D{5}1A{N8WS26{M7`62$WXC copy

JIANGXI AGRICULTURAL UNIVERSITY

**本 科 毕 业 论 文（设 计）**



**题目： 基于Redis缓存的生活服务平台管理系统的设计与实现**

**学 院： 软件学院**

**姓 名： 罗紫东**

**学 号： 20152028**

**专 业： 软件工程**

**班 级： 软件1501**

**指导教师： 袁露 职 称：**

**二0一九 年 二 月**

**目录**

[摘要 1](#_Toc5616005)

[Abstract 2](#_Toc5616006)

[1 绪论 3](#_Toc5616007)

[1.1 背景与意义 3](#_Toc5616008)

[1.2 现状和发展趋势 3](#_Toc5616009)

[1.3 研究方案 5](#_Toc5616010)

[2 技术简介 6](#_Toc5616011)

[2.1 Spring简介 6](#_Toc5616012)

[2.2 Maven简介 6](#_Toc5616013)

[2.3 Redis简介 7](#_Toc5616014)

[2.4 MySQL简介 8](#_Toc5616015)

[2.5 AngularJS简介 9](#_Toc5616016)

[2.5 本章小结 10](#_Toc5616017)

[3 需求分析 10](#_Toc5616018)

[3.1 技术可行性分析 10](#_Toc5616019)

[3.2 运行可行性分析 11](#_Toc5616020)

[3.3 生活服务平台后台管理系统需求分析 11](#_Toc5616021)

[3.3.1 权限管理 11](#_Toc5616022)

[3.3.2 会员管理 12](#_Toc5616023)

[3.3.3 商品管理 12](#_Toc5616024)

[3.3.4 门店管理 12](#_Toc5616025)

[3.3.5 订单管理 13](#_Toc5616026)

[3.3.6 板块管理 13](#_Toc5616027)

[3.3.7 数据统计 13](#_Toc5616028)

[3.3.8 系统设置 13](#_Toc5616029)

[3.4 基于Redis缓存的生活服务平台管理系统总体结构 14](#_Toc5616030)

[3.5 基于Redis缓存的生活服务平台管理系统用例图 15](#_Toc5616031)

[4 概要设计 16](#_Toc5616032)

[4.1 概要设计简介 16](#_Toc5616033)

[4.2 功能设计目标 16](#_Toc5616034)

[4.3 内部结构设计 16](#_Toc5616035)

[4.4 E-R图设计 16](#_Toc5616036)

[4.4.1 菜单信息E-R图 18](#_Toc5616037)

[4.4.2 会员信息E-R图 18](#_Toc5616038)

[4.4.3 门店信息E-R图 19](#_Toc5616039)

[4.4.4 订单信息E-R图 19](#_Toc5616040)

[4.4.5 商品信息E-R图 20](#_Toc5616041)

[4.4.6 管理员信息E-R图 20](#_Toc5616042)

[4.4.7 角色信息E-R图 21](#_Toc5616043)

[4.4.8 城市信息E-R图 21](#_Toc5616044)

[4.4.9 商品类别信息E-R图 22](#_Toc5616045)

[4.5.0 日志信息E-R图 22](#_Toc5616046)

[4.5.1 APP信息E-R图 23](#_Toc5616047)

[4.5.2 权限信息E-R图 23](#_Toc5616048)

[4.5.3 活动信息E-R图 24](#_Toc5616049)

[4.5.4 浏览信息E-R图 24](#_Toc5616050)

[4.5 数据库表设计 24](#_Toc5616051)

[5 详细设计 30](#_Toc5616052)

[5.1 详细设计简介 30](#_Toc5616053)

[5.2 模块描述 30](#_Toc5616054)

[5.2.1 模块功能清单 30](#_Toc5616055)

[5.2.2 登陆校验模块 30](#_Toc5616056)

[5.2.3 权限管理模块 31](#_Toc5616057)

[5.2.4 角色关联管理模块 32](#_Toc5616058)

[5.2.5 后台用户管理模块 32](#_Toc5616059)

[5.2.6 会员管理模块 33](#_Toc5616060)

[5.2.7 门店信息管理模块 34](#_Toc5616061)

[5.2.8 订单管理模块 34](#_Toc5616062)

[5.2.9 商品管理模块 35](#_Toc5616063)

[5.2.10 特定活动管理模块 36](#_Toc5616064)

[5.2.11 数据统计模块 37](#_Toc5616065)

[5.2.12 日志模块 37](#_Toc5616066)

[5.2.13 项目管理模块 38](#_Toc5616067)

[5.3 Redis缓存设计 38](#_Toc5616068)

[5.3.1 设计原则 38](#_Toc5616069)

[5.3.2 缓存设计 38](#_Toc5616070)

[5.3.3 架构设计 39](#_Toc5616071)

[5.3.4 缓存设计的不足 40](#_Toc5616072)

[6 系统主要功能 41](#_Toc5616073)

[6.1 系统主要功能特征 41](#_Toc5616074)

[6.1.1 用户角色权限管理 41](#_Toc5616075)

[6.1.2 平台基础功能管理 41](#_Toc5616076)

[6.2 系统特色与指标 41](#_Toc5616077)

[6.2.1 前后端分离 41](#_Toc5616078)

[6.2.2 AOP日志统计 42](#_Toc5616079)

[6.2.3 服务动态负载均衡实现弹性计算 42](#_Toc5616080)

[6.2.4 服务性能统计分析支持基于数据的服务运维 43](#_Toc5616081)

[6.2.5 集群节点动态伸缩提升基础设施弹性 43](#_Toc5616082)

[7 程序测试与评价 44](#_Toc5616083)

[7.1 程序调试 44](#_Toc5616084)

[7.2 程序的测试 44](#_Toc5616085)

[7.2.1 测试的重要性及目的 44](#_Toc5616086)

[7.2.2 测试的步骤 46](#_Toc5616087)

[7.2.3 测试的主要内容 46](#_Toc5616088)

[7.2.4 测试用例 46](#_Toc5616089)

[7.3 系统的特点和优点 46](#_Toc5616090)

[7.4 系统存在的不足和改进方案 46](#_Toc5616091)

[7.5 设计收获和心得 46](#_Toc5616092)

[8 结论 47](#_Toc5616093)

[参考文献 48](#_Toc5616094)

[致谢 49](#_Toc5616095)

# 摘要

近几年来，随着移动互联网的疾速发展，本地同城生活服务平台、同城生活服务相关的应用也越来越多，通过这些可以在一个一个城市交易的APP软件，不但让用户们感受到了方便的、服务周到的生活服务平台服务。在给商家们提供一个多样性的推广平台的同时，还体验到了更接地气的商品售卖模式，可谓是口碑与经济双赢。经过十多年的发展，中国人在网购方面，有了世界话语权。在众多网络平台的的互联网环境下，很多用户都可以在平台上进行注册会员，然后成立自己的独特店铺。进入2019年这一行业也有了新的重要发展趋势。生活服务平台的影响会继续扩大，本地生活服务平台也可以和传统电子商务共存在这互联网大潮中。作为零售商和品牌商的另一个销售渠道，生活服务平台变得越来有重要。在相应的的平台下，其背后必定有运营管理团队，本系统便是给各大管理人员提供可靠的管理平台，方便各个层次的管理人员和相应的运营市场人员进行相应的部署规划和市场预测。生活服务平台管理系统加之以高效的缓存服务层，给各个用户提供高效的操作性。由于缓存层的存在，持久层数据库基本可以空载与系统中，很大层面上提高了系统的吞吐量；负载均衡提供流量划分，能够承受大流量的冲击。

关键词：互联网；生活服务；网站性能；高并发；缓存架构；

# Abstract

Recently, mobile Internet has developed rapidly and many times, and more and more applications related to local small town related life service platforms and smart living services. Through these APP software that can be traded in one city, not only users feel convenient. service and thoughtful service platform services. While providing a variety of promotion platforms for merchants, they also experienced a more grounded sales model, which is a win-win situation between word of mouth and economy. After more than a decade of development, the Chinese have the right to speak in the world in online shopping. In the Internet environment of many online platforms, many users can register members on the platform and then set up their own unique stores. Into the industry in 2019, there has also been a new important development trend. The impact of the life service platform will continue to expand, and the local life service platform can also coexist with the traditional e-commerce in this Internet tide. As another sales channel for retailers and brands, life service platforms are becoming more and more important. Under the corresponding platform, there must be an operation management team behind it. This system provides a reliable management platform for all managers to facilitate the deployment planning and market forecasting of managers at all levels and corresponding operational market personnel. . The life service platform management system is combined with an efficient cache service layer to provide efficient operation for individual users. Due to the existence of the cache layer, the persistent layer database can be basically empty and in the system, which improves the throughput of the system on a large scale; load balancing provides traffic division and can withstand the impact of large traffic.

**Keywords:** Internet; life service; website performance; high concurrency; cache architecture;

# 1 绪论

## 背景与意义

当前，信息时代已经到来，互联网发展一直处在蓬勃发展的阶段，未来依然以互联网发展为主，大量的数据信息在互联网中的传输已经由缓慢式变成了蔓延式；不能再像以前的互联网传输，以小量、简短的文本信息为主要内容，而目前的互联网传输则是海量的动态流式信息，当然也包含了很多流媒体信息。同时，随着网络的智慧发展，人们已经完全融入了互联网生活；科技不断提高， 5G时代已经降临，万物互联模式即将引爆时代。在使用网络时所需流量逐年递增，业务范围也在逐年扩大，在当前的互联网市场下，越来越频繁的购物疯抢不断出现，高扩展、微服务化、高并发、高流量、大数据量的网络新特性正在考验着当前的互联网。

为了解决高并发点击、高流量冲击，各种互联网网站都面临着大量数据的归档处理的难题，高稳定性运转等一系列的问题，大型互联网公司在业务发展以及技术实践中提出了很多解决方案，各个公司公开的理论和实践资料越来越多也越来越成熟，各种开源的组件、工具、框架越来越丰富，相对于的社区维护越来越活跃，相应的都有了更成熟的解决方案。在不同的领域，在领先的互联网公司架构的研究，无论是国外还是国内，都有它们独特和本地化的地方，为了实现高性能网站高可用性、易扩展性、可扩展性、安全性和其他技术指标体系，网站的架构模型也在不断演化与完善。

作为一名刚入门的互联网开发工程师，能亲身经历一个网站从小到大的架构演化，从功能结构单一到复杂业务的开发过程是非常难得的，而往往只有这样才能深刻体会了解到符合特定业务场景的技术架构对于网站性能、安全性、扩展性是如此的重要，更需要我们积极地深入研究与探索。

