

东 华 大 学

毕业设计（论文）

课 题 名 称 : 香港粗菜馆在线订座系统的开发

学 院 : 计算机科学与技术学院

专 业 : 软件工程

姓 名 : 马希磊

学 号 : 161310414

指 导 教 师 : 韦俊银

二 0 二 0 年 六 月 十 日

香港粗菜馆在线订座系统的开发

摘要

随着经济社会的发展，传统的电话预定方式因为复杂繁琐，管理成本较大已经逐渐不能满足用户和商家的需求。越来越多的人开始选择网上订餐，再加上用户订餐的时间比较集中，使得系统在某一时间段内需要处理的用户数据发生了爆炸性的增长。传统的单体服务常常不能很好地满足高并发情况下的性能需求。选用微服务的架构将各模块拆分成一个独立的服务单元。在满足了性能要求的同时，有效地降低了系统的耦合度，可以有效地拓展、复用某些服务。

本系统因此定位为基于 Spring Boot 和 Spring Cloud 开发一个微服务架构的在线订座系统。用户可以提前在线查看香港粗菜馆的用餐环境，订购特色菜品，同时选择自己的就餐时间、人数、位置偏好，订餐之后还可以查看自己的排队进度。同时商家可以通过后台管理界面直观简便的查看顾客的消费走势与人员分布特征。可以有效节省店家和顾客双方的时间。提高用户就餐体验，节省商家运营管理成本。

后续通过 docker 对系统进行了部署，使得在不同的环境下无需更改系统的相关环境和配置便可以直接启动 docker 容器。有效解决了部署开发环境的困难。

关键词：在线订座，微服务，Spring Boot，Spring Cloud

DEVELOPMENT OF ONLINE RESERVATION SYSTEM FOR HONG KONG ROUGHAGE RESTAURANT

ABSTRACT

With the development of the economy and society, the traditional telephone reservation method has gradually been unable to meet the needs of users and merchants because of its complicated and cumbersome and large management cost. More and more people are choosing to order food online, due to the time for users to order food are relatively concentrated, the user data that the system needs to process within a certain period has exploded. Traditional individual services sometimes can no longer meet the performance requirements under high concurrency. The micro-service architecture is used to split each module into an independent service unit. While meeting the performance requirements, it effectively reduces the coupling degree of the system, and can effectively expand and reuse certain services.

Therefore ,our main work is to develop an online reservation system with the microservice architecture based on Spring Boot and Spring Cloud. Users can check the dining environment of the Hong Kong restaurant online in advance, order special dishes, and choose their own dining time, number of people and location preference. After ordering, they can view their queue progress. At the same time, merchants can view customer consumption trends and personnel distribution characteristics through the background management interface. The system can effectively save the time of both the store and the customer. Improving user's dining experience and saving business's operation and management costs.

Subsequently, the system may be deployed through docker, and the docker container can be directly started without changing the relevant environment and configuration of the system in different environments. Effectively solved the difficulty

of deploying development environment.

Key words: Online reservations, microservices, Spring Boot, Spring Cloud

目 录

摘要.....	i
ABSTRACT	ii
1 绪论	1
1.1 研究背景及意义.....	1
1.2 国内外研究现状.....	2
1.3 研究内容.....	3
1.4 本章小结.....	3
2 相关技术及开发环境.....	4
2.1 MySQL 数据库	4
2.2 Redis 数据库	4
2.3 SpringBoot.....	5
2.4 SpringCloud.....	5
2.5 Bootstrap.....	8
2.6 Layui.....	8
2.7 开发环境.....	9
2.8 本章小结.....	9
3 系统分析与设计	10
3.1 需求分析	10
3.1.1 功能需求.....	10
3.1.2 性能需求.....	10
3.2 系统总体架构设计	11
3.3 系统各功能模块设计	14
3.3.1 登录与注册模块.....	14
3.3.2 菜品管理模块.....	15
3.3.3 订单管理模块.....	16
3.4 系统数据库设计	17

3.5 本章小结	20
4 系统实现和部署测试	21
4.1 微服务环境搭建	21
4.1.1 服务注册中心 Eureka 环境搭建	21
4.1.2 服务配置中心环境搭建	22
4.1.3 服务消费者环境搭建	23
4.1.4 服务提供者环境搭建	24
4.2 系统各功能模块实现与展示	26
4.2.1 用户管理模块	26
4.2.2 菜品管理模块	30
4.2.3 订单管理模块	32
4.3 系统部署	36
4.4 系统测试	38
4.5 本章小结	38
5 总结和展望	40
5.1 总结	40
5.2 展望	40
参考文献	42
致谢	43
译文及原文	44

1 绪论

1.1 研究背景及意义

当前,我国的经济保持了持续稳定发展,居民的消费水平不断提高,消费观念也逐渐有了新的变化,传统的餐饮服务已经难以满足群众日益增长的高端化、特色化需求。在市场的调解作用下,我国的餐饮服务市场正在经历着巨大而复杂的深刻变革市场空间得到进一步拓展,市场机会逐渐增多,服务模式日趋多样化、智能化^[1]。为了节省就餐所需要的排队时间,用户开始选择提前预约的方式。传统的电话预定方式因为复杂繁琐,管理成本较大已经逐渐不能满足用户和商家的需求。在线订座系统可以使用户随时随地了解餐厅的人流状况,合理的选择自己的用餐时间,挑选自己喜欢的座位,并能实时查看自己的排队进度,减少实地排队浪费的时间提高用户消费体验。商家可以根据客流状况自主选择接单或者拒单,并根据实际效果调整用餐时长规定,以达到利润最大化。为香港粗菜馆开发一套线上订座系统可以给顾客提供更好地用餐体验,有效实现用餐高峰分流。同时对餐厅来说也是一种有效的线上宣传方式。

伴随着互联网的发展和普及,越来越多的人开始选择网上订餐,再加上用户订餐的时间比较集中,使得系统需要处理的用户数据发生了爆炸性的增长。传统的单体服务已经不能满足高并发情况下的性能需求。选用集群的方式可以有效地进行负载均衡处理。集群方式下各个节点功能复杂且相同,而实际情况下用户对各功能模块的访问频率却有较大差异,这就造成了资源的浪费。此外各节点都包含众多复杂的功能也使系统的耦合性增高,不利于系统的维护和扩展。系统选择采用微服务的架构将系统的各个功能模块拆分成一个个独立的服务单元,各个服务单元之间通过 RPC 的方式相互通信。在满足了性能要求的同时,有效地降低了系统的耦合度,可以有效地拓展、复用某些服务。

本系统因此定位为一个微服务架构的在线订座系统。用户可以提前在线查看香港粗菜馆的用餐环境,订购特色菜品,同时选择自己的就餐时间、人数、位置偏好,订餐之后还可以查看自己的排队进度。同时商家可以通过后台管理界面直

观简便的查看顾客的消费走势与人员分布特征。可以有效节省店家和顾客双方的时间。提高用户就餐体验，节省商家运营管理成本，可以给食堂管理员提供配餐数量信息，精准备餐，避免浪费。扩大餐馆知名度，实现利润最大化。

1.2 国内外研究现状

由于智能手机的普及，已经有很多比如大众点评这样的 app 提供了线上订餐、订座等服务。用户预订座位后可以随时查看自己的排队进度。但这些应用大多功能不够完善，比如没有充分考虑到用户的用餐人数，预计用餐时间，就餐位置偏好，以及超时过号未到的处理方式，还有用户即将到号时提醒用户，商家无法确认用户是否可以到达。有些商家选择直接线上预订排号的方式，但没有考虑用户取号后不去就餐的情况，导致过多人拍号，真正需要就餐的人又因为看到等待时间过长放弃就餐。一些商家为避免上述情况只提供实地排号服务，但这又同时给用户带来了额外的不便。

本系统在提供基本的订座软件功能的同时。着眼于解决用户长时间排队体验较差和商家可能损失潜在用户的问题。用户可以在线填写用餐人数、预计用餐时间以及座位偏好等信息，同时预定后可实时查看排队进展。同时用户还可以提前浏览特色菜品与订餐。

虽然近几年来微服务架构成为了一个越来越流行的词汇，不过它的实现的想法却并不是完全新的。其实微服务架构和 SOA 模式十分相似。微服务和 SOA 都致力于通过把应用程序拆分为一个个功能更加单一的更小的简单服务，以达到实现更高效地开发、维护和扩展程序的目的。尽管目前看来 SOA 已经成为了一个被人们普遍了解的概念，SOA 也可以用来实现很多系统，不过微服务的体系结构则给出了一种使用一系列比较微小和单一的小的服务构建应用程序的特定方式，每个微服务都专注于实现比较单一的功能，做好特定的一件事情。另外因为近几年来类似 Netflix 等许多大型主流互联网公司在采用微服务体系结构进行开发维护系统方面取得了巨大的成功，这也正给那些考虑使用微服务架构的公司和团队带来了更多的希望和信心。现在主流的微服务开发框架 Spring Cloud 包含了一系列微服务开发所需要的相关框架。它通过利用 Spring Boot 的约定大于配置等特性给系统开发带来的方便，使得微服务系统中各个基础功能模块的开发变得更加方便高效。利用 Spring Boot 可以方便的实现和部署微服务系统中服务的发

现注册、消息总线、负载均衡、服务配置中心、数据监控等。Spring Cloud 也被许多开发者认为是最好的微服务开发框架。因此本系统选用 Spring Cloud 为开发框架。

1.3 研究内容

香港粗菜馆在线订座系统作为一个在线订座平台,旨在为广大用户提供一个美观、便利、可靠的在线预订平台。用户可以根据自己的不同情况通过在线平台提前预定满足自己要求的就餐时间、餐位、就餐人数等个性化餐位。同时用户可以通过在线平台了解餐厅环境、历史文化、附近交通等信息。特色菜品列表可供用户提前订餐。以便商家可以根据用户需求进行提前准备,提高效率,节约成本。用户预订座位之后可以实时查看自己的预订就餐时间,避免现场排队所需要的巨大时间开销。商家可以根据店内就餐人数状况选择接受或拒绝用户的订座请求,可以从后台添加、修改、删除菜品信息,对用户信息进行分析管理,以及通过后台数据对客流状况,菜品销售状况等进行统计分析。以便合理调整经营策略,提高餐厅收入。

1.4 本章小结

本章从研究背景与意义入手,分析介绍了目前线上订座系统的发展状况与开发本系统的必要性,然后介绍了线上订座及微服务架构的研究现状,最后介绍了香港粗菜馆在线订座系统的主要功能。本章是对整个论文的一个概述。

2 相关技术及开发环境

2.1 MySQL 数据库

本系统选用 MySQL 作为存储数据库。

MySQL 是一个关系型数据库管理系统，关联数据库选择将数据保存在不同的数据表里边，而并不选择把所有数据存储在一个比较大的数据仓库之中，通过这种方式可以提高数据存取的速度并且增加了操作的灵活性。MySQL 的开发语言为偏向底层的 C 和 C++ 语言，还进行了充分的环境测试，这些保证了其具有良好的可移植性。MySQL 可以在各种主流操作系统下运行并且还为各种编程语言提供了大量丰富的 API 操作接口。MySQL 采用客户机/服务器的架构方式实现，是一个多用户、多线程的数据库服务系统。可以有效和快捷、安全地处理大量的数据。相对于 Oracle 等数据库来说，MySQL 的使用非常简单。MySQL 软件分为社区版和商业版两个版本，由于其体积小、速度快，并且开放源代码，开放源代码数据库管理系统已经成功地应用到很多系统中，MySQL 是其中较出色的一个^[2]。现在大多数中小型网站的开发都选择 MySQL 作为数据库。

