理 13

4.3.2 菜品管理 14

4.3.3 分类管理 14

4.3.4 套餐管理 15

4.3.5 购物车 15

4.3.6 订单管理 15

4.4 系统性能设计 15

4.4.1 硬件配置 15

4.4.2 负载均衡 16

4.4.3 数据库 16

4.4.4 缓存设计 16

4.5 用户界面设计 16

4.5.1 后台管理界面 16

4.5.2 用户界面设计 17

5 系统实现 18

5.1 环境配置 18

5.1.1 IntelliJ IDEA 18

5.1.2 服务器配置 18

5.2 模块开发 19

5.2.1 登录模块 19

5.2.2 菜品管理模块 21

5.2.3 订单管理模块 22

5.2.4 购物车管理模块 23

5.2.5 地址管理模块 25

5.3 项目部署 26

5.3.1 部署架构 26

5.3.2 部署Nginx服务器与前端项目 27

5.3.3 部署Tomcat服务器与后端项目 27

5.3.4 部署MySQL数据库 28

5.3.5 部署redis数据库 29

6 系统测试 31

致 谢 34

参 考 文 献 35

# 绪论

## 1.1 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。这个框架组合的设计思想是基于各个框架之间的优势相互结合，以实现Web应用程序的开发和部署，从而大大提高了Java Web应用程序的开发效率和可维护性。

Spring是一个轻量级的IoC（控制反转）和AOP（面向切面编程）框架。IoC容器可以自动装配对象和管理对象之间的依赖关系，从而大大简化了开发人员的编码工作。而AOP可以使开发人员将业务逻辑和系统级服务（如日志、安全、事务等）分离出来，从而提高了代码的复用性和可维护性。

Spring MVC是基于Spring的MVC（模型-视图-控制器）框架，它可以帮助开发人员快速开发Web应用程序。Spring MVC可以将应用程序分为模型层、视图层和控制层，从而实现业务逻辑、数据处理和用户交互的分离。这种分离可以大大简化应用程序的开发和维护工作。

MyBatis是一个轻量级的持久化框架，它可以帮助开发人员简化与数据库的交互工作。MyBatis采用了面向SQL语句的开发方式，开发人员可以直接编写SQL语句，从而更加灵活地处理数据。MyBatis还提供了一些高级功能，如分页查询、缓存管理等，可以帮助开发人员更加高效地处理数据。

综合上述三个框架的特点，SSM框架组合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框架组合可以提供以下优点：

* 1. 高效：SSM框架组合可以大大简化Java Web应用程序的开发工作，从而提高开发效率和代码质量。
  2. 稳定：SSM框架组合可以提供高性能和可靠性，从而确保应用程序的稳定运行。
  3. 可扩展：SSM框架组合可以支持应用程序的快速扩展，从而适应不同的业务需求。
  4. 易维护：SSM框架组合可以提供良好的代码结构和注释，从而方便后续的维护工作。

因此，在SSM框架中，Spring作为核心框架，负责管理和协调各个模块的组件，提供了IoC和AOP等基础设施，方便开发者进行业务组件的开发和管理。Spring还提供了对Web应用的支持，如Web MVC框架，使得开发者能够更加方便地构建Web应用。

MyBatis作为持久层框架，通过提供一种简单的编程方式，使得开发者能够更加便捷地访问和操作数据库。MyBatis不仅提供了对象-关系映射（ORM）功能，还支持基于XML和注解两种方式的SQL映射，使得开发者能够更加灵活地定义和管理SQL语句。同时，MyBatis还提供了缓存机制，能够显著提高系统的性能。

SpringMVC作为Web框架，负责处理HTTP请求和响应，并将业务逻辑和数据模型进行整合和组织。SpringMVC提供了MVC模式的支持，使得开发者能够更加方便地进行Web应用的开发和管理。同时，SpringMVC还提供了很多特性，如表单处理、文件上传、RESTful风格的Web服务等。

通过将Spring、MyBatis和SpringMVC三个框架集成在一起，SSM框架能够实现前端展示、数据持久化和业务逻辑的高效组织和协调。同时，SSM框架也具有灵活性和可扩展性，使得开发者能够根据实际需求进行个性化开发和定制化配置。

而本设计用到的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。其次用到的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结合起来使用，可以进一步简化开发流程，提高开发效率和代码质量。Spring Boot提供了自动配置和集成的能力，可以方便地整合Mybatis-Plus，而Mybatis-Plus提供了许多便捷的方法和注解，可以帮助开发者更加快速地完成数据库操作。同时，Spring Boot和Mybatis-Plus都具有很好的可扩展性，可以方便地进行功能扩展和自定义配置。因此，Spring Boot和Mybatis-Plus已成为Java开发中常用的框架之一。

## 1.2 研究背景和意义

当今社会，外卖行业正处于高速发展期，越来越多的人通过外卖平台订餐。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。

而价格是消费者选择的主要因素之一，餐饮企业需要承担平台的一定手续费用，同时还要保证菜品价格具有竞争力。这使得企业的利润空间受到限制，难以实现真正的盈利。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，导致价格战愈演愈烈，企业盈利更加困难。

而且外卖平台上涉及大量的个人信息，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。

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。

在设计和开发过程中，本设计借鉴了现代软件开发方法，如软件开发生命周期模型、用例分析和类图设计等，以确保平台具有较好的可维护性和可扩展性。同时，本设计还采用了MySQL主从复制机制实现读写分离，利用Nginx反向代理和负载均衡技术实现多个服务器协同工作，从而提高系统的性能、稳定性和可用性。

最后，本设计的研究意义在于，为外卖服务业的数字化转型提供了有益的借鉴和指导，同时也为软件开发领域的技术创新和进步做出了贡献。

## 1.3 国内外研究现状

### 1.3.1 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。

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。

### 1.3.2 国内研究现状

以下为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。

# 开发工具介绍

## 2.1 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，如Java、Kotlin、Groovy、Scala、Spring、Hibernate、Maven等等。

IntelliJ IDEA的编辑器具有强大的智能感知功能，可以根据当前的上下文自动推测代码结构和关系，并提供相关的代码提示和补全。它还支持多种语法高亮、代码折叠、快捷键、自动缩进等功能，可以大大提高编码效率。

除了编辑器外，IntelliJ IDEA还提供了丰富的调试和测试工具。它支持多种调试模式，如普通调试、远程调试等，可以帮助开发者快速定位代码问题。同时，它还内置了JUnit和TestNG等测试框架，可以帮助开发者编写和运行各种测试用例。

IntelliJ IDEA还提供了强大的代码重构和优化功能，可以帮助开发者快速重构和优化代码结构和逻辑，提高代码质量和可维护性。此外，它还支持各种版本控制系统，如Git、SVN等，可以轻松管理和协作开发项目。

## SQLyog

SQLyog是一款流行的数据库管理工具，可以让开发人员更加高效地管理和维护MySQL数据库。SQLyog提供了直观的用户界面，支持多个MySQL服务器连接，并支持在不同的服务器之间进行数据传输和同步。SQLyog还提供了一个可视化的SQL编辑器，使开发人员可以轻松地编写和调试SQL语句。SQLyog的主要特点包括：

数据库管理：可以方便地管理和维护MySQL数据库，包括创建和修改数据库、表、字段、索引、触发器等操作。

数据导入和导出：可以将数据从一个MySQL服务器导出到另一个MySQL服务器，也可以将数据导出为SQL脚本、CSV文件、Excel文件等格式。

SQL编辑器：提供了一个可视化的SQL编辑器，支持语法高亮、代码自动完成、代码折叠、代码格式化等功能。

数据库备份和还原：可以轻松地备份和还原MySQL数据库，以防止数据丢失或意外删除。

数据库同步：可以将数据从一个MySQL服务器同步到另一个MySQL服务器，以确保数据的一致性。

总之，SQLyog是一款功能强大的数据库管理工具，可以提高开发人员的工作效率，让开发人员更加轻松地管理和维护MySQL数据库。

## VMware Workstation pro

VMware Workstation Pro是一款基于虚拟化技术的桌面虚拟机软件。它允许用户在一台计算机上运行多个操作系统，包括Windows、Linux和其他操作系统，而不需要对物理计算机进行重启或进行分区。这样可以方便地在同一台计算机上进行多个操作系统的测试、开发和部署等任务，节省了成本和时间。

VMware Workstation Pro支持多种操作系统，可以在虚拟机中安装和运行Windows、Linux、Mac OS X等操作系统。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，与在物理计算机上使用操作系统一样方便。

VMware Workstation Pro具有丰富的功能和工具，例如虚拟化网络、虚拟化存储、虚拟化CPU等，以及可视化的虚拟机管理界面。用户可以创建、启动、停止、暂停、克隆和删除虚拟机，还可以配置虚拟机的硬件和网络设置等。

总之，VMware Workstation Pro是一款功能强大的虚拟机软件，为用户提供了一种方便、高效的虚拟化解决方案，帮助用户节省了时间和成本，同时提高了工作效率。

