**香港粗菜馆在线订座系统开发**

摘要

均质充量压缩着火（HCCI）燃烧，作为一种能有效实现高效低污染的燃烧方式，能够使发动机同时保持较高的燃油经济性和动力性能，而且能和碳烟排

**1 绪论**

* 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的进行负载均衡处理。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。选用微服务的架构将各模块拆分成一个独立的服务单元，它们之间通过RPC方式通信。在满足了性能要求的同时，有效的降低了系统的耦合度，可以有效的拓展、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. 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。

微服务架构是近年来流行的一个新词汇，但其背后的想法并不新鲜。事实上，它与几年前非常流行的SOA模式类似。微服务和SOA都是将应用程序分解为更小的服务，以便更高效地扩展和管理应用程序生命周期。虽然SOA是一个非常普遍的概念，可能意味着很多事情，但微服务体系结构描述了使用非常小的服务构建应用程序的特定方式，每种方法都专注于做好一件事情。由Netflix领导的大型互联网公司在实施微服务体系结构方面的成功，对于考虑切换的人来说起到了很大的激励作用与动力。Spring Cloud是一系列框架的有序集合。它利用Spring Boot的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用Spring Boot的开发风格做到一键启动和部署。Spring Cloud 也被开发者认为是最好的开发框架。因此本系统选用Spring Cloud为开发框架。

* 1. 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. 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。

1. **相关技术**
   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，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。。MySQL使⽤C和C++编写，并使⽤了多种编译器进⾏测试，保证了源代码的可移植性。MySQL支持多种操作系统并为多种编程语言提供了丰富的API。MySQL是一个真正的多用户、多线程SQL数据库服务器。它是以客户机/服务器结构实现的，由一个服务器守护程序以及很多不同的客户程序和库组成。它能够快捷、有效和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主要目标是快速、便捷和易用。MySQL 软件采⽤了双授权政策，它分为社区版和商业版，由于其体积⼩、速度快、总体拥有成本低，尤其是开放源码这⼀特点，⼀般中⼩型⽹站的开发都选择MySQL作为⽹站数据库。

·2.2 Redis数据库

本系统选为Redis实现分布式情况下的Session会话共享，保存用户的登录状态信息。

redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis 是一个高性能的key-value数据库。 redis的出现，很大程度补偿了memcached这类key/value存储的不足，在部 分场合可以对关系数据库起到很好的补充作用。它提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

* 1. SpringBoot

 Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。spring boot其实不是一个新的框架，它默认配置了很多框架的使用方式，就像maven整合了所有的jar包，spring boot整合了所有的框架。

  Spring Boot 简化了基于 Spring 的应用开发，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为 Spring 平台及第三方库提供开箱即用的设置，这样你就可以有条不紊地开始。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。采用 Spring Boot 可以大大的简化你的开发过程，所有的常用框架，它都有对应的组件支持。

SpringBoot可以快速创建独立运行的Spring项目以及与主流框架集成。使用嵌入式的Servlet容器，应用无需打成WAR包。有效解决了J2EE笨重的开发、繁多的配置、低下的开发效率、 复杂的部署流程、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。Spring Boot 的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，会给我们的开发带来非常大的便利，有效提高开发效率与质量。

* 1. SpringCloud

Spring Cloud 是一系列框架的有序集合，它利用 Spring Boot 的开发便利性简化了分布式系统的开发，比如服务发现、服务网关、服务路由、链路追踪等。Spring Cloud 并不重复造轮子，而是将市面上开发得比较好的模块集成进去，进行封装，从而减少了各模块的开发成本。换句话说：Spring Cloud 提供了构建分布式系统所需的“全家桶”。

相比于Dubbo 只是实现了服务治理而言， Spring Cloud 实现了微服务架构的方方面面。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 Netflix 开发的，一个基于 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。