## 现状和发展趋势

电子商务，17年来取得了长足的发展，截止2018年6月，参与网络消费的网民规模已达8亿多。电子商务对实体经济市场产生了重要影响，给实体经济带来了巨大的抨击。但还是有一部分传统经济市场仍然不在网络经济领域。 O2O(Online To Offline)就是适应电子商务发展趋向而产生的一种立足于当地生活效能的电子商务形式。 O2O是将在线消费者带到真正的商店--在线支付离线服务，然后上网享受服务。团购的的意思就是一种集群式消费，同时也是集团购买市场的爆发，促进了初级生活服务电子商务的消费体验，开启了中国电子商务的本地化服务时代。运用价值网络理论模型构建O2O商业模式生态学。离线经济模式价值的要素包括消费者、商品供应方（门店）、O2O运营平台和平台管理系统以及第三方支付服务。对离线商务经济业务模型进行相应划分，追踪当前的离线商务模式应用案例。分析出了四种不同类型的应用案例。针对当前电子信息运营商面临的转型挑战，分析了电子信息运营商在离线经济业务模中的优势和机遇。线下商业经济、地方生活服务、电子商务是解决线下智慧生活的一种新的电子商务理论模型。这种业务模式和线下实体业务是一种良性的互动的业务模式，可以相互补充和相互盈利。例如，在国外，有一个本地服务化的一个商务网站，允许在线购买商品之外，还能够进行个性化定制，该公司在全国雇佣了很多个个性化设计师傅和专业的团队来安排与客户会面。到达客户地点后，他们会量尺寸，然后拿出许多面料让你选择，帮助你挑选出最适合自己的类型。然后客户就只需要在网站上输入自己的尺码，面料等信息就可以在一段时间后收到定制的服装了。未来，这个市场是多样化的，可以让很多种经济模式、经济体共存的一个市场状况，以及可以共通发展。

随着互联网的发展，移动应用的服务端开发技术也经历了几次大的变迁。很早之前呢，网页服务器就是简简单单的地响应用户从浏览器发出的请求（简称URL），然后返回静态的页面渲染信息。曾经的一段时间，由JSP主导前端市场，但是现在不一样了，各种各样的前端技术爆发，比如像Angular、Vue等框架，给前后端交互带来了很大的便利。直到现在，Java语言成为主流，成为主流的原因有很多。Java的特征之一是跨平台，这一点给很多大型网站公司所利用，他们讲服务器搭建在Linux环境下，这不仅利于维护，还能提高安全性。各种大型网站的架构都是依赖于Java强大的成熟 的体系，和Apache, MySQL, Linux共同组成一个强大的Web开发平台，被称作LAMJ。既然CGI程序擅长处理请求信息，而服务器页面擅长构造响应页面，那么将两者结合起来就是当下常见的MVC（模型-视图-控制器）模式，使用MVC模式可以很好地分离模型与视图，使二者完全解耦，互相影响降到最低。

模型层和视图层的分离，给开发和维护带来了很大的便利性，例如流行的Spring开源框架。利用Spring和SpringMVC的优势，讲服务器端的代码结构进行相应的划分统一，达到功能上的解耦。所有的类（Class）只需要由框架中的管理器来统一管理即可，然后用配置文件做上相应的指示标识，就可以达到功能的依赖，而且每一个实例都是以单例模式进行创建，这也给服务端系统带来了很有利的帮助。技术上的迭代不仅依托于时代，而且还取决于业务的发展规模。在本地服务业务的不断扩张，服务端的性能需求将不断加大，全国各地的城市信息，地点信息，商品信息都会不断的涌入服务器。如果采用传统的单服务器和单数据库的架构，肯定是行不通的。所以考虑现实情况、需求业务发展的趋势，管理系统的网站架构也会随着平台的架构演进而不断改善。将服务从从主系统中拆分，可以到解耦的效果，还可以带来服务的高可用。比如数据的缓存层横向插入原有架构，给原本数据库缓解了高连接数的压力。因此，一个成熟的大型网站技术架构往往需要经过亿万用户的实时检测，互联网技术的发展也在朝着这个方向而不断迈进。

## 研究方案

研究方式主要以生产实际（社会实践）的方式，先详细了解目前企业计算环境所面对的各类问题和挑战，重点针对于企业的目前存在的痛点，以及想要达成的效果设计出项目需求。再以数据运营支撑系统推导理论，根据数据运营支撑系统的运营过程和客户要求，设计出具有通用性，健壮性，扩展性的统一的企业应用生态环境。

系统核心目标是能够提供服务运行支撑管理平台，让服务开发者专注于业务和业务代码。其他一切都由支撑平台提供。然后相应的管理人员进行对平台上的相关信息进行管理，包括如下小项：

(1) 提供服务全生命周期管理能力：创建、发布、部署等一系列状态的切换。

(2) 提供强大服务支撑：服务部署、服务访问控制、监控、负载均衡、服务限流、日志分析、服务调用统计分析。

(3) 服务滚动升级和故障回滚。

(4) 提供相应的缓存架构，提升管理上的高效性。

计算平台具有通用性，健壮性，扩展性。有利于培养开放的统一的企业应用生态环境。

根据各个管理运维人员对系统的反馈情况来证实系统架构的可行和运行的稳定性起到至关重要的作用。

# 2 技术简介

## 2.1 Spring简介

Spring Framework一个流行的开源应用程序框架，这解决了现有系统中的许多问题。 Spring使得实现轻量级，可扩展的J2EE架构变得特别容易。它提供了我们基本的架构构建模块的开箱即用的功能推荐。 Spring提供了一种结构化的一致方式应用程序，并提供了许多中间层功能，可以使J2EE开发变得更加容易和灵活在传统方法中。

  基本动机，对于Spring来说是：解决其他框架没有很好服务的领域。有针对J2EE基础架构特定领域的众多优秀解决方案：Web框架，持久性解决方案，远程处理工具等。但是，将这些工具集成到一个综合体系结构中可能会涉及到。重大努力，可以成为一种负担。Spring旨在提供端到端的解决方案，将专业化的框架集成到一个连贯的整体基础设施。 Spring也涉及一些领域其他框架没有。例如，很少有框架可以解决通用事务管理，数据访问对象的实现，以及将所有这些内容粘合到一个应用程序中，同时在每个区域中进行最佳选择。因此，我们称之为Spring应用程序框架，而不是Web框架，IOC或AOP。

为了方便采用，框架应该干净地分层，允许使用个人功能而不强加整体关于应用程序的世界观。许多Spring功能，例如JDBC抽象层或Hibernate集成，可以在库中使用风格或作为Spring端到端解决方案的一部分。提供易用性，正如我们所指出的那样，J2EE开箱即用比较难以解决许多常见问题。良好的基础架构框架应该使简单的任务变得简单，没有强制权衡未来的复杂要求，如应用程序开发人员。这应该允许开发人员使用JTA等J2EE服务适当的，但在它们存在的情况下避免依赖它们不必要的复杂。为了更容易应用最佳实践，Spring旨在降低遵循最佳实践的成本，例如编程接口而不是类。然而，它为开发者选择构建风格。非侵入性，应用程序对象应该对框架的依赖性最小。如果利用特定的Spring功能，对象应仅依赖于该特定功能，无论是通过实现回调接口还是将框架用作类库。IOC和AOP是避免使用的关键技术框架依赖。

## 2.2 Maven简介

Maven很方便，可以通俗的讲，它就是一个模板式项目创建的一个工具，管控项目，构建项目，部署项目以及报告和文档等更多功能。它指的是项目对象模型（POM）的概念。在短期内，它是一个构建工具。

运用maven带来的利益：（1）依赖管理，对jar或war的一致性管理，以达到节省空间。提到依赖，那什么是依赖呢？依赖是指：一个java项目可能需要使用到第三方的jar包才能运行，那么我们说这个java项目依赖于第三方jar包。比如：SSM(Spring、SpringMVC、 Mybatis)，SSH(Struts2、Spring、hibernate）。（2）一键构建，如果是web项目构建的是war包，Java项目构建的jar包，具有编码->编译->测试->打包，运行打包部署。（3）跨平台。（4）应用在大型的项目中可以提高开发效率。商城项目分析：用户模块，订单模块，地址模块，支付模块；maven分模块开发，父子工程，servlet - service(业务)-dao-entity；互联网中项目：安装业务来分模块；传统项目：在JavaWeb项目中按层来分：entity，dao，service，web。

Maven构建生命周期，对于构建项目的人来说，这意味着只需要学习一小组命令来构建任何Maven项目，POM将确保他们获得所需的结果。

有三个内置的构建生命周期：默认，清理和站点。默认生命周期处理项目部署。这些构建生命周期中的每一个都由不同的构建阶段列表定义，其中构建阶段表示生命周期中的阶段。

## 2.3 Redis简介

Redis是当前比较热门的NOSQL系统之一，它是一个开源的使用ANSIC语言编写的key-value存储系统（区别于MySQL的二维表格的形式存储。）。Redis缓存存储是管理会话和应用程序缓存的最佳选择之一，因为Redis缓存存储提供数据安全性，锁定和密钥感知。它还可以用于缓存Moodle config.php 配置文件中的用户会话。

Redis的是作为一个缓存存储之前，你必须用一个单一的应用程序的Moodle节点架构使用时，您的Moodle服务器上安装Redis的服务，包括本地或外部，如果您使用的Moodle节点的集群。只有这样才能将Redis配置为应用程序或会话缓存存储。

如果运行Moodle服务器/ Moodle节点集群，则应在外部服务器上安装Redis服务，并且所有Moodle节点（服务器/实例）都应指向此外部Redis。如果用户使用任何Moodle节点并且连接未绑定到特定的Moodle节点，则所有用户数据都可用。外部Redis服务可以与MySQL / MariaDB服务一起安装在主SQL服务器上，您所要做的就是确保两个服务都有足够的内存。

Redis 优势

* 性能极高。
* 极富设计感的数据结构，每一种数据类型底层的数据结构的设计都非常的极致，都为了让查询效率提高和内存占用降低。
* 通过Lua脚本可是实现不同操作指令的封装，达到原子性的操作。