## Redis Desktop Manager

Redis Desktop Manager是一款跨平台的Redis图形化管理工具，支持Windows、macOS和Linux系统。它提供了一个直观、易于使用的用户界面，使得开发人员和管理员可以方便地管理Redis服务器、数据库、键、值等信息。它的主要功能包括：

连接管理：可以轻松添加、编辑和删除Redis服务器的连接信息，并支持SSL加密连接和SSH隧道连接。

数据库浏览：可以查看Redis服务器中的所有数据库，以及它们所包含的键、值等信息。

键值操作：可以对键进行添加、编辑、删除、查找等操作，也支持Redis常见的数据结构类型如字符串、哈希、列表、集合、有序集合等的操作。

数据导入导出：可以方便地将Redis数据导入或导出为JSON、CSV、XML等格式。

监控与诊断：可以实时查看Redis服务器的性能指标、内存占用情况、连接数等信息，并可以通过slowlog、monitor等命令进行诊断。

安全性管理：可以设置密码、SSL证书等安全相关配置来保障Redis服务器的安全性。

总之，Redis Desktop Manager提供了一个简单、直观、功能丰富的界面，方便开发人员和管理员对Redis服务器进行管理和监控。

# 系统分析

## 3.1 需求分析

### 3.1.1 需求概述

需求分析是系统分析的一部分，是指对系统的用户需求、功能需求、性能需求等进行详细的分析和描述，以便为软件开发提供明确的目标和方向。在毕业设计中，需求分析是非常重要的一步，也是系统设计的基础，它可以帮助开发人员更加清晰地了解用户的需求和系统的功能要求，进而为系统的实现提供指导。

### 3.1.2 功能需求

用户可以通过提供必要的个人信息来注册平台。注册完成后，用户可以通过用户名和密码登录平台，选择餐品并添加到购物车中，最终下订单。此外，用户可以查看历史订单、修改订单、取消订单以及评价订单等操作。用户也可以修改个人信息、查看积分、查看收藏等信息。最后，用户需要选择支付方式并填写配送信息，以完成订单。平台提供了全面的功能，以方便用户更好地享受在线订餐的服务。

商家可以通过提供必要的企业信息来注册平台。注册完成后，商家可以通过用户名和密码登录平台，并添加、修改或删除套餐等操作。此外，商家还可以查看历史订单，修改订单状态，处理订单以及评价订单等。商家也可以修改个人信息，查看销售统计数据和评价等信息。总之，商家可以在平台上进行多种操作，以便更好地管理和发展他们的业务。

### 3.1.3 非功能需求

平台应该具备以下能力：在高峰期，平台应该能够处理大量的并发请求，以保证平台的稳定。平台应该能够快速响应用户请求，以提高用户体验。平台还应该支持大量的用户和商家同时使用，以满足平台的扩展需求。总之，这些能力是保证平台高效、稳定运行的必要条件。

为确保平台高效、稳定地运行，平台应该具备高可靠性和可用性，以确保24/7的服务可用性。同时，平台应该具备处理崩溃和系统故障的能力，以确保数据不会丢失。这些能力是保障平台顺利运行的必要条件，平台开发者应该尽可能地优化平台的性能、提高平台的可靠性和可用性，以便满足用户和商家的需求，提供更好的服务体验。

### 3.1.4 系统接口需求分析

当设计一个系统时，需要考虑用户的功能需求。用户的需求包括许多方面，比如登录、注册、个人信息管理、购物车、订单管理、支付等。因此，在设计系统时，需要确保系统能够支持这些功能，并且这些功能能够正常运行，以便满足用户的需求。

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。因此，设计系统时需要考虑商家的需求，确保商家能够在系统上完成相关的操作，如管理商铺信息、发布商品信息、查看订单信息等。同时，系统需要与第三方平台集成，例如支付宝等支付平台。这些第三方平台需要提供短信接口、支付接口等，以实现系统的在线支付功能和数据传输的安全性。

最后，系统的接口规范应该符合RESTful接口规范，使用JSON格式传输数据等。这些规范能够确保不同系统之间的互操作性和兼容性，以便实现不同系统之间的数据传输和信息共享。

总之，设计系统时需要考虑不同用户的需求，并确保系统能够正常运行。此外，还需要与第三方平台集成，确保系统的在线支付功能和数据传输的安全性。最后，需要确保系统的接口规范符合RESTful接口规范，以便实现不同系统之间的数据传输和信息共享。

### 3.1.5 数据需求分析

为了有效管理食品送餐平台，需要收集关于该流程各个方面的数据。例如，需要收集顾客的下单和支付行为，包括下单时间、订单金额、商品信息、支付方式等。此外，还需要收集餐厅的订单接收和处理情况，包括接单时间、处理时间、商品制作情况等。另外，还需要收集配送员的送餐路线和时效，包括接单时间、出发时间、送达时间等。

为确保订单准确无误，平台需要收集顾客下单信息，包括下单时间、订单金额、商品信息、支付方式等。同时，平台需要确保餐厅订单信息完整，包括接单时间、处理时间、商品制作情况等，以确保每个订单都有正确的商品信息和处理状态。此外，平台还需要收集配送员的行程信息，包括接单时间、出发时间、送达时间等，以确保送餐路线和送达时间的可靠性。这些信息的准确性和完整性对于平台的顺利运营和用户的满意度至关重要。

为了进行后续数据分析和业务决策，需要将顾客和餐厅的信息整合在一起。同时，对订单信息进行归类和统计，以生成销售报表和营收分析报告。除此之外，需要对配送员的行程进行路线优化，以提高配送效率和服务质量。

为确保用户的隐私权和数据安全，需要对用户的个人信息进行保护。在数据采集、传输、存储和处理的过程中，需要采用加密技术和备份机制，以确保数据不会被非法获取、篡改或丢失。此外，还需要制定相关的安全措施和政策，对员工进行安全意识教育，防止内部数据泄露。

### 3.1.6 报表和分析需求

系统需要对订单数据进行全面的统计和分析，包括订单量、销售额、平均客单价、订单满意度等指标，以帮助业务进行精细化运营。基于这些数据，还需要生成销售报表和营收分析报告，以便评估业务运营状况和制定营销策略。此外，系统还需要运用数据挖掘和机器学习技术，以便预测用户行为、优化服务流程等，从而提高用户体验和增加收益。

### 3.1.7 安全性需求分析

平台的安全性是至关重要的，需要确保用户和商家的个人信息不会被泄露、篡改或盗用。在用户和商家登录、注册、支付等关键过程中，所有数据应该进行加密传输，以防止信息被黑客攻击者截获。此外，系统也需要具备抵御常见的网络攻击的能力，例如SQL注入、跨站点脚本攻击等。通过建立安全性强、漏洞少的系统，可以保证平台运营的可靠性和稳定性。

## 数据分析

### 3.2.1 用户数据分析

需要进行用户地域分布、行为和评价分析。用户地域分布可以分析用户所在城市、区域和行政区划，以便为不同地域的用户提供个性化的服务。用户行为分析可以分析用户的活跃程度、下单频次、订单金额和消费偏好等，以便为不同类型的用户提供更加贴心的服务。同时，用户评价分析可以分析用户的评价内容、评分和评价频率，以便发现平台服务的问题并及时改进。

### 3.2.2 订单数据分析

为了优化平台的运营策略，需要对订单数据进行分析。这包括订单趋势分析，通过分析订单的时间、数量和金额趋势，来发现优化平台运营的机会。同时，还需要对不同类型的订单进行分析，如外卖订单、到店消费订单、预定订单等，以便为不同类型的订单提供更好的服务。此外，还需要进行订单满意度分析，通过分析订单的满意度、退单率和投诉情况，来发现平台服务的问题并及时改进。

### 3.2.3 菜品数据分析

对菜品销售数据进行综合分析，包括销售量、销售额、销售额占比等指标，以便确定营销策略和调整库存管理。同时，还需要综合考虑用户的口味和需求，推出更符合市场需求的新菜品，以提高用户满意度。在菜品评价分析方面，需要对用户对菜品的口味、质量、服务、环境等各方面进行综合评估，以便及时发现问题并改进菜品和服务质量。对于菜品成本分析，需要对菜品的成本、售价和毛利润等进行综合分析，以便确定合理的菜品定价和提高菜品毛利润率。

## 设计约束分析

为了遵守技术约束，系统需要基于Spring Boot和Mybatis-Plus框架进行开发，并使用MySQL数据库存储数据和Redis实现缓存等。同时，为了满足性能约束，系统需要考虑高并发和低延迟等性能要求，并保证系统的可扩展性和稳定性。最后，为了保安全约束，系统必须确保用户信息和数据的安全性，以避免遭受黑客攻击和数据泄露等安全问题。

