

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](#_Toc133052880)

[1.1 SSM概述 1](#_Toc133052881)

[1.2 研究背景和意义 2](#_Toc133052882)

[1.3 国内外研究现状 3](#_Toc133052883)

[1.3.1 国外研究现状 3](#_Toc133052884)

[1.3.2 国内研究现状 3](#_Toc133052885)

[2 开发工具介绍 4](#_Toc133052886)

[2.1 IntelliJ IDEA 2022.1 4](#_Toc133052887)

[2.2 SQLyog 4](#_Toc133052888)

[2.3 VMware Workstation pro 4](#_Toc133052889)

[2.4 Redis Desktop Manager 5](#_Toc133052890)

[3 系统分析 6](#_Toc133052891)

[3.1 需求分析 6](#_Toc133052892)

[3.1.1 需求概述 6](#_Toc133052893)

[3.1.2 功能需求 6](#_Toc133052894)

[3.1.3 非功能需求 6](#_Toc133052895)

[3.1.4 系统接口需求分析 6](#_Toc133052896)

[3.1.4 数据需求分析 7](#_Toc133052897)

[3.1.5 安全性需求分析 7](#_Toc133052898)

[3.2 数据分析 8](#_Toc133052899)

[3.2.1 用户数据分析 8](#_Toc133052900)

[3.2.2 订单数据分析 8](#_Toc133052901)

[3.2.3 菜品数据分析 8](#_Toc133052902)

[3.3 设计约束分析 8](#_Toc133052903)

[3.4 过程分析 9](#_Toc133052904)

[3.4.1 用户下单流程 9](#_Toc133052905)

[3.4.2 平台运营流程 9](#_Toc133052906)

[3.4.3 决策流程 9](#_Toc133052907)

[3.5 环境分析 9](#_Toc133052908)

[3.5.1 外部环境分析 9](#_Toc133052909)

[3.5.2 内部环境分析 9](#_Toc133052910)

[3.6 风险分析 10](#_Toc133052911)

[3.7 成本效益分析 10](#_Toc133052912)

[3.7.1 成本方面 10](#_Toc133052913)

[3.7.2 效益方面 10](#_Toc133052914)

[3.8 总结 11](#_Toc133052915)

[4 系统设计 12](#_Toc133052916)

[4.1 系统架构设计 12](#_Toc133052917)

[4.2 数据库设计 12](#_Toc133052918)

[4.2.1 用户表（User）：存储用户基本信息 12](#_Toc133052919)

[4.2.2 地址表（address\_book）：存储用户地址信息 12](#_Toc133052920)

[4.2.3 菜品分类表（Category）：存储菜品分类信息 13](#_Toc133052921)

[4.2.4 菜品表（Dish）：存储菜品基本信息 13](#_Toc133052922)

[4.2.5 菜品口味表（DishFlavor）：存储菜品口味信息 13](#_Toc133052923)

[4.2.6 订单表（Order）：存储订单基本信息 13](#_Toc133052924)

[4.2.7 订单明细表（OrderDetail）：存储订单详细信息 14](#_Toc133052925)

[4.2.8 员工表（Employee）：存储员工基本信息 14](#_Toc133052926)

[4.2.9 套餐表（Setmeal）：存储套餐基本信息 14](#_Toc133052927)

[4.2.10 套餐详细信息表（SetmealDish）：存储套餐详细信息 14](#_Toc133052928)

[4.2.11 购物车表（ShoppingCart）：存储购物车信息 15](#_Toc133052929)

[4.3 接口设计 15](#_Toc133052930)

[4.3.1 用户与员工信息管理 15](#_Toc133052931)

[4.3.2 菜品管理 15](#_Toc133052932)

[4.3.3 分类管理 16](#_Toc133052933)

[4.3.4 套餐管理 16](#_Toc133052934)

[4.3.5 购物车 17](#_Toc133052935)

[4.3.6 订单管理 17](#_Toc133052936)

[4.4 系统性能设计 17](#_Toc133052937)

[4.4.1 硬件配置 17](#_Toc133052938)

[4.4.2 负载均衡 18](#_Toc133052939)

[4.4.3 数据库 18](#_Toc133052940)

[4.4.4 缓存设计 18](#_Toc133052941)

[4.5 用户界面设计 19](#_Toc133052942)

[4.5.1 后台管理界面 19](#_Toc133052943)

[4.5.2 用户界面设计 19](#_Toc133052944)

[5 系统实现 20](#_Toc133052945)

[5.1 环境配置 20](#_Toc133052946)

[5.1.1 IntelliJ IDEA 20](#_Toc133052947)

[5.1.2 服务器配置 22](#_Toc133052948)

[5.2 模块开发 23](#_Toc133052949)

[5.2.1 登录模块 23](#_Toc133052950)

[5.2.2 菜品管理模块 25](#_Toc133052951)

[5.2.3 订单管理模块 26](#_Toc133052952)

[5.2.4 购物车管理模块 27](#_Toc133052953)

[5.2.5 地址管理模块 29](#_Toc133052954)

[6 系统测试 31](#_Toc133052955)

[致 谢 32](#_Toc133052956)

[参 考 文 献 33](#_Toc133052957)