## 2.4 MySQL简介

可以公平地说，MySQL是最流行的开源数据库。它有一个非常庞大的安装基础和数量的用户。让我们看看MySQL之所以如此流行的原因，以及它目前的状况，也许还会触及到它的一些未来(尽管预测未来是非常罕见的)。MySQL是无处不在的，当你登录到你的热门论坛或博客时，很可能它有MySQL作为它的后端数据库。传统上，MySQL的两个特点，简单的使用和性能，使它得到了这样的普及。除此之外，在非常广泛的平台(包括windows)和内置复制(为只读客户端提供了一个简单的扩展解决方案)上的可用性，给用户带来了更多的吸引力和产品部署。MySQL的简单性有很简单的证据：在15分钟或更短的时间内，您就可以安装、拥有一个可工作的数据库，并开始运行查询和存储数据。从早期阶段起，MySQL与大多数流行的Web开发语言都有一个很好的接口。PHP和Perl，以及Java和ODBC连接器。MySQL中有两个最著名的存储引擎：MyISAM和InnoDB(我这里不讨论NDB集群；这是一个完全不同的故事)。MyISAM是默认的存储工程，历史上它是最古老的，但InnoDB兼容ACID，并提供事务、行级锁定、MVCC、自动恢复和数据损坏检测。这使得它成为您希望为应用程序选择的存储引擎。另外，还有第三方事务存储引擎pbxt，它的特性类似于InnoDB，它包含在MariaDB发行版中。

MySQL数据库因其高性能，高可靠性和易用性而成为世界上最受欢迎的开源数据库。 它也是构建在LAMP堆栈（Linux，Apache，MySQL，PHP / Perl / Python）上的新一代应用程序的首选数据库。

MySQL运行在20多个平台上，包括Linux，Windows，Mac OS，Solaris，IBMAIX，为您提供了一种让您掌控的灵活性。 无论您是数据库技术新手还是经验丰富的开发人员或DBA，MySQL都能提供全面的数据库工具、支持、咨询服务，帮助您取得更好的使用体验。

MySQL发展至今，已经组合了很多种数据存储引擎，例如：InnoDB、MyISAM。目前使用的主流引擎是InnoDB，因为其支持事务并且含有MVCC的实现，以至于有很多种事务隔离级别的使用，读未提交、不可重复读、可重复读、串行化。

## 2.5 AngularJS简介

AngularJS是一个开发动态Web应用的框架。AngularJS不是一个库而是一个JavaScript框架，它将HTML扩展为更具表现力和可读性的格式。它允许您使用与JavaScript同步的特殊标记来装饰HTML，让您编写应用程序逻辑而不是手动更新视图。无论您是要扩充现有的JavaScript应用程序还是利用框架的全部功能来创建丰富的交互式SPA，Angular都可以帮助您编写更清晰，更高效的代码。

这个看起来很明显，但重要的是要记住许多（并非所有）框架都是由开源社区的业余爱好者制作的。虽然激情和驱动力已经形成了框架，如Cappucino和Knockout，但Angular是由专业（并且非常有才华的）Google工程师构建和维护的。这意味着您不仅需要一个大型的开放社区来学习，而且您还拥有熟练的，高度可用的工程师，他们的任务是帮助您解决Angular问题。

这不是谷歌首次尝试JavaScript框架;他们首先开发了全面的Web Toolkit，它将Java编译成JavaScript，并被Google Wave团队广泛使用。随着HTML5，CSS3和JavaScript的兴起，作为前端语言和后端语言，谷歌意识到网络并不是纯粹用Java编写的。

通常，我们通过以下手段来解决动态应用和静态文档之间不匹配的问题：

* 类库 - 一些在开发WEB应用时非常有用的函数的集合。你的代码起主导作用，并且决定何时调用类库的方法。例如：jQuery等。
* 框架 - 一种WEB应用的特殊实现，你的代码只需要填充一些具体信息。框架起主导作用，并且决定何时调用你的代码。例如：knockout, ember等。

Angular另辟蹊径，它尝试去扩展HTML的结构来弥合以文档为中心的HTML与实际Web应用所需要的HTML之间的鸿沟。Angular通过指令（directive）扩展HTML的语法。例如：

* 通过{{}}进行数据绑定。
* 支持表单和表单验证。
* 将逻辑代码关联到DOM元素上。

## 2.5 本章小结

本次研究主要为生活服务平台的后台管理系统，和传统的后台管理系统类似，都是以数据统计为主，提供灵活的可操作性。在架构技术的选择上，后台架构已经有很多完善稳定的体系，为提高可视化和操作的效率，选取了Spring主流MVC框架和主流的Redis高速缓存服务器与经典的MySQL数据库作为重点研究对象。采取前后端分离技术搭配AngularJS。Linux下搭建服务容器Tomcat，并部署项目，线上环境测试、日志监测等。

# 3 需求分析

互联网生活服务平台和普通的线上服务不一样，介于纯电商贸易和线下本地零售业之间的一种互联网经济模式，适用于本地化的超市、水果商店、小卖铺等等小型贸易体。打通快速的同城本地交易，生活服务平台移动端应用即可满足这种市场，平台以会员管理系统、商家管理系统、渠道管理系统为基础，以手机客户端为重点、门户网站为辅助。生活服务平台的后台管理系统也就应运而生。

和现存的本地外卖服务类似，但是所涵盖的不仅仅是食物外卖的单一商品，包含超市大量商品种类。后台管理系统将移动应用所产生的信息统计到数据库中，并提供灵活的需求视图，为满足大量的市场运营人员分析数据，装配高速缓存技术，准确无误的提供数据保障。

## 3.1 技术可行性分析

可行性分析所要描述的任务,并不是所研究问题都需要有简单明显的解决办法，事实上，许多问题不可能在预定的系统规模之内解决。如果出现问题，解决方案不是一个可行的，然后在项目的开发过程中将会带来很多方面的不必要的让费；从时间上来说，开发周期不确定；从人力上来说，开发人员安排不够妥当等等。可行性研究的目的是在尽可能短的时间内确定问题是否可以用最低的成本解决。所以，可行性研究的目的不是去解决问题，而是去判断和确定问题是不是可以解决。

作为一个服务于交易平台的后台管理系统，从技术可行性分析，业务的逐渐扩张，越来越多的传统型行业也在逐步向互联网转型，后台管理系统的服务压力会变得越来越大并且服务量也会逐渐增加，服务的解耦和分离以及降低持久层的压力尤为重要，所以采用高可用高并发量的技术架构的实现是可行的。

## 3.2 运行可行性分析

部署生活服务平台后台管理系统，最重要的是最关键的是硬件资源，每日的活跃用户都会产生大量的数据信息，充足的硬件设备尤为重要，虽然说对于目前架构而言，可以进行横向无线扩容，但是资源的紧凑实用也是有必要的，相应的硬件资源如下：

运行环境：

1. 操作系统：Ubuntu 16.4
2. 数据库：MySQL5.7
3. 开发工具：IDEA
4. 缓存服务器：Redis
5. 运行服务器：Tomcat

硬件环境：

1. 处理器：Intel i5
2. 内存：16G
3. 硬盘容量：1T
4. 其它：服务器主机

## 3.3 生活服务平台后台管理系统需求分析

### 3.3.1 权限管理

1. 功能需求：菜单管理

描述：对后台界面以特定用户显示的菜单管理

1. 功能需求：角色管理

描述：对后台用户的角色进行管理，角色所授予的菜单进行管理

1. 功能需求：查看后台用户

描述：查看后台用户（用户的激活状态、用户的创建时间更新时间等），并可以对用户进行修改，以及对用户关联角色、删除、禁用和激活等操作

### 3.3.2 会员管理

1. 功能需求：会员管理

描述：对生活服务平台的会员用户进行管理，主要是查看会员的状态信息，可以对会员账号进行启用和禁用，会员登录主要是在移动端进行登录使用。

### 3.3.3 商品管理

1. 功能需求：商品分类管理

描述：对平台上上架的商品分类进行管理，平台商品分有一级、二级、三级，可以对所有类别进行查找，并可以对其进行修改，然后也能够进行重命名操作，也可以删除。

1. 功能需求：商品明细管理

描述：对平台上的所有商品统一管理，显示商品分页列表（名称、主页图片、创建时间、简介详情），对商品进行明细变更操作（变更商品所属类的信息、商品的各项规格信息、简介、详情、主图、配图等）。

### 3.3.4 门店管理

1. 功能需求：门店管理

描述：入驻生活平台的商家门店进行统一管理，显示门店主要信息（门店编号、门店名称、门店地址、门店状态），点击相应的门店地址标题，可以在百度地图上显示GIS地理位置，并且可以对详细信息进行编辑修改。

1. 功能需求：门店商品条目管理

描述：对每一个门店下的所有商品进行一致性管理，列表显示所有的门店，二级页面显示改门店下的所有商品，显示商品的主要详情信息和是否在售的状态（在售或者不在售），并可以对商品进行上线、下线、更改现有价格（提现原价）、变更库存数量等操作；更改价格操作页则可以显示原价和现价，每一次修改价格都会有所记录。

### 3.3.5 订单管理

1. 功能需求：订单管理

描述：生活服务平台用户在平台上进行所有交易所产生的订单的显示，跟踪订单的状态信息，并提供信息更新、错误信息更正的功能，运营人员、客服可以对客户反馈的订单号给用户提供订单详细信息反馈。显示所有订单详情，包括各种状态的订单列表。能够根据时间段查询相应时间段的订单记录。订单详情页显示订单详细信息（订单编号、门店名称、订单时间、客户名称、电话、订单商品信息）。

### 3.3.6 板块管理

1. 功能需求：板块管理

描述：对生活管理平台上架的板块进行后台管理，可以人工上线活动板块，人工下架过期活动等。例如：限时抢购活动板块，可以对这一块的商品进行查看，对可以向这板块中添加商品，删除商品。修改板块内容，删除板块。

### 3.3.7 数据统计

1. 功能需求：数据统计

描述：通过大数据方对后台数据统计后写入后台系统的数据库，然后通过后台管理平台可以进行相应的图形化展示，可以清晰客观的分析用户的交易画像。这一块主要是对数据方统计出的数据信息进行图像化显示，有各种各样的图形模型显示，对商品销售量在不同模型下有不同的展示，方便入驻商家进行商品商家调整。