## 过程分析

### 3.4.1 用户下单流程

用户登录平台或注册账号，只需输入手机号，发送验证码并填写正确后即可自动注册。接下来用户可以浏览菜单，选择心仪的菜品并添加到购物车中。

在进入结算页面前，用户需要填写收货地址、联系方式和支付方式等信息，以便平台为用户提供更贴心的服务。用户确认订单并完成支付后，平台将订单信息推送给餐厅，餐厅开始准备菜品。

当餐厅完成菜品准备后，将菜品交给配送员进行配送。配送员将根据订单信息送达用户所在地，并将菜品交给用户。用户收到菜品后需要确认收货并对菜品进行评价，以便帮助平台和餐厅了解用户的口味偏好和提供更好的服务。

### 3.4.2 平台运营流程

用户确认订单并完成支付。平台接收用户下单信息并推送给餐厅。平台接收餐厅的订单确认和菜品准备情况，并通知配送员进行取货。配送员根据订单信息送达用户所在地，并将菜品交给用户。平台接收配送员的取货信息和配送情况，并通知用户订单状态。用户收到菜品后确认收货并对菜品进行评价。平台根据用户评价和订单情况进行数据分析，优化平台运营策略。

### 3.4.3 决策流程

平台根据用户需求和行为，分析用户的消费偏好和订单趋势等信息，制定运营策略和推广计划。同时，平台根据菜品成本和销售情况等信息，调整菜品价格和供应商等决策，以提高平台的盈利和用户的满意度。此外，平台还根据用户评价和订单满意度等信息，改进平台的服务流程和用户体验，不断提升平台的品质和服务水平，满足用户的需求和期望。这一系列的运营和改进策略，为平台的长期发展和用户的持续满意打下坚实的基础。

## 环境分析

### 3.5.1 外部环境分析

在外卖服务行业中，政治环境、经济环境、社会环境和技术环境是影响其发展的重要因素。政策法规对外卖服务行业的影响，包括政府对商业税收和监管等方面的政策影响，对于行业发展有着深远的影响。消费水平和人均收入的变化，竞争对手的价格战和市场占有率的争夺等经济环境因素也对外卖服务行业产生影响。此外，消费者对于外卖服务的需求和偏好，对于品质、安全和快速配送等方面的要求和期待，以及互联网技术的不断进步和创新、智能化和自动化技术的应用和发展，都是社会环境和技术环境中重要的因素。同时，数据隐私和信息安全等技术环境问题也需要引起注意。因此，外卖服务企业需要全面考虑这些因素，以制定有效的发展策略和应对措施。

### 3.5.2 内部环境分析

平台的发展除了受到外部环境的影响外，还受到内部因素的制约和促进。其中人力资源是关键因素之一，平台需要有一支素质高、能力强的员工团队，并建立合理的组织架构和职责分工。资金是平台发展的另一个重要因素，平台的融资和资金运营情况直接关系到平台的发展速度和规模。技术创新是平台发展的重要支撑，平台需要持续研发和应用先进技术，提升平台的应用效果和维护水平。市场营销是平台的重要策略，平台需要建设自己的品牌形象，采取有效的市场推广和客户维护措施，提高平台的知名度和用户黏性。

## 风险分析

外卖平台所面临的风险十分复杂多样。首先，供应链风险是平台所涉及的供应商、配送员、餐厅等方面的供应链关系可能出现延误、质量问题、安全问题等情况，导致服务质量下降、客户流失等问题。其次，数据安全风险是平台所涉及的大量客户和商家信息可能会面临数据泄露、网络攻击等问题，导致客户信息和商业机密的泄露和损失。

同时，法律合规风险是平台所涉及的餐饮安全、交通安全、税收等方面的法律和政策要求可能会发生变化或者平台未能及时遵守，导致违规处罚、信誉受损等问题。此外，市场竞争风险是市场上存在众多竞争对手，以及新进入者的挑战，可能会导致平台的市场份额下降、营收减少等问题。

另外，用户需求变化风险也是一个十分重要的因素，消费者的需求和偏好不断变化，可能会导致平台所提供的服务无法满足用户需求，进而导致用户流失和品牌形象下降。此外，战略决策风险是平台的决策制定可能会存在失误，导致资源投入不足、业务扩张不当等问题，影响平台的长期发展。

最后，预算控制风险是平台预算控制可能存在问题，导致运营成本过高、现金流问题等，影响平台的盈利能力和财务稳健性。因此，外卖平台应该全面评估并有效应对这些风险。

## 成本效益分析

### 3.7.1 成本方面

平台的运营需要投入大量成本，其中包括技术开发成本、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，需要投入大量的资金来开发和维护系统，包括雇佣软件开发人员、购买设备、租用服务器等。

同时，平台也需要进行大量的市场宣传和推广，包括广告投放、促销活动等，以吸引更多的用户和商家入驻平台。另外，平台还需要支付员工工资、租金、水电费等日常开支，并需要承担维护和管理供应链、配送网络等方面的费用。在控制这些成本的同时，平台还需要寻找新的收益来源，以确保平台的长期盈利能力和可持续发展。

### 3.7.2 效益方面

外卖平台的成功很大程度上依赖于用户体验。为此，平台需要提供快捷、便利的外卖服务，以提高用户满意度和留存率。

同时，营销和推广活动也是提高平台知名度和影响力的重要手段。这将有助于吸引更多商家入驻平台，从而增加平台的营业收入。除了商家佣金和配送费用等收入，平台还需要考虑如何控制成本，以保证收益的可持续性。平台的成功除了依赖于营收和成本控制，还需要不断创新和优化，以满足用户需求和市场变化，进一步提高用户满意度和忠诚度。

## 总结

外卖服务平台旨在提供快捷、便利的外卖服务，以满足用户需求。为了实现商业目标，平台需要优化用户体验、增加商家入驻量、提高订单量等方面。为了深入了解用户和商家需求，平台需要收集和分析用户数据、商家数据、订单数据、配送数据等各类数据。在系统设计方面，平台需要考虑如何实现快捷、稳定、安全的服务，同时要提供用户友好的界面和完善的功能。在落实系统分析的基础上，平台应该不断寻求降低成本、提高效益的策略，重视品牌建设和推广营销，以实现长期的稳健发展。

# 系统设计

## 系统架构设计

一个外卖平台需要依赖多个技术要素来实现其业务逻辑和功能。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，包括应用服务器、数据库服务器、缓存服务器等。此外，平台需要与第三方系统进行数据交互，如支付接口、短信接口、地图接口等。同时，平台还需要与供应商进行数据交互，如餐厅、骑手、物流公司等接口。为了提高系统性能和可扩展性，消息队列用于异步消息通信。

最后，平台需要重视系统安全，包括用户认证、访问控制、数据加密、系统备份和恢复等方面，以保障系统的安全性和可靠性。所有这些技术要素相互配合，才能够实现一个高效、安全、稳定的外卖平台。

## 数据库设计

### 数据库需求分析

本设计需要设计多张表来存储相关数据，包括用户表、地址表、菜品表、菜品分类表、菜品口味表、订单表、订单明细表、员工表、套餐表和购物车信息表。用户表用于存储用户信息，地址表用于存储用户的收货地址信息，菜品表用于存储菜品的信息，菜品分类表和菜品口味表用于记录不同种类的菜品和口味，订单表和订单明细表用于存储订单信息，员工表用于记录平台的员工信息，套餐表和套餐详细信息表用于存储不同套餐的信息和详细内容，购物车信息表用于存储用户的购物车信息。对于每个表，需要确定存储的数据类型和属性，并考虑数据的访问模式和数据的约束条件，以保证数据的一致性和完整性。

### 数据库模型设计

该数据库包含多个实体，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。用户实体包含用户ID、用户名、密码、手机号等属性；地址实体包含地址ID、用户ID、收货人姓名、联系电话、省份、城市、街道地址等属性；菜品实体包含菜品ID、菜品名称、菜品描述、菜品图片、价格、菜品分类ID等属性；菜品分类实体包含分类ID、分类名称等属性；菜品口味实体包含口味ID、口味名称等属性；订单实体包含订单ID、用户ID、下单时间、订单状态等属性；订单明细实体包含订单明细ID、订单ID、菜品ID、数量、单价等属性；员工实体包含员工ID、员工姓名、联系电话、职位等属性；套餐实体包含套餐ID、套餐名称、套餐描述、套餐价格等属性；购物车信息实体包含购物车ID、用户ID、菜品ID、数量等属性。

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。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，级联操作可以选择“删除用户时同时删除其关联的地址信息”。

### 数据库逻辑设计

在外卖服务平台的数据库逻辑设计中，我们可以将模型设计中的实体转换为关系表，而在每个关系表中，我们需要定义该表的属性和约束。例如，在用户表中，我们可以定义用户的ID、用户名、密码、电子邮件、电话和性别等属性，其中用户名和电子邮件需要定义为唯一约束。在订单表中，我们可以定义订单的ID、用户ID、地址ID、下单时间、订单状态和总价等属性，其中用户ID和地址ID需要定义为外键，与用户和地址关系表建立联系。