# 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

#### 3.1.2.1 用户端需求

1. 用户注册：用户可以通过提供必要的个人信息注册平台。
2. 用户登录：已注册用户可以通过用户名和密码登录平台。
3. 在线点餐：用户可以在平台上选择餐品、添加到购物车中，并下订单。
4. 订单管理：用户可以查看历史订单、修改订单、取消订单、评价订单等。
5. 用户管理：用户可以修改个人信息、查看积分、查看收藏等。
6. 支付配送：用户可以选择支付方式并填写配送信息，以完成订单。

#### 3.1.2.2 商家端需求

1. 商家注册：商家可以通过提供必要的企业信息注册平台。
2. 商家登录：已注册商家可以通过用户名和密码登录平台。
3. 套餐管理：商家可以添加套餐、修改套餐、删除套餐等。
4. 订单管理：商家可以查看历史订单、修改订单状态、处理订单、评价订单等。
5. 商家管理：商家可以修改个人信息、查看销售统计、查看评价等。

### 3.1.3 非功能需求

#### 3.1.3.1 性能需求

1. 在高峰期，平台应能够处理大量的并发请求。
2. 平台应能够快速响应用户请求。
3. 平台应能够支持大量的用户和商家同时使用。

#### 3.1.3.2 可用性需求

1. 平台应该具有高可靠性和可用性，确保24/7的服务可用性。
2. 平台应能够处理崩溃和系统故障，确保数据不会丢失。

### 3.1.4 系统接口需求分析

1. 用户接口需求：

描述系统对用户的功能需求，包括用户登录、注册、个人信息管理、购物车、订单管理、支付等功能。例如，用户需要能够通过注册功能注册成为平台会员，能够在平台上浏览商品信息，将商品添加到购物车中，能够进行下单并支付等。

1. 商家接口需求：

描述系统对商家的功能需求，包括商家登录、商铺管理、商品管理、订单管理等功能。例如，商家需要能够通过商家登录功能登录平台，管理自己的商铺信息、发布商品信息，查看订单信息等。

1. 第三方接口需求：

描述系统需要与第三方平台集成的接口需求，包括短信接口、支付接口等。例如，系统需要与支付宝等支付平台集成，实现在线支付功能。

1. 系统接口规范：

描述系统接口的规范、协议、数据格式等，以确保不同系统之间的互操作性和兼容性。例如，系统接口需要采用RESTful接口规范，使用JSON格式传输数据等。

### 数据需求分析

#### 3.1.4.1 业务需求

1. 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。
2. 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。
3. 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。

#### 3.1.4.2 数据质量需求

1. 顾客下单信息需要准确无误，确保订单不会出现漏单或错单情况。
2. 餐厅订单信息需要完整，确保每个订单都有正确的商品信息和处理状态。
3. 配送员行程信息需要可靠，确保送餐路线和送达时间不会出现错误。

#### 3.1.4.3 数据处理需求

1. 需要对顾客和餐厅的信息进行整合，以便后续数据分析和业务决策。
2. 需要对订单信息进行归类和统计，生成销售报表和营收分析报告。
3. 需要对配送员的行程进行路线优化，提高配送效率和服务质量。

#### 3.1.4.4 安全和隐私需求

1. 需要对用户的个人信息进行保护，确保订单数据不会被非法获取和使用。
2. 需要对数据进行加密和备份，确保数据在采集、传输、存储和处理过程中不会泄露、篡改或丢失。

#### 3.1.4.5 报表和分析需求

1. 需要对订单数据进行统计和分析，包括订单量、销售额、平均客单价、订单满意度等指标。
2. 需要生成销售报表和营收分析报告，用于评估业务运营状况和制定营销策略。
3. 需要进行数据挖掘和机器学习，以便预测用户行为、优化服务流程等。

### 安全性需求分析

1. 平台应该具有高度的安全性，确保用户和商家的个人信息得到保护。
2. 用户和商家在登录、注册、支付等过程中的数据应该进行加密传输。
3. 系统应该能够抵御常见的网络攻击，例如SQL注入、跨站点脚本攻击等。

## 数据分析

### 3.2.1 用户数据分析

1. 用户地域分布

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。

1. 用户行为分析

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。

1. 用户评价分析

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

1. 订单趋势分析

分析订单的时间、数量和金额趋势，以便优化平台的运营策略。

1. 订单类型分析

分析不同类型的订单，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。

1. 订单满意度分析

分析订单的满意度、退单率和投诉情况，以便发现平台服务的问题并及时改进。

### 3.2.3 菜品数据分析

1. 菜品销售排行分析

分析不同菜品的销售数量和销售额排行，以便了解用户的消费偏好和推广热门菜品。

1. 菜品评价分析

分析用户对菜品的评价、评分和评价频率，以便发现菜品的优劣和改进空间。

1. 菜品成本分析

分析菜品的成本、售价和毛利润等，以便优化菜品的定价和利润率。

## 设计约束分析

1. 技术约束

系统必须基于Spring Boot和Mybatis-Plus框架进行开发，使用MySQL数据库存储数据，使用Redis实现缓存等。

1. 性能约束

系统必须满足高并发、低延迟等性能要求，同时需要考虑系统的可扩展性和稳定性。

1. 安全约束

系统必须保证用户信息和数据的安全性，防止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

1. 用户登录平台或注册账号，只要输入手机号，发送验证码并填写正确后自动注册。
2. 用户浏览菜单，选择菜品并添加到购物车中。
3. 用户进入结算页面，填写收货地址、联系方式和支付方式等信息。
4. 用户确认订单并完成支付。
5. 平台将订单信息推送给餐厅，餐厅开始准备菜品。
6. 餐厅完成菜品准备，将菜品交给配送员。
7. 配送员根据订单信息送达用户所在地，并将菜品交给用户。
8. 用户收到菜品后确认收货并对菜品进行评价。

