中国地质大学（北京）

现代远程教育

本 科 毕 业 设 计

题　　目 基于微服务下的新零售电商平台管理系统设计

英文题目 Design of new retail e-commerce platform management system based on microservice

学生姓名 王刚　　**批　　次** 1809

专　　业 计算机科学与技术　　**学　　号** 1829010120029

指导教师 张鑫睿　　**职　　称** 中级网络工程师

学习中心 知金北京学习中心

2020 年 11 月

中国地质大学（北京）

现代远程教育

本 科 毕 业 设 计

题　　目 基于微服务下的新零售电商平台管理系统设计

英文题目 Design of new retail e-commerce platform management system based on microservice

学生签名： 王刚

指导教师签名：

2020 年 11 月

中文摘要

随着互联网时代的发展和创新，现有的项目管理系统越来越庞大，访问量和流量的增加使得当前系统出现了瓶颈。为了解决该问题，一种新型的理念被提了出来，那就是“微服务”。微服务是一种新型的软件架构，也是一种新理念和解决方案。微服务的理念是把一个大型的单个应用程序和服务拆分成数十个的支持服务。每个服务运行在其独立的自己的进程中，服务之间互相协调、互相配合，为用户提供最终的价值。服务之间采用轻量级的通信机制互相沟通。每个服务都围绕着具体业务进行构建，并且能独立的部署到生产环境、类生产环境等。从而达到应用之间的耦合性降低，一般会按照不同的业务逻辑来拆分。微服务设计的原则：1、各司其职 2、服务高可用和扩展性。微服务的核心就是将 传统的一站式应用，根据业务拆分成一个一个的服务，彻底的去耦合。每一个微服务提供单个业务功能的服务，一个服务做一件事，从技术角度看就是一种小而独立的处理过程，类似进程的概念，能够自行单独启动或销毁，拥有自己独立的数据库。另外，尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建，可以有一个非常轻量级的集中管理来协调这些服务，可以使用不同语言来编写，也可以使用不同的数据存储。本项目将采用 Java 语言为编写，并结合 SpringCloud 来完成微服务的架构。数据存储将采用 MySql 数据库来进行数据管理。核心技术框架：SpringCloud、SpringBoot、SpringData、Kafka、MyBatis。

关键字： 微服务； Java； SpringCloud； SpringBoot

ABSTRACT

With the development and innovation of the Internet era, the existing projectmanagement system is becoming more and more huge, and the increase of accessand traffic makes the current system appear bottleneck. In order to solve this problem, a new concept has been put forward, which is "micro service". Microservice is a new software architecture, and also a new concept and solution. The concept of microservice is to divide a large single application and service into dozens of supporting services. Each service runs in its own independent process, and services coordinate with each other to provide the final value for users. Services communicate with each other through lightweight communication mechanism. Each service is built around specific business, and can be independently deployed to production environment, class production environment, etc. In order to reduce the coupling between applications, it is generally split according to different business logic. The principles of microservice design are as follows: 1. Perform their own duties; 2. High availability and scalability of services. The core of microservice is to divide the traditional one-stop application into one service according to the business, and thoroughly decouple it. Each microservice provides a service with a single business function, and a service does one thing. From a technical point of view, it is a small and independent processing process, similar to the concept of process. It can start or destroy independently by itself, and has its own independent database. In addition, the unified and centralized service management mechanism should be avoided as far as possible. For a specific service, the appropriate language and tools should be selected according to the business context to build it. There can be a very lightweight centralized management to coordinate these services, which can be written in different languages or different data stores. This project will be written in Java language and combined with spring cloud to complete the microservice architecture. The data storage will use MySQL database for data management. Core technology framework: springcloud, springboot, springdata, Kafka, mybatis.

Key words: Microservices; Java; SpringCloud; SpringBoot

目　　录

引 言

随着互联网的技术和架构发展，越来越多的现有的项目追顺的时代的步伐进行升级，在互联网时代的冲击下，许多优秀的框架百花争放，我们选择主流的语言和主流的框架来完成本次项目的完成。自 2018 年来，我国越来越多的互联网公司从传统项目架构，转移到了微服务的架构中。微服务的出现的解决了很多问题。其中的有很多优点：1、对于单个服务足够的内聚也足够小并且易于管理，代码容易维护和理解这样聚焦一个指定的业务功能或业务需求；2、开发简单，开发效率高，一个服务可能就是专一的干一件事；3、由于单个微服务足够小，所以它可以由小规模的团队人员开发，这个小团队不需要是很多人，可以是 2 到 5 个人的开发人员组成；4、微服务出现的目的就是解耦的，是有功能意义的服务，无论在开发阶段还是部署阶段都是相互独立互不干扰；5、非常容易于和第三方继承，微服务允许容易且灵活的方式集成自动部署，通过集成工具，比如 Jenkins 和 Kubernets；6、非常容易维护单个的服务相对来说；7、每个单独的微服务都有自己数据存储能力，比如有自己的数据库，当然也可以共用统一的数据库；其缺点：1、由于是微服务架构，这样就的要求开发人员要处理并且兼顾分布式系统的复杂性；2、服务多了运维难度倍数增加，随着服务数量的增加，运维的压力也在一直增大；3、服务部署也变的增多，且依赖复杂；4、服务与服务之间的通讯的成本也增加了；5、数据一致性；6、系统集成测试；7、服务器的数量增加，成本高。

1、 微服务下新零售电商平台概述

1.1、 新零售电商平台概念及特点

随着互联网的发展，我们越发离不开互联网给我们带来的便利和快捷省事。当我们出差旅游入住酒店时，常常会碰到需要买些生活用品或一些其他商品，当我们需要买这些东西的时候，就的需要去酒店前台购买或酒店附近的商品或超市。这大大增加了用户的时间成本并且用户体验也很不友好。本平台就是为了解决用户的这些痛点，以方便用户体验和服务提升。真正做到用户服务体验的一个大提升，让用户体验的互联网所带来的快捷和方便。

1.2、 新零售电商平台作用及其内容

平台具有的服务很多，购买商品只是其中一个模块，其中还包括酒店部门的管理、部门人员的管理、酒店信息的维护、酒店的服务单、酒店房价的 wifi 密码连接……

我们会把酒店入住本平台，为该酒店创建部门，为每个部门创建房间，为每个房间生成一个独一无二的房间二维码。该二维码可以贴到用户显眼的地方，并提示用户可以扫码购物。这里扫码购物主要借助微信小程序，当用户通过扫码打开小程序会提示给用户当前所在酒店，所在的房间，并且还可以告知用户该房间的 wifi 密码。当用户下单并支付了，平台会给酒店商家接单端进行消息的推送。商家听到语音信息播报，进行接单操作，并为用户配送到房间。这里用户收货地址是选填的，因为下单的时候，平台会根据二维码锁定用户的房间号，用户填写了按照用户填写的配送，用户不填写按照用户扫码的房间进行配送。

2、 系统需求分析

2.1、可行性分析

2.1.1、操作可行性

该系统开发用了三个多月内完成的。前期主要是以学习未目的以及收集一些重要资料为主，接下来就是对系统的技术选型和系统的架构分析。业务需求分析并设计数据库，数据库表结构严格按照三范式而设计。前后端分离，面向接口编程开发。

2.1.2、经济可行性

经济可行性要是对该项目的经济能力和情况做一个客观的评价。目前，具备了运行网络平台的 MIS 的硬件基础，该系统的核心开发是本人惊醒设计编写，免费帮助设计并实施的。因此开发、设计这套系统的支出费用是学校可以承担的，所以在经济上是行得通的。

2.1.3、技术可行性

技术上的可行性主要考虑将来采用的硬件和软件技术能否满足用户提出的要求。基于当前的计算机网络技术和数据技术已成熟，而且管理信息系统（MIS）的各种开发技术也已经相当成熟，并且在各个领域都不乏成熟的案例。都有落地成熟的技术方案，所以开发一套网络平台的多用户共享信息的在线考试系统在技术上是可行的

2.2、方案的设计与对比

2.2.1、C/S 设计架构和 B/S 设计结构比较

Client/Server 模式,Browser/Server 模式是目前网络应用软件中运行的两大主要模式。C/S 模式的主要的缺点就是维护不方便、升级也相对来说麻烦。不管是 C/S 还是 B/S 在我们日常生活中都是很常见。，B/S 模式其实就是客户端一个标准的浏览器，服务器端就是 web server，而 Web server 与数据库和应用服务器的紧密结合，因此这种模式的应用范围在不

断的扩大，上网查询早已不是最基本最简单的了。事实上有很多企业业务系统，企业的 MIS 系统都是才用这种模式这是因为它扩展方便而且升级也相对简单不需要开发专门的客户端，大大降低了维护成本，对客户端的要求就是只要有浏览器就可以了。

2.2.2、系统模式的设计

系统分为三端，用户端是小程序，平台端是 PC 端，酒店既有 PC 后台端也有 app 接单端。开发人员面向接口编程，采用微服务架构，按照业务功能进行项目拆分，以达到解耦作用。

2.2.3、系统设计采用技术选型

处于安全性和健壮性我们采用 Java 作为编程语言，数据库采使用免费版的 MySql 关系型数据库。开发框架使用 SpringBoot+SpringCloud，消息采用 Kafka，各个服务之间调用采用 OpenFeign，注册中心和配置文件使用 Nacos，熔断机制使用 sentinel，网关采用 GateWay，系统部署使用虚拟化容器技术 Docker。整体架构图如图 1 所示：

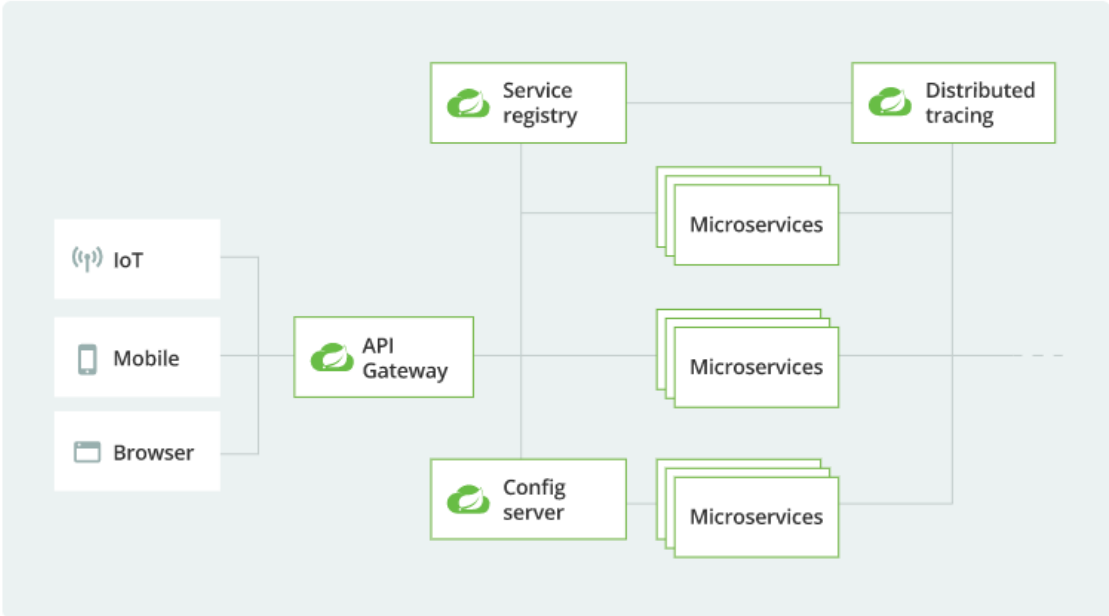

图 1：SpringCloud 架构图

3、 技术介绍

3．1、Java

3.1.1、发展史

Java 是一种语言，具有自己的语法和数据结构。Java 来 Sun(斯坦福大学网络)，Sun 为在 1991~1995 为了占领智能电子产品市场，由 James Goling 负责该项目，来开发 Oak 语言。1995 年 Oak 改名为 Java。1996 和 1997 陆续发布了 JDK1.0 和 JDK1.1，到了 1998 年发布 JDK1.2，将该版本命名为 j2SDK，将 Java 更名为 java2。1999 年将 Java 分为三大块：J2SE（Java 标准版）、J2EE（Java 企业版）、J2ME（Java 微型版）。2000 年和 2002 发布了 J2SE1.3 和 J2SE1.4。到了 2004 年此时不再叫 J2SE1.5，而是 5.0。2005 年 Java 10 周年，将 J2SE 改为 Java SE、将 J2EE 改为 JAVA EE、将 J2ME 改为了 Java 并发布了 Java SE 6。2009 年甲骨文公司宣布收购 Sun。2011 年甲骨文公司举行了全球性的活动，以庆祝 Java7 的推出。2014 年甲骨文公司发布了 Java8 正式版。