2.2 Redis 数据库

本系统选用 Redis 实现分布式情况下的 Session 会话共享，在多个微服务之间保存用户的登录状态信息。

Redis 是一款开源的、网络化的、基于内存的、可进行数据持久化的 Key-Value 存储系统。^[4]相比于 Memcached，它可以存储的数据类型更加丰富多样，包括链表、集合、字符串、哈希表以及有序集合等常用数据结构。这些数据类型可以很好地支持添加、删除及取交集并集和差集等更加丰富多样的常用操作，并且这些操作都可以保证是原子性的。除此之外 redis 还可以通过各种不同方式的实现对数据的高效排序。和 memcached 一样 redis 的数据也都存储在内存里，这就保证了对数据的各种操作可以获得很高的速度。和 memcached 不同的是 redis 会定期的把修改操作写入相关的记录文件或者把更新的数据同步写入到磁盘中，并且在这个基础之上实现了数据的主从同步。

Redis 具有很好的性能表现和丰富的数据类型，不仅仅在很大程度上弥补了 memcached 这类 key/value 数据库的不足，而且在一些特定的环境和应用之中也可以很好地配合关系型数据库使用。Redis 可以很好地支持主从同步。数据能够从主服务器上向从服务器上同步，从服务器也可以是与其它从服务器关联的主服务器。因为实现了发布/订阅机制，从数据库在任何时候进行数据同步操作时，都可以通过订阅一个频道实现对主服务器发布的记录和消息的完整的接收。主从同步对读取操作的扩展和保证系统数据安全性都具有重要的意义。

2.3 SpringBoot

SpringBoot 是由 Pivotal 团队设计提供的一套框架，其目的是为了使 Spring 应用的搭建过程得到简化，方便其开发过程。因为开发人员平时的开发工作有很多固定重复性的工作，该框架使用了特定的方式对一些常用的配置进行了约定的默认配置，从而简化了开发人员的开发流程。SpringBoot 给 Spring 项目以及 web 开发带来了非常大的影响和改变。^[6] SpringBoot 默认配置了很多框架的配置信息以及使用方式。

Spring Boot 使得 Spring 的应用系统开发得到了极大的简化，开发人员仅仅需要通过少量的代码就能创建一个独立的、产品级别的 Spring 应用。Spring Boot 的核心思想就是约定大于配置，多数 Spring Boot 应用只需要很少的 Spring 配置。Spring Boot 为 Spring 平台及第三方库提供开箱即用的设置。采用 Spring Boot 可以大大的简化你的开发过程，它基本对所有的常用框架都有对应的组件支持。

SpringBoot 使用嵌入式的 Servlet 容器，有效解决了 J2EE 笨重的开发、配置繁多、开发效率低下、部署流程复杂、第三方技术集成难度大等问题。

SpringBoot 使编码、配置、部署、监控都变得非常简单。这些特性使其能够非常方便、快速构建独立的微服务。所以使用 Spring Boot 开发项目，会使我们的开发工作更加简单规范，有效提高开发效率与质量。

2.4 SpringCloud

Spring Cloud 其实是一套框架的集合，它利用 Spring Boot 的便利性成功使得分布式系统的开发得到了简化。Spring Cloud 并没有重复造轮子，而是将目前

各种主流模块进行了封装和集成，从而减少了系统的开发时间和成本。相比于 Dubbo 只实现微服务的服务治理而言，Spring Cloud 成功实现了微服务架构中的方方面面。提供了开发分布式系统所用到的“全家桶”。Spring Cloud 框架是当前实现微服务架构的一个优秀方案。^[9]

香港粗菜馆在线订座系统中主要运用到了 Spring Cloud 的如下组件：

（1）Eureka

Eureka 是一个由 Netflix 开发的，基于 REST 服务的，服务注册与发现的组件。

它主要包括两个组件：Eureka Server 和 Eureka Client。

单个微服务启动的时候，会通过 Eureka Client 将自己向 Eureka Server 注册，Eureka Server 可以存储各个服务的信息。

Eureka 的工作原理如图 2-1。

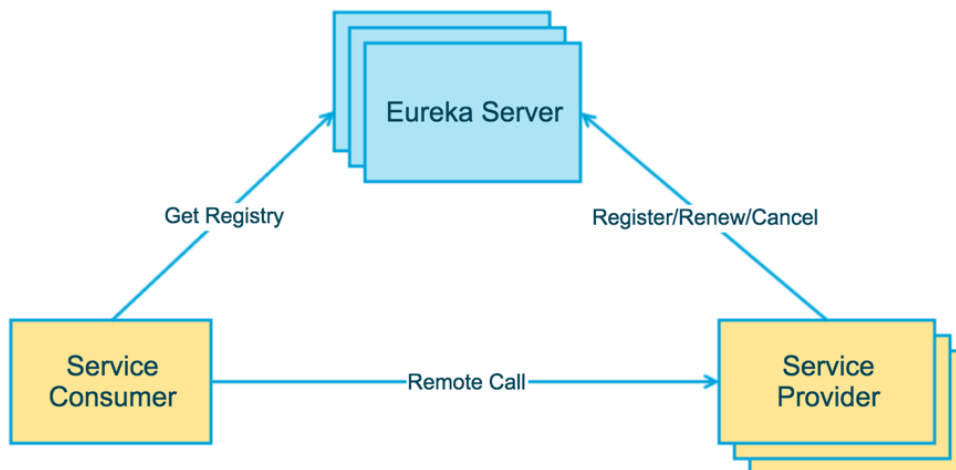


图 2-1 Eureka 工作原理架构图

（2）SpringCloudConfig

SpringCloudConfig 是 SpringCloud 提供的分布式系统和微服务架构的集中配置中心。它分为客户端和服务端两部分。服务端也称为分布式配置中心，是一个独立的服务应用，可以连接配置仓库并且客户端可以从服务端获取配置信息。客户端在启动的时候从配置中心获取和加载配置信息。引入配置中心后可以对各个微服务的配置进行集中统一的管理。并且可以实现在不重启微服务的前提条件下对微服务的配置进行更改。避免了传统情况下修改配置需要重启系统的巨大开销。SpringCloudConfig 可以将配置文件存储在本地，也可以将配置文件存储在远程

Git 仓库，可以创建 ConfigServer，通过它集中统一管理所有的配置文件。SpringCloudConfig 架构如图 2-2。

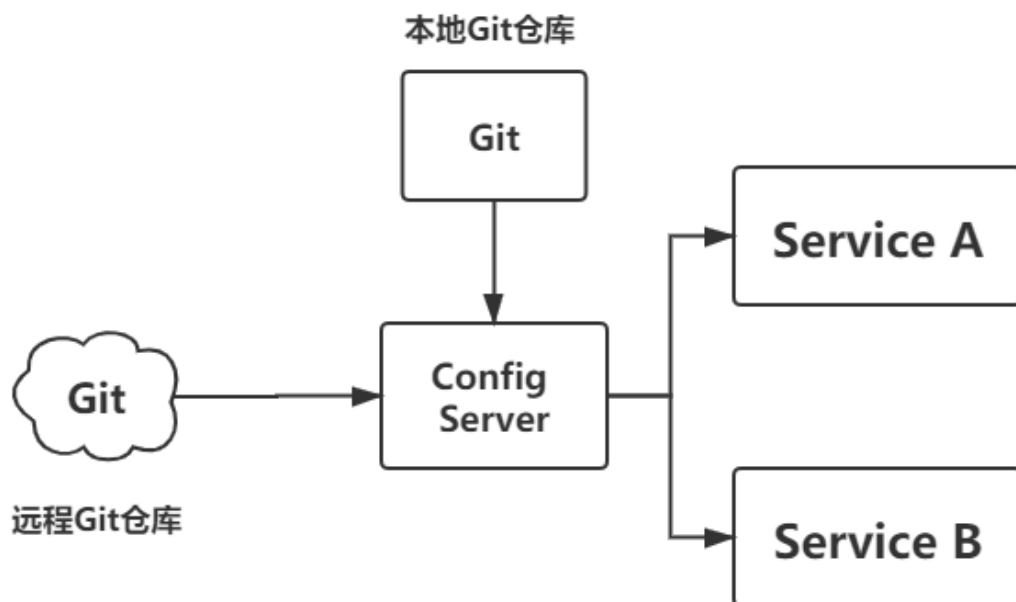


图 2-2 SpringCloudConfig 架构图

(3) Feign

Feign 不仅默认集成了 Ribbon，并且和 Eureka 有效结合，很好的实现了客户端负载均衡的效果。

在进行 Spring Cloud 微服务开发的时，各个服务之间大多是通过 HTTP 接口的形式对外提供服务的。因此消费者在进行系统调用时，底层就是通过 HTTPClient 的这种方式进行访问。层就是可以通过 HTTPClient 的这种方式进行数据访问。您可以选择通过 JDK 本地的 URLConnection，Apache 的 HTTP 客户端，Netty 的异步 HTTP 客户端，RestTemplate 来实现服务之间的调用。但是利用 Spring Cloud Open Feign 进行服务间的调用才是最方便、最优雅的方式。因为 Feign 被 Spring Cloud 进行了增强，可以支持 Spring Mvc 的注解，并且整合了 Ribbon，所以开发者可以更加简单的对 Feign 进行使用。

Feign 作为一个 Web Service 客户端，它是声明式的。Web Service 客户端的开发也因为它的出现而变得简单了起来。Feign 是一种模板化 HTTP 客户端。使用它之后仅仅只需要创建一个接口加上对应的注解便可以实现对相关功能的

调用。通过使用 Feign, 可以做到像是在调用本地方法一样的使用 HTTP 请求访问远程服务, 开发者不会感觉到自己在进行远程方法调用, 更感知不到在访问 HTTP 请求。它整合了 Hystrix 和 Ribbon, 所以开发者不需要再对他们进行整合。Feign 还提供了模板化的 HTTP 请求, 开发者通过编写简单的接口和注解, 就可以对 HTTP 请求的参数等信息进行良好的定义。HTTP 的请求会完全被 Feign 代理, 我们只需要依赖注入依赖的 Bean, 然后传递相关参数调用对应的方法即可。使用起来非常方便简单。

2.5 Bootstrap

Bootstrap 作为一款前端框架, 其强大的功能使得 Web 开发变得比以前更加快捷方便。Bootstrap 包含了很多组件, 包括: 导航, 工具条, 标签, 按钮等可以供开发者方便使用。内置 jQuery 插件, 支持 html5 css3, 支持 less 动态样式, 跨设备, 跨浏览器, 响应布局。代码优雅简洁, 界面美观大方, 是当下最为流行的前端框架之一。

2.6 Layui

本系统选用 Layui 进行管理员后台管理页面的开发。

Layui 遵循了原生 HTML/CSS/JS 的书写与组织形式, 是一款采用自身模块规范编写的前端 UI 框架, 十分简单易用, 学习成本极低。其体积小巧轻便, 组件丰富完善, 看起来似乎简单朴素, 内部功能却十分饱满, 从核心代码到 API 的每一处细节都经过了精心的研究设计。是进行快速前端界面开发的良好选项。在 2016 年秋季发布的 Layui 第一版便将自己与那些基于 MVVM 底部的 UI 框架区分开来。但这恰恰是它返璞归真的体现, 并不能看作为逆道而行。比较而言, 它更适合进行后端管理系统的前台界面开发, 开发者不用考虑各种前端工具的复杂配置, 使得开发过程变得十分简单高效。