### 3.4.2 平台运营流程

1. 平台接收用户下单信息并推送给餐厅。
2. 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。
3. 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。
4. 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。

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。

平台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。

## 环境分析

### 3.5.1 外部环境分析

1. 政治环境：政策法规对外卖服务行业的影响，政府对商业税收和监管等方面的政策影响。
2. 经济环境：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因素。
3. 社会环境：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。
4. 技术环境：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，以及数据隐私和信息安全等方面的问题。

### 3.5.2 内部环境分析

1. 人力资源：平台人员的素质和能力水平，以及组织架构和职责分工等方面的因素。
2. 资金：平台融资和资金运营情况，包括资金来源、投入和收益等方面的因素。
3. 技术：平台技术的研发和应用情况，包括技术创新、应用效果和维护等方面的因素。
4. 市场营销：平台品牌建设、市场推广和客户维护等方面的因素，包括广告宣传、促销活动和客户服务等方面的因素。

## 风险分析

1. 供应链风险

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。

1. 数据安全风险

平台所涉及的大量客户和商家信息，包括个人信息、财务信息等，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。

1. 法律合规风险

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。

1. 市场竞争风险

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。

1. 用户需求变化风险

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。

1. 战略决策风险

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。

1. 预算控制风险

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。

## 成本效益分析

### 3.7.1 成本方面

1. 技术开发成本

平台需要投入大量资金来开发和维护系统，包括软件开发人员工资、设备购买、服务器租用等。

1. 营销和推广成本

平台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。

1. 运营成本

平台需要支付员工工资、租金、水电费等日常开支，同时需要维护和管理供应链、配送网络等方面的费用。

### 效益方面

1. 用户体验

平台通过提供快捷、便利的外卖服务，提升了用户的购物体验，进而增加了用户满意度和留存率。

1. 品牌影响力

通过营销和推广活动，平台能够增强品牌的知名度和影响力，进而增加商家入驻平台的意愿，形成良性循环。

1. 营业收入

平台通过收取商家佣金和配送费用等方式获得收益，随着平台规模的扩大和用户数量的增加，平台的营业收入也会不断增加。

## 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

1. 客户端

包括Web端和移动端，提供用户注册、登录、下单、支付等功能。

1. 服务器

包括应用服务器、数据库服务器、缓存服务器等，承担平台业务逻辑处理和数据存储。

1. 第三方接口

包括支付接口、短信接口、地图接口等，提供与外部系统的数据交互。

1. 供应商接口

包括餐厅、骑手、物流公司等接口，提供与供应商之间的数据交互。

1. 消息队列

用于异步消息通信，提高系统性能和可扩展性。

1. 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，保障系统的安全性和可靠性。

## 数据库设计

### 用户表（User）：存储用户基本信息

* 用户ID（UserID）
* 用户名（UserName）
* 性别（sex）
* 头像（avatar）
* 手机号码（PhoneNumber）
* 邮箱（Email）
* 注册时间（RegisterTime）

### 地址表（address\_book）：存储用户地址信息

* 用户ID（UserID）
* 收货人姓名（Consignee）
* 手机号码（PhoneNumber）
* 省级名称（ProvinceName）
* 市级名称（CityName）
* 区级名称（DistrictName）
* 详细地址（Detail）
* 地址标签（Label）
* 是否默认地址（IsDefault）
* 创建时间（CreateTime）

### 菜品分类表（Category）：存储菜品分类信息

* 分类类型（Type）：1为菜品分类，2为套餐分类
* 菜品名称（Name）
* 顺序（Sort）
* 创建时间（CreateTime）
* 修改时间（UpdateTime）
* 创建人（CreateUser）
* 修改人（UpdateUser）

### 菜品表（Dish）：存储菜品基本信息

* + 菜品ID（DishID）
  + 菜品名（Name）
  + 分类ID（CategoryID）
  + 菜品价格（Price）
  + 菜品码（Code）
  + 菜品图片（Image）
  + 菜品描述（Description）
  + 售卖状态（Status）：1表示在售，0表示不在售
  + 创建时间（CreateTime）
* 修改时间（UpdateTime）
* 创建人（CreateUser）
* 修改人（UpdateUser）

### 菜品口味表（DishFlavor）：存储菜品口味信息

* 菜品ID（DishID）
* 口味名称（Name）
* 口味数据（Value）
* 创建时间（CreateTime）
* 修改时间（UpdateTime）
* 创建人（CreateUser）
* 修改人（UpdateUser）