3.1.2、语言特性

Java 的语言特性有很多比如：简单性、面向对象、分布性、编译和解释性、可移植性、健壮性（自动垃圾回收机制，GC）、多线程性、动态性，做到一次编译，到处运行。

3.1.3、加载和执行

流程图：

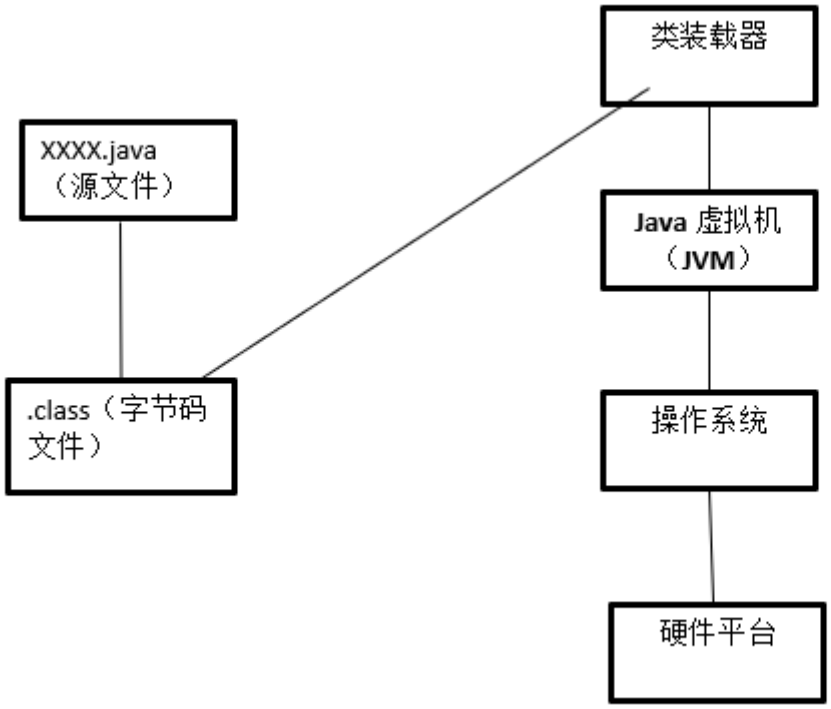

图 2：Java 加载执行图

3.1.4、编程环境

JDK(Java Development Kit)被称为 Java 的开发包也有的被称为 Java 开发工具包，它是一个编译 Java 的开发环境。JDK 是 Java 整个的核心，包含了 Java 运行环境(Java Runtime Environment)。由于 JDK 更新很快，从 1.8 以后就开始对 lamda 进行了支持，lamda 的出现大大提高了代码的质量和数量，本次用的是 11 作为环境。

JRE 和 JDK 的区别就是一个是运行环境一个是开发环境。如果是编写 Java 代码就安装 JKD，如果是只运行不编写代码那么只需要安装个 JRE 就完全够了。JDK 里面是包含 JRE 的，也就是说了只要安装了 JDK，就可以操作 Java 的代码同时也可以正常运行 Java 程序。但是还是建议根据自身的情况来采用不用的安装，如果只是运行建议安装 JRE，如果是开发编写建议请安装 JDK。

3.1.5、技术应用领域

Java 目前的应用领域是最多而且三方对它的支持也是最大的，已经渗入到我们生活的

方方面面：航空航天、移动通信、日常购物、推荐系统、支付、网站开发，科学应用……

3.2、Spring

3.2.1、概述

Spring 的兴起于 2003 年，它是一个框架一个轻量级的 Java 开发框架。它的出现大大解决了企业开发的复杂度。Spring 框架的核心就是控制反转(IoC)和面向切面编程(AOP)。

Spring 的核心作用就是要解决业务代码的“解耦”，降低代码和代码间的依赖性和耦合度。根据业务或则功能的不同，拆分为主业务逻辑与系统级业务逻辑两类。主业务逻辑和系统级业务它们各自具有鲜明的特点!其中主业务的联系非常紧密代码，有具体的专业业务应用场景，复用性相对较低;系统级业务相对功能独立，没有具体的专业业务应用场景，主要是为主业务提供系统级服务，如日志、安全、事务等，复用性强。

Spring 由于会以代码的功能，从而为降低耦合度的方式拆分为两大类: IoC 与 AOP。IoC 主要是在主业务的互相调用过程中不再自我维护关系了，也就是说自己需要在创建对象了。对象的创建都是由 Spring 容器来进行维护和管理，自动“注入”。因此 AOP 会让系统级服务能够得到最大的利用，而不是再需要程序员手工将系统级服务“参与”到主业务的逻辑中了，而是由 Spring 容器统一完成“ 织入”。

3.2.2、体系结构

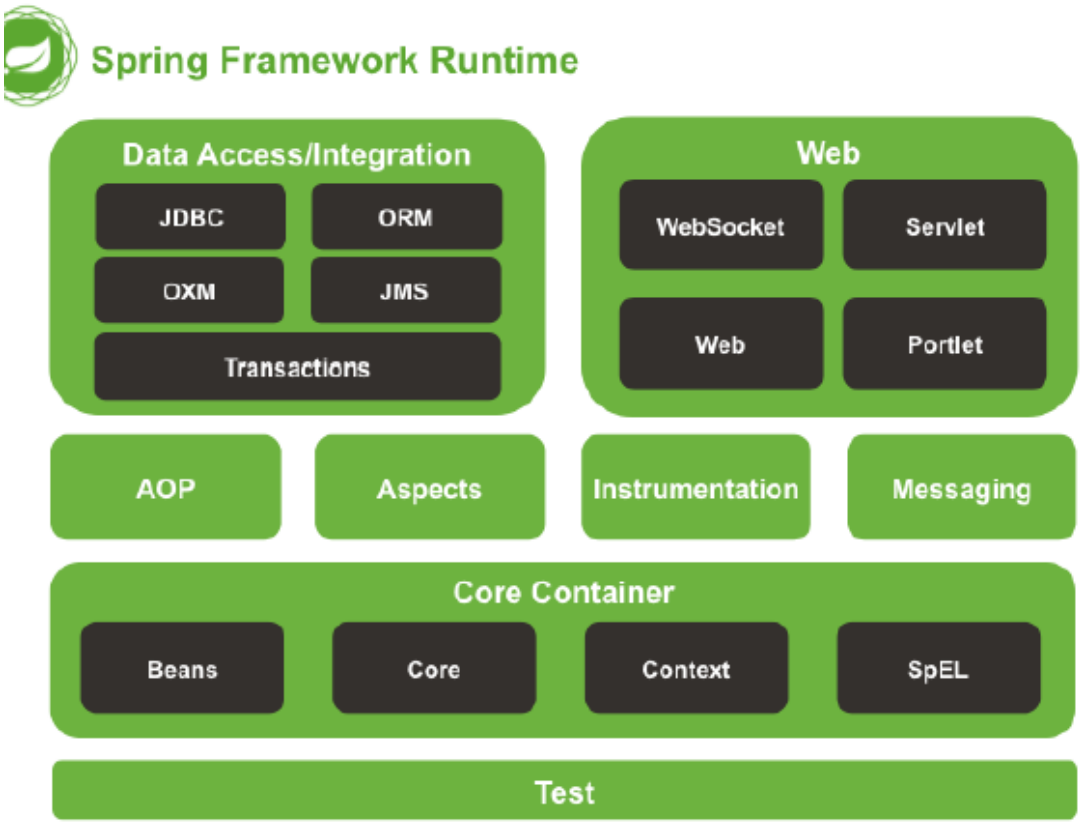

图 3：Spring 体系结构图

3.2.3、IoC

控制反转(Ioc,Inversion of Controller)，是一种概念，也是一种编程思想。Bean 的创建都是由程序员手动 new 出来，现在 Bean 的创建直接交付给容器来管理，通过容器来维护 Bean 的生命周期。控制反转说的就是对象的控制器转移，从手动到容器一个转变。

Ioc 不仅仅是一个概念，同事也是一种编程思想，虽然市面上的实现各种各样，但是就目前比较流行的实现方式有两种：依赖注入和依赖查找。依赖注入方式更为广泛。

依赖查找：Dependency Lookup，DL，组件是由容器提供接口和结合上下文，程序代码则需要提供具体查找方式。比较典型的是依赖于 JNDI 服务接口（Java Naming and e Directory Interface）的查找。

依赖注入：Dependency Injection，DI，程序代码不做定位查询，容器会帮我们自动完成这些工作。程序在运行过程依赖注入期间如果需要调用其他对象协助时，不需要在代码中

创建被调用者，通过容器依赖注入进来，容器创建成功后交付给程序。调用和被调用者在Srping容器来说没有任何具体的细节的要求，并且完全支持POJO依赖的管理关系维护。

依赖注入可以说在当前最为优秀没有之一的解耦方式。依赖注入通过配置文件和注解的方式让Spring和Bean之间组合在一起，而并没有以传统的硬编码的方式强制耦合在一起的。

3.2.4、AOP

AOP（Asect Orient Programming），面向切面编程，是面向对象编程OOP的一种补充。面向对象编程是从静态角度考虑程序的结构，面向切面编程是从动态角度考虑程序运行过程。

AOP底层，就是采用动态代理模式实现的。AOP的动态代理采用了两种：分别采取了JDK的动态代理和 CGLIB 的动态代理。面向切面编程，就是讲交叉业务逻辑封装成切面，利用AOP容器的功能将切面织入到主业务逻辑中。所谓交叉业务逻辑是指，通用的、与主业务逻辑无关的代码，如安全检查、事务、日志等。若不使用AOP，则会出现代码纠缠，即交叉业务逻辑与主业务逻辑混合在一起。这样会主业务逻辑变得的混合不清。

例如，转账，在真正业务逻辑前后，需要权限控制，日志管理、加载事务、结束事务等交叉业务逻辑，而这些业务逻辑与主业务逻辑并无直接关系。但它们的代码量所占比重能达到总代码量的一半甚至还多。它们的存在，不仅产生了大量的“冗余”代码，还大大干扰了主业务逻辑——转账。

3.2.5、总结

Spring的出现大大降低了开发的成本，也降低了代码的耦合度，从而做到真正的解耦，其源码的设计思想非常巧妙，值得我们深入学习和了解。

3.3、Spring MVC

3.3.1、简介

SpringMVC也叫Spring web mvc，属于表现层框架。SpringMVC是Spring框架的一部分，是在Spring3.0后发布的。

3.3.2、执行流程

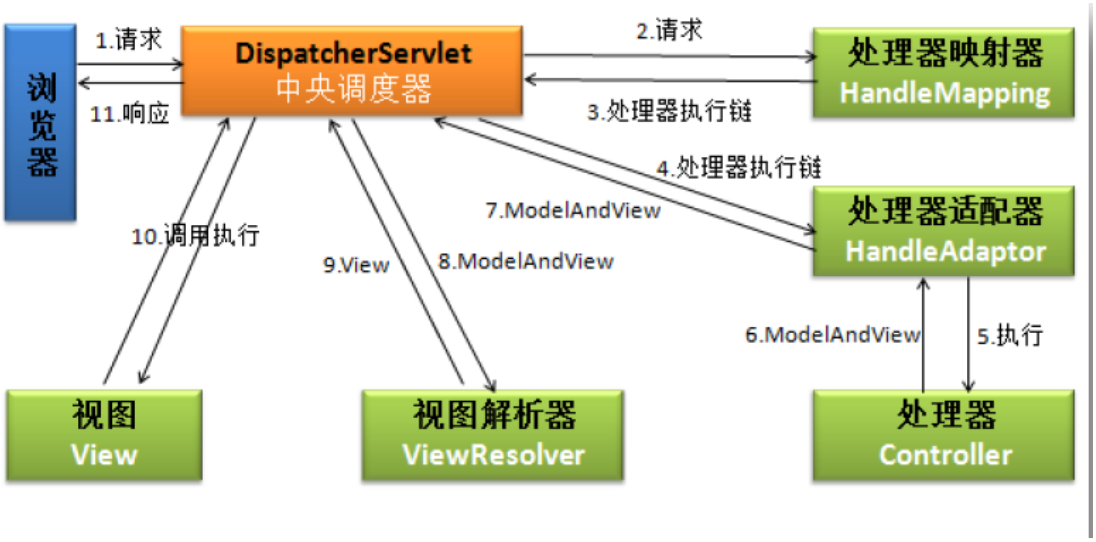

图 4：SpringMVC 执行流程图

浏览器提交请求到中央处理器，中央处理器直接将请求转给处理器映射器。处理器映射器根据请求会找到处理该请求的处理器，并将其封装为处理器执行链后返回给中央调度器。中央调度器根据传递过来的处理器执行链中的处理器，找到能够执行该处理器的对应的处理器适配器。处理器适配器直接将结果返回给中央调度器。中央调度器回去调用视图解析器，将 ModelAndView 中的视图进行封装成视图对象。中央调度器调用视图对象，让其自己进行渲染，即行数据填充，形成响应对象。中央调度器响应浏览器。

3.4、MyBatis

3.4.1、概述

MyBatis 原来是 apache 基金会下的一个开源项目 iBatis。2010 年这个项目从 apache 就迁移到了 Google 并且更名为 MyBatis。2013 年从 Google 又迁移到 GitHub。

3.4.2、简介

MyBatis 框架是一个很优秀的持久层框架并且基于 Java。由于它二封装了 JDBC，我们开发人员只管编写 SQL 语句既可，不需要再去关注驱动、配置 Statement、创建 Connection 等一切繁琐的过程，从而剩下时间更关注业务 SQL 的编写。

MyBatis 通过 xml 或注解的方式将要执行的各种 statement（statement、preparedStatement 等）配置起来，并通过 Java 对象和 Statement 中 SQL 的动态参数进行映射生成最终执行的 SQL 语句，最后 MyBatis 框架执行 SQL 并将结果映射成 Java 对象并返回。

3.4.3、MyBatis 与 Hibernate

Hibernate 框架是提供了全面的数据库封装机制的“全自动”ORM，即实现了 POJO 和数据库表之间的映射，以及 SQL 的自动生成和执行。

相对于此，MyBatis 只能算作是“半自动”ORM。其着力点，是在 POJO 类与 SQL 语句之间的映射关系。也就是说，MyBatis 并不会为程序员自动生成 SQL 语句。具体的 SQL 需要程序员自己编写，然后通过 SQL 语句映射文件，将 SQL 所需的参数，以及返回的结果字段映射到指定 POJO。 因此，MyBatis 成为了“全自动”ORM 的一种有益补充。