Eureka Client：一个Java客户端，用于简化与 Eureka Server 的交互（通常就是微服务中的客户端和服务端）。

Eureka Server：提供服务注册和发现的能力（通常就是微服务中的注册中心）。

各个微服务启动时，会通过 Eureka Client 向 Eureka Server 注册自己，Eureka Server 会存储该服务的信息。也就是说，每个微服务的客户端和服务端，都会注册到 Eureka Server。

Eureka的工作原理如图2-1。

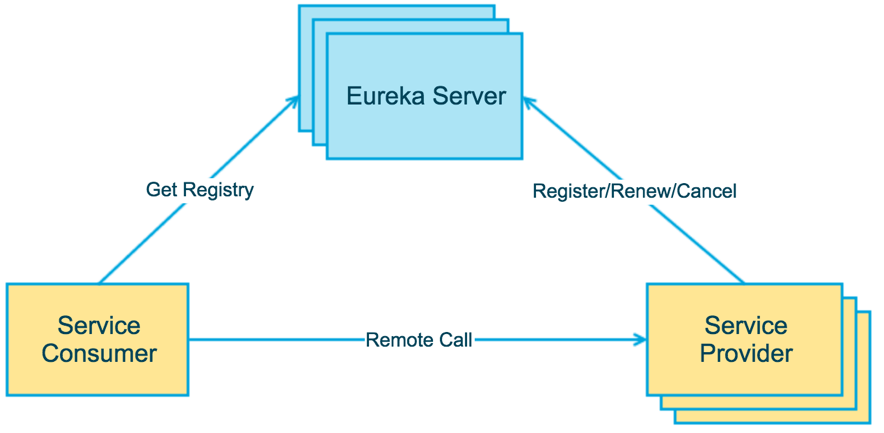


图2-1 Eureka工作原理架构图

1. SpringCloudConfig

SpringCloudConfig是SpringCloud创建的用来为分布式系统中的基础设施和微服务应用提供集中化的外部配置支持，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，是一个独立的微服务应用，用来连接配置仓库并为客户端提供获取配置信息，加密/解密信息等访问接口。而客户端则是微服务架构中各微服务应用或基础设施，通过指定的配置中心来管理应用资源与业务相关的配置内容，并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。

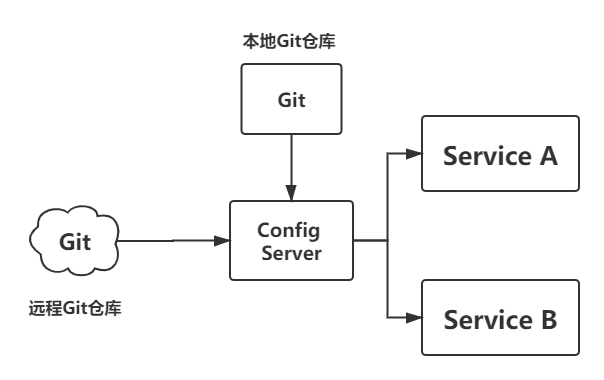


图2-2 SpringCloudConfig架构图

1. Feign

Feign默认集成了Ribbon，并和Eureka结合，默认实现了客户端负载均衡的效果。

在开发 Spring Cloud 微服务的时候，服务之间都是以 HTTP 接口的形式对外提供服务的，因此消费者在进行调用的时候，底层就是通过 HTTP Client 的这种方式进行访问。可以使用JDK原生的 URLConnection、Apache 的 HTTP Client、Netty 异步 Http Client，Spring 的 RestTemplate 去实现服务间的调用。但是最方便、最优雅的方式是通过 Spring Cloud Open Feign 进行服务间的调用 Spring Cloud 对 Feign 进行了增强，使 Feign 支持 Spring Mvc 的注解，并整合了 Ribbon 等，从而让 Feign 的使用更加方便。