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，在订单表和订单明细表之间，我们可以定义一个级联删除操作，以确保在删除订单时，相关的订单明细也会被删除。在购物车信息表中，我们可以定义一个唯一约束，以确保每个用户只能有一个购物车信息记录。具体设计如表1所示。

表1 数据库模型设计

Table.1 Database Model 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

### 数据库物理设计

在外卖服务平台中，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，需要考虑主库和从库的分布和数据同步机制。我们可以将主库和从库分别部署在不同的服务器上，以实现分布式的数据存储和读写请求处理。同时，需要设置MySQL主从同步机制，以确保数据的同步和一致性。具体实现方式包括使用MySQL复制协议和二进制日志等技术。

在Redis服务器中，需要考虑Redis实例的数量和数据分片机制。我们可以将Redis实例部署在不同的服务器上，并使用Redis集群技术来实现数据的分片和负载均衡。具体实现方式包括使用Redis Cluster集群技术和一致性哈希算法等技术。

为确保数据的安全性和可靠性，需要制定数据库备份和恢复策略。我们可以使用MySQL和Redis提供的备份和恢复工具，如mysqldump、redis-cli等，来定期备份数据库和数据文件。同时，还可以使用分布式存储技术和云备份等技术，来实现数据的长期备份和恢复。

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。我们可以使用MySQL和Redis提供的性能监控和优化工具，如MySQL Enterprise Monitor、Redis Monitor等，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。同时，还可以使用缓存技术、分布式事务等技术，来优化数据库的读写性能和并发访问能力。

## 接口设计

### 用户与员工信息管理

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。这些接口设计旨在提供方便的员工管理和用户登录/注册功能。具体如表12所示。

表2 用户与员工信息管理的相关接口信息

Table.2 Interface information related to user and employee information management.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

### 菜品管理

新增菜品操作使用POST /dish请求，并需要提供Dish实体类作为请求参数，包括菜品名称、菜品分类ID、菜品价格、商品码、图片、描述信息、售卖状态、口味、套餐分类等信息。响应数据为“新增菜品成功”。

菜品信息分页查询操作使用GET /dish/page请求，并需要提供当前页数page、每一页大小pageSize、菜品名字name作为请求参数。响应数据包括菜品列表和菜品信息。

根据id查询菜品信息和对应的口味信息使用GET /dish/{id}请求，并需要提供菜品ID作为请求参数。响应数据包括菜品信息。

修改菜品操作使用PUT /dish/update请求，并需要提供Dish实体类作为请求参数。响应数据包括支付状态和订单状态。

根据条件查询对应的菜品数据使用GET /dish/list请求，并需要提供Dish实体类作为请求参数。响应数据包括菜品信息。

### 分类管理

针对分类管理的接口设计如下：用户可以通过POST方法新增分类，请求参数是分类的名称、顺序、创建时间、更新时间、创建人和修改人等信息，响应数据为“新增分类成功”。通过GET方法分页查询所有分类信息，请求参数是当前页数page和每页大小pageSize，响应数据为分类信息。通过DELETE方法删除某个分类信息，请求参数是分类ID，响应数据为“分类信息删除成功”。也可以通过GET方法根据条件查询对应的分类数据，请求参数是Category实体，响应数据是分类信息。

### 套餐管理

新增套餐可通过发送POST请求到/setmeal，请求参数为Category实体类包含套餐名称、套餐价格、状态、编码、描述信息、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page、每一页大小pageSize以及套餐名字name。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。

### 购物车

用户可以通过发送POST请求到/shoppingCart/add接口将商品添加到购物车中。请求参数应该包含购物车实体类，其中包括名称、用户ID、菜品ID、套餐ID、口味、数量、金额、图片和添加时间。响应数据将包含购物车信息。

要查看购物车中的所有商品，用户可以向/shoppingCart/list发送GET请求，不需要请求参数。响应数据将包含购物车信息。

用户可以通过向/shoppingCart/clean发送GET请求来清空购物车。这个接口不需要请求参数，响应数据将返回“清空购物车成功”的信息。

### 订单管理

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。

## 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。

### 负载均衡

为了保证平台的高并发支持，需要使用负载均衡器来快速、准确地分发请求，确保每台应用服务器的负载尽可能平均。同时，为了保证平台的高可靠性，负载均衡器需要具备自动故障切换的能力，避免单点故障。为了避免网络瓶颈成为性能瓶颈，负载均衡器需要足够的网络带宽来支持高并发的请求和响应。同时，为了保证平台的高性能，负载均衡器需要具备高性能的处理能力和转发速度，能够快速地处理请求并将请求分发到应用服务器上。最后，为了应对平台规模的增大，负载均衡器需要具备易扩展性，以支持更高的并发量和更多的应用服务器。

### 数据库

为了在高并发的情况下确保数据库性能，可以采用一系列优化措施。

首先，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。

### 缓存设计

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。

此外，为了避免缓存中的数据过期而产生脏数据，可以设置缓存过期时间。对于一些频繁更新的数据，可以设置较短的过期时间，保证缓存中的数据与数据库中的数据保持一致。通过合理地运用缓存技术，可以有效提高系统的性能和稳定性。

## 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。

具体的功能模块设计可以根据业务需求和用户使用习惯来定制，每个模块都应该包含相关的输入框、按钮和表单，以便用户进行条件搜索、添加信息和编辑信息等操作。当一页无法完全展示所有信息时，翻页功能可以提供更好的信息浏览和操作体验。

除了以上设计方案，根据实际需求还可以考虑其他的导航菜单设计方案，例如面包屑导航、标签导航等，以满足不同用户群体的需求。最终的导航菜单设计应该简洁明了、易于理解和操作，以提高用户满意度和系统的易用性。

### 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。

商家详情页设计应该包括商家基本信息、商品列表、商家公告和底部栏。商家基本信息应该包括商家图片、名称、销售量、距离、配送费、预计时长等信息。商品列表应该根据商品分类展示各类商品的图片、名称、价格、月售量等信息，并支持选择规格并添加到购物车。商家公告应该展示商家公告和优惠活动信息。底部栏应该包括购物车、结算等功能入口。

购物车设计应该包括商品列表、结算信息和底部栏。商品列表应该展示用户已添加的商品列表，支持修改数量、删除商品等操作。结算信息应该展示商品数量、总价、配送费、预计送达时间、优惠券等结算信息。底部栏应该包括备注、支付等功能入口。

个人中心设计应该包括我的订单、个人信息、收货地址和底部栏。我的订单应该展示用户的最新订单、历史订单等。个人信息应该展示用户的头像、昵称、联系方式等信息，并支持修改个人信息。收货地址应该展示用户的收货地址列表，支持添加、修改和删除地址。底部栏应该包括退出登录。

以上几个部分是电商系统设计的重要组成部分，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。

# 系统实现

## 环境配置

### 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，为了使用mybatis-plus框架，需要在pom.xml文件中添加mybatis-plus依赖。同时，Java编程语言的核心类库也需要被引入。为了简化代码并提高开发效率，引入Lombok库是非常有必要的。同时项目需要使用MySQL数据库，还需要添加mysql依赖。最后，为了更好地管理数据库连接，使用德鲁伊数据库连接池是比较常见的做法。编辑pom.xml文档并添加以上依赖可以使得项目具备相应的功能和特性，从而满足项目需求。

application.yml是Spring Boot应用程序的配置文件，其中包含了一些关键的配置信息，例如数据库连接信息和服务器端口号等。首先，需要设置服务器端口号，以确保应用程序能够正确地监听网络请求。其次，还需要配置Spring中的数据源（datasource）以及Redis，以便应用程序能够与这些服务进行交互。最后，还需要配置MyBatis，以便应用程序能够与MySQL数据库进行交互。