和 Hibernate 做比较，MyBatis 具有以下几个特点：(1)在 XML 文件中配置 SQL 语句，实现了 SQL 语句与代码的分离，给程序的维护带来了很大便利。(2)由于可以开发人员自行编写 SQL，这样灵活度就大大提升，即使复杂的业务多表多条件也能自我完成复杂的查询，也就体现出比 Hibernate 等全自动 ORM 框架更高的查询效率。(3) 简单，易于学习，易于使用，上手快，因此国内使用的 MyBatis 的人群远远要大于使用 Hibernate。

3.4.4、体系结构

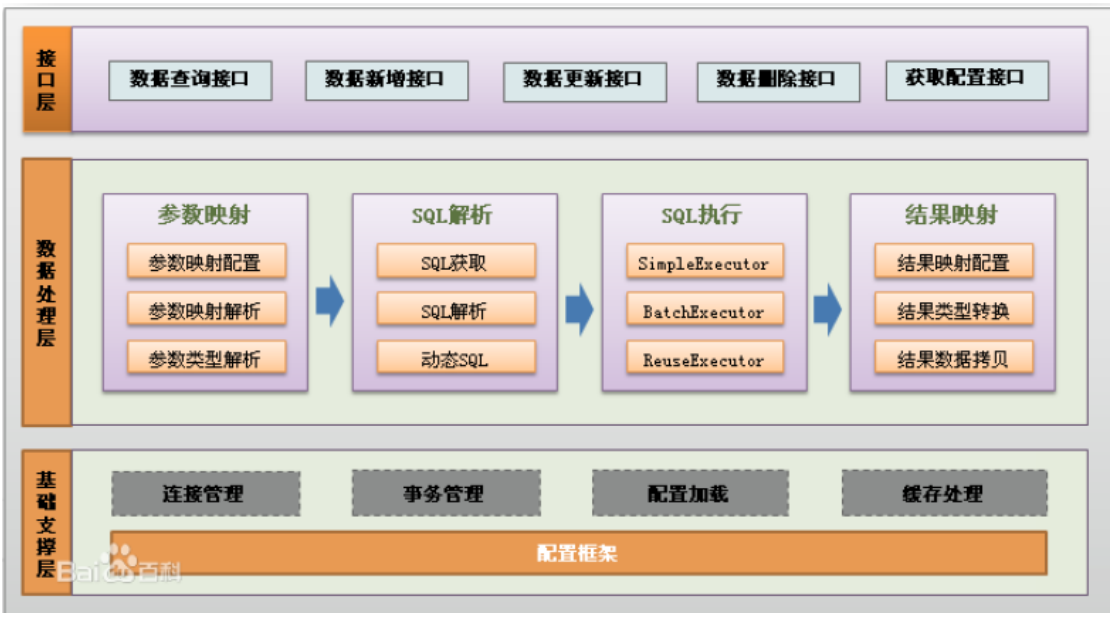

图 5：MyBatis 结构图

3.4.5、工作原理

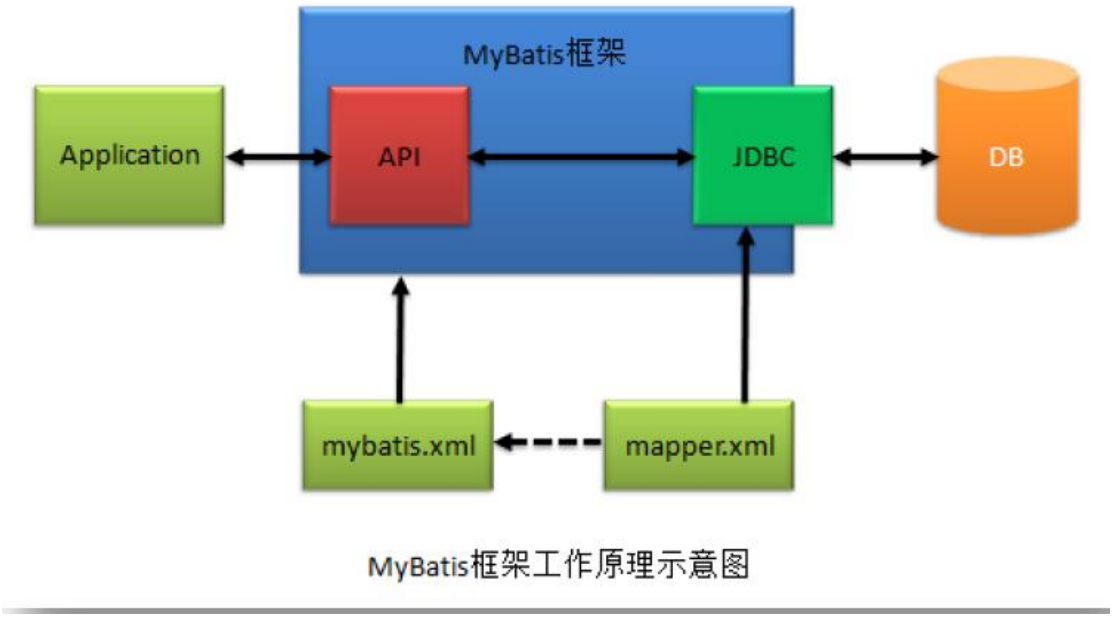

图 6：MyBatis 工作原理示意图

3.5、Spring Boot

3.5.1、简介

Spring Boot 是 Spring 家族中的一个全新的框架，它用来简化 Spring 应用程序的创建和开发过程，也可以说它能简化我们之前采用 SpringMVC+Spring+MyBatis 框架进行开发的的过程。在以往我们上述三大框架整合并搭配环境需要做很多工作，比如配置 web.xml 文件，配置 Spring，配置 MyBatis，并将他们整合在一起。而 Spring Boot 的框架出现对此过程进行了革命性的颠覆，抛弃了繁琐的 xml 配置过程，采用大量的默认配置简化我们的开发过程。所以采用 Spring Boot 可以非常容易和快速创建基于 Spring 框架的应用程序，它让编码变简单了，配置变简单了，部署也变得简单，更容易维护。正因为 Spring Boot 它化繁为简，让开发变得极其简单和快速，所以在业界备受关注。

3.5.2、特性

它能快速创建独立的 Spring Applications。能够直接使用 Java main 方法启动内嵌的 Tomcat，Jetty 运行 Spring Boot 程序，不需要部署 war 包文件。提供约定的 starter pom 来简化 Maven 配置，让 Maven 的配置变得的简单。根据项目的 Maven 依赖配置，Spring Boot 自动配置 Spring。提供了程序的健康检查等功能。基本可以完全不使用 xml 配置文件，采用注解配置。

3.5.3、核心

Spring Boot 的核心就是自动配置：针对很对 Spring 应用程序和常见的应用功能，Spring Boot 自动提供相关配置；起步依赖：告诉 Spring Boot 需要什么功能，它就能引入需要的依赖库。

3.6、Spring Cloud

3.6.1、概述

就目前而言，微服务并没有一个标准的标杆，对于微服务大家并没有一个统一的、标准的定义通常而言，微服务本身是一种思想同事也是一种架构，这种架构风格提倡将一个大的应用拆分出来，每个运行在自己独立的进程，然后每个服务与服务之间互相协调、互相配合，为用户提供最终价值。服务与服务通信都是采用轻量级机制一般都是基于 HTTP 的 RESTful API。业务的构建都是围绕着服务来说，由于每个服务足够单一，同事也独立开发部署，似的整理足够解耦。尽管如此那我们应该也尽量避免集中式的服务管理机制。就具体的服务来说应该按照业务逻辑或技术维度选择其更合适的开发工具和语言来构建，解决方案没有最合适的只有相对来说更合适的。根据业务的不同我们可以考虑采用是统一数据库管理也可以采取单个服务单个数据库，看业务和需求场景选择最合适自己当前的方案。

微服务化的核心就是将一个大的项目拆分成一个一个独立的项目，彻底的去耦合，每一个微服务提供单个业务功能的服务， 一个服务做一件事，从技术角度看就是一种小而独立的处理过程，类似进程的概念，能够自行单独启动或销毁，拥有自己独立的数据库。

SpringCloud，是在 SpringBoot 提供了一套解决方案和时间，微服务的和子组件包括配置中心、注册中心、网关，负载均衡，服务熔断，链路追踪。SpringCloud 利用 SpringBoot 的开发便利性巧妙地简化了分布式系统基础设施的开发，SpringCloud 为开发人员提供了快速构建分布式系统的一些工具包括配置管理、服务发现、断路由、微代理、事件总线、全局锁、决策竞选、分布式会话等，它们都可以用 SpringBoot 的开发风格做到一键启动和部 SpringBoot 并没有重复制造轮子，它只是将目前各家公司开发的比较成熟、经得起实际考验的服务框架组合起来，通过 SpringBoot 风格进行再封装屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包。SpringCloud = 分布式微服务架构下的一站式解决方案，是各个微服务架构落地技术的集合体，俗称微服务全家桶。

3.6.2、微服务与微服务架构

微服务是微服务架构的实现。微服务具体指的是服务大小，更多关注的其中一个点，是具体解决某一个问题/提供落地对应服务的一个服务应用，狭义的看，可以看做 Eclipse 里

面的一个个微服务工程/或者 Moudel。 微服务架构是一种全新的概念，与早年的单机版不同，现在的微服务更具有挑战性，服务服务之间的通信，数据传递，数据的一致都是需要重新去解决，但是微服务整体又是集群，容错性很高，服务也健壮一般我们集群都是奇数，对于高并发场景我们也可以在网关层做限流。

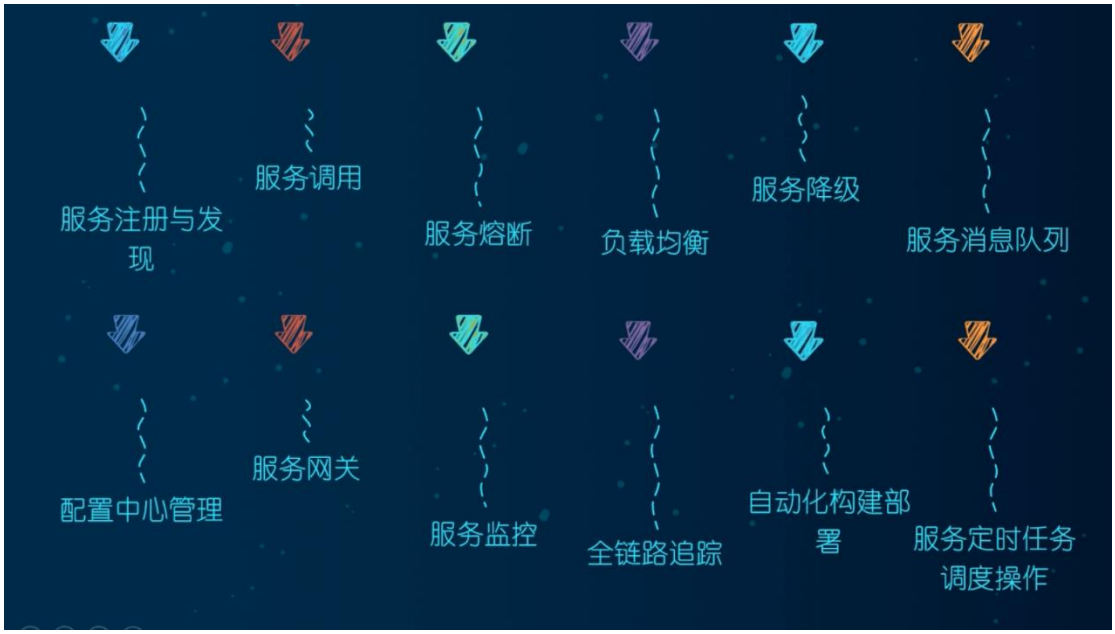

图 7：微服务架构图

3.6.3、Spring Cloud 和 Spring Boot 关系

SpringBoot 专注于快速方便的开发单个个体微服务。SpringCloud 是关注全局的微服务协调整理治理框架，它将 SpringBoot 开发的一个个单体微服务整合并管理起来，为各个微服务之间提供，配置管理，服务发现，断路由，路由，微代理，事件总线，全局锁，决策竞选，分布式会话等等集成服务。SpringBoot 可以离开 SpringCloud 独立开发项目，但是 SpringCloud 离不开 SpringBoot，属于依赖关系。SpringBoot 专注于快速、方便开发单个微服务个体，SpringCloud 关注全局的微服务治理框架。

3.6.4、优缺点

优点：每个服务就是一个独立的项目，代码依赖降低，单个业务或单个需求能成为一个需求、开发简单，开发效率高，一个服务可能就是专一的干一件事、 开发团队不需要很大，不管是开发还是部署都是独立，同时和第三方服务来说也容易集成，随着服务的细粒度拆分，部署这块都是采用 Kubectl 来进行一个服务的部署和维护，Kubectl 是服务的趋势同时也是