在组织形式上 Layui 采用了几年前比较流行的以浏览器为宿主的类 AMD 模块管理方式, 它自己拥有独特的模式, 更多的面向于系统后端开发者, 更加轻量化和简单。尽管 Layui 定义为“经典模块化”, 但这是有意避开当下 JS 社区的主流方案, 试图以尽可能简单的方式去诠释高效, 并非是刻意强调“模块”理念本身。正是对于返璞归真的执念, 使得它成为了经典。

Layui 提供了可通过模块化的方式按需加载丰富的内置模块。它几乎兼容正在使用的全部浏览器, 是进行 PC 端后台系统与前台界面的速成开发的优秀选择。

2.7 开发环境

(1) 硬件环境:

处理器: Intel Core i7-7700HQ 四核处理器

内存: 8G

硬盘: 256G SSD

(2) 软件环境:

操作系统: Microsoft Windows 10、Centos 7

数据库: MySQL 5.6、Redis

开发语言: Java jdk 1.8

开发工具: IntelliJ IDEA、Navicat

2.8 本章小结

本章主要对香港粗菜馆在线订座系统所用的一些主要关键技术和系统开发所使用的平台环境进行了简要介绍。分别介绍了数据库、分布式系统后台以及前端所用的相关技术。

3 系统分析与设计

3.1 需求分析

3.1.1 功能需求

香港粗菜馆在线订座系统是基于微服务架构开发的一个在线订座平台，目的是在于为用户提供方便快捷的在线订座服务。主要分为用户订座和管理员后台两大部分。系统的主要功能需求如下：

(1) 用户登录注册

用户可根据用户名、密码和手机号进行注册。

用户登录时系统可以检查用户名和手机号是否已经被注册避免重复注册，同时对用户密码经过 MD5 加密后存储到数据库中，保证系统的安全性。

根据用户名和密码进行登录。

(2) 用户管理

管理员可以通过后台页面查看、修改、删除用户信息，以及分析用户性别比例。

(3) 菜品管理

管理员可以上传菜品图片以及价格和菜品描述等信息。修改、删除菜品信息。客户可以通过前端页面浏览特色菜品信息并且进行提前预定。

(4) 订单管理

用户登陆之后可以选择自己的就餐时间、就餐位置、就餐人数等相关信息后进行座位预订。预定座位的同时可以提前点餐。用户下单后可以通过在线平台查看自己的预定就餐时间以及订座信息。管理员可以分析每月的订单数量变化状况。

3.1.2 性能需求

(1) 实时性：香港粗菜馆在线订座系统对于实时性的要求较低，允许在用户可以接受的时间限度内有一定的延时效果。但不可影响用户使用体验。

(2) 安全性：考虑到网络环境的潜在威胁，要求系统有较好的安全防护措施：包括密码安全性、用户验证、权限拦截、系统封装等，并使用安全机制和数

据加密技术。同时要求系统保留升级接口，为之后的系统升级和安全维护提供便利。

(3) 易用性：香港粗菜馆在线订座系统应该具有美观简洁的前端页面，符合用户的操作习惯，保证用户可以方便的通过系统挑选菜品以及预定满足自己特定需求的座位。

(4) 健壮性：整个系统在一周内的使用不能出现两次以上的故障。每个功能点经过反复测试后出现的错误小于 3 个，故障可以在一天之内恢复。

(5) 易维护性：香港粗菜馆在线订座系统各个功能模块之间采用微服务的形式进行定义开发。各个功能模块之间耦合度很低，接口界限明确，每个微服务内部高度聚合，且功能相对简单，便于系统维护。

(6) 可扩展性：香港粗菜馆在线订座系统的各个微服务可以按照实际需求进行扩展，每个功能模块只含有部分核心功能方便扩展。

(7) 可移植性：香港粗菜馆在线订座系统采用 java 语言开发，具有天然的可移植性。此外系统的各个独立功能模块均可以以 jar 包的形式运行，方便部署。

3.2 系统总体架构设计

香港粗菜馆在线订座系统采用微服务的架构进行设计开发。系统的总体架构如图 3-1。

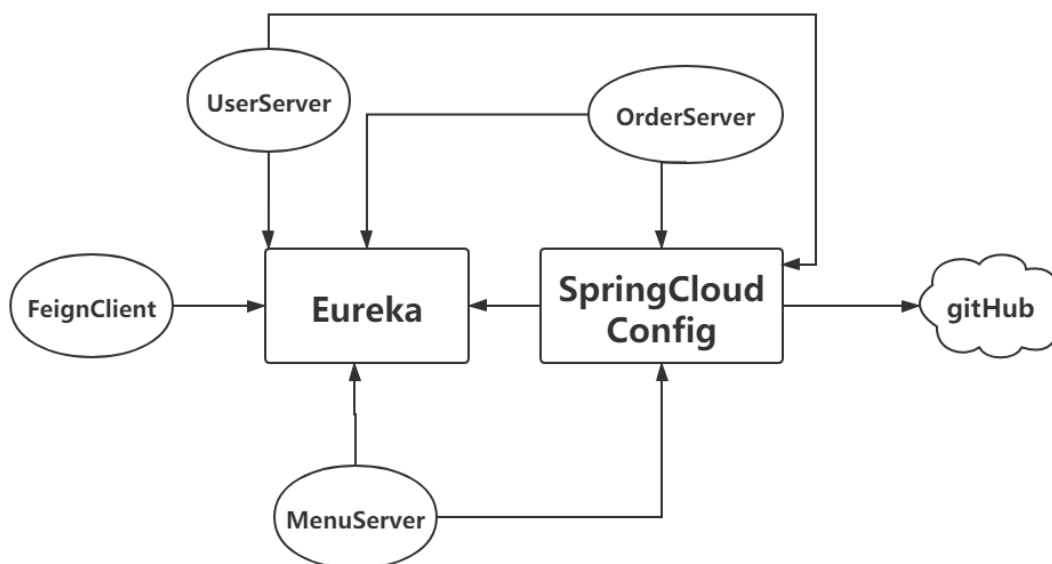


图 3-1 香港粗菜馆在线订座系统架构图

如图 3-1 所示，香港粗菜馆在线订座系统主要通过用户服务、菜品服务以及订单服务三个微服务提供者保证实现系统的主要功能。采用了 Eureka 作为服务

的注册管理中心，对各个微服务进行统一管理。选用 SpringCloudConfig 作为配置中心，主动拉取远程 Git 仓库中的配置文件信息。可以实现对配置文件的集中统一管理。并且可以在不需要重新启动系统的前提条件下实现配置文件的更新。客户端服务消费者选用 Feign 进行了负载均衡，有效地提高了系统的性能。保证了系统的稳定性。