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等。配置完成后，保存并关闭文件即可。

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。

### 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。

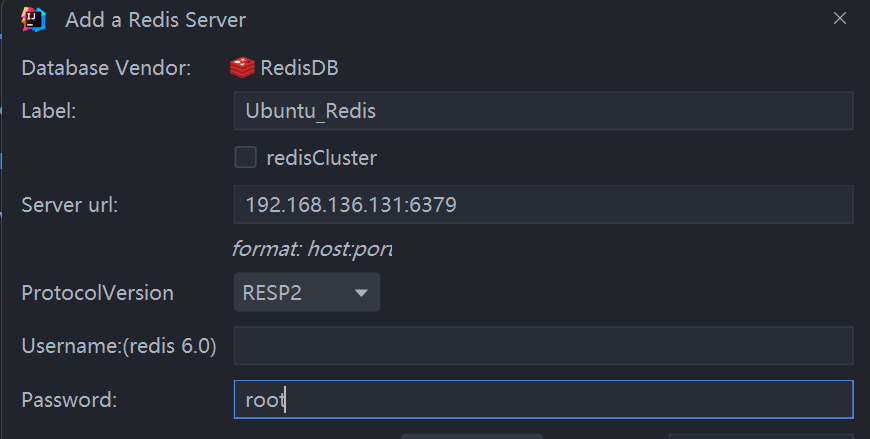


图2 IDEA中Redis插件的配置

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。

## 模块开发

### 登录模块

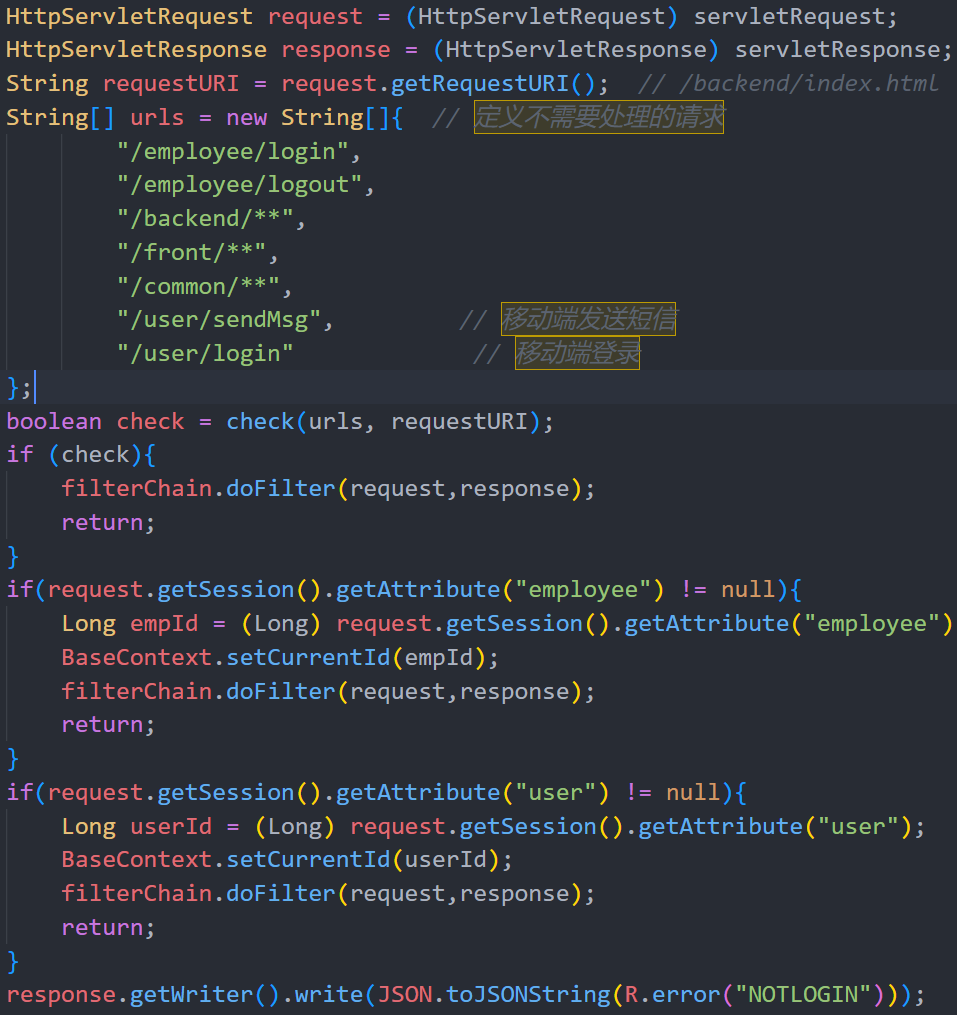
在创建登录页面时，可以使用HTML、CSS和JavaScript等前端技术。需要创建一个HTML页面，在该页面中添加用户名和密码的输入框，并且在底部放置一个登录按钮。可以使用CSS样式美化页面，并使用JavaScript编写交互逻辑，比如验证用户名和密码是否符合要求，或者对登录按钮添加点击事件等。登录按钮绑定一个函数，通过AJAX或者其他方式向后端发送请求，进行身份验证，根据身份验证的结果，进行页面跳转或者提示用户登录失败等信息。

然后编辑后端代码逻辑。首先，创建Controller，用于处理登录请求和返回登录页面。包括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。从Redis缓存中获取存储在该手机号码下的验证码，然后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。然后将该用户的ID存储在HttpSession中，并将其返回给客户端。如果验证码不匹配，则返回一个包含"登录失败"消息的错误响应。



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，以便在用户未登录的情况下禁止访问其他页面。

具体来说，拦截器需要实现Filter接口，重写doFilter方法，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。



### 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。

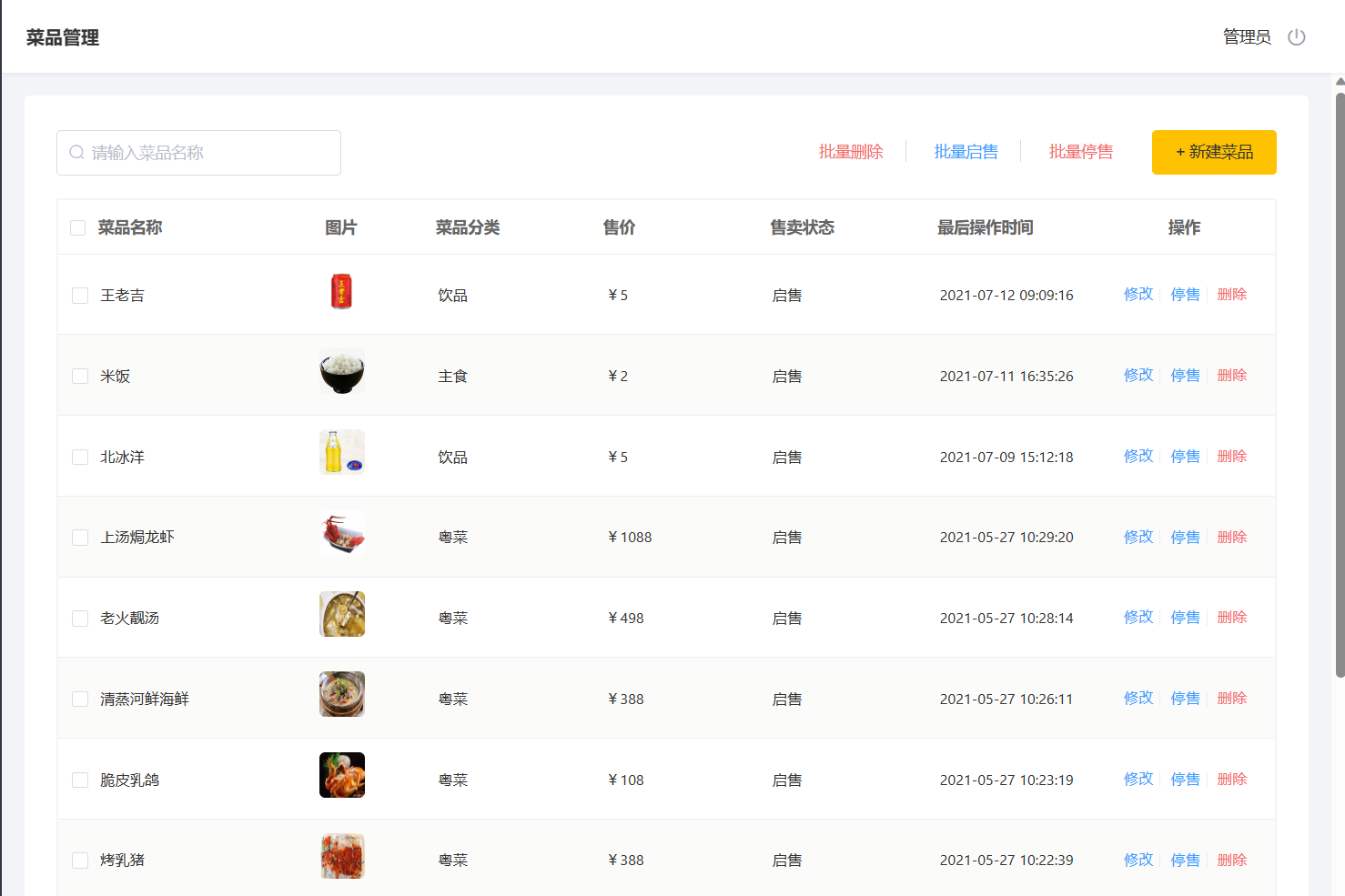


图3 菜品管理页面

Fig.3 Dish management page.