大势所趋。微服务是一种理念更是一种思想因此它的实现不局限某一种特定的开发语言。每个微服务可以有自己独立的数据库也可以和各个微服务共用一个数据库。

缺点：维护的成本变大、运维的难度增大、服务之间通信成本变高、开发的成本也大数据一致性、系统集成测试、系统监控、服务器数量的增加，成本高。

4、 系统的设计

4．1、系统结构图

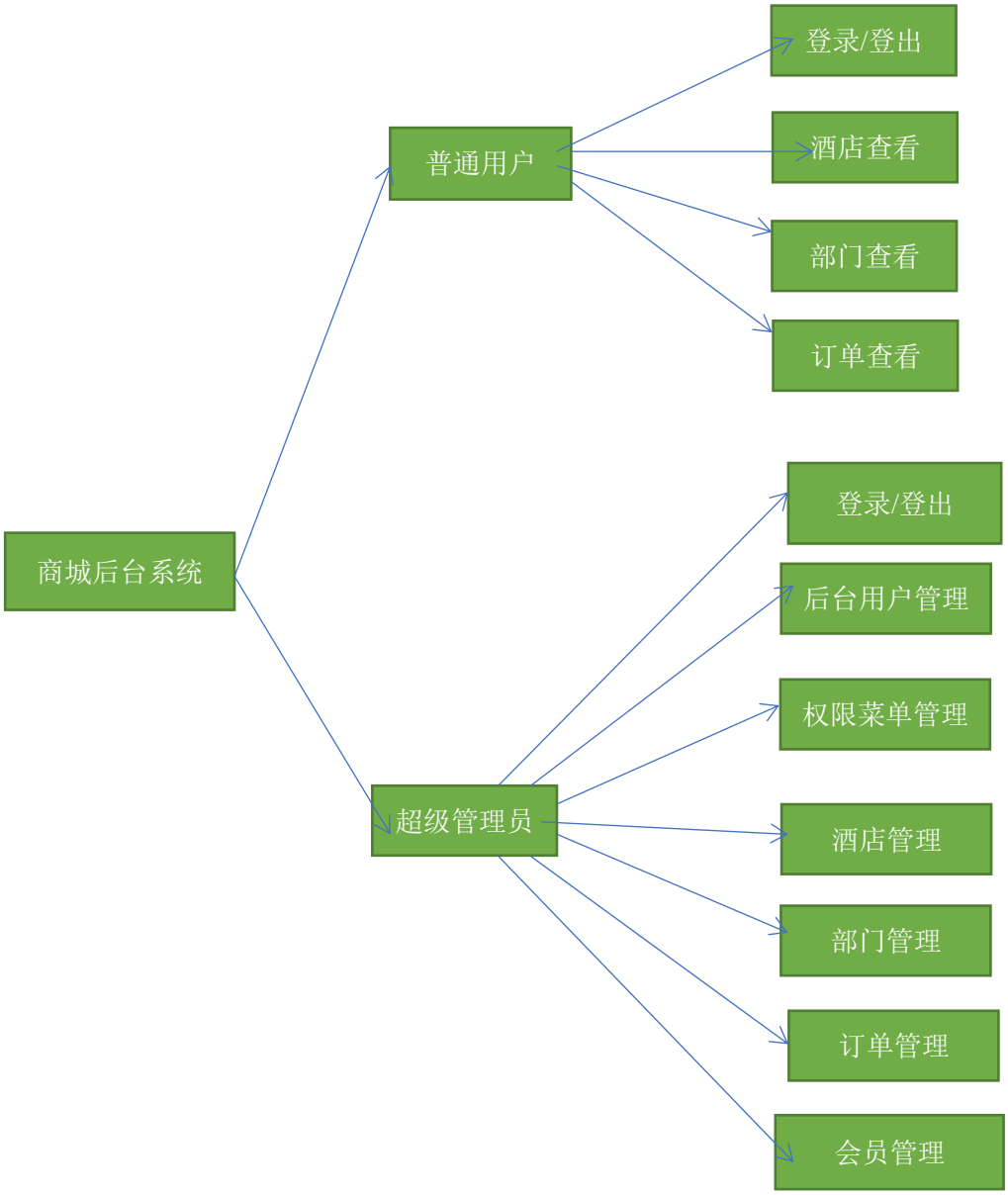

图 8：系统结构图

4．2、数据库表

(1)后台用户表：

后台用户表		
id主键	bigint(20)	<pk>
用户名	varchar(18)	
密码	varchar(18)	
启用标识:0启用;1禁用	bit(1)	
删除标识:0未删除;1已删除	bit(1)	
创建时间	datetime	
修改时间	datetime	
乐观锁	bigint(20)	

图 9：系统后台用户表图

(2)后台角色表：

后台角色表		
id主键	bigint(20)	<pk>
角色名称	varchar(20)	
角色备注	varchar(20)	
创建时间	datetime	
修改时间	datetime	
乐观锁	bigint(20)	

图 10：系统后台角色表图

(3)后台权限表：

后台权限表		
id主键	bigint(20)	<pk>
菜单编码	varchar(20)	
菜单名称	varchar(20)	
模快路径	varchar(50)	
请求路径	varchar(50)	
排序号	bigint(3)	
父级id	bigint(20)	
创建时间	datetime	
修改时间	datetime	
删除标识:0未删除;1已删除	bit(1)	
乐观锁	bigint(20)	

图 11：系统后台权限表图

(4)后台用户角色关联关系表：

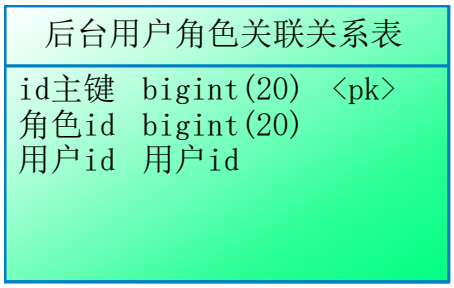

图 12：系统后台权用户角色关联关系表图

(5)后台角色权限关联关系表：

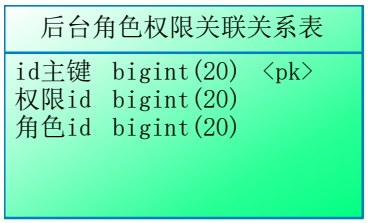

图 13：系统后台角色权限关联关系表图

(6)企业表：

图 14：系统后台企业表图

(7)企业部门表：

企业部门表		
id主键	bigint(20)	<pk>
所属企业id	bigint(20)	
部门名称	varchar(20)	
部门展示名称	varchar(20)	
服务电话	varchar(20)	
营业时间段	varchar(30)	
挂账支付:1是;2不是	varchar(2)	
现金支付:1是;2不是	varchar(2)	
配送费抽成	int(6)	
平台商品流水抽成	int(6)	
全局挂账金额	decimal(19,2)	
员工让利	int(6)	
创建时间	datetime	
修改时间	datetime	
删除标识:0未删除;1已删除	bit(1)	
乐观锁	bigint(20)	

图 15：系统后台企业部门表图

(8)企业部门楼层表：

企业部门表		
id主键	bigint(20)	<pk>
所属企业id	bigint(20)	
部门名称	varchar(20)	
部门展示名称	varchar(20)	
服务电话	varchar(20)	
营业时间段	varchar(30)	
挂账支付:1是;2不是	varchar(2)	
现金支付:1是;2不是	varchar(2)	
配送费抽成	int(6)	
平台商品流水抽成	int(6)	
全局挂账金额	decimal(19,2)	
员工让利	int(6)	
创建时间	datetime	
修改时间	datetime	
删除标识:0未删除;1已删除	bit(1)	
乐观锁	bigint(20)	

图 16：系统后台企业部门楼层表图

(9)企业部门房间表：

企业部门房间表		
id主键	bigint(20)	<pk>
所属企业id	bigint(20)	
所属企业部门id	bigint(20)	
楼层id	bigint(20)	
房间号	varchar(20)	
房间类型:0标间;1套房	char(1)	
房态:0空房;1住客房	char(1)	
删除标识:0未删除;1已删除	bit(1)	
创建时间	datetime	
修改时间	datetime	
乐观锁	bigint(20)	

图 17：系统后台企业部门房间表图

(10)企业银行账户表：

企业银行账户表		
id主键	bigint(20)	<pk>
所属企业id	bigint(20)	
开户名	varchar(25)	
银行卡号	varchar(35)	
排序号	int(5)	
开户行名	varchar(35)	
收款单位	varchar(50)	
开户支行	varchar(50)	
银行卡正反面	varchar(255)	
默认银行卡:0是;1不是	bit(1)	
审核状态	varchar(10)	
审核备注	varchar(50)	
审核通过时间	datetime	
申请时间	datetime	
删除标识:0未删除;1删除	bit(1)	

图 18：系统后台企业银行账户表图

(11)订单主表：

订单主表		
id主键	bigint(20)	<pk>
购买人id	bigint(20)	
企业id	bigint(20)	
部门id	bigint(20)	
房间id	bigint(20)	
房间号	varchar(20)	
订单状态:created创建;payed已支付;dispatched派送中;canceled已取消;success已完成;close已关闭	varchar(8)	
订单来源:1酒店;2物业	char(1)	
支付方式:0微信;1支付宝;2银行卡;3现金支付;4挂账	char(1)	
配送费金额	decimal(19,2)	
挂账支付类型:0无;1企业;2个人	char(1)	
挂房账的备注	varchar(20)	
订单类型:0普通;1内部员工	char(1)	
商品合计价格	decimal(19,2)	
合计价格(扣除退款)	decimal(19,2)	
优惠金额	decimal(19,2)	
应分润	decimal(19,2)	
可分润金额	decimal(19,2)	
实际分润	decimal(19,2)	
酒店分润比	decimal(19,2)	
酒店分润总比	decimal(19,2)	
员工优惠比	decimal(19,2)	
接单员	bigint(20)	
接单时间	datetime	
完成时间	datetime	
删除标识:0未删除;1已删除	bit(1)	
用户删除:0未删除;1已删除	bit(1)	
创建时间	datetime	
修改时间	datetime	
支付完成时间	datetime	
乐观锁	bigint(20)	
订单渠道类型:0自营;1三方;2商户	char(1)	
订单退款类型:0未退款;1部分退款;2全部退款	char(1)	
退款状态:0退款中;1已退款;2未退款;3退款失败	char(1)	
是否是父单FLAG:0是;1不是	bit(1)	
是否分润:0分润;2不分润	bit(1)	
主订单标识,所有不拆单的订单和父单:0是;1不是	bit(1)	
订单父ID,父级填0	bigint(20)	
实际支付金额	decimal(19,2)	
订单退款金额	decimal(19,2)	
备注	varchar(20)	
退款时间	datetime	
过期时间	datetime	
订单类型:1酒店;2:商户	char(1)	
配送方式	varchar(20)	
快递单号	varchar(20)	
收货地址id	bigint(20)	

图 19：系统后台订单主表图

(12)订单子表：

子单表		
id主键	bigint(20)	<pk>
父单id	bigint(20)	
部门id	bigint(20)	
商品id	bigint(20)	
商品名称	varchar(20)	
商品价格	decimal(19,2)	
商品实付价格	decimal(19,2)	
数量	int(6)	
合计价格	decimal(19,2)	
删除标识:0未删除;1已删除	bit(1)	
创建时间	datetime	
修改时间	datetime	
乐观锁	int(6)	
退款状态:0退款中;1已退款;2已完成;3退款失败	char(1)	
退款数量	int(6)	
退款理由	varchar(255)	
退款时间	datetime	
供货价格	decimal(19,2)	
餐盒费	decimal(19,2)	
优惠后的包装费	decimal(19,2)	
内部员工让利比例	decimal(19,2)	