### 3.3.8 系统设置

1. 功能需求：系统设置

描述：对后台管理系统进行相关的系统设置

1. 功能需求：APP版本管理

描述：显示移动端APP的更新进程，显示APP当前的版本号，并可做相关的信息调整。

1. 功能需求：操作日志

描述：对后台用户请求日志进行统计，超级管理员可以对访问情况进行查看，提高系统的安全性。

## 3.4 基于Redis缓存的生活服务平台管理系统总体结构

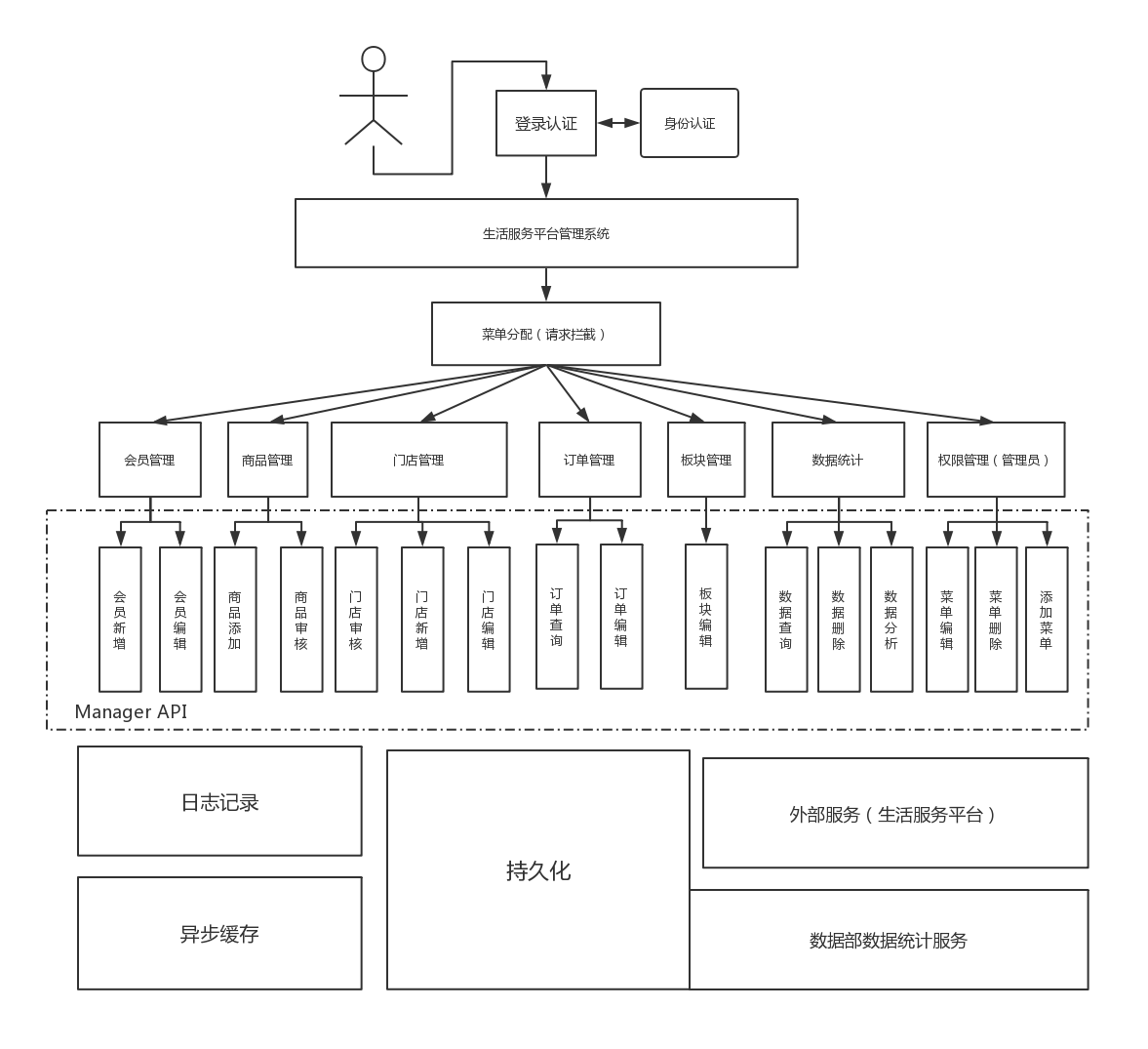
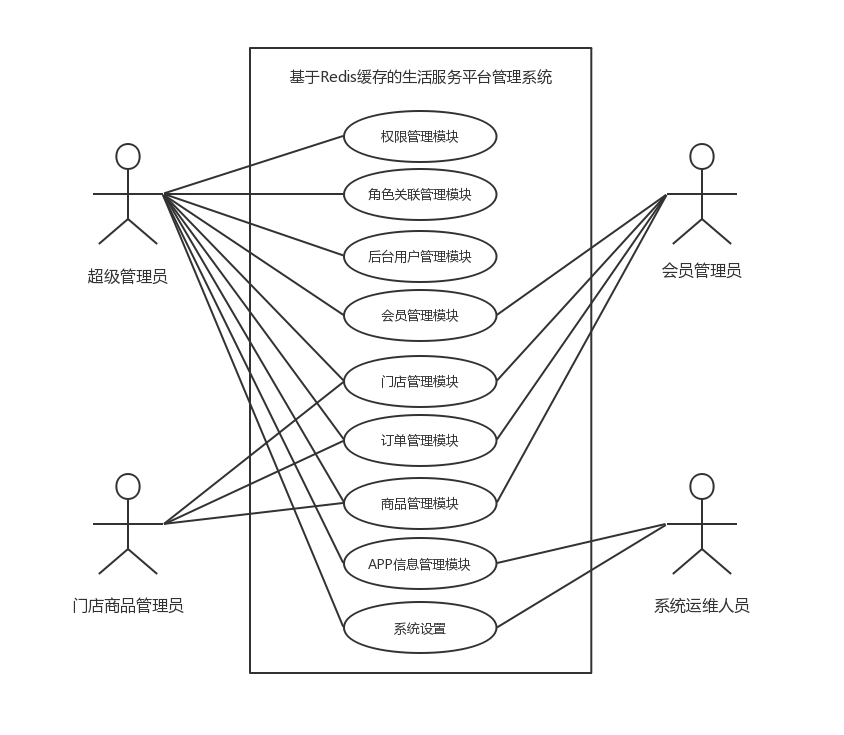


图3-1 基于Redis缓存的生活服务平台管理系统总体结构图

## 3.5 基于Redis缓存的生活服务平台管理系统用例图

用例分析是从用例模型到分析模型的过程，是介于需求分析与系统设计之间的一座桥梁。作为一个管理系统而言，主要用例来自于各种数据的吞吐，通过与后台管理系统的交互操作以及大数据部批量统计数据的写入，将各类信息入库。

基于Redis缓存的生活服务平台管理系统用例图如图3-2所示：

图3-2 基于Redis缓存的生活服务平台管理系统用例图

# 4 概要设计

## 4.1 概要设计简介

概要设计主要是描述本系统中的基本情况，比如说系统的总体设计情况，总体架构情况。概要设计是处于需求分析和详细设计这两个阶段的一个重要的阶段。概要设计的主要任务就是把前一个阶段的所设计的系统图形转换成相应的数据结构模型，以及明确数据库各个表的设计情况。

## 4.2 功能设计目标

本次课题为基于Redis缓存的生活服务平台后台管理系统统，该阶段主要表现后台管理系统的绝大部分功能、项目内部的结构设计以及E-R图等。

## 4.3 内部结构设计

生活服务平台后台管理系统内部结构如图xxx所示：

## 4.4 E-R图设计

E-R图叫做实体-联系图(Entity Relationship Diagram)，E-R图可以认为是关系模型的雏形，每个实体是一张表，实体与实体之间的关系可以合并到其中一个实体中，也可以是另外一张表，关系表和相关的实体表是通过主键或外键来联系的。其表示方法为：

1. 实体型：用矩形标识，代表和数据库中对应的某一实体
2. 属性：椭圆形表示，框内写上具体属性名称
3. 联系：菱形表示，菱形框内为联系的名称，代表实体与实体之间的联系，同时在联系的相应边旁标上联系的类型，有1:1 1:n n:m。