系统的功能主要靠用户服务、菜品服务以及订单服务三个微服务提供者提供具体的支持。各个微服务的具体功能如下：

用户微服务所能提供的服务功能主要如图 3-2。

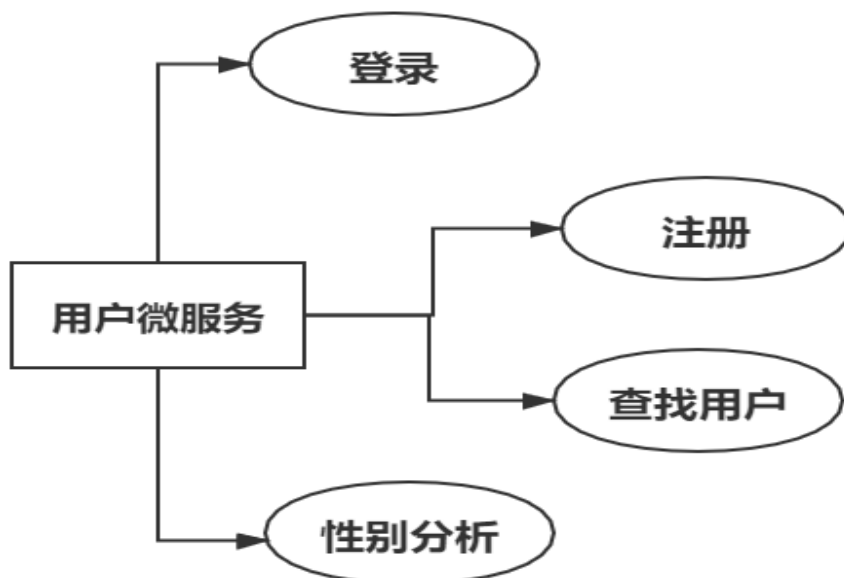


图 3-2 用户微服务功能结构图

菜品微服务所能提供的服务功能主要如图 3-3。

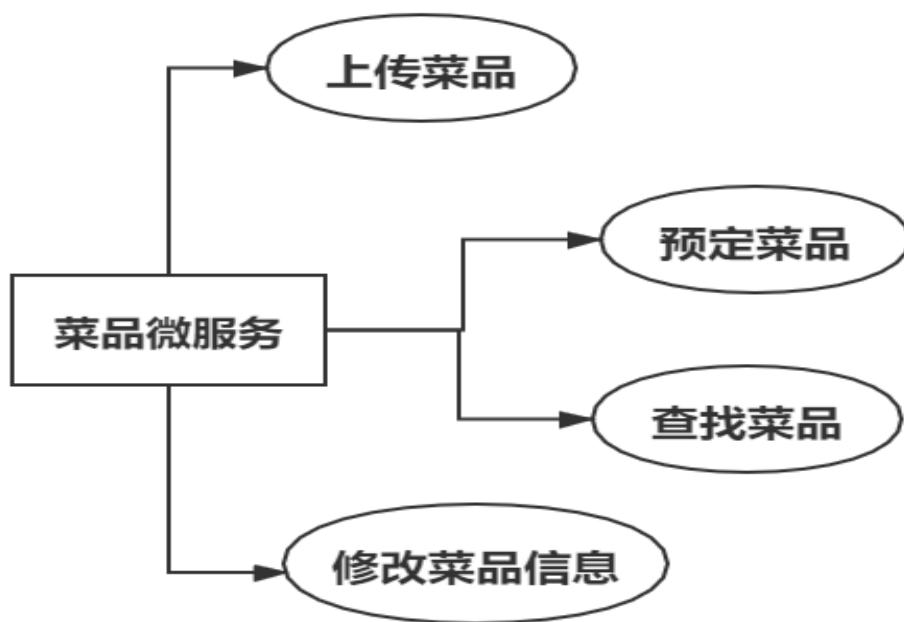


图 3-3 菜品微服务功能结构图

订单微服务所能提供的服务功能主要如图 3-4。

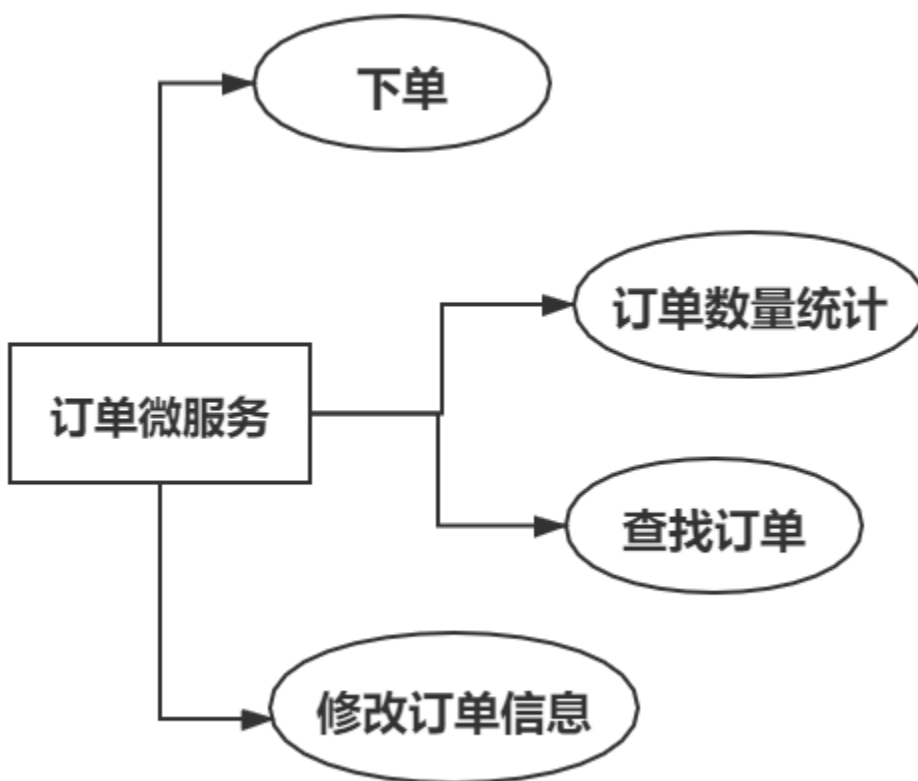


图 3-4 订单微服务功能结构图

香港粗菜馆在线订座系统主要的使用者分为普通用户和管理员两类。并为不同的

用户提供了不同的权限和功能。不同用户的相对权限如图 3-5。

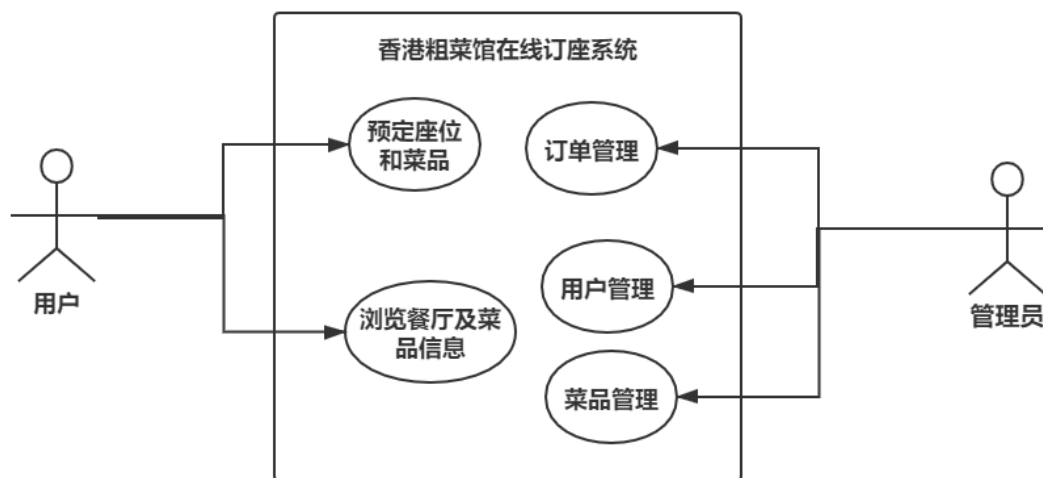


图 3-5 香港粗菜馆在线订座系统用例图

3.3 系统各功能模块设计

3.3.1 登录与注册模块

(1) 功能描述

用户登录模块功能：用户或管理员可以输入用户名和密码进行登录, 未注册的用户需要首先注册, 用户名和密码正确后可以进入到首页, 不正确则提示“用户名或密码错误”，继续重新登录。

用户的注册模块功能：用户或管理员可以输入用户名、密码和电话号码进行注册, 注册成功后会自动跳转到登录界面提示“注册成功, 请登录”, 若该用户名已存在, 会提示“用户名已存在”, 继续重新注册。

(2) 算法思想

用户登录成功后将用户信息存储到 Session 对象中, 用于识别当前用户, 以及后续功能的正常进行, 用户进行后续相关操作时会首先被拦截器拦截, 只有登陆后检测到对应 Session 对象才可以正常进行后续访问与操作。由于分布式环境条件下不同微服务中的 Session 对象无法共享, 所以将用户登陆后的状态信息存储到 Redis 数据库中, 实现不同微服务之间登录状态信息的共享。

(3) 逻辑流程

登录与注册的流程如图 3-6。

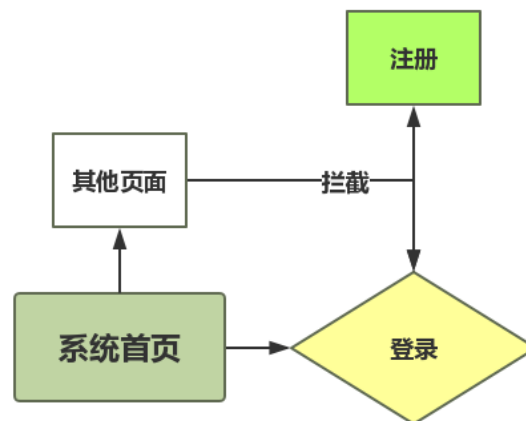


图 3-6 登录与注册流程图

3.3.2 菜品管理模块

(1) 功能描述

上传菜品功能:管理员登陆后可以通过后台菜品上传页面上传菜品的名称、价格、菜品详细描述、菜品图片等菜品信息。进行菜品的更新上传。及时更新特色菜品信息。

菜品浏览功能:用户可以通过前端界面浏览查看菜品的图片价格等相关信息。登录之后可以添加菜品到购物车，提前预定自己喜欢的特色菜品。

编辑菜品功能:管理员登陆后可以通过后台编辑删除页面查找相关菜品，并且可以修改菜品的相关信息。还可以删除相关菜品。

(2) 算法思想

用户登录成功后将用户信息存储到 Session 对象中,用于识别当前用户,以及后续功能的正常进行,用户进行后续相关操作时会首先被拦截器拦截,只有登陆后检测到对应 Session 对象才可以正常进行后续访问与操作。进行相关操作之前需要首先进行用户的角色判定,保证只有具有相关权限的用户才可以进行相应的合法操作。

(3) 逻辑流程

菜品管理相关功能的流程如图 3-7。

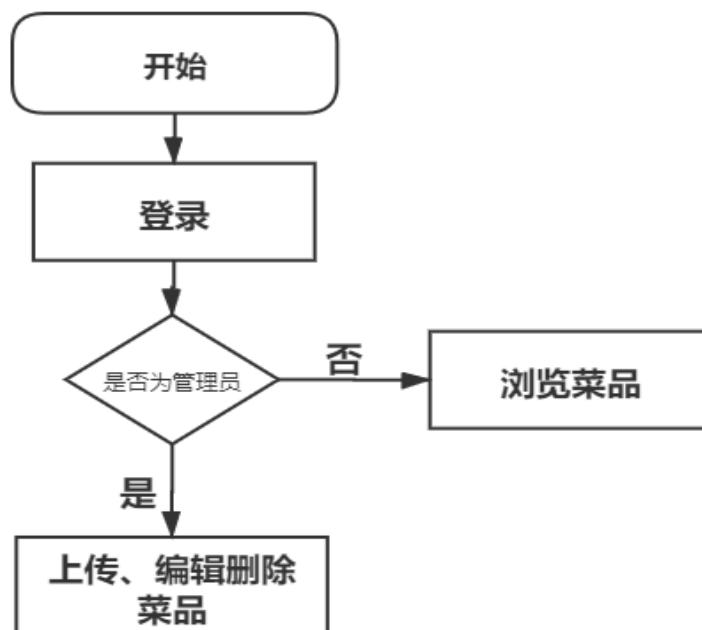


图 3-7 菜品管理流程图

3.3.3 订单管理模块

(1) 功能描述

用户订座功能：用户登陆后可以预定自己的就餐座位和时间。选择自己的就餐时间、就餐人数、就餐位置偏好等信息，预定符合自己特定需要的座位。用户预定座位的同时还可以选择自己喜欢的特色菜品。

分析订单数据功能：管理员登陆后可以通过后台管理页面以柱状图和折线图的形式清晰地观察到每月的订单数量以及订单走势。根据数据分析原因，制定合理的经营管理策略以求有效地提高利润。