图 20：系统后台订单子表图

(13)用户表：

图 21：系统用户表图

(14)用户收货地址表：

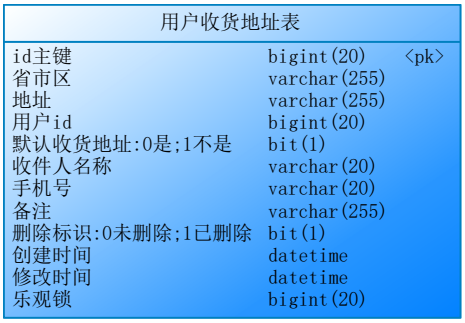

图 22：系统用户收货地址表图

(15)商品表：

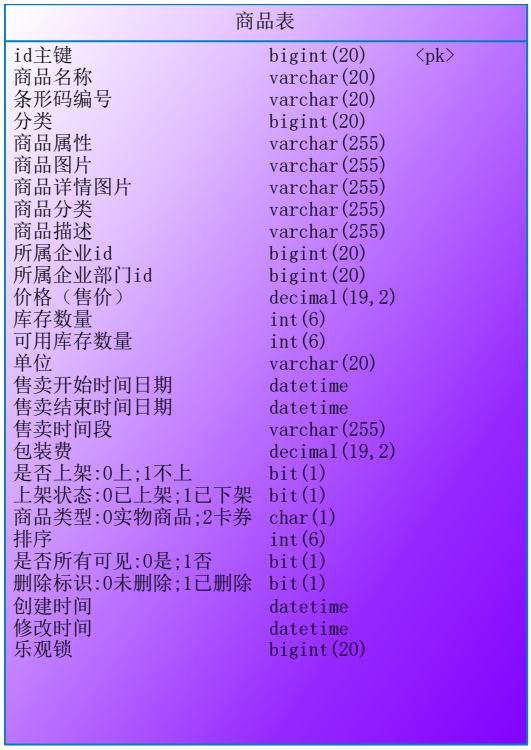

图 23：系统商品表图

(16)商品扩展表：

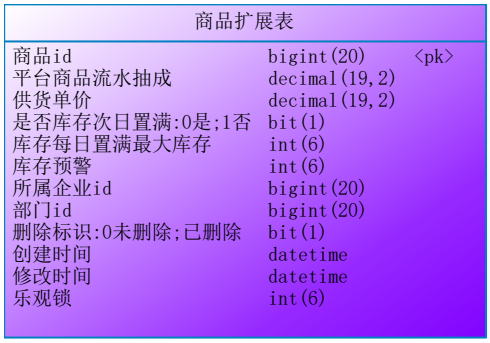

图 24：系统商品扩展表图

(17)分类表：

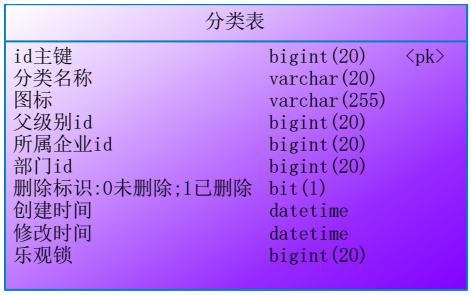

图 25：系统商品分类表图

(18)商品分类关联关系表：

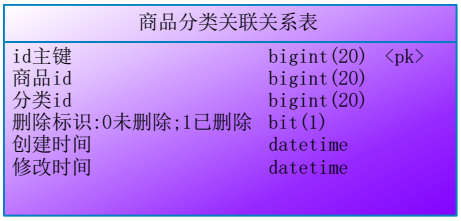

图 26：系统商品分类关联关系表图

(19)库存锁定记录表：

图 27：系统库存锁定记录表图

4．3、数据库表总体预览

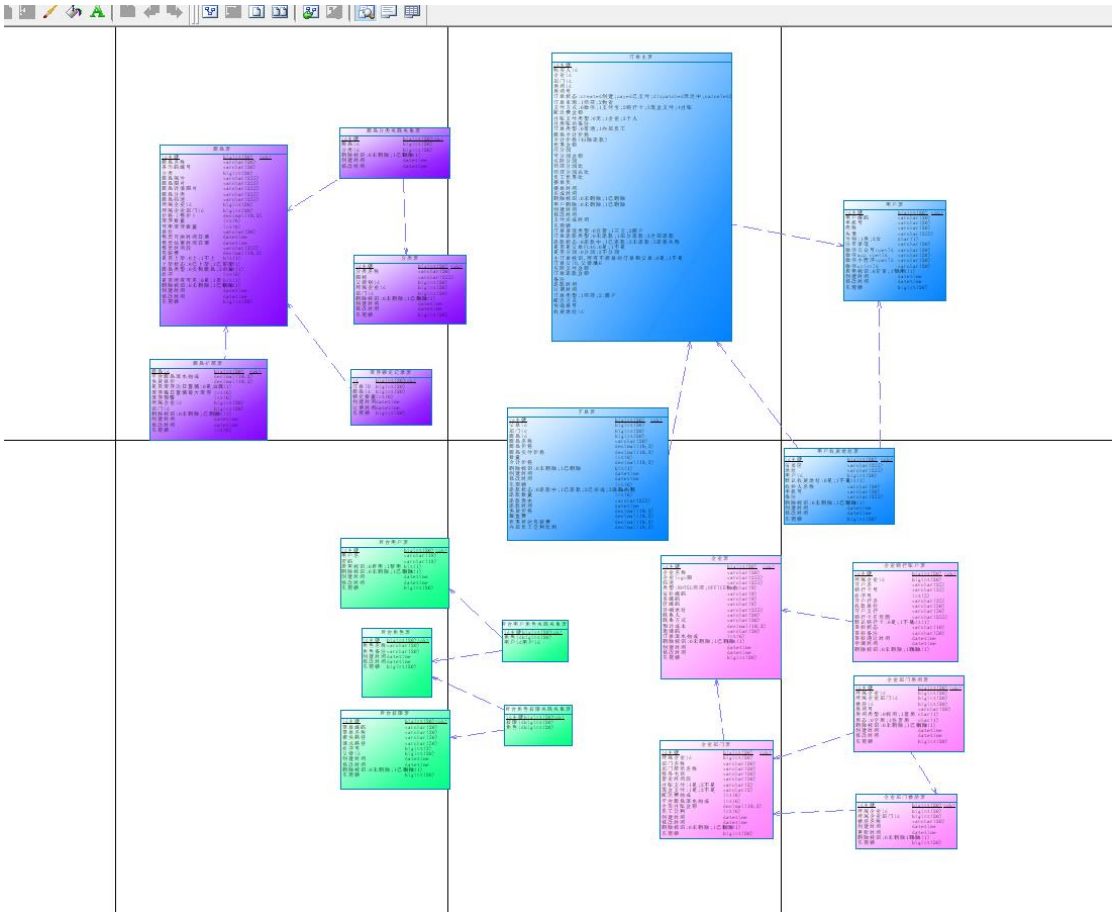

图 28：数据库整体关系图 a

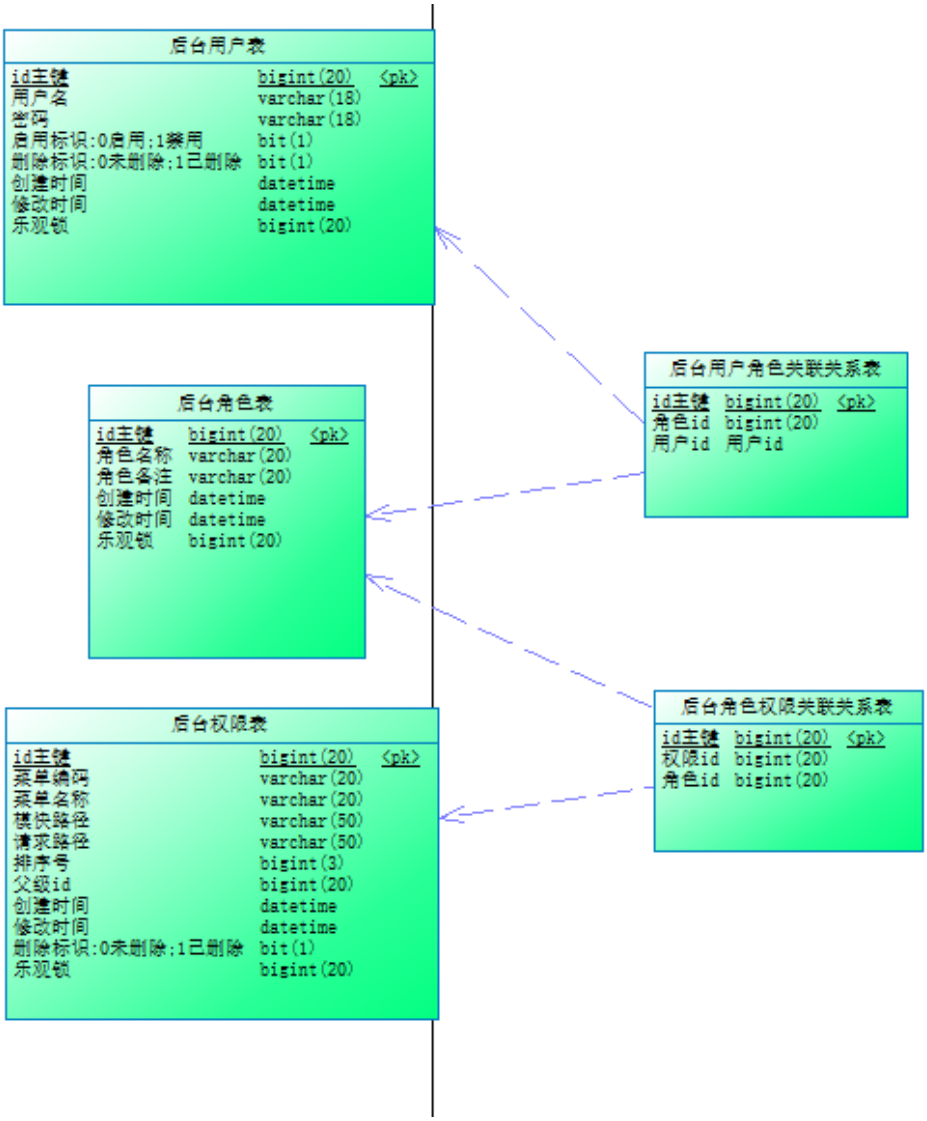

图 29：数据库整体关系图 b

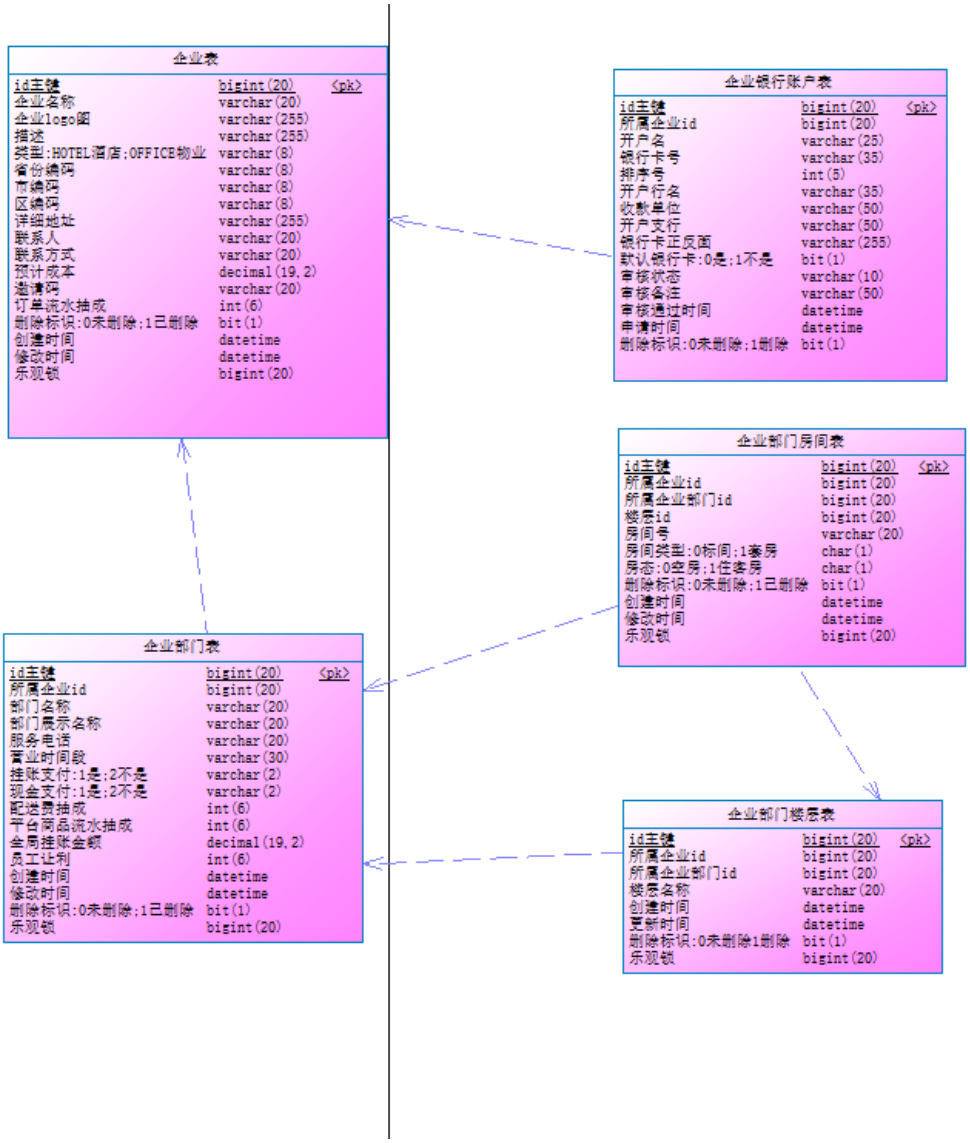

图 30：数据库整体关系图 c

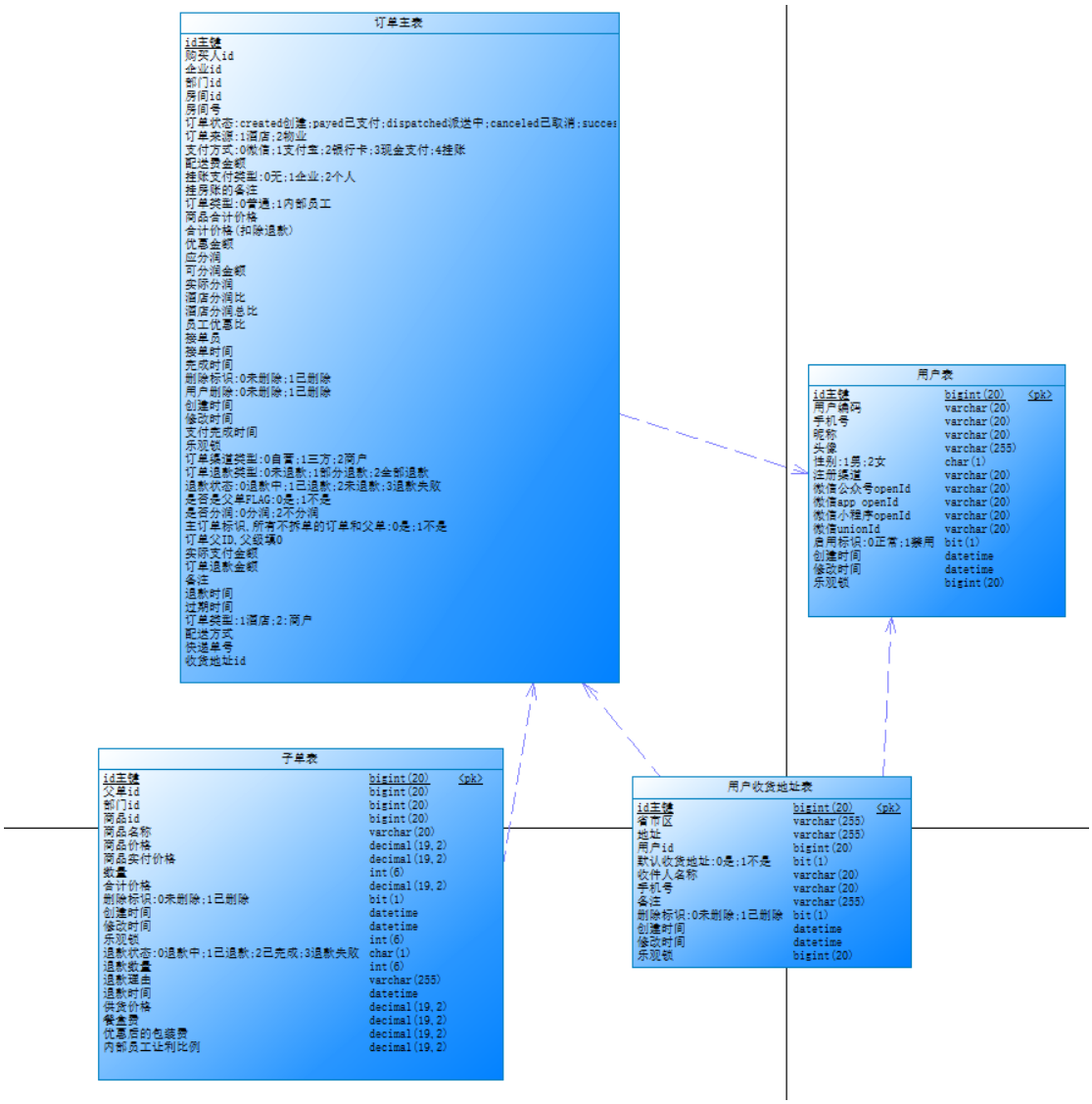

图 31：数据库整体关系图 d

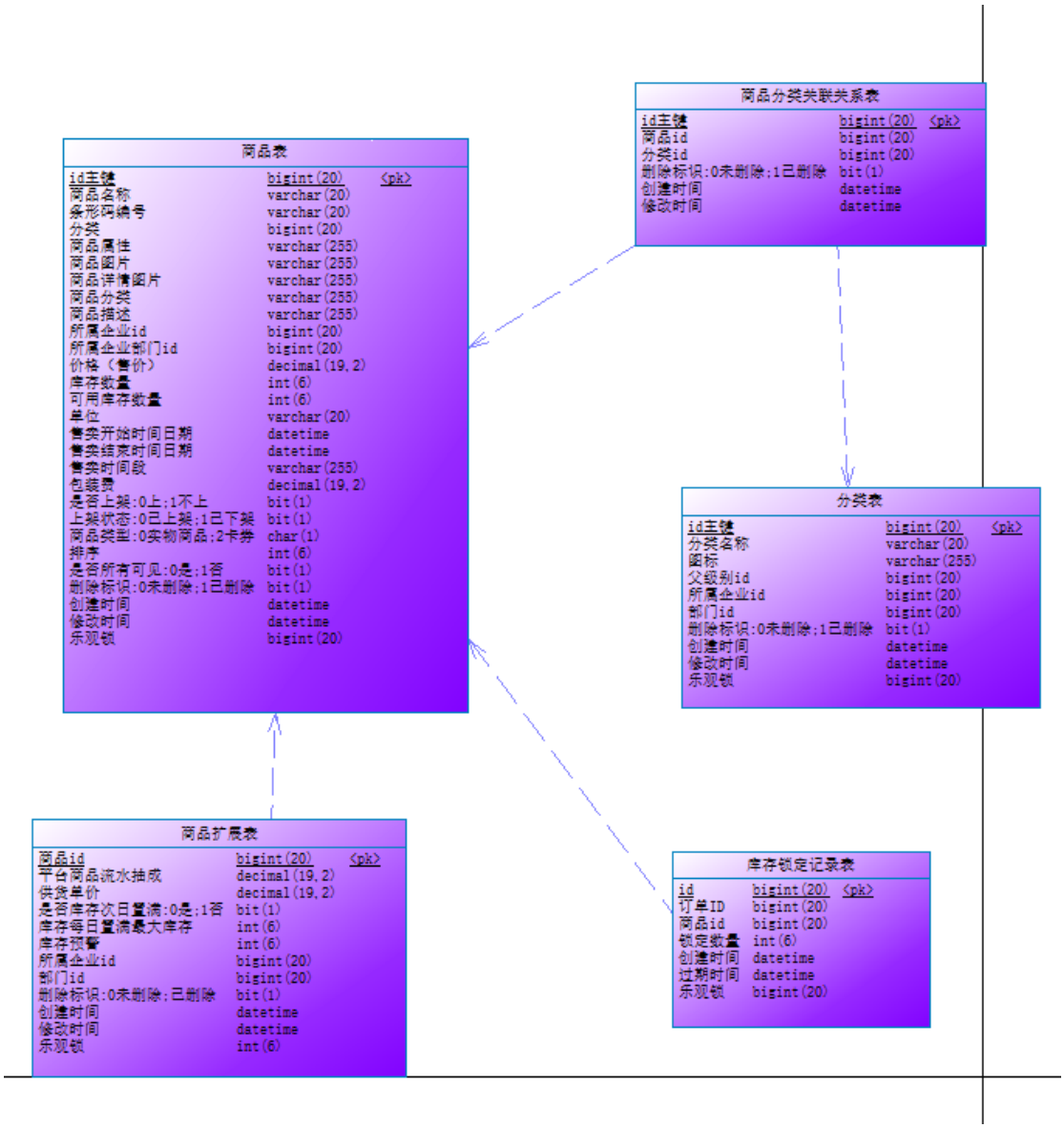