Feign 是一个声明式的 Web Service 客户端。它的出现使开发 Web Service 客户端变得很简单。使用 Feign 只需要创建一个接口加上对应的注解。Feign 是一种声明式、模板化的 HTTP 客户端。在 Spring Cloud 中使用 Feign，可以做到使用 HTTP 请求访问远程服务，就像调用本地方法一样的，开发者完全感知不到这是在调用远程方法，更感知不到在访问 HTTP 请求。它整合了 Ribbon 和 Hystrix，从而不需要开发者针对 Feign 对其进行整合。Feign 还提供了 HTTP 请求的模板，通过编写简单的接口和注解，就可以定义好 HTTP 请求的参数、格式、地址等信息。Feign 会完全代理 HTTP 的请求，在使用过程中我们只需要依赖注入 Bean，然后调用对应的方法传递参数即可。

* 1. Bootstrap

Bootstrap是基于HTML、CSS、JavaScript开发的简洁、直观、强悍的前端开发框架，使得Web开发更加快捷。Bootstrap提供了实用性很强的组件，包括：导航，标签，工具条，按钮等供开发者使用。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。灵活简洁，代码优雅，美观大方，是当下最流行的前端框架。

* 1. Layui

本系统选用layui进行管理员后台管理页面的开发。

layui 是一款采用自身模块规范编写的前端 UI 框架,遵循原生 HTML/CSS/JS 的书写与组织形式,门槛极低,拿来即用。其外在极简,却又不失饱满的内在,体积轻盈,组件丰盈，从核心代码到 API 的每一处细节都经过精心雕琢，非常适合界面的快速开发。layui 首个版本发布于2016年秋，它区别于那些基于 MVVM 底层的 UI 框架，却并非逆道而行，而是信奉返璞归真之道。准确地说，它更多是为服务端程序员量身定做，开发者无需涉足各种前端工具的复杂配置，只需面对浏览器本身，使得开发十分便捷。

layui在组织形式上毅然采用了几年前的以浏览器为宿主的类 AMD 模块管理方式，更多的是面向于后端开发者，它拥有自己的模式，更加轻量和简单。layui 定义为“经典模块化”，并非是刻意强调“模块”理念本身，而是有意避开当下 JS 社区的主流方案，试图以尽可能简单的方式去诠释高效！它的所谓经典，是在于对返璞归真的执念，它以当前浏览器普通认可的方式去组织模块！ layui 认为这种轻量的组织方式，仍然可以填补 WebPack 以外的许多场景。所以它坚持采用经典模块化，也正是能让人避开工具的复杂配置，重新回归到原生态的 HTML/CSS/JavaScript本身。

layui 提供了丰富的内置模块，他们皆可通过模块化的方式按需加载。它兼容除了IE6/7外正在使用的全部浏览器，可作为 PC 端后台系统与前台界面的速成开发方案。

2.7 本章小结

本章主要对香港粗菜馆在线订座系统所用到的一些主要关键技术进行了简要介绍。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**

3.1 可行性分析

3.1.1 资源可行性

本次开发可利用的开发资源主要是基于windows系统的个人笔记本电脑，电脑安装并配置了IntelliJ IDEA和MySQL数据库，可利用Navicat工具进行数据库设计。并且在VmWare虚拟机上边安装了Redis数据库。以上都是较为成熟的开发工具。

开发过程中可以查阅到各方面的书籍，涉及软件工程、前端设计、用户界面、架构开发、数据库原理等。再借助网络和图书馆，在可查阅资源上十分充足。

本系统预计的学习、开发、产品测试所需时间为7个月。有相对足够的时间资源可保证香港粗菜馆在线订座系统按质量完成。

香港粗菜馆为一家真实存在的餐馆，可以在网络上方便的查找到餐馆的位置、历史、环境以及特色菜品等各种真实详细的信息，保证了香港粗菜馆在线订座系统的数据来源可靠，数据量充足。

综合上述，对于完成系统实现在软硬件环境、可参考资料信息以及开发时间等资源上具有可行性。

3.1.2 技术可行性

香港粗菜馆在线订座系统主要是基于SpringBoot和SpringCloud开发的一个微服务架构的在线订座系统。系统开发工具主要是IntelliJ IDEA和MySQL数据库。