订单状态管理：管理员登录之后可以查看订单的详细信息，并且具有权限修改订单的状态。当用户到店就餐之后订单的状态应该更新为已完成。

查看就餐时间功能：用户预订座位后可以通过在线系统查看自己的订单时间以及距离自己的就餐时间还有多久，避免错过时间或者到达店里时间过早需要排队造成自己的时间浪费。

(2) 算法思想

用户登录成功后选择自己的就餐时间、就餐人数、餐位信息，填写预订者的姓名和联系电话后点击“立即预定”可以进行下单操作。管理员登陆后

可以通过可视化图表方便的观察到按月份统计好的订单数量以及订单数量走势。还可以及时根据实际情况按需要修改订单的状态。

(3) 逻辑流程

订单管理的流程如图 3-8。

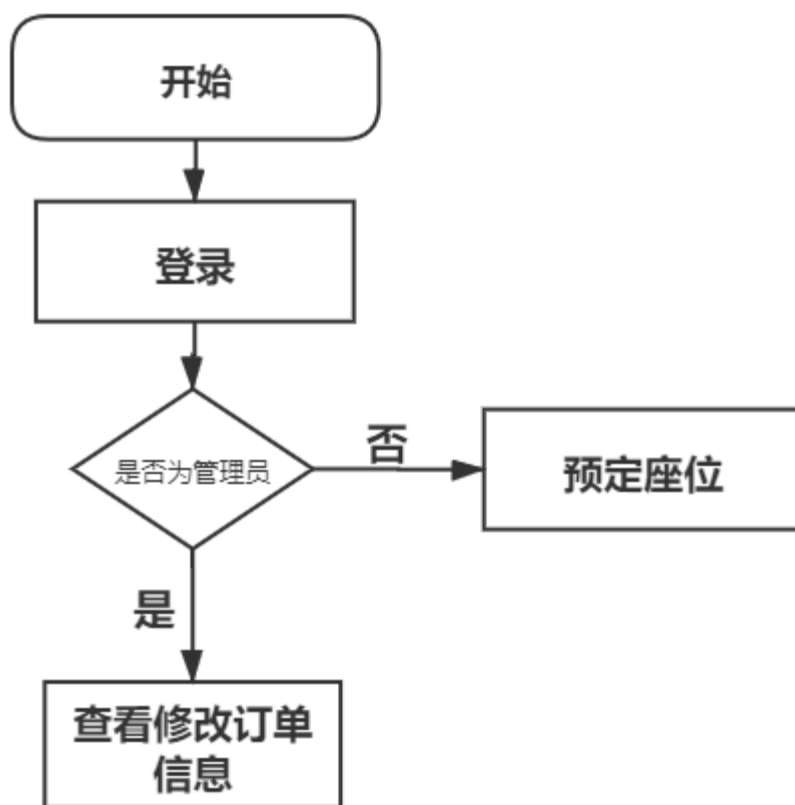


图 3-8 订单管理流程图

3.4 系统数据库设计

香港粗菜馆在线订座系统的数据库主要包含了用户（user）表、菜品（menu）表、评论（comment）表和订单(ordert)表。系统选用 MySQL 数据库。

根据各数据表项之间的对应关系。可以建立实体-联系模型图（E-R 图），如图 3-9。

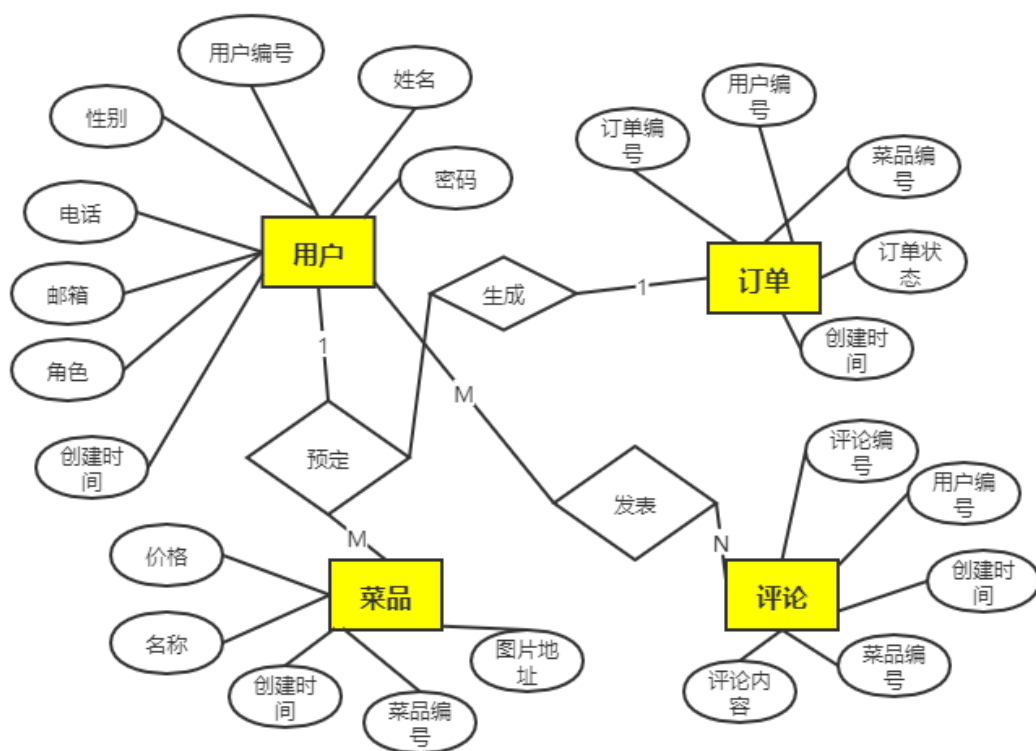


图 3-9 香港粗菜馆在线订座系统 E-R 图

系统数据库中各数据表详细信息如下。

用户信息表（user）如表 3-1。

表 3-1 user 用户信息表

字段名	类型	说明
id	int	用户编号
username	varchar	用户姓名
gender	varchar	用户性别
password	varchar	用户密码
email	varchar	用户邮箱
phone	varchar	用户电话
role	int	用户角色，0 为管理员，1 为普通用户
create_time	datetime	创建时间

表 3-1 user 用户信息表保存了用户的相关信息。管理员和普通用户共用一张

表。两者身份通过 role 字段进行区分。

当用户注册时生成相应的数据表项，用户登录时通过匹配用户名和密码是否与数据库表中的一致来判定用户是否可以合法登录。还可以根据 gender 字段对用户的性别组成进行分析。

菜品信息表（menu）如表 3-2。

表 3-2 menu 菜品信息表

字段名	类型	说明
id	int	菜品编号
name	varchar	菜品名称
image	varchar	菜品图片地址
detail	varchar	菜品详情
price	varchar	菜品价格
scorces	int	菜品总分
freq	int	菜品被评价次数
average	int	菜品平均评分
create_time	datetime	创建时间
update_time	datetime	更新时间

表 3-2menu 菜品信息表保存了菜品的相关信息。管理员可以上传菜品的相关信息，并且进行相应的编辑和修改。菜品图片上传后数据库表中会记录菜品图片的路径信息。用户评价后 freq 和 scores 字段会发生更新，并且可以根据这两个字段的新值重新计算得到菜品的新的平均评分。

订单信息表（ordert）如表 3-3。

表 3-3 ordert 订单信息表

字段名	类型	说明
id	int	订单编号
uid	int	用户编号
mid	int	菜品编号
state	int	订单状态

name	varchar	预订者姓名
phone	varchar	预订者电话
people	int	就餐人数
reserve_time	datetime	预定就餐时间
create_time	datetime	创建时间
update_time	datetime	更新时间

表 3-3 ordert 订单信息表保存了订单的相关信息。用户登录后可以选择自己喜欢的菜品进行预定。同时可以挑选自己的就餐时间、就餐人数、就餐位置偏好等信息进行提前订座。预定座位时需要填写订座人姓名和联系电话。管理员可以根据实际状况修改订单状态 state 字段。0 为预定，1 为订单完成。

评论信息表（comment）如表 3-4。

表 3-4 comment 评论信息表

字段名	类型	说明
id	int	评论编号
uid	int	用户编号
mid	int	菜品编号
score	int	菜品评分
content	varchar	评论内容
create_time	datetime	创建时间

表 3-4 comment 评论信息表保存了评论的相关信息。用户登录后可以对菜品进行评价，给菜品打分。

3.5 本章小结

本章首先从功能需求和非功能需求两方面介绍了系统应该完成的功能和应达到的一些非功能目标要求。

然后主要先介绍了香港粗菜馆在线订座系统的概要设计给出了系统的总体架构图，之后详细介绍了系统各个功能模块的设计思路以及对应的流程图。最后进行了数据库设计。通过 ER 图可以了解系统的主要实体以及他们之间的各种联系。本章所进行的各种设计为之后系统的开发实现提供了良好的指导和规范。

4 系统实现和部署测试

4.1 微服务环境搭建

香港粗菜馆在线订座系统选用微服务的架构进行设计开发。系统主要运用 Spring Boot 和 Spring Cloud 进行搭建。其中主要包含用户服务、菜品服务和订单服务三个服务提供者。系统选用 Eureka 作为服务的注册与发现中心，采用 Spring Cloud Config 作为系统的配置管理中心，对系统的配置进行统一管理。此外各个微服务还有部分配置保存在自己的独立服务模块内。在负载均衡方面本系统选用 Feign 作为客户端负载均衡的有效实现方式，实现了类似于面向接口调用的效果。可以有效地提高开发效率，便于系统的维护和扩展升级。

本小结将介绍微服务整体环境的搭建和配置。

4.1.1 服务注册中心 Eureka 环境搭建

系统采用 Eureka 作为服务注册中心，其环境搭建步骤如下：

(1) 在 pom.xml 文件中导入 Eureka 依赖

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
  <version>2.0.2.RELEASE</version>
</dependency>
```

(2) 配置 application.yml 配置文件

```
server:
  port: 7001
eureka:
  instance:
    hostname: localhost
  client:
    register-with-eureka: false //自身不必在 Eureka-server 注册
    fetch-registry: false //不需要从在 Eureka-server 拉取服务信息
    service-url:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

将服务注册中心的端口设置为 7001，主机名为默认的 localhost，register-with-eureka 和 fetch-registry 的值均设置为 false 表明此为服务的注册中心。为服务端。

(3) 创建主启动类 EurekaServerApplication7001

```
package com.leilei.eurekaServer;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication7001 {
    public static void main(String[] args) {
        SpringApplication.run(EurekaServerApplication7001.class,args);
    }
}
```

@EnableEurekaServer 标识 EurekaServerApplication7001 为服务注册中心 EurekaServer 的主启动类。

(4) Eureka 启动测试

上述配置成功后启动 eurekaServer 工程，在浏览器中输入 localhost: 7001，可以看到服务注册中心成功启动的效果如图 4-1。

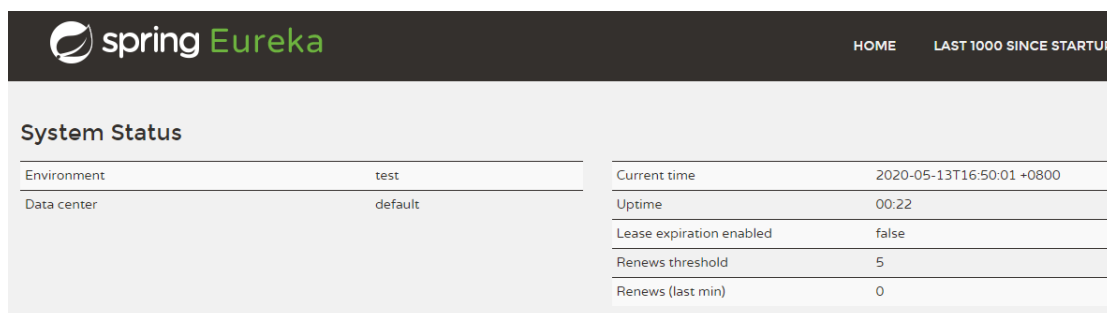


图 4-1 服务注册中心搭建成功界面

4.1.2 服务配置中心环境搭建

系统采用 Spring Cloud Config 作为服务配置中心，其环境搭建步骤如下：

- (1) 在 pom.xml 文件中导入 spring-cloud-config-server 依赖

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-config-server</artifactId>
</dependency>
```

- (2) 配置 application.yml 配置文件

```
server:
    port: 3344
spring:
    application:
```

```
name: configServer
cloud:
  config:
    server:
      git:
        uri: https://github.com/xileigit/reservationSystemConfig.git
```

将服务配置中心的端口设置为 3344，git 仓库地址为本人的 github 远程仓库 <https://github.com/xileigit/reservationSystemConfig.git>。

(3) 创建主启动类 ConfigServerApplication3344

```
@SpringBootApplication
@EnableConfigServer
public class ConfigServerApplication3344 {
    public static void main(String[] args) {
        SpringApplication.run(ConfigServerApplication3344.class,args);
    }
}
```

@EnableConfigServer 标识 ConfigServerApplication3344 为服务配置中心的主启动类。

(4) 服务配置中心启动测试

上述操作成功完成后，服务配置中心环境搭建完成。启动 configServer 工程，在浏览器输入 <http://localhost:3344/application-dev.yml> 可以看到提前存储在 git 仓库中的 application-dev.yml 配置文件中的内容，证明服务配置中心搭建成功。效果如图 4-2。

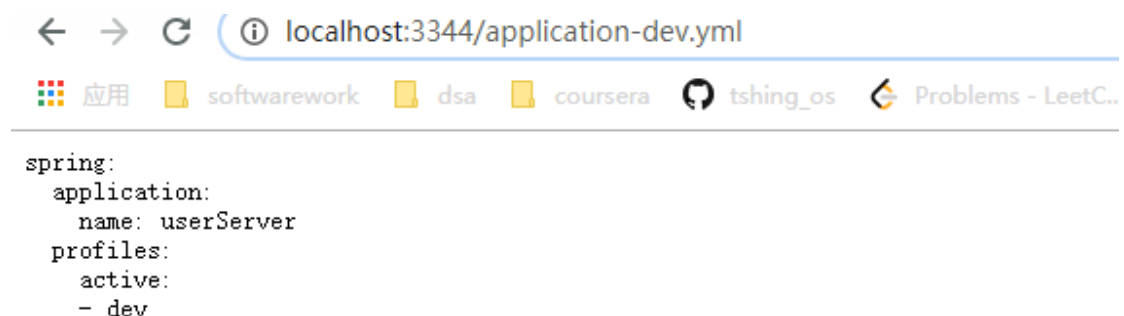


图 4-2 服务配置中心搭建成功界面

4.1.3 服务消费者环境搭建

系统采用 Spring Cloud Feign 实现客户端负载均衡，其环境搭建步骤如下：

- (1) 在 pom.xml 文件中导入 spring-cloud-starter-openfeign 依赖

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

(2) 配置 application.yml 配置文件