### 订单表（Order）：存储订单基本信息

* 订单ID（OrderID）
* 订单号（Number）
* 订单状态（Status）：1表示待付款，2表示待派送，3表示派送中，4表示已完成，5表示已取消。
* 用户ID（UserID）
* 地址（AddressID）
* 支付方式（PaymentMethod）
* 实收金额（Amount）
* 下单时间（OrderTime）
* 备注（Remark）

### 订单明细表（OrderDetail）：存储订单详细信息

* 明细表ID（ID）
* 订单ID （OrderID）
* 菜品ID（DishID）
* 菜品名（Name）
* 套餐ID（SetmealID）
* 菜品预览图（Image）
* 金额（Amount）

### 员工表（Employee）：存储员工基本信息

* 员工ID（ID）
* 员工姓名（Name）
* 密码（Password）
* 手机号码（Phone）
* 性别（Sex）
* 身份证号（IDNumber）
* 状态（Status）：0禁用，1正常
* 创建时间（CreateTime）
* 修改时间（UpdateTime）
* 创建人（CreateUser）
* 修改人（UpdateUser）

### 套餐表（Setmeal）：存储套餐基本信息

* 套餐ID（ID）
* 分类ID（CategoryID）
* 姓名（Name）
* 套餐价格（Price）
* 售卖状态（Status）：1启售，0停售
* 描述信息（Description）
* 套餐预览图（Image）
* 创建时间（CreateTime）
* 修改时间（UpdateTime）
* 创建人（CreateUser）
* 修改人（UpdateUser）

### 套餐详细信息表（SetmealDish）：存储套餐详细信息

* 套餐ID（SetmealID）
* 菜品ID（DishID）
* 菜品名（Name）
* 价格（Price）
* 创建时间（CreateTime）
* 修改时间（UpdateTime）
* 创建人（CreateUser）
* 修改人（UpdateUser）

### 购物车表（ShoppingCart）：存储购物车信息

* 菜品ID（ID）
* 菜品名（Name）
* 预览图象（Image）
* 用户ID（UserID）
* 套餐ID（SetmealID）
* 口味（DishFlavor）
* 数量（Number）
* 金额（Amount）
* 创建时间（CreateTime）

## 接口设计

### 用户与员工信息管理

1. 员工登录接口：POST /employee/login

请求参数：HttpServletRequest对象、Employee实体

响应数据：员工ID、员工密码password、员工状态status

1. 员工信息展示接口：GET /employee/page

请求参数：当前页数page、每一页大小pageSize、员工姓名name

响应数据：员工信息

1. 新增员工接口：POST /employee/save

请求参数：Employee实体

响应数据：“新增员工成功”

1. 用户登录/注册接口：POST /user/login

请求参数：存储用户实体的map集合、session对象

响应数据：手机号、验证码

### 菜品管理

1. 新增菜品：POST /dish

请求参数：Dish实体类（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）

响应数据：“新增菜品成功”

1. 菜品信息分页查询：GET /dish/page

请求参数：当前页数page、每一页大小pageSize、菜品名字name

响应数据：菜品列表、菜品信息

1. 根据id查询菜品信息和对应的口味信息：GET /dish/{id}

请求参数：菜品ID

响应数据：菜品信息

1. 修改菜品：PUT /dish/update

请求参数：Dish实体类

响应数据：支付状态、订单状态

1. 根据条件查询对应的菜品数据：GET /dish/list

请求参数：Dish实体类

响应数据：菜品信息

### 分类管理

1. 新增分类：POST /category

请求参数：Category实体类（分类名称、顺序、创建时间、更新时间、创建人、修改人）

响应数据：“新增分类成功”

1. 分页查询分类：GET /category/page

请求参数：当前页数page、每页大小pageSize

响应数据：分类信息

1. 删除分类：DELETE /category

请求参数：分类ID

响应数据：“分类信息删除成功”

1. 根据条件查询分类数据：DELETE /category/list

请求参数：Category实体

响应数据：分类信息

### 套餐管理

1. 新增套餐：POST /setmeal

请求参数：Category实体类（分类ID、套餐名称、套餐价格、状态、编码、描述信息、图片）

响应数据：“新增套餐成功”

1. 套餐分页查询：GET /setmeal /page

请求参数：当前页数page、每一页大小pageSize、套餐名字name

响应数据：套餐信息

1. 删除套餐：DELETE /setmeal

请求参数：套餐ID

响应数据：“套餐数据删除成功”

1. 根据条件查询套餐数据：GET /setmeal /list

请求参数：当前页数page、每一页大小pageSize、套餐名字name

响应数据：套餐信息

1. 启售或停售套餐：POST /setmeal /0

请求参数：套餐ID

响应数据：套餐信息

### 购物车

1. 添加购物车：POST /shoppingCart/add

请求参数：购物车实体类（名称、用户ID、菜品ID、套餐ID、口味、数量、金额、图片、添加时间）

响应数据：购物车信息

1. 查看购物车：GET /shoppingCart/list

请求参数：无

响应数据：购物车信息

1. 清空购物车：GET /shoppingCart/clean

请求参数：无

响应数据：“清空购物车成功”