生活服务平台后台管理系统E-R图如图4-2所示：

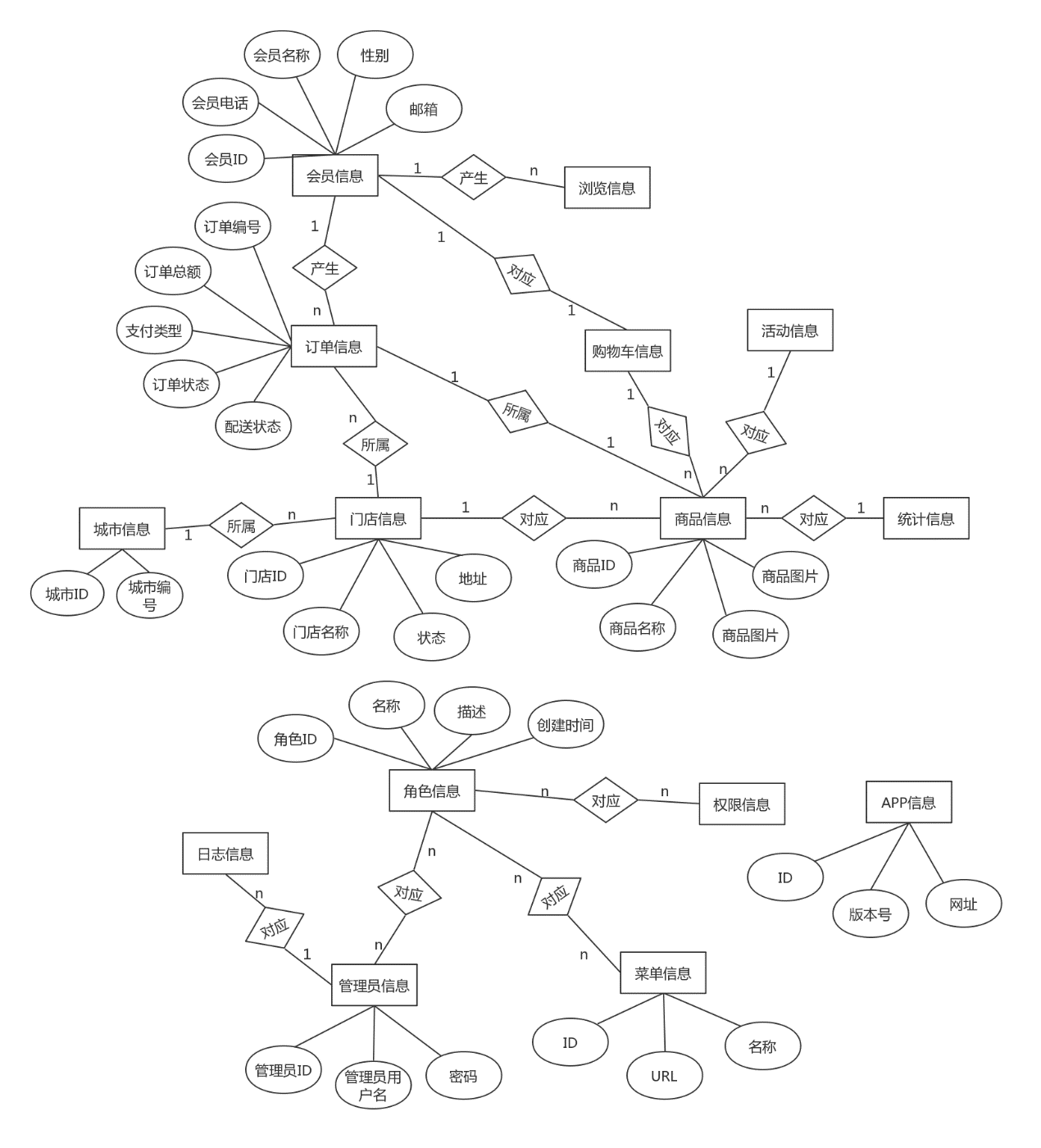
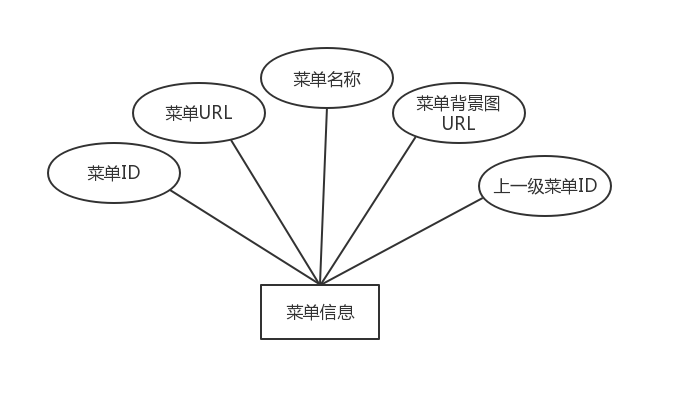
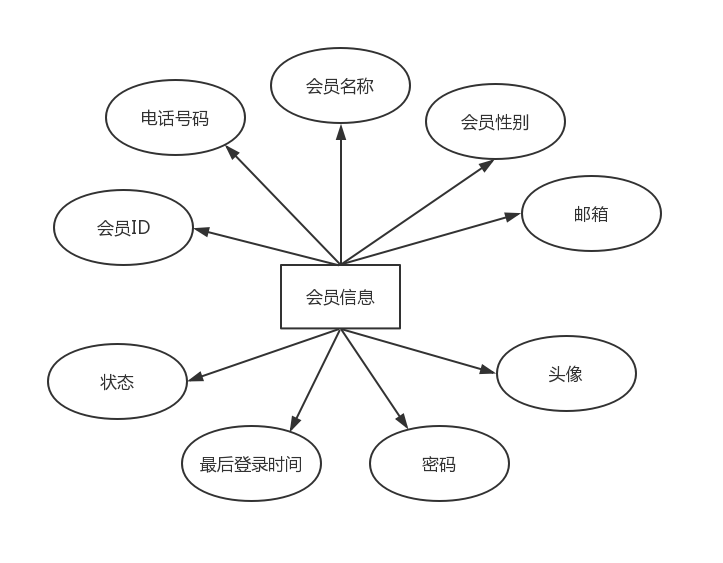