图 32：数据库整体关系图 e

4．4、总结

数据库在设计表结构的时候结合业务场景并严格遵守三范式来设计数据库系统，每张表都有自己的主键，严格禁止使用主外键关联，同时也严格禁止使用存储过程。每张表的主键都是使用长整型，使用长整型的好处就是排序可以通过主键来排序并保证使用到主键的索引来提高效率。MySql 在 5.7 版本以后就支持了 json 字段的类型，json 利用好了会解决很多问题。

5、 系统的实现

5.1、 后台登录

登录很简单只需要输入正确账号和正确的密码，登录失败有错误信息提示。登录成功后台会生成 token，做为前端交互的凭证。

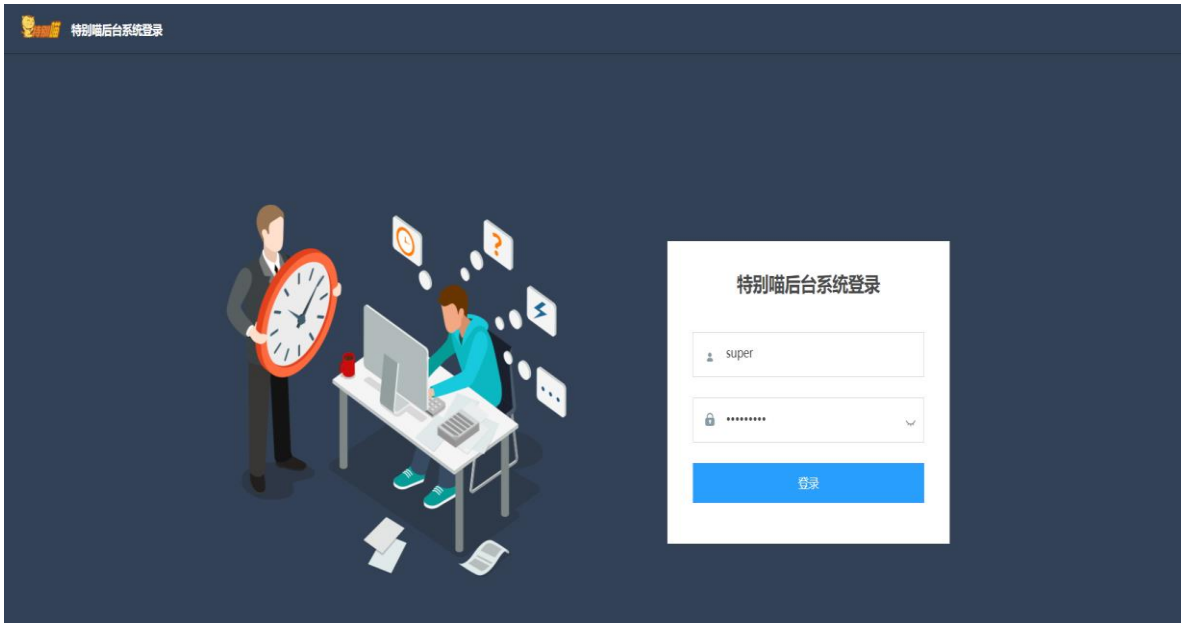

图 33：后台登录图

```
@PostMapping("/login")
@NotLoginRequired
@ApiOperation(value = "登录",notes = "", httpMethod = "POST")
public Result<Map<String, String>> login(@RequestBody LoginParam param) {
    log.info("userInfo: {}", param);
    Map<String, String> info = new HashMap<>();

    String token = userService.getToken(param);

    info.put("token", token);
    return Result.ok(info);
}
```