### 订单管理

1. 用户下单：POST /order/submit

请求参数：订单实体类（订单号、订单状态（待付款、待派送、已派送、已完成、已取消））

响应数据：“下单成功”

1. 订单明细：REQUEST /orderDetail

请求参数：无

响应数据：外卖名称、订单ID、菜品ID、套餐ID、口味、数量、金额、图片

## 系统性能设计

### 4.4.1 硬件配置

1. 主数据库服务器：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。因为主数据库服务器需要更强的处理能力、更大的内存和更快的磁盘速度，以应对高并发的写操作。
2. 从数据库服务器：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。因为从数据库服务器需要较大的内存和足够快的磁盘速度，以支持高并发的读操作。
3. 负载均衡服务器：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）。因为负载均衡服务器需要足够强的处理能力和网络带宽，以处理高流量的请求并快速分发请求。
4. 应用程序服务器：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）。因为应用程序服务器需要足够的内存和处理能力，以保证应用程序的快速响应和高并发处理。

### 负载均衡

1. 高并发支持：平台需要支持大量的并发请求，负载均衡器需要能够快速、准确地分发请求，确保每台应用服务器的负载尽可能平均。
2. 高可靠性：负载均衡器作为整个系统的入口，必须保证高可靠性，避免单点故障，当出现故障时能够快速自动切换到备份负载均衡器。
3. 网络带宽：负载均衡器需要足够的网络带宽，以支持高并发的请求和响应，避免网络瓶颈成为性能瓶颈。
4. 高性能：负载均衡器需要具有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，以保证平台的响应速度和稳定性。
5. 易扩展性：当平台的规模增大时，需要能够方便地扩展负载均衡器，以支持更高的并发量和更多的应用服务器。

### 数据库

1. 数据库分库分表：在高并发的情况下，数据库容易成为瓶颈，采用分库分表可以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。
2. 索引优化：建立合理的索引可以提高查询效率，减少全表扫描的时间。
3. SQL语句优化：编写高效的SQL语句可以减少数据库的查询时间，提高查询效率。
4. 内存优化：增加数据库内存缓存可以提高查询效率。
5. 读写分离：将读操作和写操作分离，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。

### 缓存设计

1. 缓存热点数据：例如将手机验证码放入Redis缓存，减轻数据库的访问压力，提高系统性能。
2. 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，避免缓存中的数据过期而导致脏数据的产生。

## 用户界面设计

### 4.5.1 后台管理界面

1. 导航菜单设计

* 侧边栏菜单：位于页面左侧，包含不同的功能模块，例如员工管理、菜品管理、订单明细等。
* 顶部栏：包括当前操作的模块名，当前的登录用户

1. 具体模块设计

* 输入框：可以输入相关条件进行全文搜索。
* 按钮：可以用来添加信息。
* 表单：包含每个模块的一些具体信息。
* 翻页：当信息在一页展示不完时，可以分页展示。

### 用户界面设计

1. 商家详情页设计

* 商家信息：展示商家图片、名称、销售量、距离、配送费、预计时长等基本信息。
* 商品列表：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，支持选择规格并添加到购物车。
* 商家公告：展示商家公告和优惠活动信息。
* 底部栏：购物车、结算等功能入口。

1. 购物车设计

* 商品列表：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：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：包括备注、支付等功能入口。

1. 个人中心设计

* 我的订单：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，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：退出登录。

# 系统实现

## 环境配置

### IntelliJ IDEA

1. 创建Spring Initializr项目
2. 设置项目名：takeout\_service
3. 选择项目类型：maven
4. 选择jdk版本：JDK1.8.0\_291
5. 选择Spring Boot版本：3.0.5
6. 添加Spring Web和jdbc API工具
7. 编辑pom.xml文档，添加依赖
8. 添加mybatis-plus依赖

<dependency>  
 <groupId>com.baomidou</groupId>  
 <artifactId>mybatis-plus-boot-starter</artifactId>  
 <version>3.5.2</version>  
</dependency>

1. 增强 Java 编程语言的核心类库

<dependency>  
 <groupId>commons-lang</groupId>  
 <artifactId>commons-lang</artifactId>  
 <version>2.6</version>  
</dependency>

1. 添加Lombok库依赖

<dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
</dependency>

1. 添加mysql依赖

<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <scope>runtime</scope>  
</dependency>

1. 添加德鲁伊数据库连接池依赖

<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid-spring-boot-starter</artifactId>  
 <version>1.2.15</version>  
</dependency>

1. 编辑application.yml文档
2. 设置服务器端口号

server:  
 port: 8080

1. 设置spring

spring:  
 application:  
 name: takeout\_service  
 datasource:  
 druid:  
 driver-class-name: com.mysql.cj.jdbc.Driver  
 url: jdbc:mysql://localhost:3306/imautakeoutservice?serverTimezone=Asia/Shanghai&useUnicode=true&characterEncoding=utf-8&zeroDateTimeBehavior=convertToNull&useSSL=false&allowPublicKeyRetrieval=true  
 username: root  
 password: root  
 redis:  
 host: 127.0.0.1  
 port: 6379  
 password:  
 database: 0  
 cache:  
 redis:  
 time-to-live: 1800000 *#设置缓存的过期时间*