图4-2 基于Redis缓存的生活服务平台管理系统E-R图

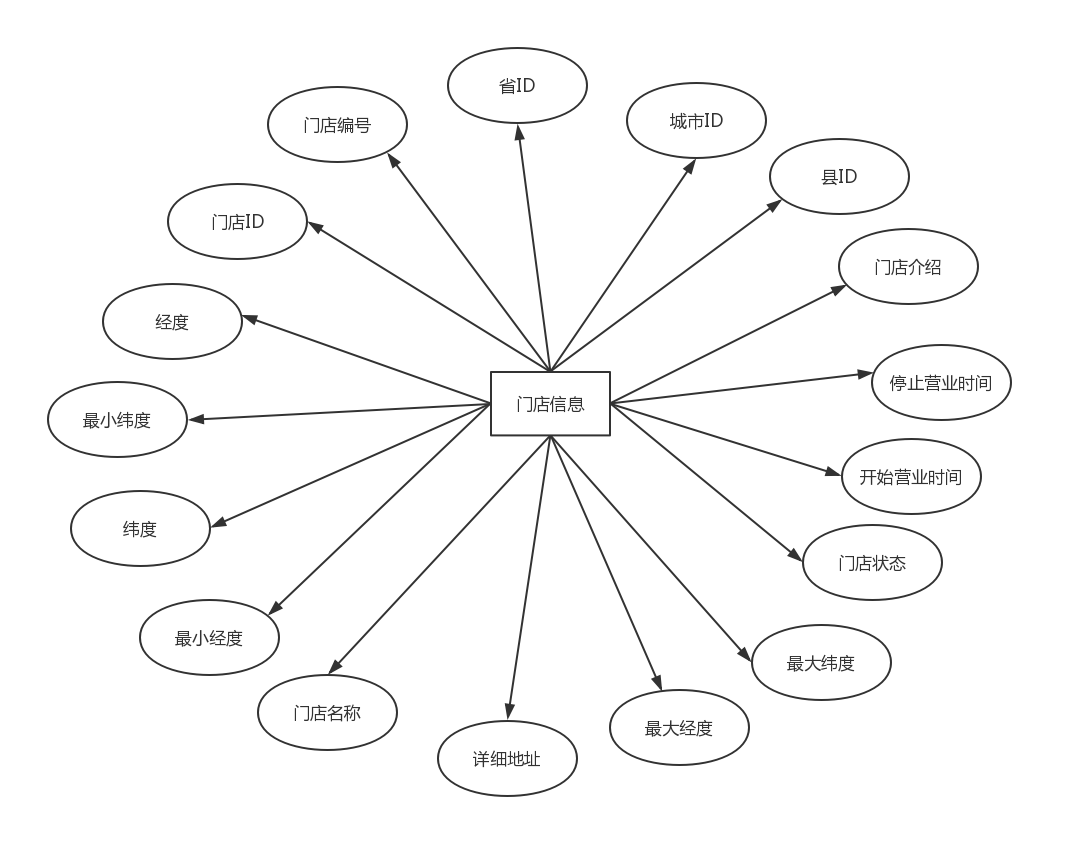
### 4.4.1 菜单信息E-R图



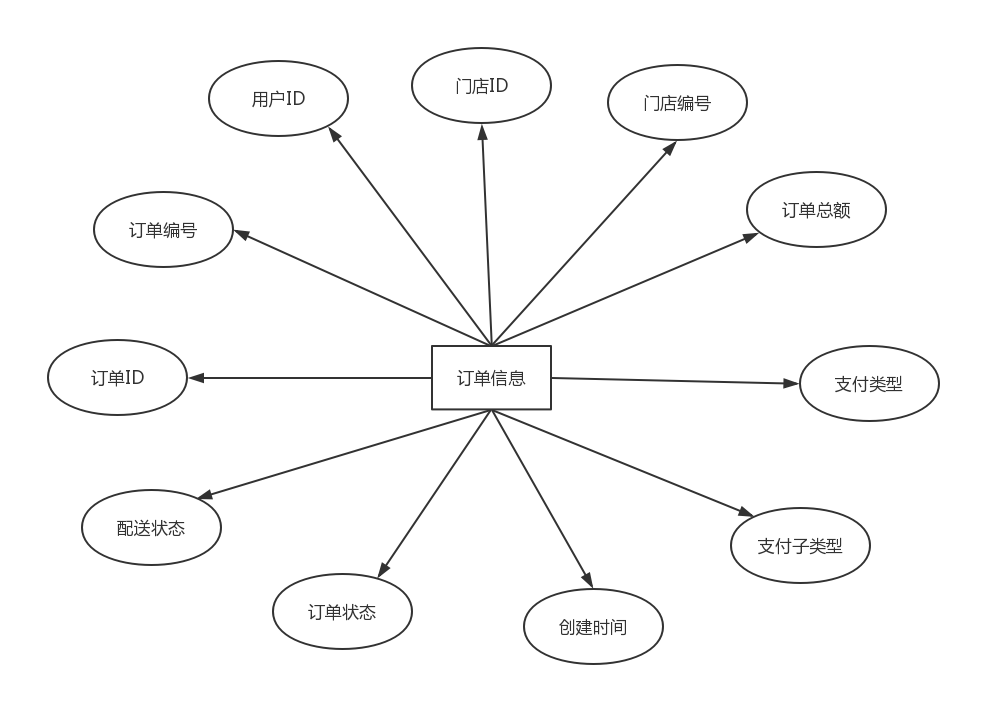
### 4.4.2 会员信息E-R图



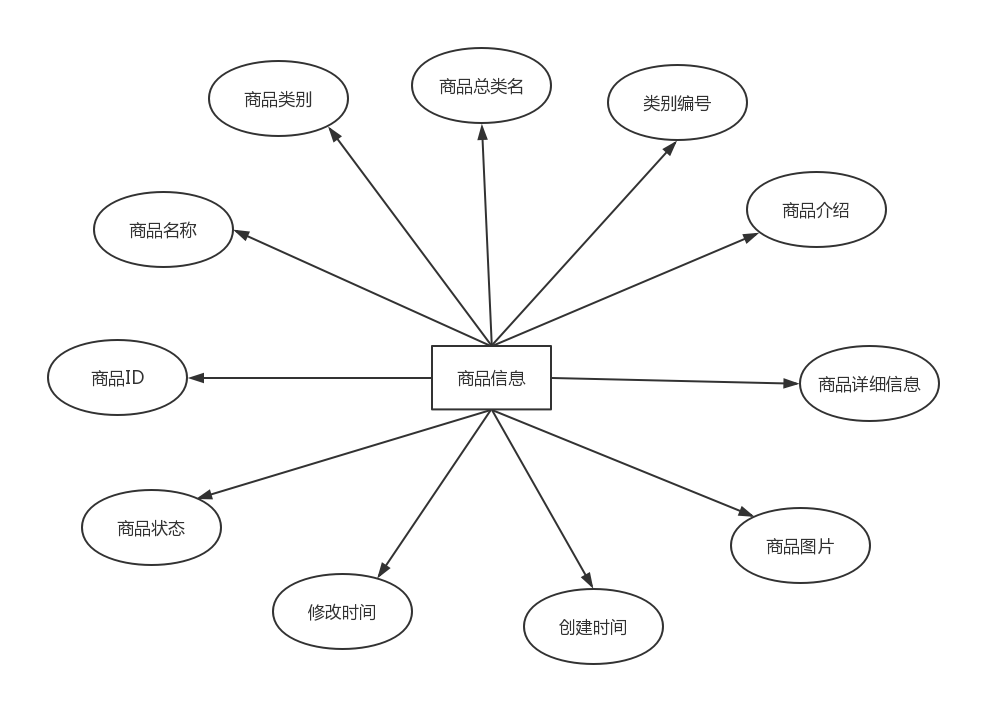
### 4.4.3 门店信息E-R图



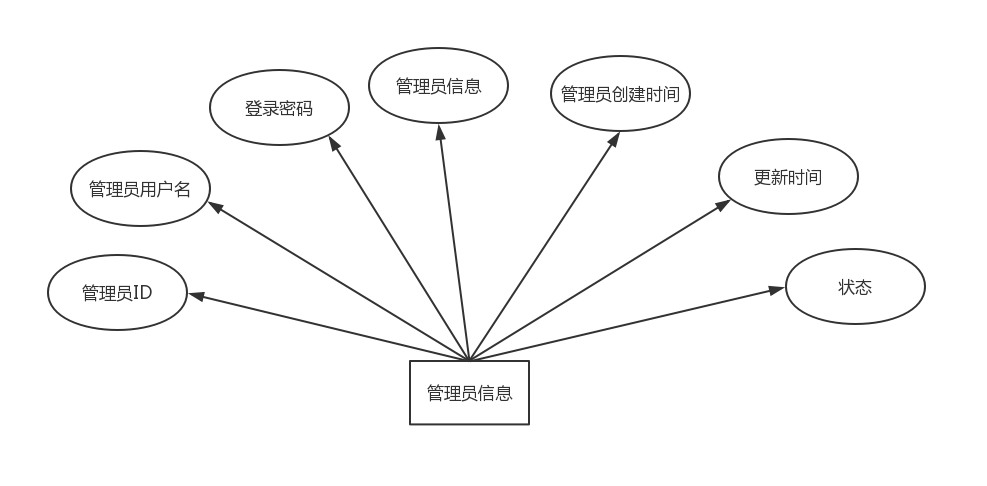
### 4.4.4 订单信息E-R图



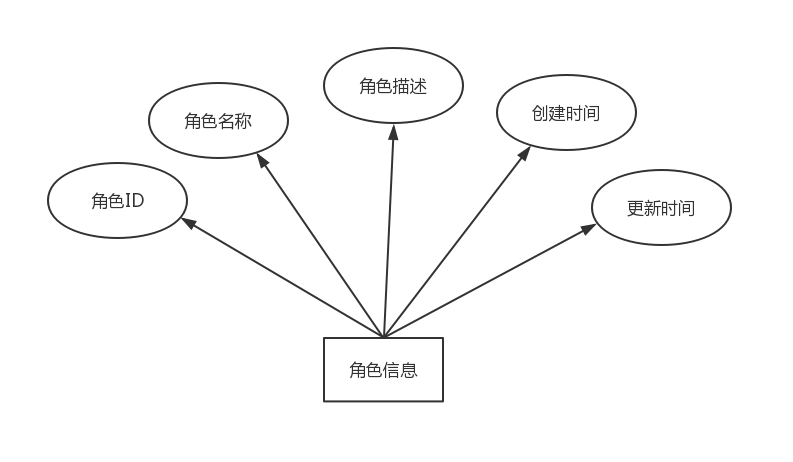
### 4.4.5 商品信息E-R图



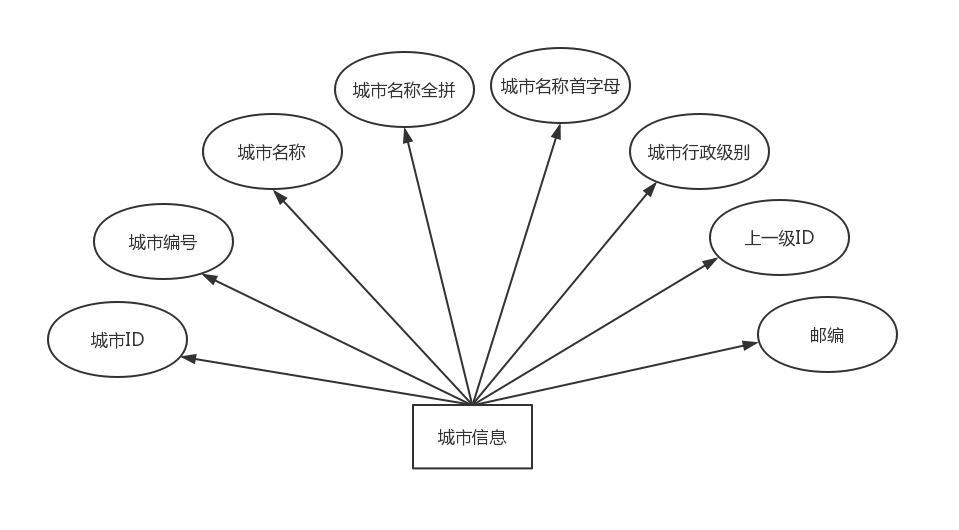
### 4.4.6 管理员信息E-R图



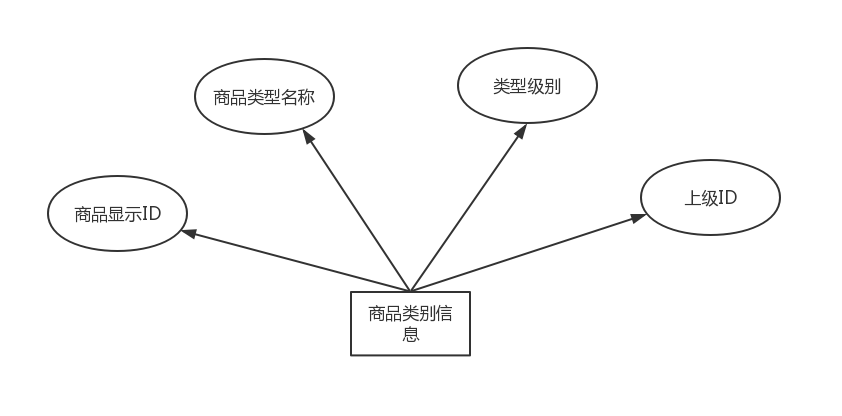
### 4.4.7 角色信息E-R图



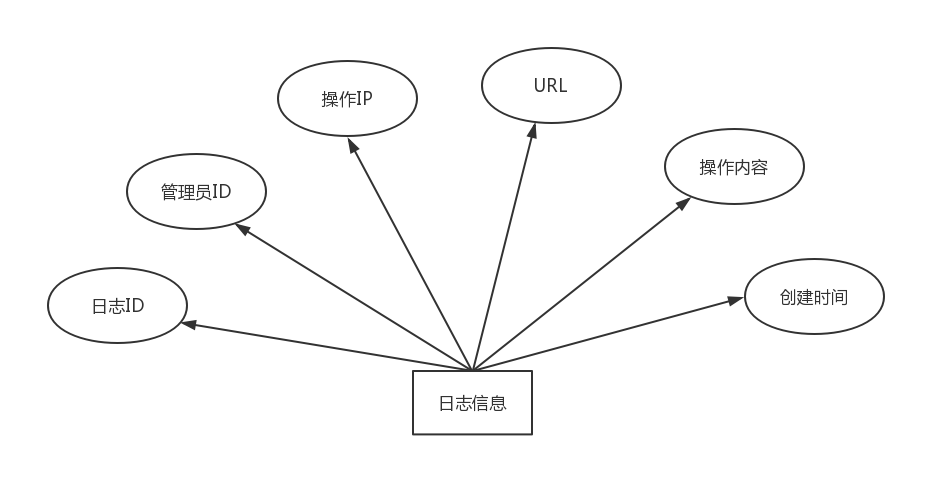
### 4.4.8 城市信息E-R图



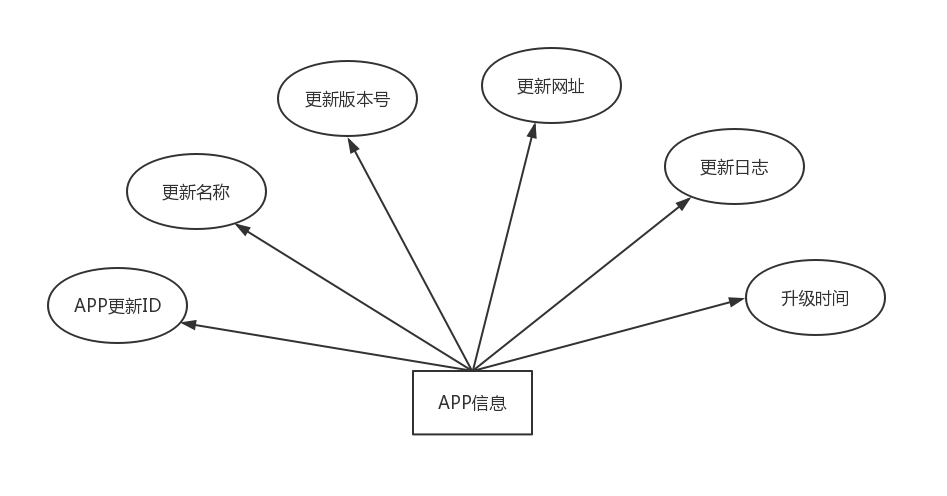
### 4.4.9 商品类别信息E-R图



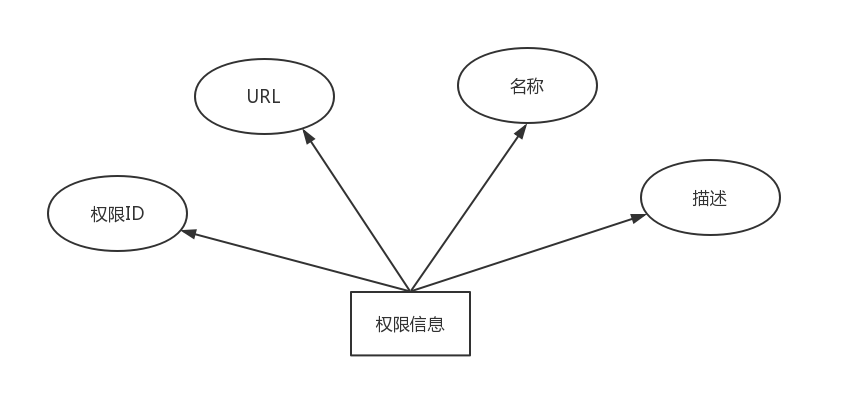
### 4.5.0 日志信息E-R图



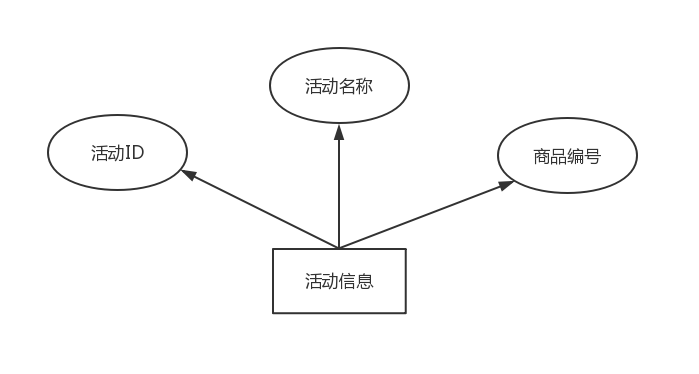
### 4.5.1 APP信息E-R图



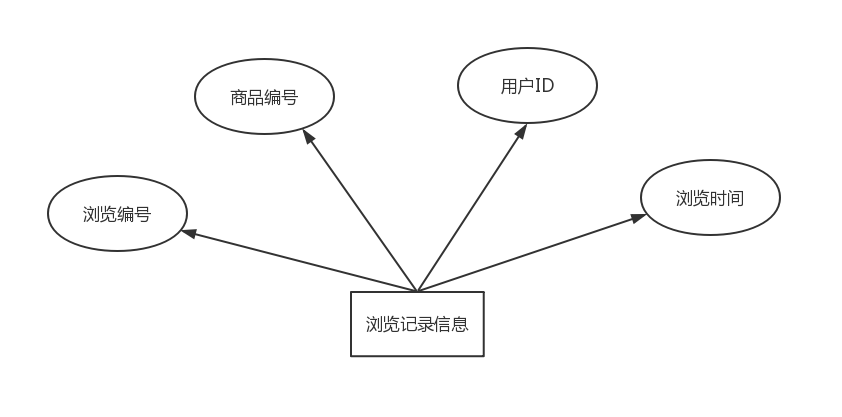
### 4.5.2 权限信息E-R图



### 4.5.3 活动信息E-R图



### 4.5.4 浏览信息E-R图



## 4.5 数据库表设计

菜单信息表，如表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| Id | int | 10 | 是 | 菜单ID |
| url | varchar | 200 | 否 | 菜单URL |
| Name | varchar | 50 | 否 | 菜单名称 |
| Backurl | varchar | 200 | 否 | 菜单背景图 |
| Parented | int | 10 | 否 | 上级菜单ID |

会员信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| Id | varchar | 50 | 是 | 用户ID |
| Phone | varchar | 20 | 否 | 手机号 |
| Username | varchar | 20 | 否 | 真实姓名 |
| Sex | char | 2 | 否 | 性别 |
| Mail | varchar | 50 | 否 | 邮箱 |
| Portrait | varchar | 200 | 否 | 头像 |
| Password | varchar | 100 | 否 | 密码 |
| LastloginTime | datetime | 0 | 否 | 最后登录时间 |
| Status | int | 2 | 否 | 状态 |

门店信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 门店ID |
| store\_no | varchar | 20 | 否 | 门店编号 |
| province\_id | int | 10 | 否 | 省ID |
| citys\_id | int | 10 | 否 | 城市ID |
| county\_id | int | 10 | 否 | 县ID |
| storename | varchar | 50 | 否 | 门店名称 |
| storeaddress | varchar | 100 | 否 | 详细地址 |
| maxlongitude | double | 20 | 否 | 最大经度 |
| maxlatitude | double | 20 | 否 | 最大纬度 |
| minlongitude | double | 20 | 否 | 最小经度 |
| minlatitude | double | 20 | 否 | 最小纬度 |
| longitude | double | 20 | 否 | 经度 |
| latitude | double | 20 | 否 | 纬度 |
| introduce | varchar | 100 | 否 | 门店介绍 |
| starttime | time | 0 | 否 | 营业时间 |
| closetime | time | 0 | 否 | 休业时间 |
| status | int | 1 | 否 | 状态 |

订单信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 11 | 是 | 订单ID |
| order\_no | varchar | 20 | 否 | 订单编号 |
| user\_id | varchar | 50 | 否 | 用户ID |
| store\_no | varchar | 20 | 否 | 门店编号 |
| totalamount | double | 0 | 否 | 订单总额 |
| paymenttype | varchar | 20 | 否 | 支付类型 |
| paymentsubtype | varchar | 20 | 否 | 支付子类型 |
| deliverytype | varchar | 20 | 否 | 配送类型 |
| states | int | 11 | 否 | 状态 |
| createtime | datetime | 0 | 否 | 创建时间 |

商品信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 商品ID |
| commodity\_no | varchar | 20 | 是 | 编号 |
| name | varchar | 50 | 否 | 名称 |
| category\_id | int | 11 | 否 | 品类ID |
| introduce | varchar | 200 | 否 | 介绍 |
| detailed | varchar | 5000 | 否 | 商品详细信息 |
| pictureurl | varchar | 200 | 否 | 商品图片 |
| createtime | datetime | 0 | 否 | 创建时间 |
| updatetime | datetime | 0 | 否 | 修改时间 |
| status | int | 1 | 否 | 状态 |

管理员信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 管理员ID |