```
server:
  port: 8009
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:7001/eureka/
    register-with-eureka: false    //不需要注册
```

将服务消费者的端口设置为 8009，服务消费者可以从服务注册中心发现注册进其中的各个服务提供者，服务消费者本身不需要向服务注册中心注册，只要具有从中发现服务的能力即可。

(3) 创建主启动类 ClientApp

```
@SpringBootApplication
@EnableFeignClients
@EnableRedisHttpSession
public class ClientApp {
    public static void main(String[] args) {
        SpringApplication.run(ClientApp.class,args);
    }
}
```

@ EnableFeignClients 标识 ClientApp 为服务消费者以及客户端负载均衡实现的主启动类。

4.1.4 服务提供者环境搭建

香港粗菜馆在线订座系统主要具有用户服务、菜品服务以及订单服务三个微服务提供者。它们的搭建配置过程一致，只是部分参数会有所区别。接下来以用户微服务 UserServer 为例介绍服务提供者的环境搭建流程。之后给出三个微服务提供者之间在具体参数上的差异。

用户服务 UserServer 环境搭建过程如下。

(1) 在 pom.xml 文件中导入相关依赖

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-config-client</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

引入 `spring-cloud-config-client` 依赖包保证服务提供者 `UserServer` 具有服务配置中心的客户端的相关依赖可以从服务配置中心的服务端拉取相关配置信息。
`spring-cloud-starter-netflix-eureka-client` 依赖包保证服务提供者 `UserServer` 具有服务注册中心的客户端的相关依赖可以向服务注册中心的服务端进行注册。

(2) 配置 `application.yml` 配置文件

```
server:
  port: 8001
spring:
  application:
    name: UserServer
eureka:
  client:
    service-url:
      defaultZone: http://localhost:7001/eureka
  instance:
    instance-id: UserServer8001
    prefer-ip-address: true
```

将用户微服务的服务提供者的端口设置为 8001，微服务名称为 `UserServer`。
并且将用户微服务注册到服务注册中心 `http://localhost:7001/eureka`。

(3) 创建主启动类 `UserServer8001`

```
@SpringBootApplication
@EnableEurekaClient
@EnableRedisHttpSession
public class UserServer8001 {
    public static void main(String[] args) {
        SpringApplication.run(UserServer8001.class,args);
    }
}
```

`@EnableEurekaClient` 标识 `UserServer8001` 为服务注册中心的客户端，启动后会将自己注册到服务注册中心。作为一个微服务提供者可以被其他服务消费者

直接通过自己在服务注册中心注册的服务名称调用。

三个服务提供者的端口号如下：

微服务名称	端口号
UserServer	8001
MenuServer	8002
OrderServer	8003

三个微服务提供者环境搭建完成之后，依次启动 UserServer、MenuServer 和 OrderServer 三个微服务提供者工程。保证之前已经启动了服务注册中心，则可以通过在浏览器中输入 localhost:7001 来检验三个微服务提供者是否已经启动并注册成功。效果如图 4-3。

Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
MENUSERVER	n/a (1)	(1)	UP (1) - MenuServer8002
USERSERVER	n/a (1)	(1)	UP (1) - UserServer8001
ORDERSERVER	n/a (1)	(1)	UP (1) - OrderServer8003

图 4-3 服务提供者启动成功界面

4.2 系统各功能模块实现与展示

4.2.1 用户管理模块

(1) 用户注册

用户可以通过用户注册页面填写自己的用户名、密码、联系电话等个人信息后进行注册。注册时会检查用户填写的用户名、电话以及邮箱等信息是否已经被注册，如果已经被注册则会给出相关提示。检测过后，用户的密码将会通过 MD5 算法进行加密过之后存入数据库之中，用户注册成功。

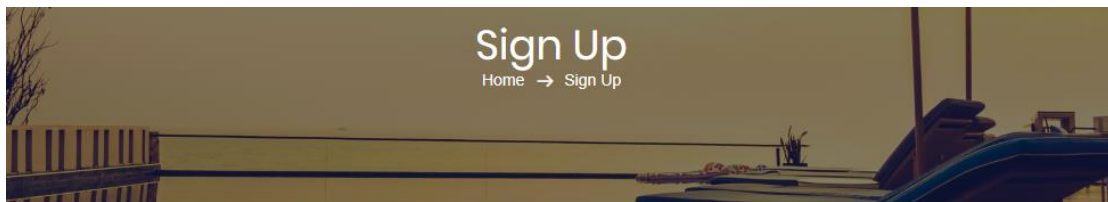
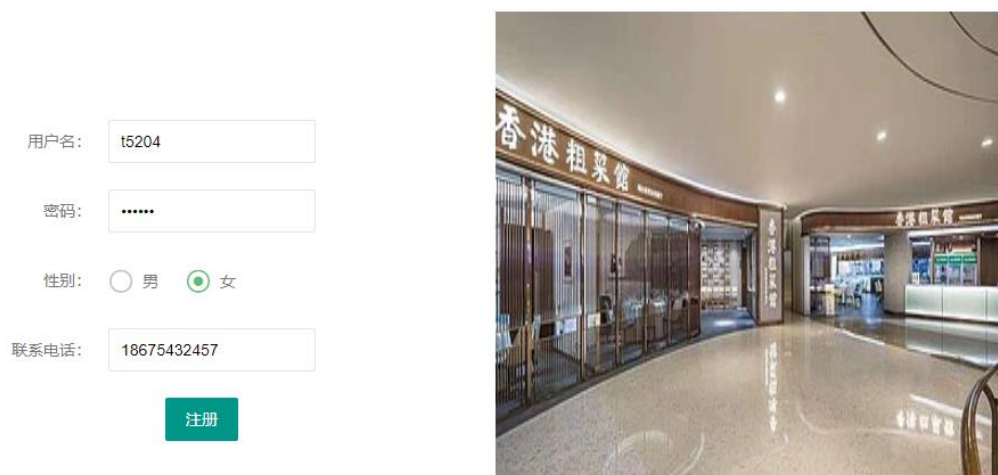
用户注册主要实现代码如下

```
public ResponseResult<String> register(User user) {  
    //检查用户名是否已经存在  
    int resultCount = userMapper.checkUsername(user.getUsername());  
    if(resultCount > 0){  
        return ResponseResult.createByErrorMessage("用户已存在");  
    }  
    //检查邮箱是否已经存在
```



```
resultCount = userMapper.checkEmail(user.getEmail());
if(resultCount > 0 ){
    return ResponseResult.createByErrorMessage("邮箱已存在");
}
//检查电话是否已经存在
resultCount = userMapper.checkPhone(user.getPhone());
if(resultCount > 0 ){
    return ResponseResult.createByErrorMessage("电话已存在");
}
//对用户密码进行 MD5 加密
user.setPassword(DigestUtils.md5DigestAsHex(user.getPassword().getBytes()));
//用户注册
resultCount = userMapper.addUser(user);
if(resultCount == 0){
    return ResponseResult.createByErrorMessage("注册失败");
}
return ResponseResult.createBySuccess("注册成功",null);
}
```

用户注册界面如图 4-4。

用户注册界面包含以下元素：

- 用户名：t5204
- 密码：*****
- 性别：
 ☐ 男
 ☒ 女
- 联系电话：18675432457
- 注册按钮

图 4-4 用户注册界面

(2) 用户登录

用户通过登录页面输入自己的用户名和密码之后可以进行登录。登录时会检查用户名和密码是否合法。检验成功后将用户密码设置为空以保证安全性。

```
public ResponseResult<User> login(String username, String password) {  
    //检查用户是否存在  
    int resultCount = userMapper.checkUsername(username);  
    if(resultCount == 0){  
        return ResponseResult.createByErrorMessage("用户名不存在");  
    }  
    //密码进行 MD5 加密  
    String md5Password = DigestUtils.md5DigestAsHex(password.getBytes());  
    //登录检验  
    User user = userMapper.login(username,md5Password);  
    if(user == null){  
        return ResponseResult.createByErrorMessage("密码错误");  
    }  
    //用户密码置空  
    user.setPassword(Strings.EMPTY);  
    return ResponseResult.createBySuccess("登录成功",user);  
}
```

用户登录界面如图 4-5。

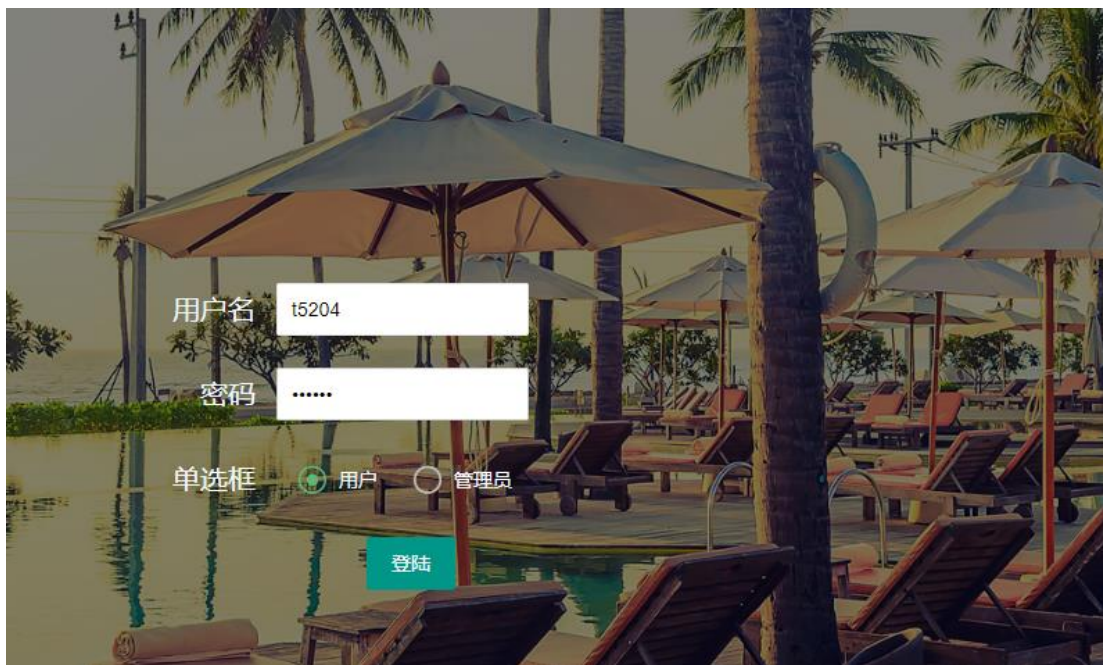


图 4-5 用户登录界面

(3) 用户性别比例分析

管理员可以通过后台管理界面以饼状图的形式查看到系统的男女用户性别比例分布。

新建用户性别类 UserGender

```
@Data
public class UserGender {
    private int male;
    private int female;
}
```

然后分别查询到男性和女性用户数量注入到 UserGender 对象的相应属性之中。将对象返回到前端页面。

```
@GetMapping("/gender")
public UserGender gender(){
    int male,female;
    //查询男性数量
    male=userService.getMaleCnt();
    //查询女性数量
    female=userService.getFemaleCnt();
    UserGender userGender = new UserGender();
    userGender.setMale(male);
    userGender.setFemale(female);
    return userGender;
}
```

前端通过 ajax 请求得到数据后，取出相应的男女人数。