1. 设置mybatis

mybatis-plus:  
 configuration:  
 *# 映射实体时，下划线命名和驼峰命名对应* map-underscore-to-camel-case: *true* log-impl: org.apache.ibatis.logging.stdout.StdOutImpl  
 global-config:  
 db-config:  
 id-type: ASSIGN\_ID

1. 编辑Spring Boot启动类
2. 添加扫描驱动的注解：@ServletComponentScan
3. 添加事务管理注解：@EnableTransactionManagement
4. 添加Lombok日志注解：@Slf4j

### 服务器配置

1. 安装JDK
2. 官网下载jdk1.8.0\_371
3. 使用如下命令解压安装包:

tar -zxvf jdk-8u371-linux-x64.tar.gz -C /usr/local/

1. 配置环境变量，编辑/etc/profile文件，添加如下配置到文末：

JAVA\_HOME=/usr/local/jdk1.8.0\_371

JRE\_HOME=/usr/local/jdk1.8.0\_371/jre

PATH=$JAVA\_HOME/bin:$PATH

1. 查看是否安装成功，输入Java -version：

java version "1.8.0\_371"

Java(TM) SE Runtime Environment (build 1.8.0\_371-b11)

Java HotSpot(TM) 64-Bit Server VM (build 25.371-b11, mixed mode)

1. 安装Tomcat
2. 官网下载tomcat8.5.88
3. 使用如下命令解压安装包：

tar -zxvf apache-tomcat-8.5.88.tar.gz -C /usr/local/

1. 启动tomcat:输入命令./bin/startup.sh，出现如下则安装成功：

Using CATALINA\_BASE: /usr/local/apache-tomcat-8.5.88

Using CATALINA\_HOME: /usr/local/apache-tomcat-8.5.88

Using CATALINA\_TMPDIR: /usr/local/apache-tomcat-8.5.88/temp

Using JRE\_HOME: /usr/local/jdk1.8.0\_371/jre

Using CLASSPATH: /usr/local/apache-tomcat-8.5.88/bin/bootstrap.jar:/usr/local/apache-tomcat-8.5.88/bin/tomcat-juli.jar

Using CATALINA\_OPTS:

Tomcat started.

1. 配置mysql
2. 安装命令：sudo apt install mysql-server
3. 登录：sudo mysql -u root -p
4. 配置主从复制功能，在后文详述
5. 安装Nginx
6. 安装命令：sudo apt-get install nginx
7. 启动：sudo systemctl start nginx
8. 添加80端口：sudo ufw allow 80/tcp
9. 安装maven

sudo apt-get install maven

1. 安装redis
2. 安装命令：sudo apt-get install -y redis-server
3. 设置密码：打开/etc/redis/redis.conf文件

修改requirepass root

1. 重启服务：service redis restart
2. 进入redis：redis-cli

输入密码：auth root

1. 安装git
2. sudo apt-get install git
3. 生成SSH密钥：ssh-keygen -t rsa -b 4096 -C "[248107148@qq.com](mailto:248107148@qq.com)"
4. 查看公钥：cat ~/.ssh/id\_rsa.pub
5. 将公钥添加到github账户：

复制公钥，然后在GitHub网站上打开你的账户设置页面。选择“SSH and GPG keys”，然后点击“New SSH Key”。在“Key”字段中粘贴你复制的公钥，并为该密钥提供一个描述性名称。最后，点击“Add SSH Key”按钮。

1. 配置git全局变量

git config --global user.name "Ray"

git config --global user.email "[248107148@qq.com](mailto:248107148@qq.com)"

## 模块开发

### 登录模块

1. 创建登录页面：创建一个登录页面，包含用户名和密码输入框，以及登录按钮。

<el-form-item *prop*="username">  
 <el-input *v-model*="loginForm.username" *type*="text" *auto-complete*="off" *placeholder*="账号" *maxlength*="20"  
 *prefix-icon*="iconfont icon-user" />  
</el-form-item>  
<el-form-item *prop*="password">  
 <el-input *v-model*="loginForm.password" *type*="password" *placeholder*="密码" *prefix-icon*="iconfont icon-lock" *maxlength*="20"  
 *@keyup.enter.native*="handleLogin" />  
</el-form-item>  
<el-form-item *style*="width:100%;">  
 <el-button *:loading*="loading" *class*="login-btn" *size*="medium" *type*="primary" *style*="width:100%;"  
 *@click.native.prevent*="handleLogin">  
 <span *v-if*="!loading">登录</span>  
 <span *v-else*>登录中...</span>  
 </el-button>  
</el-form-item>

1. 创建Controller：创建一个Controller类，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。

*@PostMapping*("/login")  
 *public* R<User> login(*@RequestBody* Map map, HttpSession session){  
 log.info(map.toString());String phone = map.get("phone").toString();String code = map.get("code").toString();Object codeInSession = redisTemplate.opsForValue().get(phone); *if*(codeInSession != *null* && codeInSession.equals(code)){LambdaQueryWrapper<User> queryWrapper = *new* LambdaQueryWrapper<>();  
 queryWrapper.eq(User::getPhone,phone);  
 User user = userService.getOne(queryWrapper);  
 *if*(user == *null*){user = *new* User();  
 user.setPhone(phone);  
 user.setStatus(1);  
 userService.save(user);  
 }  
 session.setAttribute("user",user.getId());redisTemplate.delete(phone);  
 *return* R.success(user);  
 }  
 *return* R.error("登录失败");  
 }