5.2、 首页

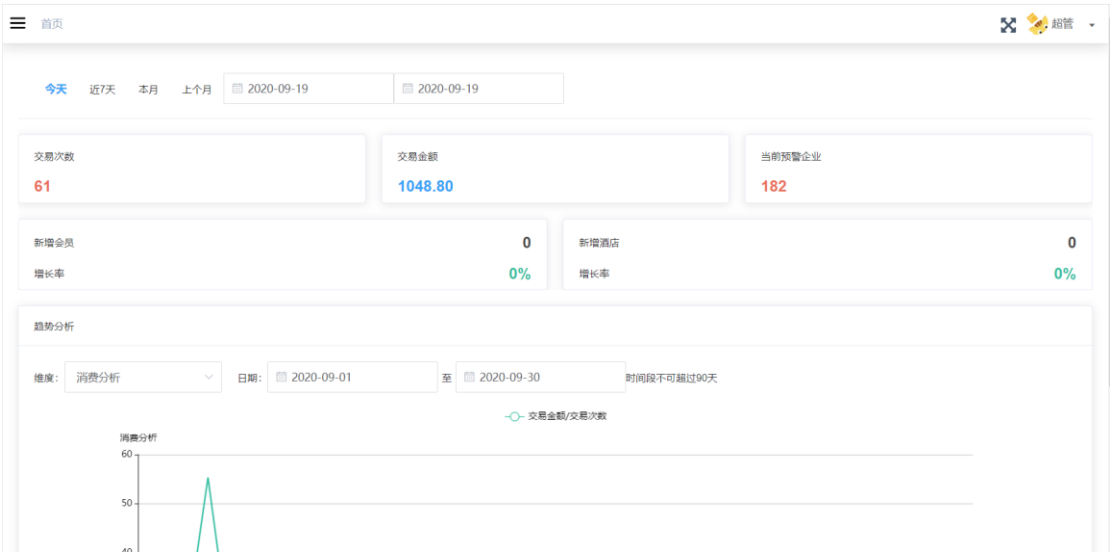

图 34：登录成功后首页图

5.3、 酒店管理

酒店是入住平台的会员，其实酒店相关功能包括基本信息，银行账户、部门管理、内部人员、代理信息、商品列表、合作商户、小程序首页、专题页、轮播图、公告/通知、提货点。

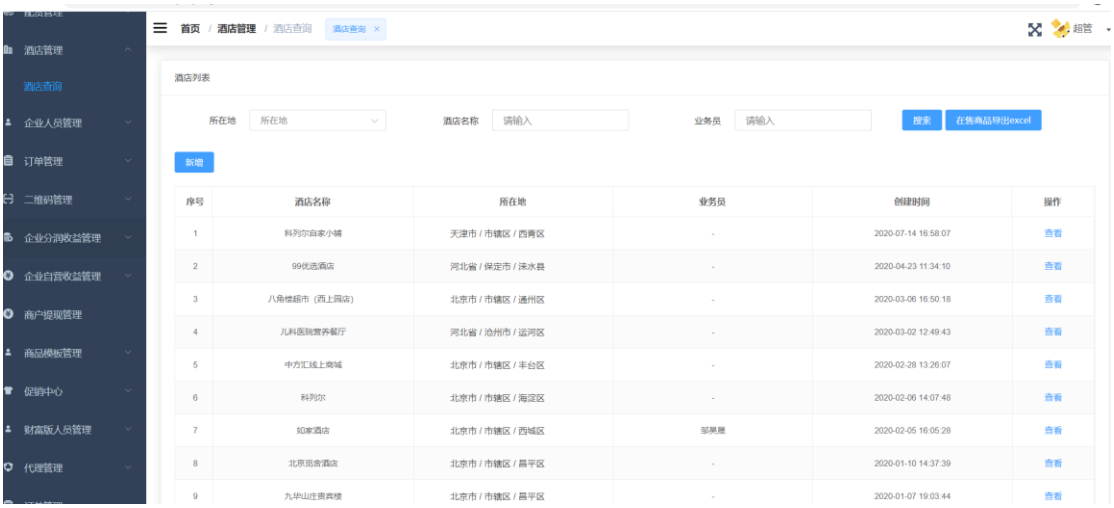

图 35：酒店分页图

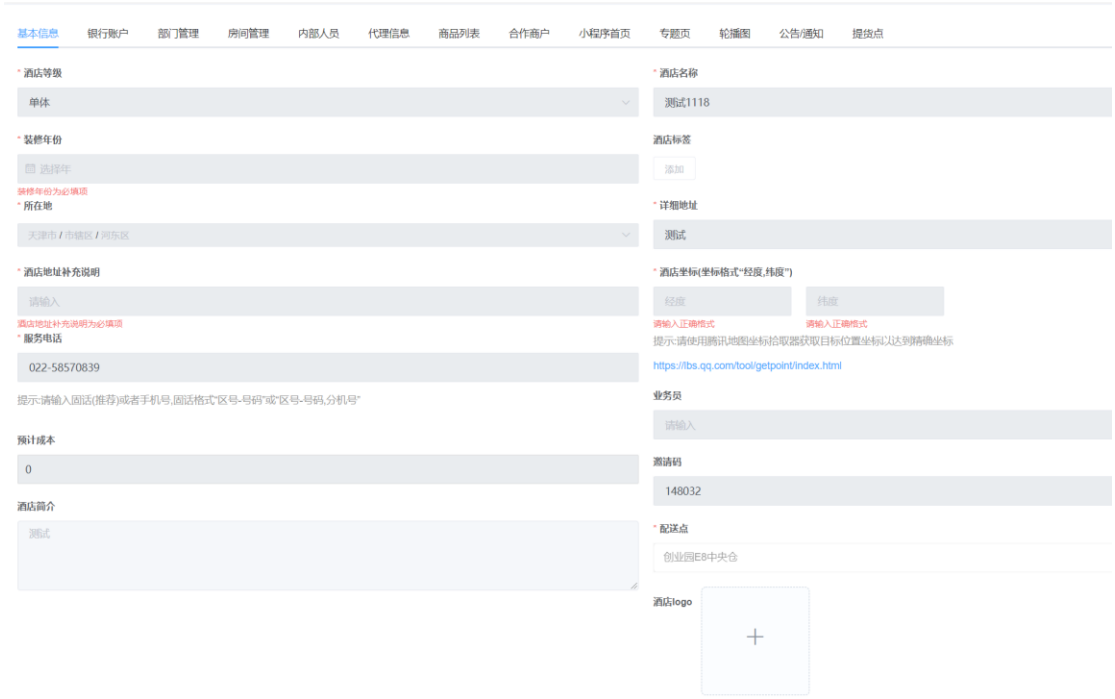

图 36：酒店基本信息图

```
@GetMapping
@ApiOperation(value = "酒店分页列表查询", httpMethod = "GET")
public Result<Page<FirmSimpleVo>> hotels(FirmQueryModel firmQueryModel) {
    validate(firmQueryModel.getCreatorIdEQ(), NOT_BLANK, "当前用户 id 不能为空");
    Page<FirmSimpleVo> page = firmService.queryFirmsPagesBy(firmQueryModel);
    return Result.ok(page);
}
public Page<FirmSimpleVo> queryFirmsPagesBy(FirmQueryModel firmQueryModel) {
    firmQueryModel.setIsDeletedEQ(Boolean.FALSE);

    if (firmQueryModel.getPageNumber() == null) {
        firmQueryModel.setPageNumber(1);
    }

    if (firmQueryModel.getPageSize() == null) {
        firmQueryModel.setPageSize(20);
    }

    Page<FirmSimpleVo> page = new Page<>();
    page.setPageNumber(firmQueryModel.getPageNumber());
    page.setPageSize(firmQueryModel.getPageSize());

    //权限划分明细
    if ("10000".equals(firmQueryModel.getCreatorIdEQ())) {
```

```
        firmQueryModel.setCreatorIdEQ(null);
    } else {
        //调用 spkitty 项目，获取登录用户对应的组织信息
        Result<OrgDetailVo> orgDetailVoResult =
spkittyApi.orgDetailByUserId(firmQueryModel.getCreatorIdEQ());
        OrgDetailVo data = orgDetailVoResult.getData();
        if (data != null) {
            if (data.getOrgType().equals(OrgDetailVo.OrgType.配送点)) {
                firmQueryModel.setDeliverFirmIdEQ(data.getId());
            }

            if (data.getOrgType().equals(OrgDetailVo.OrgType.子公司)) {
                firmQueryModel.setSubsidiaryIdEQ(data.getId());
            }

            firmQueryModel.setCreatorIdEQ(null);
        } else {
            page.setTotal(0L);
            List<FirmSimpleVo> firmSimpleVoList = new ArrayList<>();
            page.setList(firmSimpleVoList);
            return page;
        }
    }

    final long count = firmMapper.count(firmQueryModel);
    firmQueryModel.setOrderBy("createdTime");
    page.setTotal(count);
    if (count > 0) {
        List<Firm> collect = firmMapper.findAll(firmQueryModel);
        List<FirmSimpleVo> firmSimpleVoList = new ArrayList<>();
        for (Firm firm : collect) {
            FirmSimpleVo vo = new FirmSimpleVo();
            vo.setId(firm.getId());
            vo.setAddress(firm.getAddress());
            vo.setContactName(firm.getContactName());
            vo.setRegion(firm.getRegion());
            vo.setCreator(firm.getCreator());
            vo.setCreatorId(firm.getCreatorId());
            vo.setName(firm.getName());
            vo.setType(firm.getType());
            vo.setContactPhone(firm.getContactPhone());
            vo.setCreatedTime(firm.getCreatedTime());
            firmSimpleVoList.add(vo);
        }
```

```
        page.setList(firmSimpleVoList);
    }
    return page;
}
```

5.4、 部门管理

每个酒店都会有很多个部门，每个部门负责的事不同，每个部门下面的房间也不同，商品也不同：

图 37：酒店部门图

```
@ApiOperation(value = "获取企业部门分页列表", notes = "",
httpMethod = "GET")
public Result<Page<FirmDeptSimpleVo>> page(FirmDeptQueryModel queryModel) {
    Page<FirmDeptSimpleVo> page =
firmDeptService.getFirmDeptListPage(queryModel);
    return Result.ok(page);
}
public Page<FirmDeptSimpleVo> getFirmDeptListPage(FirmDeptQueryModel queryModel)
{

    queryModel.setDeletedEQ("1");

    if (queryModel.getPageNumber() == null) {
        queryModel.setPageNumber(1);
    }

    if (queryModel.getPageSize() == null) {
        queryModel.setPageSize(20);
```

```
    }

    final long count = firmDeptMapper.count(queryModel);
    queryModel.setOrderBy("createdTime");
    queryModel.setDirection(Direction.ASC);
    Page<FirmDeptSimpleVo> page = new Page<>();
    page.setTotal(count);
    page.setPageNumber(queryModel.getPageNumber());
    page.setPageSize(queryModel.getPageSize());

    if (count > 0) {
        List<FirmDept> deptList = firmDeptMapper.findAll(queryModel);
        List<FirmDeptSimpleVo> voList = new ArrayList<>(deptList.size());
        for (FirmDept firmDept : deptList) {
            FirmDeptSimpleVo firmDeptSimpleVo = new FirmDeptSimpleVo();
            firmDeptSimpleVo.setId(firmDept.getId());
            firmDeptSimpleVo.setName(firmDept.getName());
            firmDeptSimpleVo.setFirmId(firmDept.getFirmId());
            firmDeptSimpleVo.setShelfId(firmDept.getShelfId());
            firmDeptSimpleVo.setCreatedTime(firmDept.getCreatedTime());
            firmDeptSimpleVo.setCreator(firmDept.getCreator());
            voList.add(firmDeptSimpleVo);
        }
        page.setList(voList);
    }

    return page;
}
```

5.5、 房间管理

房价主要是那个部门的那个房间号，每个房间都有自己的二维码，也有自己房间的 wifi 相关系信息，通过扫码打开小程序，如果该房间有 wifi，小程序会弹框提示用户是否连接该 wifi：

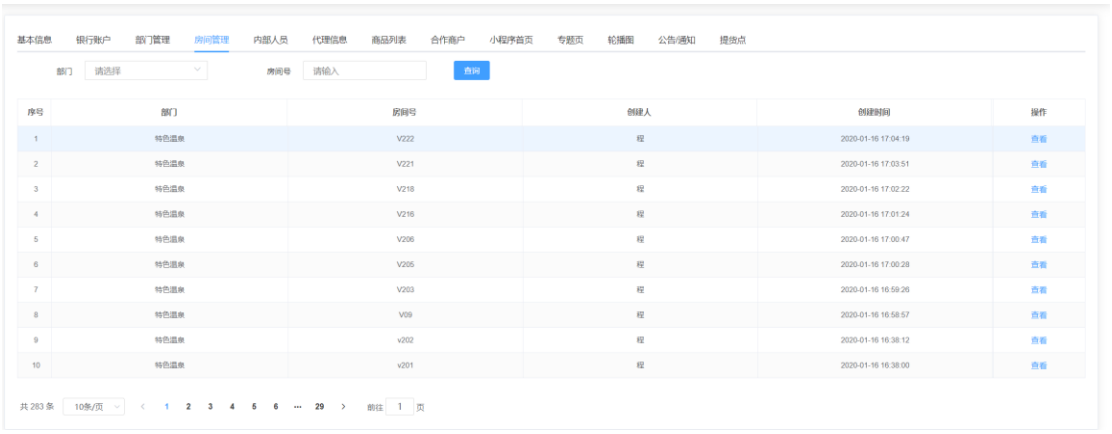

图 38：酒店部门房间分页图

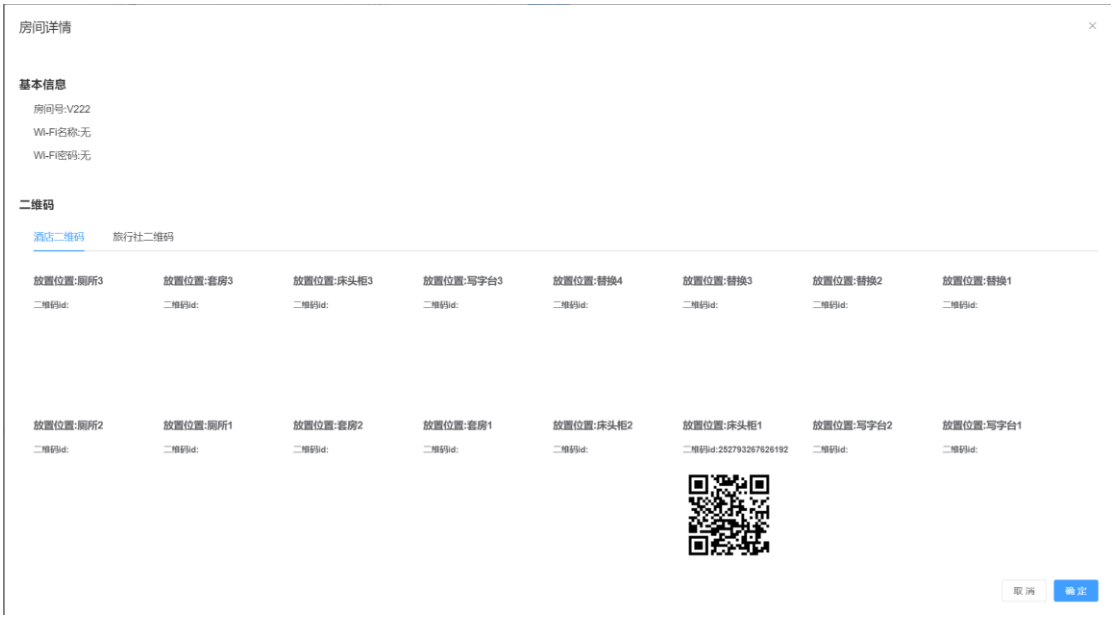

图 39：酒店部门房间详情图

```
@GetMapping("/page")
@ApiOperation(value = "企业酒店房间分页(前端使用)", httpMethod = "GET")
public Result<Page<HotelRoomVo>> page(HotelRoomQueryModel queryModel) {
    validate(queryModel.getFirmIdEQ(), NOT_BLANK, "参数异常");
    Page<HotelRoomVo> page = hotelRoomService.getHotelRoomPage(queryModel);
    return Result.ok(page);
}
public Page<HotelRoomVo> getHotelRoomPage(HotelRoomQueryModel queryModel) {
    queryModel.setIsDeleteEQ(Boolean.FALSE);
    if (queryModel.getPageNumber() == null) {
        queryModel.setPageNumber(1);
    }
    if (queryModel.getPageSize() == null) {
        queryModel.setPageSize(20);
```

```
    }
    long count = hotelRoomMapper.count(queryModel);
    Page<HotelRoomVo> page = new Page<>();
    page.setPageNumber(queryModel.getPageNumber());
    page.setPageSize(queryModel.getPageSize());
    queryModel.setOrderBy("registerTime");
    page.setTotal(count);
    if (count > 0) {
        List<HotelRoom> rooms = hotelRoomMapper.findAll(queryModel);
        List<HotelRoomVo> hotelRoomVos = new ArrayList<>(rooms.size());
        for (HotelRoom room : rooms) {
            HotelRoomVo roomVo = new HotelRoomVo();
            FirmDept firmDept =
firmDeptService.getFirmDeptById(room.getDeptId());
            roomVo.setId(room.getId());
            roomVo.setCreateRoomUser(room.getCreateRoomUser());
            roomVo.setRegisterTime(room.getRegisterTime());
            roomVo.setRoomNo(room.getRoomNo());
            if (firmDept != null) {
                roomVo.setDeptName(firmDept.getName());
            }
            hotelRoomVos.add(roomVo);
        }
        page.setList(hotelRoomVos);
    }
    return page;
}
```