在前端技术上，本人有一定的开发经验并上过相应的课程，可以熟练使用Bootstrap前端框架进行开发。此外layui和echarts等也是十分成熟的技术，可以有力保证开发的效果，以及前端界面的兼容性。

在后端技术上，Spring Boot和Spring Cloud是现在十分流行的分布式系统开发技术。并且Spring Cloud为微服务系统的开发提供了一站式解决方案。本人在毕业设计期间对于Spring Boot和Spring Cloud的架构有了一定的了解并进行深入的学习，有足够的能力完成框架的搭建。同时大学期间学习过数据库相关原理，对数据库的设计和操作比较了解。通过学习过的软件工程，软件测试等课程，掌握了一定的系统分析、设计和测试能力。对于高质量的开发系统都提供了一定的理论和技术支撑。

综合上述，对于完成系统实现在技术上完全具有可行性。

3.1.3 操作可行性

在开始开发系统之前进行了整个系统实现的项目规划，制定了项目的操作流程，并进行了初步的需求分析和系统设计。开发过程主要基于瀑布模型，将实验步骤分为制定计划、需求分析、软件设计、程序编写、软件测试、运行维护六个步骤，在实际开发中严格按照以上过程来执行。

综合上述以及资源和技术可行性，本系统具有很强的操作可行性。

3.2 需求分析

3.2.1 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。

根据用户名和密码进行登录。

1. 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。

1. 菜品管理

管理员可以上传菜品图片，价格描述等信息。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。

1. 订单管理

用户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。管理员可以分析每月的订单数量变化状况。

3.2.2 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。

3.3 开发环境

（1）硬件环境：

处理器： Intel Corei7-7700HQ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：MySQL5.6、Redis

开发语言： Java jdk 1.8

开发工具：IntelliJ IDEA、Navicat

3.4 本章小结

本章从资源、技术和可操作性三个反面对香港粗菜馆进行了可行性分析，之后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。最后介绍了香港粗菜馆在线订座系统的软硬件开发环境。

**4 系统设计**

4.1 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4-1。

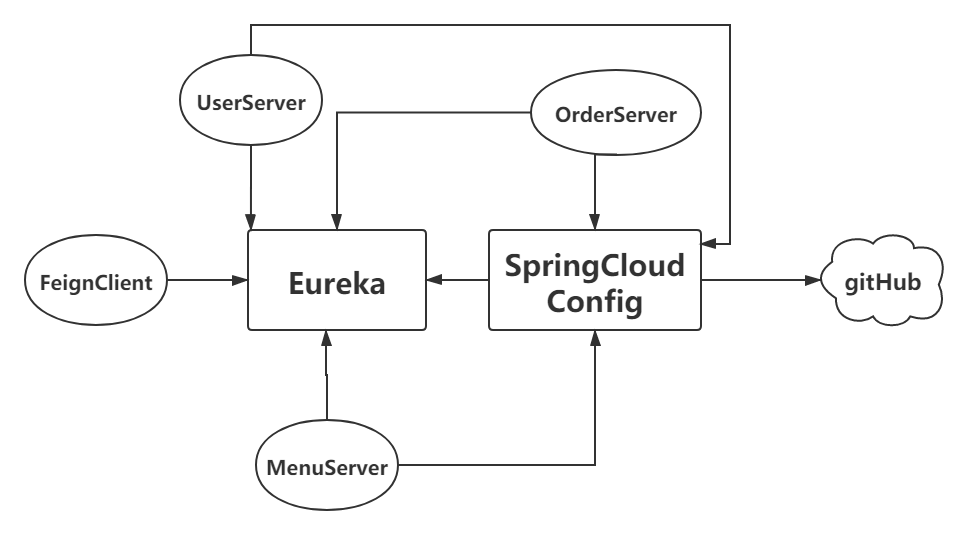


图4-1香港粗菜馆在线订座系统架构图

如图4-1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，有效的提高了系统的性能。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4-2。