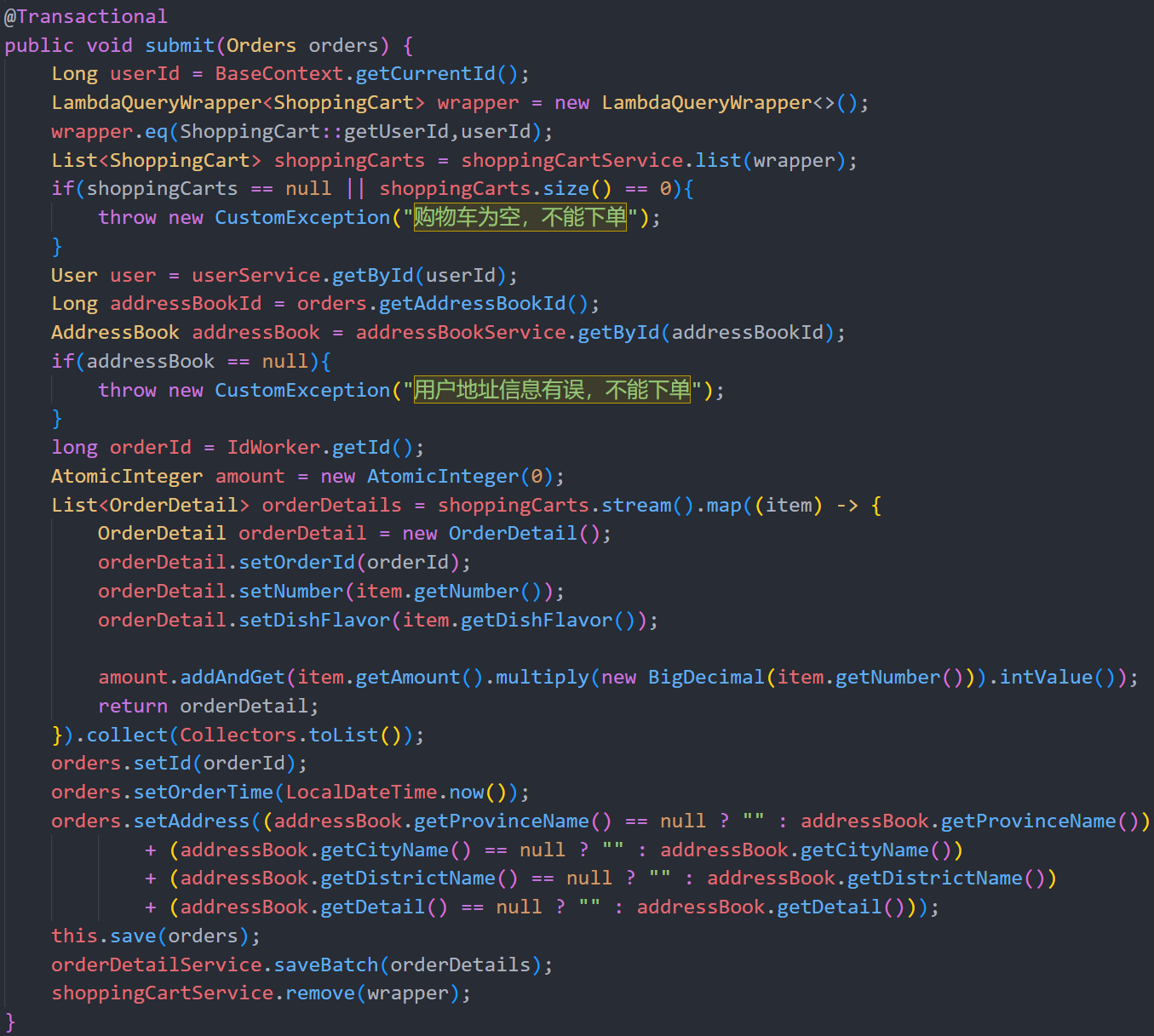
商家可以在菜品管理模块中添加、修改和删除菜品。这些数据库查询方法全部通过DishMapper继承Mybatis-plus中的BaseMapper获得。BaseMapper是MyBatis-Plus提供的一个通用Mapper接口，它封装了一些常用的数据库操作方法，如插入、查询、更新和删除等。要使用它时，先用注解@Mapper来标注这个接口是MyBatis框架需要的Mapper接口。然后指定它需要泛型类型参数为接口所对应的实体类Dish。

### 订单管理模块

在前端页面中，使用了Element UI的组件库，通过导入CSS文件实现了样式的渲染。页面上包括一个搜索栏，包含一个输入框和一个日期选择器，用于搜索订单；另外，还有一个表格用于展示订单信息，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。页面底部还有分页组件，用于翻页操作。同时，页面还有一个查看弹框，用于展示订单详细信息。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。

首先使用@Transactional注解来实现事务处理。方法首先获取当前用户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，将抛出CustomException异常。接着查询用户数据和地址数据，并判断地址数据是否存在。然后生成一个唯一的订单号，并遍历购物车数据，创建订单详情数据。通过设置订单数据和订单详情数据，创建一个订单并将其保存到订单表中。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。

### 购物车管理模块

首先，在商品详情页面，可以提供加入购物车的功能，用户可以选择商品数量并将商品加入购物车。这可以通过在CartItem实体类中添加一个quantity属性来实现。然后在购物车管理模块的控制器中编写一个添加购物车项的方法：

*@PostMapping*("/addCartItem")  
*public* String addCartItem(*@RequestParam*("productId") Long productId, *@RequestParam*("quantity") Integer quantity, HttpSession session) {  
 User user = (User)session.getAttribute("user");  
 *if* (user == *null*)   
 *return* "redirect:/login";  
 CartItem cartItem = *new* CartItem();  
 cartItem.setUserId(user.getId());  
 cartItem.setProductId(productId);  
 cartItem.setQuantity(quantity);  
 cartItemService.save(cartItem);  
 *return* "redirect:/cart";  
}

其中cartItemService是一个封装了对购物车项进行增删改查操作的服务类。在上述方法中，首先判断用户是否登录，然后创建一个CartItem对象并将其保存到数据库中，最后重定向到购物车列表页面。

在购物车列表页面，可以展示当前用户所有的购物车项，以及每个购物车项对应的商品信息和数量等。这些信息可以通过调用CartItemService中提供的查询方法来获取。编写一个CartItemService类实现一个查询当前用户所有购物车项的方法：

*@Service*  
*public class CartItemService* {  
 *@Autowired*  
 *private CartItemMapper* cartItemMapper;  
 *public List<CartItemVO>* listByUserId(Long userId) {  
 *return* cartItemMapper*.*listByUserId(userId);  
 }  
}

其中CartItemVO是一个封装了购物车项和对应商品信息的值对象。在CartItemMapper中，编写一个根据用户ID查询购物车项和对应商品信息的方法：

*public interface* CartItemMapper *extends* BaseMapper<CartItem> {  
 List<CartItemVO> listByUserId(Long userId);  
}

在购物车管理模块的控制器中，可以调用CartItemService中的listByUserId()方法获取当前用户所有的购物车项，然后将其传递给购物车列表页面进行展示。

在购物车列表页面，用户可以修改购物车项中商品的数量。这可以通过在CartItemMapper中实现一个更新购物车项数量的方法来实现。例如，可以编写一个更新购物车项数量的方法：

*public interface* CartItemMapper *extends* BaseMapper<CartItem> {  
 *int* updateQuantity(@Param("id") Long id, @Param("quantity") Integer quantity);  
}

在购物车管理模块的控制器中，可以根据前端传递过来的购物车项ID和新数量调用CartItemMapper中的updateQuantity()方法来更新购物车项的数量。编写一个修改购物车项数量的方法：

*@PostMapping*("/updateCartItem")  
*public* String updateCartItem(*@RequestParam*("cartItemId") Long cartItemId, *@RequestParam*("quantity") Integer quantity) {  
 CartItem cartItem = cartItemService.getById(cartItemId);  
 cartItem.setQuantity(quantity);  
 cartItemService.updateById(cartItem);  
 *return* "redirect:/cart";  
}

在上述方法中，首先根据购物车项ID获取购物车项对象，然后更新其数量并将其保存到数据库中，最后重定向到购物车列表页面。

删除购物车商品

在购物车管理模块的控制器中，可以根据前端传递过来的购物车项ID调用CartItemMapper中的deleteByCartItemId()方法来删除购物车项。例如，可以编写一个删除购物车项的方法：

*@PostMapping*("/deleteCartItem")  
*public* String deleteCartItem(*@RequestParam*("cartItemId") Long cartItemId) {  
 cartItemService.removeById(cartItemId);  
 *return* "redirect:/cart";  
}

在上述方法中，首先根据购物车项ID删除购物车项，然后重定向到购物车列表页面。

### 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。

首先，用户可以通过在地址映射器（AddressMapper）中实现一个插入地址的方法来添加新的地址。

接下来，在地址管理模块的控制器中，可以通过调用AddressMapper中的insert()方法来插入新的地址。例如，可以编写一个添加地址的方法，如下所示：

@PostMapping("/addAddress")

public String addAddress(@ModelAttribute("address") Address address, Principal principal) {

User user = userService.getUserByUsername(principal.getName());

address.setUserId(user.getId());

addressService.save(address);

return "redirect:/address";

}

在上述方法中，首先获取当前用户的ID，然后将其设置为地址对象的用户ID，最后将地址保存到数据库中并重定向到地址列表页面。

其次，用户可以通过在AddressMapper中实现一个更新地址的方法来修改已有的地址。在地址管理模块的控制器中，可以通过调用AddressMapper中的updateById()方法来更新地址。首先将用户修改后的地址保存到数据库中，然后重定向到地址列表页面。