1. 创建DAO：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。
2. 配置数据源：在项目中配置数据源，以便连接到数据库。
3. 配置MyBatis-Plus：在项目中配置MyBatis-Plus，以便连接到数据库。

*public interface* UserService *extends* IService<User> {  
}

1. 配置登录拦截器：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。

*@Override  
public void* doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) *throws* IOException, ServletException {  
 HttpServletRequest request = (HttpServletRequest) servletRequest;  
 HttpServletResponse response = (HttpServletResponse) servletResponse;String requestURI = request.getRequestURI(); *//* String[] urls = *new* String[]{ *// 定义不需要处理的请求* "/employee/login",  
 "/employee/logout",  
 "/backend/\*\*",  
 "/front/\*\*",  
 "/common/\*\*",  
 "/user/sendMsg", *// 移动端发送短信* "/user/login" *// 移动端登录* }; *boolean* check = check(urls, requestURI); *if* (check){  
 filterChain.doFilter(request,response);  
 *return*;  
 } *if*(request.getSession().getAttribute("employee") != *null*){Long empId = (Long) request.getSession().getAttribute("employee");  
 BaseContext.setCurrentId(empId);  
 filterChain.doFilter(request,response);  
 } *if*(request.getSession().getAttribute("user") != *null*){Long userId = (Long) request.getSession().getAttribute("user");  
 BaseContext.setCurrentId(userId);  
 filterChain.doFilter(request,response);  
 *return*;  
 }

response.getWriter().write(JSON.toJSONString(R.error("NOTLOGIN")));}

1. 注销功能：在用户注销时，删除存储在Redis中的登录token。

*@PostMapping*("/logout")  
*public* R<String> logout(HttpServletRequest request){  
 request.getSession().removeAttribute("employee");

*return* R.success("退出登录成功");  
}

### 菜品管理模块

1. 添加菜品

在菜品管理模块中，商家可以添加新的菜品。这可以通过在DishMapper中实现一个插入菜品的方法来实现。例如，可以编写一个插入菜品的方法：

public interface DishMapper extends BaseMapper<Dish> {

}

在菜品管理模块的控制器中，可以通过调用DishMapper中的insert()方法来插入新的菜品。例如，可以编写一个添加菜品的方法：

@PostMapping("/addDish")

public String addDish(@ModelAttribute("dish") Dish dish) {

dishService.save(dish);

return "redirect:/dish";

}

在上述方法中，首先将商家添加的菜品保存到数据库中，然后重定向到菜品列表页面。

1. 修改菜品

在菜品管理模块中，商家可以修改已有的菜品。这可以通过在DishMapper中实现一个更新菜品的方法来实现。例如，可以编写一个更新菜品的方法：

public interface DishMapper extends BaseMapper<Dish> {

}

在菜品管理模块的控制器中，可以通过调用DishMapper中的updateById()方法来更新菜品。例如，可以编写一个修改菜品的方法：

@PostMapping("/updateDish")

public String updateDish(@ModelAttribute("dish") Dish dish) {

dishService.updateById(dish);

return "redirect:/dish";

}

在上述方法中，首先将商家修改后的菜品保存到数据库中，然后重定向到菜品列表页面。

1. 删除菜品

在上述方法中，首先将商家修改后的菜品保存到数据库中，然后重定向到菜品列表页面。

public interface DishMapper extends BaseMapper<Dish> {

int deleteByDishId(Long dishId);

}

在菜品管理模块的控制器中，可以根据前端传递过来的菜品ID调用DishMapper中的deleteByDishId()方法来删除菜品。例如，可以编写一个删除菜品的方法：

@PostMapping("/deleteDish")

public String deleteDish(@RequestParam("dishId") Long dishId) {

dishService.removeById(dishId);

return "redirect:/dish";

}

### 订单管理模块

1. 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。
2. 在后端代码中创建Order实体类，用于与数据库中的订单表对应，并添加相关属性、getters和setters方法。

*@Data  
public class* Orders *implements* Serializable {  
 *private static final long* serialVersionUID = 1L;  
 *private* Long id; *private* Long userId; *private* Long addressBookId; *private* LocalDateTime orderTime; *private* LocalDateTime checkoutTime;

*private* BigDecimal amount;

*……*  
}

1. 在Service层中创建OrderService接口，并实现其中定义的订单相关的业务逻辑方法。

*public interface* OrderService *extends* IService<Orders> { *public void* submit(Orders orders);  
}

1. 在Controller层中创建OrderController类，处理前端传来的订单相关请求，并调用OrderService中的方法进行业务逻辑的处理。

*@PostMapping*("/submit")  
*public* R<String> submit(*@RequestBody* Orders orders){  
 log.info("订单数据：{}",orders);  
 orderService.submit(orders);  
 *return* R.success("下单成功");  
}