| name | varchar | 20 | 否 | 管理员名称 |
| password | varchar | 50 | 否 | 密码 |
| info | varchar | 200 | 否 | 管理员信息 |
| createtime | datetime | 0 | 否 | 创建时间 |
| updatetime | datetime | 0 | 否 | 操作时间 |
| status | int | 1 | 否 | 状态 |

权限表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 权限ID |
| url | varchar | 200 | 否 | 网址 |
| name | varchar | 50 | 否 | 名称 |
| description | varchar | 200 | 否 | 描述 |

角色信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 角色ID |
| name | varchar | 50 | 否 | 角色名称 |
| description | varchar | 200 | 否 | 描述 |
| createtime | datetime | 0 | 否 | 创建时间 |
| updatetime | datetime | 0 | 否 | 更新时间 |

城市信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 城市ID |
| cityid | varchar | 20 | 否 | 城市编号 |
| city | varchar | 20 | 否 | 城市名称 |
| city\_py | varchar | 200 | 否 | 城市全拼 |
| city\_pym | varchar | 20 | 否 | 名称首字母 |
| clazz | int | 1 | 否 | 行政级别 |
| parent\_id | int | 11 | 否 | 上级ID |
| postcode | varchar | 10 | 否 | 邮编 |

日志信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 日志ID |
| manager\_id | int | 10 | 否 | 操作人ID |
| ip | varchar | 20 | 否 | 操作IP |
| url | varchar | 200 | 否 | 请求URL |
| content | varchar | 200 | 否 | 操作内容 |
| createtime | datetime | 0 | 否 | 创建时间 |

商品类别表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 商品显示id |
| categoryname | varchar | 50 | 否 | 商品类型名称 |
| type | int | 2 | 否 | 类型级别 |
| parented | int | 2 | 否 | 上一级id |

APP信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | APP更新ID |
| appname | varchar | 50 | 否 | 更新名称 |
| versionnumber | varchar | 20 | 否 | 更新版本号 |
| appurl | varchar | 200 | 否 | 更新网址 |
| log | varchar | 500 | 否 | 更新日志 |
| updatetime | datetime | 0 | 否 | 升级时间 |

活动信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 20 | 是 | 活动ID |
| name | varchar | 20 | 否 | 活动名称 |
| commodityno | varchar | 20 | 否 | 活动编号 |

浏览记录信息表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **长度** | **不是null** | **备注** |
| id | int | 10 | 是 | 浏览编号 |
| commodity\_no | varchar | 40 | 否 | 商品编号 |
| user\_id | varchar | 40 | 否 | 用户Id |
| browsetime | datetime | 0 | 否 | 浏览时间 |

# 5 详细设计

## 5.1 详细设计简介

详细设计是软件工程中软件开发的一个步骤，就是对概要设计的一个细化，详细设计在软件设计阶段，为详细设计的开展提供基础。在软件开发阶段，帮助开发人员明确任务和需求，明确软件开发流程。在软件测试阶段，帮助测试人员更有条理的实施测试

## 5.2 模块描述

### 5.2.1 模块功能清单

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **模块ID** | **模块名称** | **关联描述** |
|  | 100001 | 权限管理模块 | 权限菜单管理 |
|  | 100002 | 角色关联管理 | 用户角色关联信息管理 |
|  | 100003 | 后台用户管理模块 | 后台用户管理 |
|  | 100004 | 会员管理模块 | 会员信息管理 |
|  | 100005 | 门店信息管理模块 | 门店详细经纬度信息管理 |
|  | 100006 | 订单管理模块 | 订单信息管理 |
|  | 100007 | 商品管理模块 | 商品管理 |
|  | 100008 | 商品类别管理模块 | 商品类别信息管理 |
|  | 100009 | 初始化模块 | 设置初始的登录密钥和管理员权限 |
|  | 100010 | 登陆校验模块 | 用户登录 |
|  | 100011 | 密码重置模块 | 系统用户登录后可修改 |
|  | 100012 | 日志模块 | URL请求日志及操作内容 |
|  | 100013 | 项目管理模块 | 项目管理 |

### 5.2.2 登陆校验模块

1. 功能

登录校验模块提供登录时校验凭据、获取令牌、获取账户权限以身份下的菜单列表和个人信息的功能。

1. 处理流程
2. 查询登录的浏览器是否已经有登录的用户信息，如果有用户信息直接跳转至主页面
3. 调用用户校验模块校验用户登录凭证的合法性
4. 调用用户校验模块检查用户是否有登录权限
5. 调用用户校验模块获取用户对应身份下的所属菜单列表
6. 将所有信息打包并返回给前台页面
7. 接口
8. 账户管理模块获取用户凭据接口
9. 账户管理模块获取用户权限接口
10. 账户管理模块获取个人信息接口
11. 用户校验模块获取用户菜单列表
12. 登录状态校验拦截器
13. 登录校验模块界面设计，如图xxx所示：