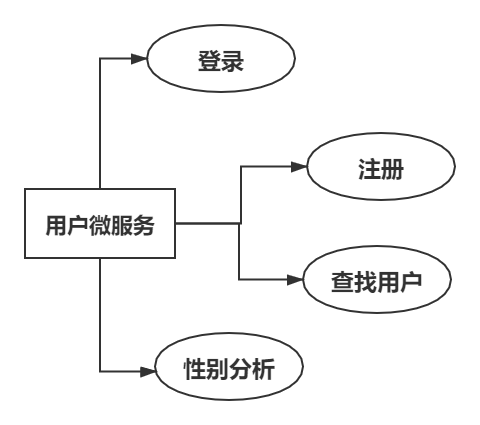


图4-2用户微服务功能结构图

菜品微服务所能提供的服务功能主要如图4-3。

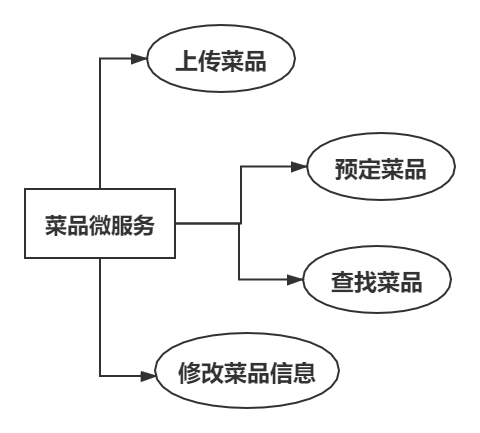


图4-3菜品微服务功能结构图

订单微服务所能提供的服务功能主要如图4-4。

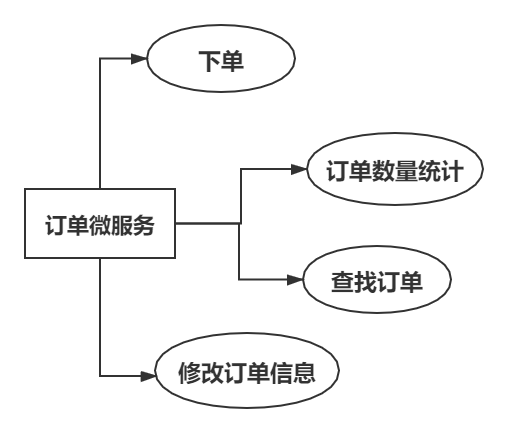


图4-4订单微服务功能结构图

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4-5。

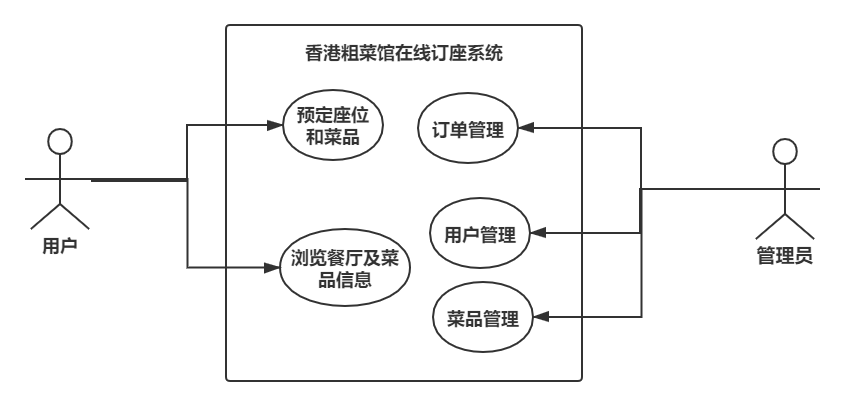


图4-5香港粗菜馆在线订座系统用例图

4.2 系统各功能模块设计

4.2.1 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

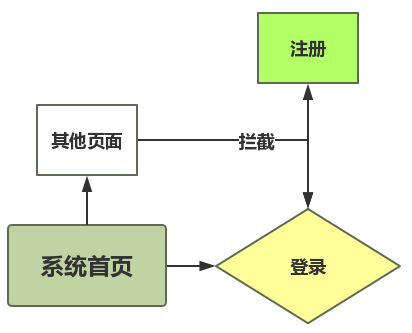
登录与注册的流程如图4-6。

图4-6登录与注册流程图

4.2.2 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4-7。

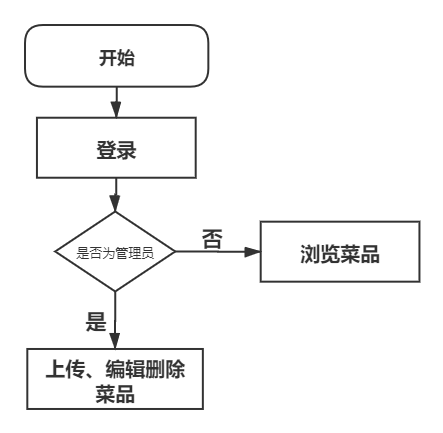


图4-7菜品管理流程图

4.2.3 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的提高利润。

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4-8。