最后，用户可以通过在AddressMapper中实现一个删除地址的方法来删除已有的地址。在地址管理模块的控制器中，可以根据前端传递过来的地址ID调用AddressMapper中的deleteByAddressId()方法来删除地址。根据地址ID删除地址，然后重定向到地址列表页面。

## 项目部署

### 部署架构

在管理端管理后台菜品数据的过程中，管理员可以通过一个后台管理系统来添加、删除、修改、查询菜品数据。后台管理系统一般由前端页面和后端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。

在后端服务器中，通常会使用主从数据库来提高读取性能和数据可用性。主数据库用于写操作和更新操作，从数据库用于读取操作。此外，为了提高查询性能和降低数据库的压力，后端服务器还会使用Redis缓存来存储一些常用的数据，比如菜品信息、用户信息等。

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4所示。

Redis

MySQL

MySQL

192.168.136.131

192.168.136.129

192.168.136.130

192.168.136.129

192.168.136.130

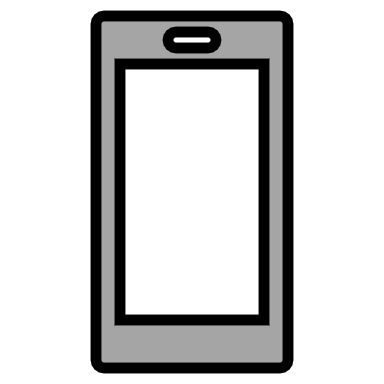
Tomcat

Nginx

用户端

Internet

管理端





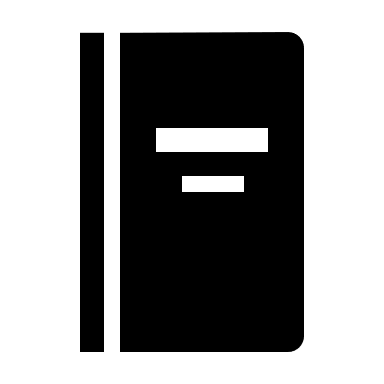
  

图4 服务器架构图

Fig.4 Server architecture diagram

### 部署Nginx服务器与前端项目

在将已上传到GitHub的项目克隆到服务器时，可以使用git clone命令进行操作。克隆完成后，需要修改nginx的配置文件，以便能够访问到该项目的静态资源。此后，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。实现效果如图5所示。



图5 Nginx部署静态资源预览图

Fig.5 Deploying Static Resource Preview Images with Nginx

### 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。执行mvn clean package -Dmaven.test.skip=true命令，打包代码。再更改到target目录，使用nohup java -jar takeout\_service-1.0-SNAPSHOT.jar &> imau\_take\_out.log &命令启动应用程序。这个命令启动Java虚拟机并运行应用程序，并将输出写入imau\_take\_out.log文件。nohup命令使应用程序在终端关闭后继续运行。最后赋予脚本文件运行权限并运行脚本文件。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

### 部署MySQL数据库

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。

在主服务器Master上打开/etc/mysql/mysql.conf.d/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=129。接着使用sudo service mysql restart命令重启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，具体命令为mysql> create user 'leidongchen'@'%' identified by 'root';mysql> grant replication slave on \*.\* to 'leidongchen'@'%';mysql> flush privileges; 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。

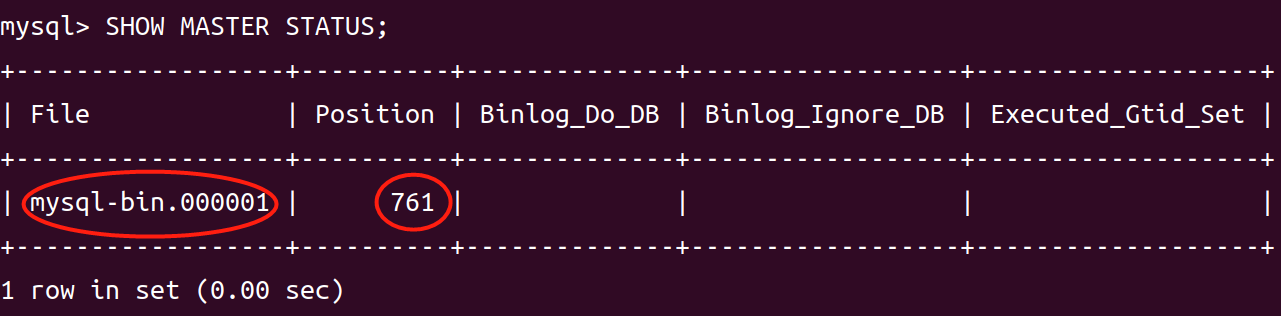


图7 主服务器数据库的二进制信息

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

在从服务器Slave上，修改MySQL的配置文件后，重启MySQL服务。然后打开MySQL，然后输入命令以修改从服务器的复制进程，将其连接到主服务器。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，然后使用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。正常情况下如图8所示。