```
$.ajax({
    url: "/user/gender",
    type: "GET",
    async: false,
    success:function(data){
        male=data.male;
        female=data.female;
    },
    error:function () {
        alert('error');
    }
});
```

之后 echarts 便可以根据数据自动绘制出饼状图。效果如图 4-6。

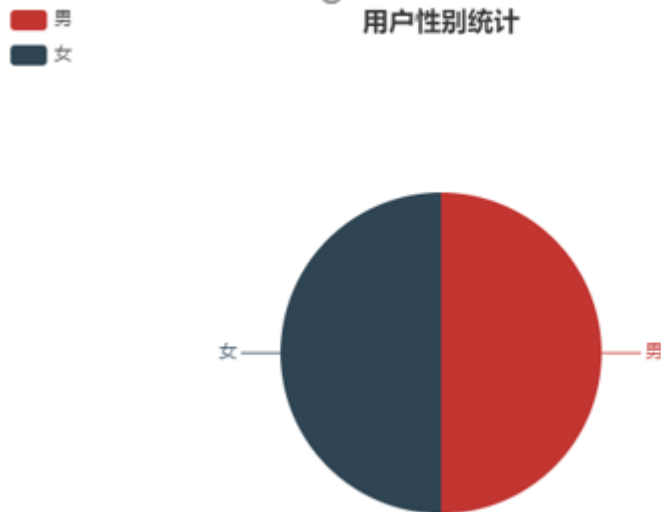


图 4-6 用户性别统计

4.2.2 菜品管理模块

管理员登陆后可以通过后台菜品上传页面上传菜品的名称、价格、菜品详细描述、菜品图片等菜品信息。进行菜品的更新上传。及时更新特色菜品信息。

菜品图片上传主要实现代码如下：

```
public Object uploadImg(@RequestParam(value = "file", required = false)
MultipartFile file, HttpServletRequest request, HttpServletResponse response) throws
Exception {
    String suffix = "";
    String dateStr = "";
    String filepath="";
    //保存上传
    OutputStream out = null;
    InputStream fileInput = null;
    try {
        if (file != null) {
            //得到上传图片文件的原始文件名
            String originalName = file.getOriginalFilename();
            //获取图片名称后缀
            suffix = originalName.substring(originalName.lastIndexOf(".") + 1);
            //获取当前时间字符串
            dateStr=new
java.text.SimpleDateFormat("yyyyMMddhhmmss").format(new Date());
            //根据系统时间和图片后缀加上新的存储位置生成新的文件路径全
            名
            filepath="D:\\reservationSystem\\client\\src\\main\\resources\\stati
c\\images\\"+ dateStr + "." + suffix;
            filepath = filepath.replace("\\", "/");
```



```
//创建新的图片文件
    File files = new File(filepath);
    if (!files.getParentFile().exists()) {
        files.getParentFile().mkdirs();
    }
    file.transferTo(files);
}
} catch (Exception e) {
} finally {
    try {
        if (out != null) {
            out.close();
        }
        if (fileInput != null) {
            fileInput.close();
        }
    } catch (IOException e) {
    }
}
}
Map<String, Object> map2 = new HashMap<>();
Map<String, Object> map = new HashMap<>();
map.put("code", 0);
map.put("msg", "");
map.put("data", map2);
//将上传的图片路径封装进 map 内并返回
String realativePath=dateStr + "." + suffix;
map2.put("src",realativePath );
return map;
}
```

添加菜品界面如图 4-7。



图 4-7 添加菜品

如图 4-7 管理员可以通过后台管理界面填写菜品的名称、价格菜品详细描述信息，上传菜品图片来添加菜品。

4.2.3 订单管理模块

用户登陆后可以预定自己的就餐座位和时间。选择自己的就餐时间、就餐人数、就餐位置偏好等信息，预定符合自己特定需要的座位。

用户需要提前登录之后才可以进行订座。用户订座界面如图 4-8。

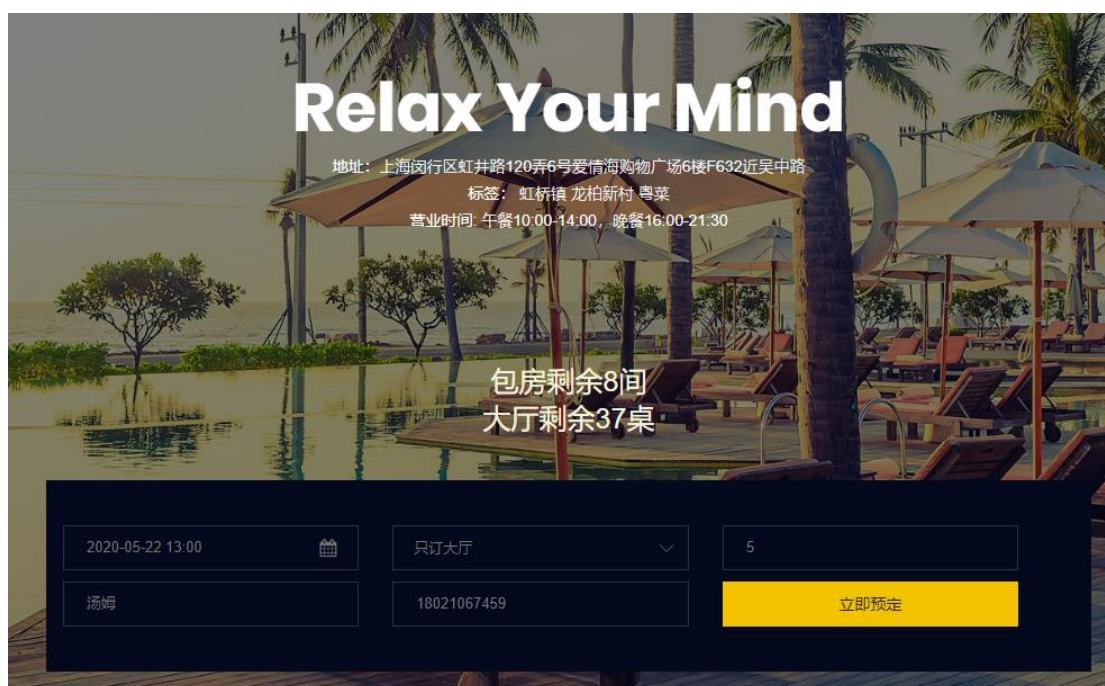


图 4-8 用户订座

如图 4-8 所示用户登录后可以进入到订座界面。在此界面可以查看到包房和大厅的就餐位置剩余数量情况。用户可以通过此页面预定座位。填写就餐人数，联系人姓名和电话等信息。选择自己的就餐位置偏好。可以选择大厅或者包房。以及预定就餐时间。

新建用户订座实体类 `SeatOrder` 用于封装用户订座信息

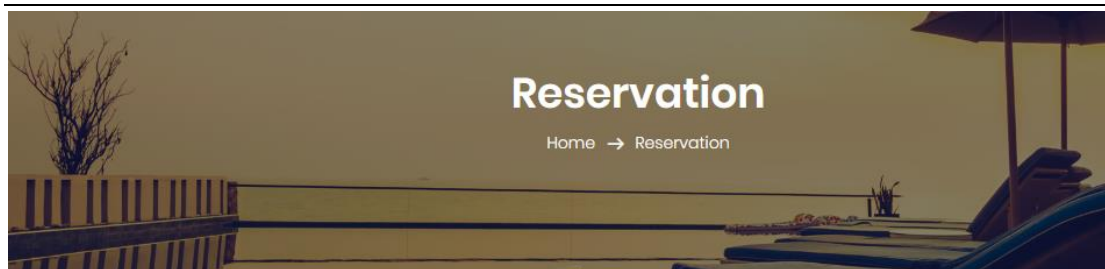
```
public class SeatOrder implements Serializable {  
    @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm")  
    private Date create_time;//就餐时间  
    private String name;//姓名  
    private String phone;//电话  
    private Integer type;//座位类型  
    private Integer personNum;//就餐人数  
};
```

用户选择相关定做信息之后点击“立即预定”按钮，相关信息会通过 form 表单封装传送到增加订单接口。

```
@PostMapping("/save")  
public String save(SeatOrder ordert, HttpServletRequest request){  
    request.getSession().setAttribute("SEAT_ORDER",ordert);  
    //改变包房数量  
    if(ordert.getType()==1){  
        orderFeignClient.decbaofangCount();  
    }  
    //改变大厅数量  
    else if (ordert.getType()==0){  
        orderFeignClient.decdatingCount();  
    }  
    return "blog";  
};
```

Save 接口获得前端传来的订单信息后将其保存到 session 中。之后根据用户预定的座位类型对相应座位的数量进行更新。之后跳转到用户订单信息界面。

因为之前已经将用户订座信息存储到了 session 之中，用户订座之后可以从相关 session 的属性之中取出订座信息的相关属性查自己的订座信息。如图 4-9。



t5204 您的订单信息如下

2020-5-22 13:00:00 5人 大厅用餐

汤姆 18021067459

距您预定的就餐时间还剩

01 天 00 时 32 分 44 秒

图 4-9 用户订座信息

如图 4-9 用户订座之后可以查看自己的订单信息。明确自己的预定就餐时间,并且可以通过倒计时功能直观有效地提醒用户不要错过自己的预定就餐时间。

管理员登陆后可以通过后台管理页面以柱状图和折线图的形式清晰地观察到每月的订单数量以及订单走势。主要实现方法如下:

新建月度销量类 MonthSales

```
@Data
public class MonthSales {
    private int month;
    private int sales;
}
```

然后按照月份分组查询每个月的销量,并将各月的销量按照月份注入到 MonthSales 对象的相关属性之中。将对象返回到前端页面。

```
<select id="getMonthSales" resultType="com.leilei.entity.MonthSales">
    select    DATE_FORMAT(create_time,'%m') as month,sum(id) as sales
    from    ordert  group  by    DATE_FORMAT(create_time,'%m')    order  by
    DATE_FORMAT(create_time,'%m') ;
```


</select>

前端通过 ajax 请求得到数据后，取出相应的月份和每个月的销量。