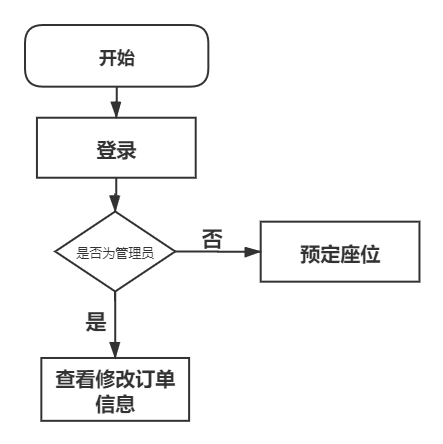


图4-8订单管理流程图

4.3 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4-9。

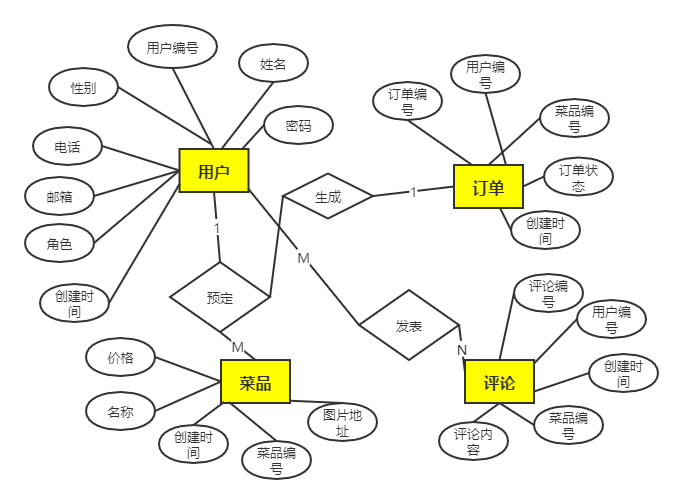


图4-9香港粗菜馆在线订座系统E-R图

系统数据库中各数据表详细信息如下。

用户信息表（user）如表4-1。

表4-1 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4-1 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4-2。

表4-2 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4-2 menu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4-3。

表4-3 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4-3 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4-4。

表4-4 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4-4 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。