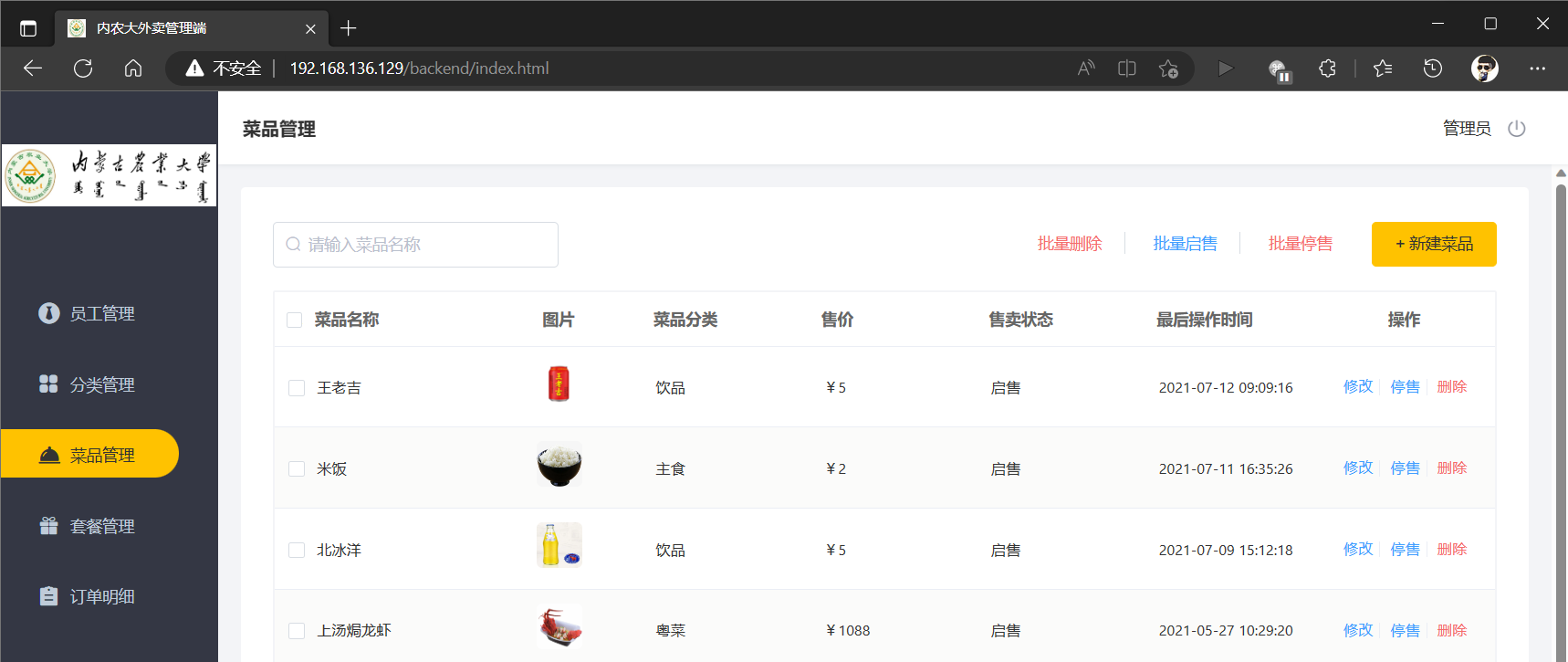


图8 后端管理平台预览图

Fig.8 Backend management platform preview image

### 部署redis数据库

首先在application.yml中添加如下配置来配置redis：

redis:  
 host: 192.168.136.131  
 port: 6379  
 password: root  
 database: 0

然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。项目全部部署成功后的预览图如图9。

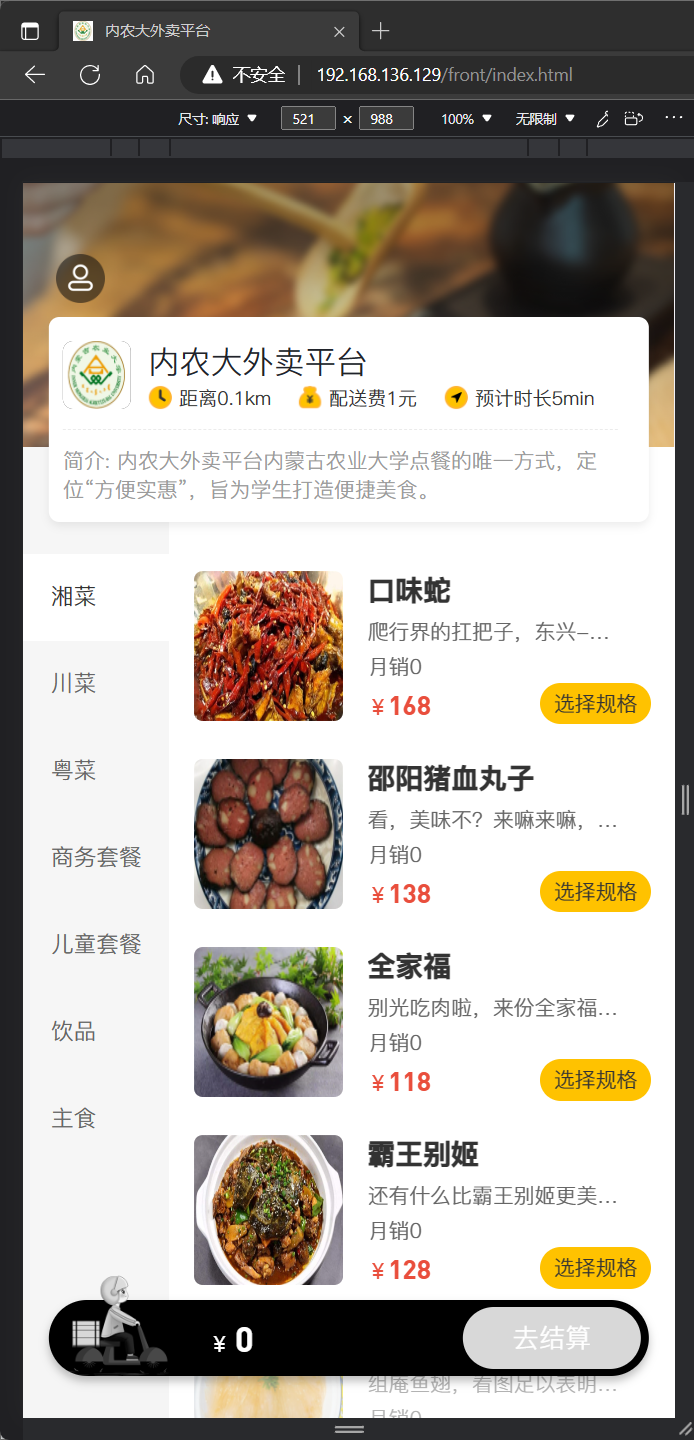


图9 项目点餐页面预览图

Fig.9 Project ordering page preview image.

# 系统测试

## 测试目标和策略

在进行系统测试时，我们需要验证系统的功能是否符合需求和设计规格，确保系统的每个功能、输入、输出和操作行为都满足预期。同时，还需要验证系统的性能、可靠性和安全性是否符合需求和设计规格。具体来说，我们需要测试系统的响应时间、吞吐量、并发性、负载能力和稳定性等性能指标，以确保系统能够在不同的负载条件下正常运行；测试系统的容错能力、恢复能力、可用性和可维护性等可靠性指标，以确保系统能够保持稳定运行，并在故障发生时快速恢复；测试系统的数据保护、访问控制、认证和授权等安全机制，以确保系统能够保护用户的数据和隐私。通过对系统功能、性能、可靠性和安全性的全面测试，我们可以发现和解决潜在的问题，确保系统的稳定和可靠运行。

为了保证系统的质量和稳定性，我们采用黑盒测试方法，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。我们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，我们采用自动化测试工具和手动测试相结合的方法。我们利用 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，我们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。我们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，我们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。

## 执行测试

本测试于2023年5月1日星期一执行，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。

### 测试登录功能

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10所示。

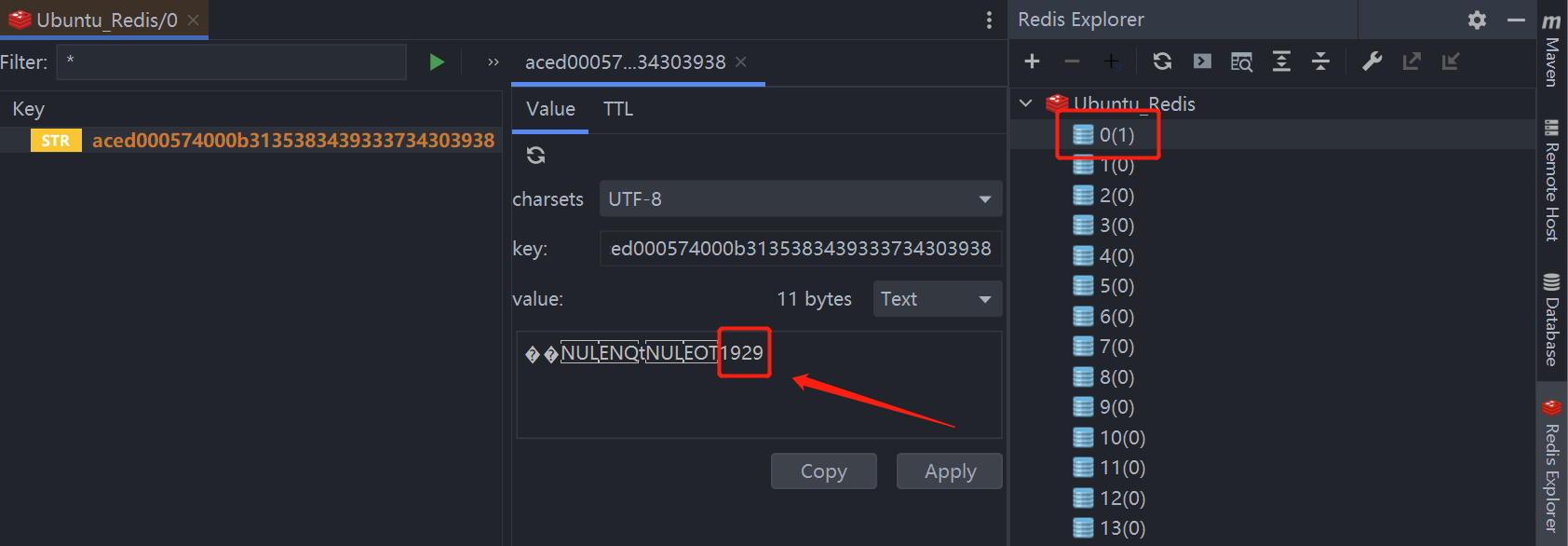


图10 redis服务器存储的手机验证码

Fig.10 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11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。

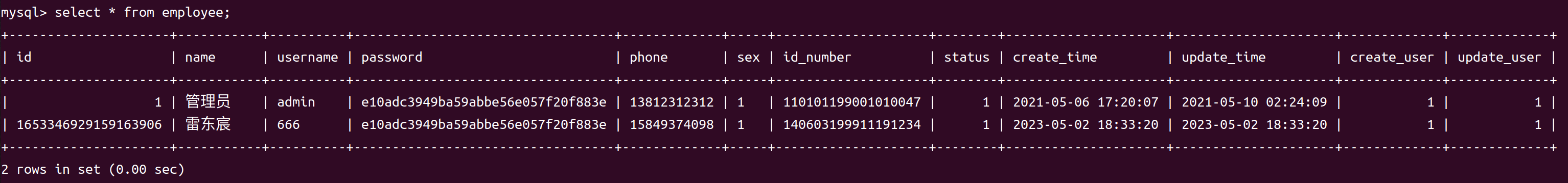


图11 数据库员工表信息

Fig.11 Employee table information in the database.

### 测试菜品管理功能

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 |  |  |
| 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” |  |  |
| 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 |  |  |
| 功能测试 | 删除菜品 | 选择一道不存在的菜品进行删除操作 | 提示“该菜品不存在” |  |  |
| 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 |  |  |
| 功能测试 | 修改菜品 | 选择一道不存在的菜品进行修改操作 | 提示“该菜品不存在” |  |  |
| 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“请正确填写信息” |  |  |
| 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 |  |  |
| 安全性测试 | 数据保护 | 在添加菜品页面添加一道菜品并保存，然后退出系统并重新登录，查看是否仍能看到已添加的菜品 | 已添加的菜品不能被查看 |  |  |
| 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” |  |  |

# 致 谢

# 参 考 文 献

* 1. 丘明. 中国外卖市场数据分析：2022年中国在线外卖市场规模增长率为19.80%

附 录