1. 在前端页面中，设计订单管理的UI界面，包括订单查询、订单修改、订单删除等功能。
2. 使用jQuery等前端框架，在前端页面中发送Ajax请求，与后端进行数据交互，完成订单管理模块的功能。

### 购物车管理模块

1. 加入购物车

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。

1. 购物车列表展示

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。

1. 修改购物车商品数量

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。

1. 删除购物车商品

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。

*public interface* CartItemMapper *extends* BaseMapper<CartItem> {  
 *int* deleteByCartItemId(Long cartItemId);  
}

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

1. 添加地址

在地址管理模块中，用户可以添加新的地址。这可以通过在AddressMapper中实现一个插入地址的方法来实现。编写一个插入地址的方法：

public interface AddressMapper extends BaseMapper<Address> {

}

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：

@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。

1. 修改地址

在地址管理模块中，用户可以修改已有的地址。这可以通过在AddressMapper中实现一个更新地址的方法来实现。例如，可以编写一个更新地址的方法：

public interface AddressMapper extends BaseMapper<Address> {

}

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。例如，可以编写一个修改地址的方法：

@PostMapping("/updateAddress")

public String updateAddress(@ModelAttribute("address") Address address) {

addressService.updateById(address);

return "redirect:/address";

}

在上述方法中，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。

1. 删除地址

在地址管理模块中，用户可以删除已有的地址。这可以通过在AddressMapper中实现一个删除地址的方法来实现。例如，可以编写一个删除地址的方法：

public interface AddressMapper extends BaseMapper<Address> {

int deleteByAddressId(Long addressId);

}

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。例如，可以编写一个删除地址的方法：

@PostMapping("/deleteAddress")

public String deleteAddress(@RequestParam("addressId") Long addressId) {

addressService.removeById(addressId);

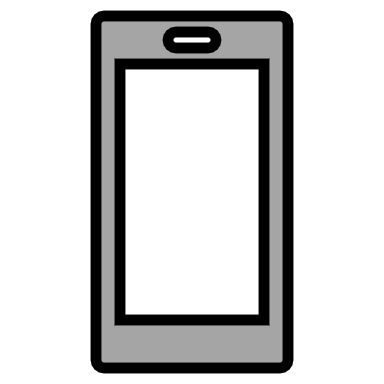
return "redirect:/address";

}

在上述方法中，首先根据地址ID删除地址，然后重定向到地址列表页面。

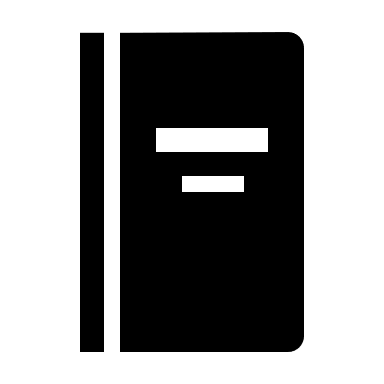
## 项目部署

### 部署架构





192.168.136.129

Internet

192.168.136.128

192.168.136.129

192.168.136.130

Tomcat

Nginx

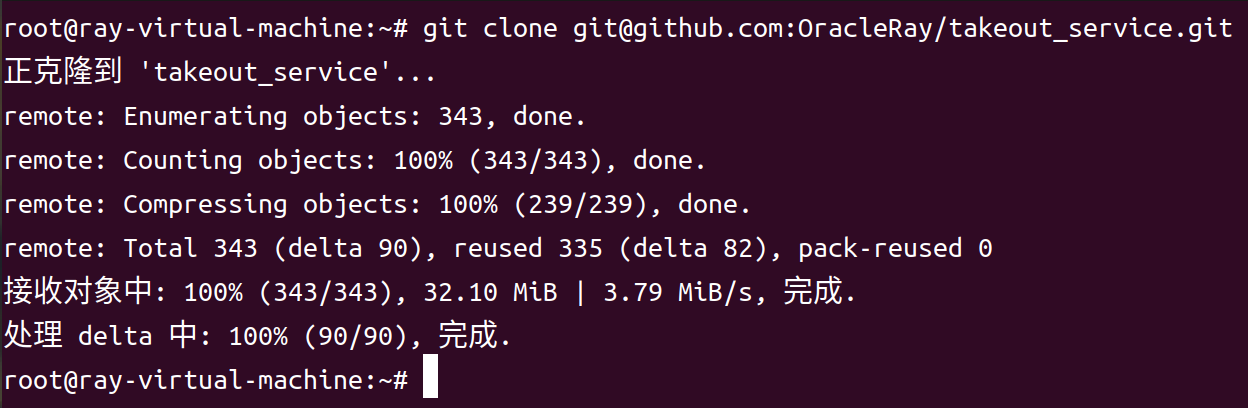
192.168.136.128

用户端

管理端

### 部署前端项目

1. 克隆已上传到github的项目到服务器



1. 修改nginx的配置文件，使其可以访问到项目静态资源



1. 本地浏览器输入IP地址，即可访问到



### 部署后端项目

1. 同样先克隆已上传到github的项目到服务器
2. 编写运行脚本文件

# 系统测试

# 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