```
$.ajax({  
    url: "/order/getMonthSales",  
    type: "GET",  
    async: false,  
    success: function (data) {  
        $.each(data, function (index, monthsale) {  
            months[index] = monthsale.month + "月";  
            sales[index] = monthsale.sales;  
        });  
    },  
    error: function () {  
        alert('error');  
    }  
});
```

之后 echarts 便可以根据每月的数据自动绘制出销量的按月份变化的柱状图如图 4-10 和折线图如图 4-11。

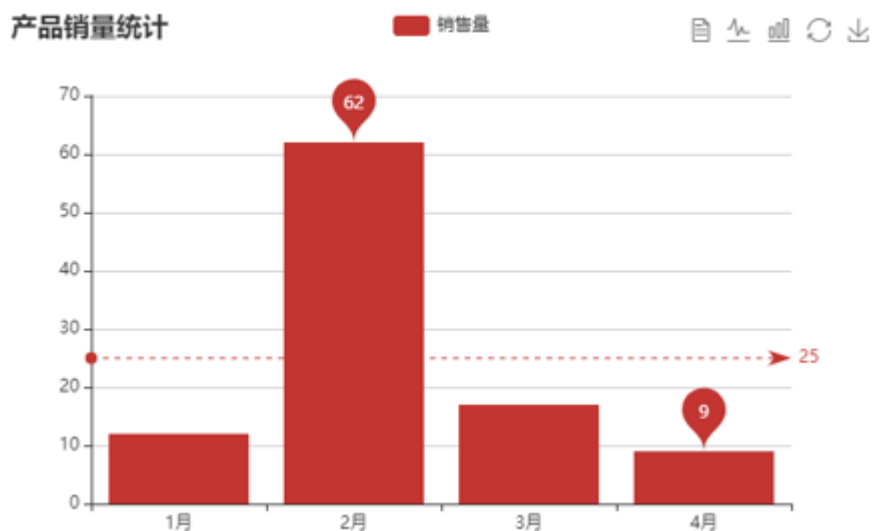


图 4-10 产品销量统计柱状图

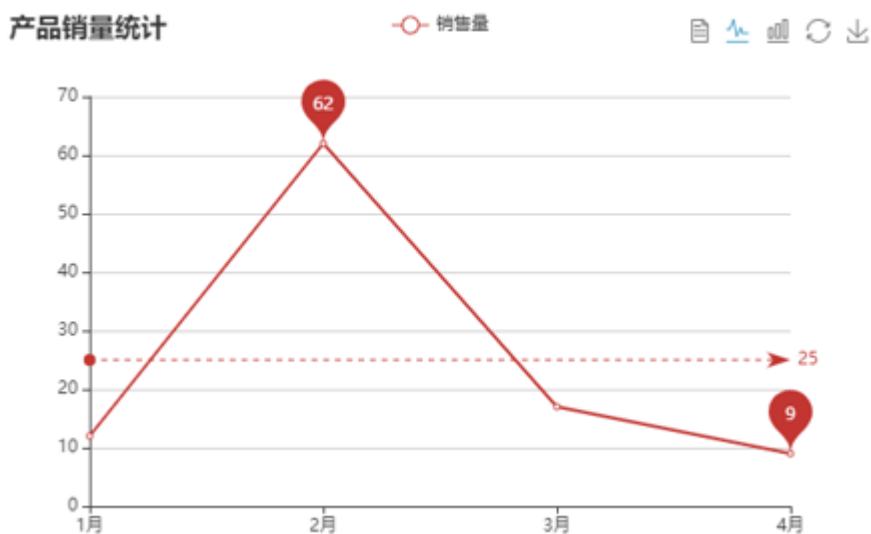


图 4-11 产品销量统计折线图

4.3 系统部署

系统利用 docker 进行了部署。有效地解决了开发环境部署困难的问题，部署之后可以直接从 docker 拉取相关镜像。启动容器后便可以运行系统，不需要在进行其他的环境配置。部署过程如下

首先在 centos7 上安装 docker

命令	说明
yum -y update	更新 yum 源
yum install -y docker-ce	安装 Docker
systemctl start docker systemctl enable docker	将 Docker 服务设置为开机自启动模式，并启动 Docker 服务

通过 docker version 命令验证 docker 是否安装启动成功。成功效果如图 4-12。

```
[xilei@localhost reservationSys]$ sudo docker version
[sudo] password for xilei:
Client: Docker Engine - Community
 Version:           19.03.11
 API version:       1.40
 Go version:        go1.13.10
 Git commit:        42e35e61f3
 Built:             Mon Jun  1 09:13:48 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:           19.03.11
  API version:       1.40 (minimum version 1.12)
  Go version:        go1.13.10
  Git commit:        42e35e61f3
  Built:             Mon Jun  1 09:12:26 2020
  OS/Arch:           linux/amd64
  Experimental:      false
```

图 4-12 Docker 安装启动成功图

将系统的各个模块利用 maven 构建工具打成 jar 包,之后利用 filezilla 上传到 centos 虚拟机。

利用 maven 打包需要在 pom 文件中加入 maven 构建工具

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

在 jar 包相同目录中创建 Dockerfile 文件

```
# 基础镜像为 java8
FROM java:8
# 作者
MAINTAINER xileima
# 将 jar 包添加到容器中并更名为 eureka-server.jar
ADD eureka-server-0.0.1-SNAPSHOT.jar eureka-server.jar
# 运行 jar 包
```

```
ENTRYPOINT ["java","-jar","eureka-server.jar"]
```

启动 docker 容器。

```
sudo docker run -p 9001:7001 eureka-server
```

其中-p 9001:7001 表示将 centos 的 9001 端口映射为 docker 容器的 7001 端口。

从本机浏览器输入虚拟机的 IP 和端口验证 eureka-server 容器是否启动成功，成功效果如图 4-13。

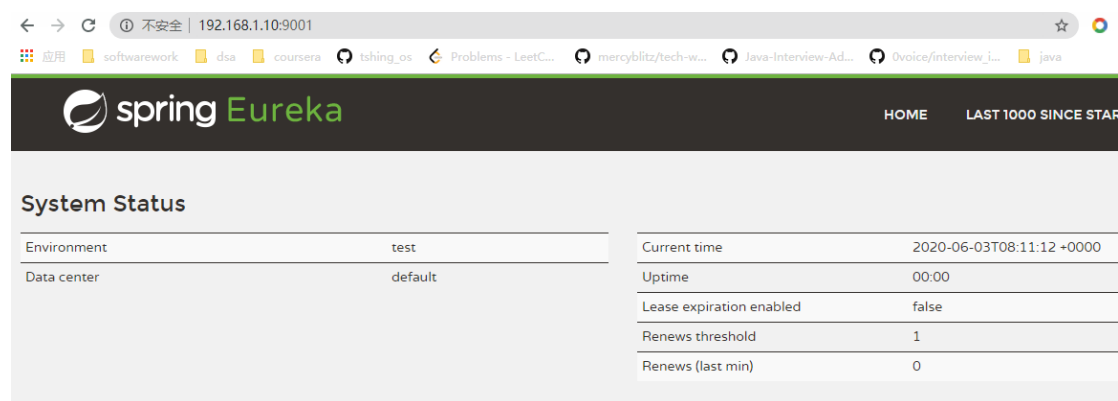


图 4-13 服务注册中心在 Docker 容器中启动成功图

4.4 系统测试

系统测试是程序的一种执行过程，目的是尽可能发现并改正系统中的错误，提高软件的可靠性。为了发现提高系统的可靠性，需要进行系统测试。在测试过程中发现香港粗菜馆在线订座系统中的错误并进行修改，不断迭代，保证整个系统的质量。

实验过程中进行了多次充分的测试，以下选取部分测试样例进行说明。

登录注册模块：分别选用没有注册的用户和已经注册的用户进行注册，验证能否检验出相同的用户名、手机号、邮箱等重复注册的问题。

菜品管理模块：上传菜品信息以及图片，验证是否可以上传菜品

销量管理模块：进行订座操作，观察能否下单成功以及月度订单数量是否发生相应的变化。

4.5 本章小结

本章主要介绍了香港粗菜馆在线订座系统的环境搭建过程以及各个主要功能实现的关键代码和思路，并且给出了各主要功能的运行效果，并对系统进行了

测试。

5 总结和展望

5.1 总结

本系统主要介绍了针对香港粗菜馆开发的在线订座系统的完整开发流程。首先分析了经济以及互联网技术高速发展的大背景 and 智能餐饮的不断普及和推广。接着分析了实地排号就餐的诸多不便引出了开发在线订座系统的必要。本人首先选取了香港粗菜馆这家餐厅作为开发的目标用户，之后通过互联网搜集到了很多餐厅的相关信息保证了系统的数据充足性和真实性。接着进行技术选型，通过对各种主流技术与架构的反复比较和分析最终选择了以微服务架构作为系统的开发架构。并且主要选用 Spring Boot 和 Spring Cloud 进行系统的开发。

经过将近半年的学习研究和老师的悉心指导。最终基本实现了最初的预期目标，为香港粗菜馆开发出了一个功能比较完整的在线订座平台。并对系统进行了充分的测试。用户可以通过此平台了解餐馆的历史和环境信息，浏览特色菜品。还可以提前预定符合自己特定需求的座位，订座之后用户可以通过在线系统查看自己的就餐时间。避免现场排号造成的大量时间浪费。管理员可以通过后台界面分析用户的特征已经餐厅的经营状况，从而节省管理成本提高工作效率。

最后通过总结实验以及参考相关资料完成了论文的书写。

5.2 展望

由于时间以及资源的限制系统还存在很多可以改进的问题。比如受实验资源的限制，系统在个人的笔记本电脑上进行开发测试，内存资源相对紧张给开发和测试过程带来了许多不必要的困难。另外系统没有上线，只是本机工作有一些实际问题可能还没有充分暴露。本来预计通过短信的方式提醒用户即将到达自己的预定就餐时间但后来发现需要企业账号才可以具有相关权限。由于硬件条件有限系统的注册中心以及各个微服务提供者均没有采用集群的方式部署，导致系统不具备微服务架构应有的高可用性。负载均衡也未能起到应有的作用。

这些问题部分在硬件条件允许的情况下可以快速简单的得到解决，倒也有部分问题是因为自己开发前期的考虑不够充分导致，更有部分问题还需要更多的技

术积累和时间才可以进一步提高与完善。

参考文献

- [1] 范文旭. 基于网络订餐平台的智能餐饮发展研究[J]. 无线互联科技, 2018 (12): 48.
- [2] 兰旭辉, 熊家军, 邓刚. 基于 MySQL 的应用程序设计[D]. , 2004.
- [3] 刘勇. 大众点评在线订座系统的设计与实现[D].
- [4] 马豫星. Redis 数据库特性分析[J]. 物联网技术, 2015, 3: 105-106.
- [5] 曾超宇, 李金香. Redis 在高速缓存系统中的应用[J]. 微型机与应用, 2013, 32(12):11-13.
- [6] 张峰. 应用 Spring Boot 改变 Web 应用开发模式[J]. 科技创新与应用, 2017, 23: 193-194.
- [7] 王悦. 基于 Spring Boot 技术的 SOA 接口研究[J]. 信息技术, 2019(6).
- [8] 郭致远, 魏银珍. 基于 Spring Cloud 服务调用的设计与应用[J]. 微型机与应用, 2019, 038(002):87-91.
- [9] 梅璇. 基于 Spring Cloud 的微服务调用研究[D]. 武汉理工大学, 2018.
- [10] Gutierrez F. AMQP with Spring Boot[M]// Spring Boot Messaging. 2017.
- [11] Felipe Gutierrez. JMS with Spring Boot[M]. Apress, 2017.
- [12] Iuliana Cosmina. Spring Boot and WebSocket[M]. Apress, 2015.
- [13] K. Siva Prasad Reddy. Web Applications with Spring Boot[M]// Beginning Spring Boot 2. 2017.
- [14] XIAO Yun. Spring Security Web-based Application Development[J]. Computer & Modernization, 2011, 1(6):158-2.

致谢

转眼间大学四年的生活已经接近了尾声。离别之际回想起自己四年来在学校的成长与收获心中充满了不舍与感激。

首先要感谢韦俊银老师在毕业设计上给我的巨大帮助。从选题开始韦老师便一直耐心的陪伴指导着我。他既善于给学生提示和建议，又积极鼓励学生的创新与探索。平时讨论他总是事先充足准备，每次都可以给我宝贵的建议和经验。在毕设期间，无论在学习和生活中我都得到了老师的巨大帮助。感谢韦老师在大学最后这段时光里的陪伴和指导。

感谢我的同学和朋友的帮助和鼓励。与他们一起学习和生活的四年时光中我得到了很大的收获和成长。

感谢学校提供的广阔自由的学习生活平台，在这里我增长了知识，开阔了视野，丰富了人格。我必将把对母校的感激转化为自己今后前进的动力。

最后感谢我的父母和家人。是他们的默默付出无私奉献，一直无条件的支持我。我才可以安心的完成自己的大学学业。认真学习，开心成长。

译文及原文