5.6、 订单列表

订单列表主要是每个酒店的房间产生订单数据，用户通过扫描房价二维码下该酒店的订单。订单一旦生成这里就会产生数据，并且实时有订单的状态：

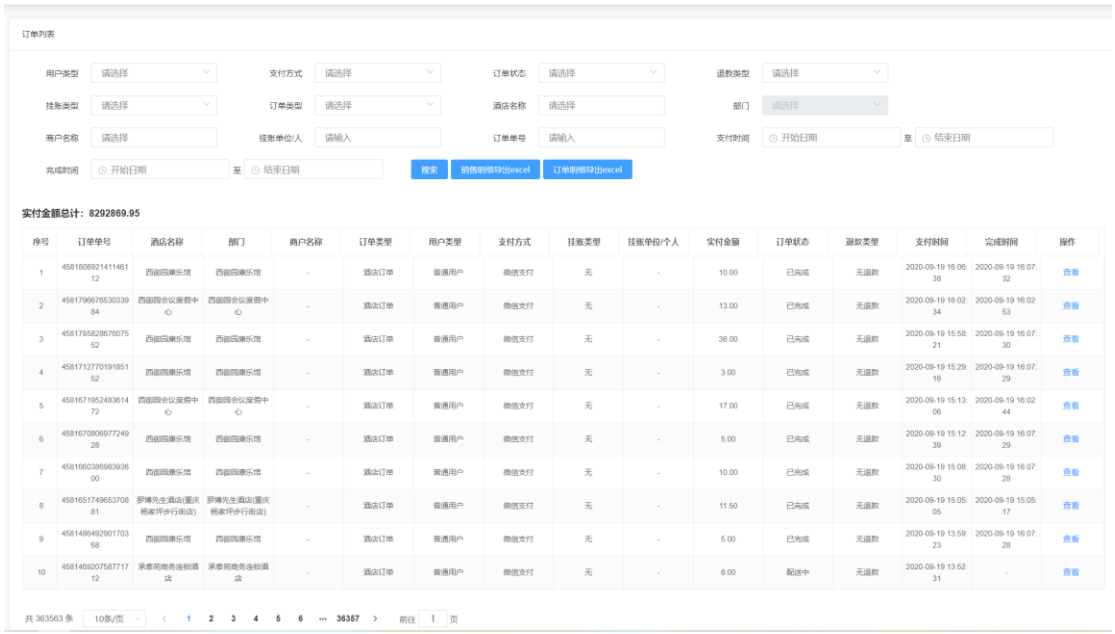

图 40：订单列表分页图

```
@GetMapping("list")
@ApiOperation(value = "订单列表查询", notes = "", httpMethod = "GET")
public Result<Page<BuyOrderListVo>> firms(@CurrentUser Principal user,
BuyOrderQueryModel buyOrderQueryModel) {
    if (CollectionUtils.isEmpty(buyOrderQueryModel.getOrderByList())) {
        List<OrderBy> orderByList = new ArrayList<>();
        orderByList.add(new OrderBy().desc("registerTime"));
        buyOrderQueryModel.setOrderByList(orderByList);
    }
    if (!"0".equals(user.getUserType()) &&
StringUtils.isNotBlank(user.getFirmId())) {

        List<Firm> firms = orgComponent.findFirmByUserId(user.getUserId());
        if (CollectionUtils.isNotEmpty(firms)) {
            if (StringUtils.isEmpty(buyOrderQueryModel.getFirmIdEQ())) {
                buyOrderQueryModel.setFirmIdIN(firms.stream().map(o ->
o.getId()).toArray(String[]::new));
            }
        } else {
            return
Result.ok(Page.<BuyOrderListVo>builder().total(0L).pageNumber(1).pageSize(20).
list(Collections.emptyList()).build());
        }
    }
    buyOrderQueryModel.setMainOrderFlagEQ(true);
    List<BuyOrderListVo> buyOrderList =
```

```
buyOrderMapper.findAll(buyOrderQueryModel).stream()
                .peek(BuyOrderComponent.PRICE_ZERO)
                .map(buyOrderConvert::orderToListVo)
                .map(orderListVo -> {
                    return addShelfName.apply(orderListVo);
                })
                .collect(Collectors.toList());

        long count = buyOrderMapper.count(buyOrderQueryModel);
        Page<BuyOrderListVo> page = new Page<>();
        page.setList(buyOrderList);
        page.setTotal(count);
        return Result.ok(page);
    }
```

6、 系统测试

6.1、 测试概述

测试软件的主要是为了保证系统的健壮和安全以免部分不良人员利用系统的漏洞对系统进行攻击以及数据爬取和窃取。因此测是软件开发中是很重要的一个环节，它的存在就是为了找出错误和问题并使系统更完整更完善。

6.2、 测试目的

软件测试的目的就是为了发现错误并纠正错误以确保程序的健壮性，测试是一个程序的执行过程，就是为了发现错误。有时候一个小小的漏洞也会让整个系统崩溃宕机，所以测试的目的就是避免这些情况。

6.3、 测试原则

测试前要根据产品的圆形定的规则为基础，在此基础上，对结果进行单元测试，测试人员避免不了与开发人之间沟通，以圆形为前提大家对此互相测试以至于达到理想的结果。测试的的计划和用例要保留好，方便以后对质和结果的保存。

6.4、 测试方法分类

由于测试方法和分类很多根据技术分类最常见的分白盒测试与黑盒测试。还有环境功能配置兼容等等，而且现如今出了很多很新的方法分类，因此需要我们实时学习。

6.4.1、白盒测试

白盒测试更多的是对程序的源代码去测试而不是根据操作页面。因此白盒测试也不是全面的测试，它也有它的局限性。

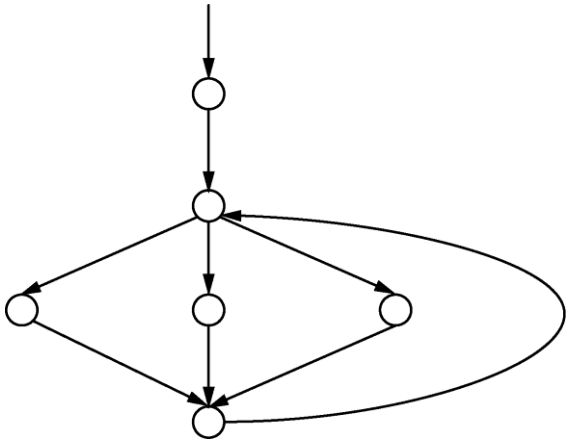

图 41：白盒测试图

6.4.2、黑盒测试

黑盒测试说白了就是测试人员在开发人员部署到测试环境后对系统进行点点操作，并且对着圆形进行测试从而对逻辑进行复盘，看看是否能达到预定结果。黑盒测试是不可能全方面对系统测试，这是由于它的局限性所致，因此一般都是白盒与黑盒一起来使用。

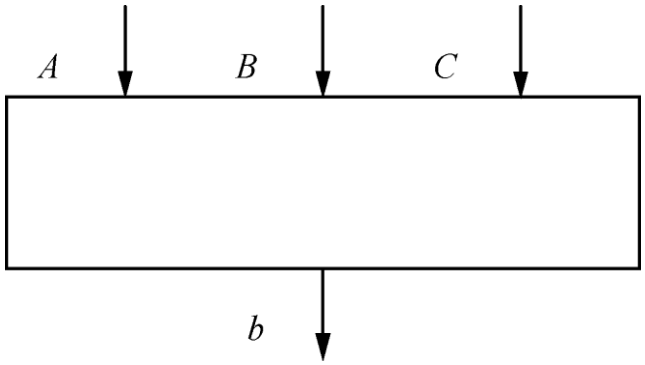

图 42：黑盒测试图

6.5、 测试用例

通过登录测试用例来看系统做出的反馈：

表 1： 系统用户登录测试用例

用例编号	测试名称	测试标题	重要级别	预置条件	测试输入	操作步骤	预期结果

1	登录	字段为空	高	进入登录界面	空	点击登录按钮	提示用户名和密码不能为空
2	登录	用户名字段为空	高	进入登录界面	密码输入123456	点击登录按钮	提示用户名不能为空
3	登录	密码为空	高	进入登录界面	用户名输入张三	点击登录按钮	提示密码不能为空
4	登录	登录成功	高	进入登录界面	正确账号密码	点击登录按钮	登录成功，进入首页

表 2： 添加系统用户测试用例

用例编号	测试名称	测试标题	重要级别	预置条件	测试输入	操作步骤	预期结果
1	用户名	用户名	高	可以添加	空	点击添加	用户名为必填项，不能为空
2	用户名	用户名	高	可以添加	admin	点击添加	用户名已存在
3	密码	密码	高	可以添加	空	点击添加	密码为必填项，不能为空。
4	密码	密码	高	可以添加	123456	点击添加	无任何提示，继续添加
5	确认密码	确认密码	高	可以添加	空和密码不一致	点击添加	提示密码不一致，无法添加
6	确认密码	确认密码	高	可以添加	123456	点击添加	添加成功

6.6、 测试总结

通过介绍测试概述、目的、原则、方法分类以及系统测试用例，我们可以得出测试对系统的重要性，它是系统的健壮稳定的一个保证。没有经过测试的系统是不能使用的，尤其涉及到数据隐私和安全问题，不测试带来的问题是很致命的，所以测试是系统的必不可少的一个环节，一定要争取保障程序的安全和健壮性。通过对测试结果的分析提高的系统的安全质量也保障了系统的稳定运行，给用户带来良好的用户体验。

结　论

从大环境出发,在电子信息如此发达的今天，几乎每个年轻人都拥有一台自己的电脑，相当部分已经使用上了智能系统的电子设备，出门也许你可以不带钱包,但你绝不会离开互联网。电脑在人们生活的地位越来越重要。借助互联网使得我们获取信息更方便了，为我们节省的时间很多，有一个很好的规划，设计一套管理系统,不但顺应大趋势,而且能随时随地为人们服务,对现在人力资源的释放起到了很重大的意思。

本系统的设计的主要目的就是帮助企业或其他人做到数据维护便捷,提升企业的工作效率，使人力的资源大大降低成本，合理的管理。同时软件设计美观，使用简单，结合了现在流行的管理系统，访问也很方便，可以实现随时随地管理。经不断努力，系统基本实现了需求分析给出的各项功能。后续的功能也会继续完善。

由于作者经验不足，系统仍然存在一些不足的地方，还需要进一步的扩展。由于时间和其他客观条件的约束，本系统实现了目前的这些基本功能，完善的工作还待以工作中继续。

通过本次系统的开发，以及其中涉及到的问题，技术的解决方案，都让我学习到了很多新的东西和新的技能。技术的变更的是快速的我们要温故知新，勇于扩展自己的知识面，多看多想，找到自己不充沛的地方加以弥补。

致　谢

毕业，总是阳光灿烂。毕业，总是曲终人散。毕业，我们拒绝伤感。花儿谢了芳芳，迎来硕果飘香。毕业带来离别，我们走向辉煌。在论文完稿之际，也是对自己的一份满意答卷。

感谢大学收获的知识和老师们的无私奉献也感谢同学之间的互相互助，大学是一个非常锻炼好磨砺人的小型社会，因为它即将面对的就是社会，感谢这大学的收货的知识友情同学师生。

在论文即将完成之际，我很欣慰，从一而终自己很认真在需求分析，代码编写，测试用例，数据库设计，技术选型，一步一个脚印，很有成就感。学习是永不停止，学习也是我们人生中的必修课，大学的学习不只有知识技能，还有为人处世，良好的交际圈和生活习惯。

在本论文截稿之际，我要特别感谢老师对我的悉心指导与鼓励，在系统的设计过程中给我提出了很多宝贵的建议，老师那严谨的治学态度也深深地感动了我，可以这么说，没有老师的无微不至的关注，我的论文也不会这么顺利地完成，为此我非常感激。此外，我还要感谢我的学校给了我一个良好的学习环境，感谢我的各位任课老师，没有他们我不会有如此大的进步，大学的经历历历在目，收货颇多，感慨也良多，学到了很多，认识很多朋友，也正因为如此我们的经历不仅丰富而且还光彩夺目。

参考文献

[1] 埃克尔，陈昊鹏译，《Java 编程思想》，机械工业出版社，2007

[2] 薛小龙，《JSP 典型系统实战与解析》，电子工业出版社，2007:90-96

[3] 管西京，《JSP+MySql 动态网站案例开发》，电子工业出版社，2008:76-101

[4] 程舒通，《学校网站动态技术的开发》，《南宁职业技术学院学报》2006 1 :13-14

[5] 马丁著，《代码整洁知道》，人民邮电出版社

[6] 张海藩 ，《软件工程导论学习辅导》[M]，清华大学出版社，2004-9-1

[7] 李刚，《疯狂 Java 讲义》（第 3 版）电子工业出版社 ，2014-7-1

[8] 李兴华，《Java Web 开发实战经典基础篇》，清华大学出版社，2010-8-1

[8] Joshua Bloch. Effective Java. 机械工业出版社,2003

[10] 耿祥义、张跃平.Java 设计模式.清华大学出版社,2009

[11] 龙中华.Spring Cloud 微服务架构实战派.电子工业出版社,2020