4.4 本章小结

本章主要首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。

**5 系统实现和测试**

5.1 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。

5.1.1 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5-1。

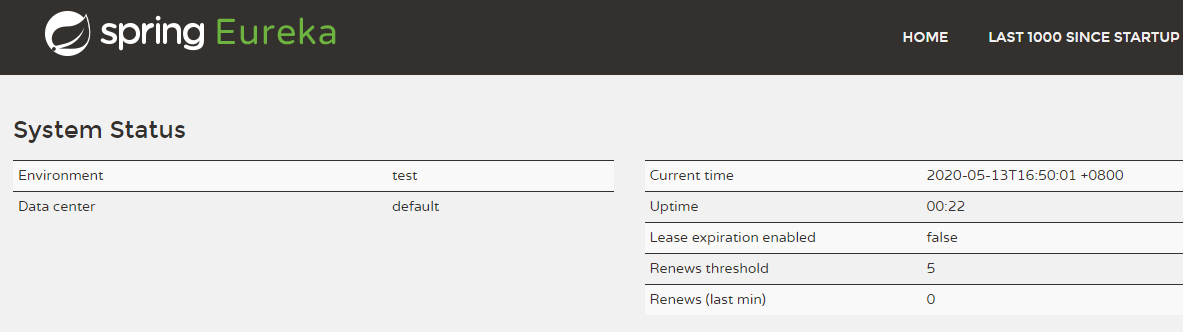


图5-1 服务注册中心搭建成功界面

5.1.2 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](http://localhost:3344/application-dev.yml%20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5-2。

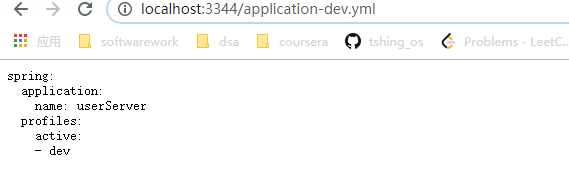


图5-2服务配置中心搭建成功界面

5.1.3 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。

5.1.4 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5-3。

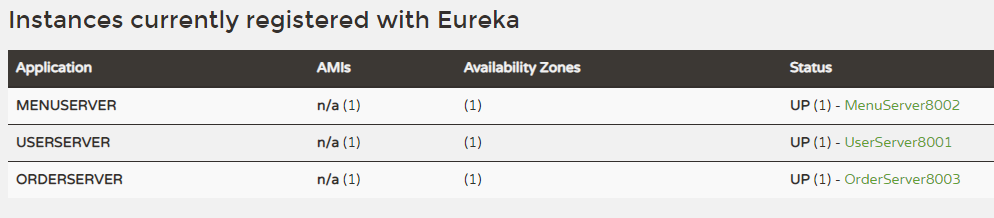


图5-3服务提供者启动成功界面

5.2 系统各功能模块实现与展示

5.2.1 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。

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);

}

（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);

}

（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5-4。

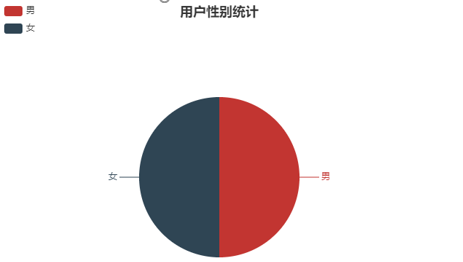


图5-4用户性别统计

5.2.2 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\\static\\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;

}

5.2.3 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。

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5-5和折线图如图5-6。

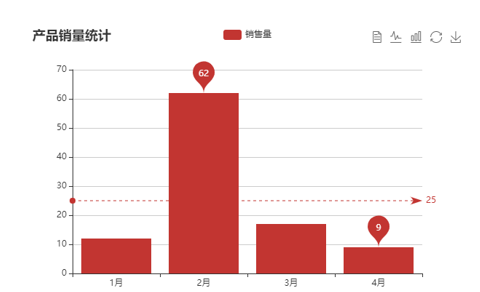


图5-5产品销量统计柱状图

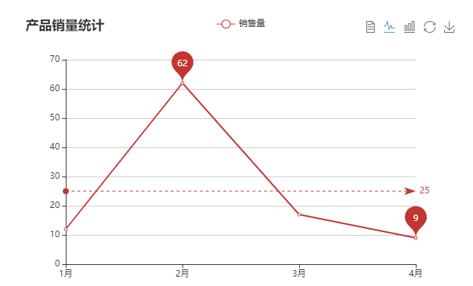


图5-6产品销量统计折线图

5.3 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。

5.4 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。

6 总结和展望

6.1 总结

本文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和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。

6.2 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很多不必要的困难。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.