### 5.2.3 权限管理模块

1. 功能

控制后台用户权限，细化用户管理方式，不同的用户有不同的身份信息，不同的身份对应不同的菜单管理；给不同的管理员提供不同的方式。

1. 处理流程
2. 查看所有权限列表，包括权限名称和权限描述
3. 通过权限名称模糊查询相应的权限条目
4. 通过权限描述模糊查询相应的权限条目
5. 查看并查找菜单列表，现实菜单名称、上级菜单、菜单背景图、URL等
6. 接口
7. 获取所有权限列表接口
8. 分页拦截器
9. 根据权限名称或权限描述查找指定权限接口
10. 获取菜单列表接口
11. 根据菜单名称查询指定菜单接口

### 5.2.4 角色关联管理模块

1. 功能

每一个角色会有对应的菜单列表，可以为一个角色分配多个菜单，所以这个角色就会拥有进入归属于它的菜单，进行相应的操作。

1. 处理流程
2. 获取所有的角色列表
3. 根据角色名称、角色描述查询指定角色
4. 添加角色信息，包含角色名称和角色描述
5. 修改角色信息，修改角色名称和角色描述
6. 为指定角色添加菜单
7. 接口
8. 查询所有角色列表接口
9. 根据角色名称、角色描述查询指定角色接口
10. 添加角色信息接口
11. 修改角色基本信息接口
12. 变更角色拥有的菜单接口

### 5.2.5 后台用户管理模块

1. 功能

后台用户是使用后台所有用户的信息管理，包括现实后台所有用户的状态和基本信息，此功能只可由超级管理员管理。

1. 处理流程
2. 查看所有的后台用户，现实后台管理员账号、名称、创建时间、状态
3. 根据管理员账号查询指定管理员信息
4. 修改管理员基本信息
5. 添加管理员，包括管理员的账号密码和描述信息，以及管理员的角色授予
6. 管理员关联角色，现实已经关联的角色信息，变更角色信息
7. 删除指定管理员
8. 为某个管理员重置密码，默认密码（123456）
9. 禁用管理员
10. 接口
11. 查询所有管理员信息列表接口
12. 根据指定账号查询管理员接口
13. 修改管理员信息接口
14. 添加管理员信息接口
15. 管理员删除接口
16. 管理员密码重置接口
17. 禁用管理员接口
18. 管理员角色关联接口

### 5.2.6 会员管理模块

1. 功能

后台管理系统主要是为整个生活服务平台服务，该模块管理整个平台所注册的会员信息。

1. 处理流程
2. 分页显示所有会员信息，包括会员名称、会员手机号码、最后登录时间以及会员的状态
3. 根据会员名称查找指定会员
4. 根据会员手机号码查询指定会员
5. 修改会员状态，启用或者禁用；禁用状态下的会员无法登录平台
6. 接口
7. 查询所有会员信息接口
8. 指定条件查询接口
9. 修改会员状态接口

### 5.2.7 门店信息管理模块

1. 功能

对整个平台所有入驻的门店进行管理

1. 处理流程
2. 分页显示门店的基本信息，包括门店的编号、门店名称、门店地址、门店状态
3. 根据门店名称和编号查询指定门店
4. 添加门店信息，需要添加门店的编号、名称、门店地址、门店介绍、开始营业时间、结束营业时间、经纬度信息、省份城市信息等
5. 地图上现实门店的具体位置
6. 修改门店信息
7. 修改门店状态
8. 显示门店商品列表
9. 对门店下的商品修改，修改商品状态、库存、价格等信息
10. 接口
11. 查询所有门店基本信息接口
12. 第三方地图接口
13. 添加门店信息接口
14. 修改门店接口
15. 修改门店状态接口
16. 获取门店下商品列表接口
17. 修改商品信息接口

### 5.2.8 订单管理模块

1. 功能

对平台上所产生的订单进行统一管理，进行订单信息核对以及对订单状态的跟踪

1. 处理流程
2. 分页显示所有订单列表信息，包括订单编号、下单时间、订单状态、订单总额、取货门店等基本信息
3. 根据订单编号查询指定订单信息
4. 根据时间范围查询订单信息
5. 查看指定订单，现实订单的详细信息，包括客户名称、客户电话号码、以及该订单下的所有商品
6. 根据订单状态查询订单列表
7. 接口
8. 查询所有订单信息接口
9. 查询指点订单编号的信息接口
10. 查询某个时间段的订单信息接口
11. 获取订单详细信息接口

### 5.2.9 商品管理模块

1. 功能

对平台上的所有商品进行统一管理，包括商品的种类管理和商品的详细信息管理

1. 处理流程
2. 分页显示所有商品种类的列表信息，包括类别ID、类别名称、分级、父级ID
3. 根据分类名称查询类别信息
4. 添加商品类别，选择类别等级（一级分类、二级分类、三级分类）、添加类别名称
5. 修改类别名称
6. 删除类别，以及删除该类别下的所有子类别
7. 显示商品列表，包括商品的编号、名称、主图片、创建时间、更新时间、简介信息
8. 修改商品详细信息，包括商品类别信息、商品名称、规格信息组、商品简介、商品详情信息、商品主图、商品配图等
9. 根据商品名称或者编号查找商品
10. 接口
11. 查询所有商品接口
12. 查询商品类别接口
13. 根据指点条件查询商品接口
14. 根据指定条件查询商品类别接口
15. 添加商品接口
16. 添加商品类别接口
17. 修改类别信息接口
18. 删除类别接口
19. 修改商品信息接口
20. 获取指定商品类别接口
21. 获取商品图片接口
22. 上传图片接口

### 5.2.10 特定活动管理模块

1. 功能

平台上会定期或者节假日推出相应的活动，该模块对活动进行统一管理，活动的上线和下线操作，以及该活动下的商品的加入和移除操作。

1. 处理流程
2. 分页显示平台上所有活动信息列表，包括活动的编号、活动名称等基本信息
3. 添加活动
4. 查看活动模块下的所有商品信息
5. 对某个活动进行添加商品
6. 对活动名称进行修改
7. 删除活动
8. 接口
9. 查询所有活动信息接口
10. 查看指定活动下的商品信息接口
11. 修改活动接口
12. 删除活动接口
13. 添加活动接口
14. 给指定活动添加商品接口

### 5.2.11 数据统计模块

1. 功能

该模块会对真个平台上的所有商品的销售情况进行统计显示，以不同的数据模型图显示销售状况，也会针对某个门店进行商品的销售信息统计。

1. 处理流程
2. 显示所有商品的销售情况
3. 显示购买次数最多的统计信息
4. 显示浏览次数最多的商品信息
5. 显示收藏次数的排行信息
6. 以柱形图显示统计信息
7. 以饼状图显示统计信息
8. 接口
9. 获取商品购买次数排行接口
10. 获取商品浏览次数排行接口
11. 获取商品收藏次数排行接口

### 5.2.12 日志模块

1. 功能

后台管理系统的登录、登出，以及后台管理员的操作记录都得通过日志的形式持久化操作，以免后期出现违规操作异常操作的追踪。

1. 处理流程
2. 分页现实日志信息
3. 通过每一次URL请求拦截操作，记录每一次的后台操作记录
4. 接口
5. 查询所有日志信息接口
6. AOPURL请求拦截
7. URL结果拦截

### 5.2.13 项目管理模块

1. 功能

项目管理模块提供项目创建和项目导入功能，整个项目是由maven管理工具进行搭建的，对项目管理直接通过pom文件进行管理即可。包括项目的编译环境，各个依赖项的版本信息等。

1. 处理流程
2. 选择新增方式，项目类型，填写相关信息，使用接口传到后台。
3. 根据所选择的新增方式创建新的项目。
4. 接口

无。

## 5.3 Redis缓存设计

缓存能够有效地加速应用的读写速度，同时也可以降低后端负载，对日常应用的开发至关重要。

### 5.3.1 设计原则

1. 加速读写：因为缓存通常都是全内存的，而存储层通常读写性能不够强悍（例如MySQL），通过缓存的使用可以有效地加速读写，优化用户体验。
2. 降低后端负载：帮助后端减少访问量和复杂计算（例如很复杂的SQL语句），在很大程度降低了后端的负载。
3. 加速请求响应：这是利用 Redis 或者 Memcache 优化 IO 响应时间。
4. 大量写合并为批量写：比如一些计数器先 Redis 累加以后再批量写入数据库。

### 5.3.2 缓存设计

1. 采用AOP面向切面的设计理念进行注解式缓存编码设计
2. 在Service层进行缓存切入，利用切面操作，在第一次操作时进行数据缓存处理，第二次便可以直接接入Redis查询
3. 基于Jedis封装Suishen-Redis缓存框架，支持分布式Redis缓存架构，目前采用单机Redis部署使用
4. 缓存Key以AOP拦截获取的Method名称以及参数列表构造唯一Key，Value以Kryo序列化对象存储
5. 缓存查询多以Pipeline方式执行，减少查询带来的网络IO消耗
6. 缓存服务器部署，将Redis部署于第三方服务器，单独与项目服务器，减少缓存服务器带来的性能影响
7. 配合第三方Redis监控组件，监控Redis状态，及时响应临时突发状况

### 5.3.3 架构设计

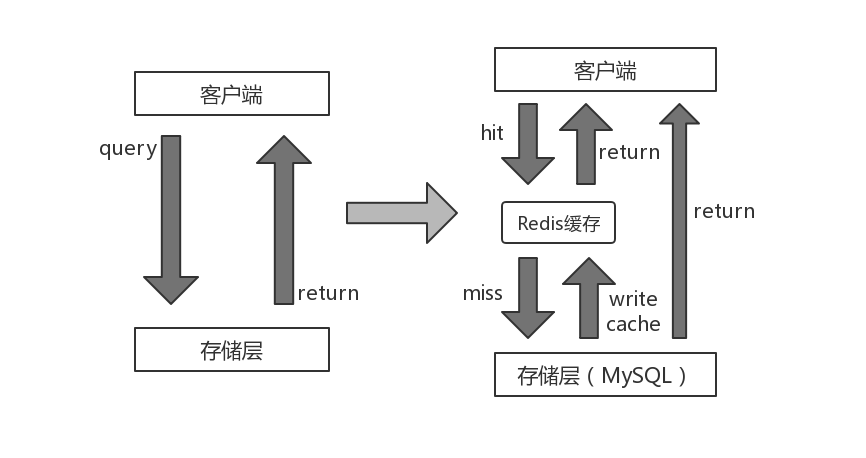
 Redis缓存架构设计如图5-1所示：

图5-1 Redis缓存架构图

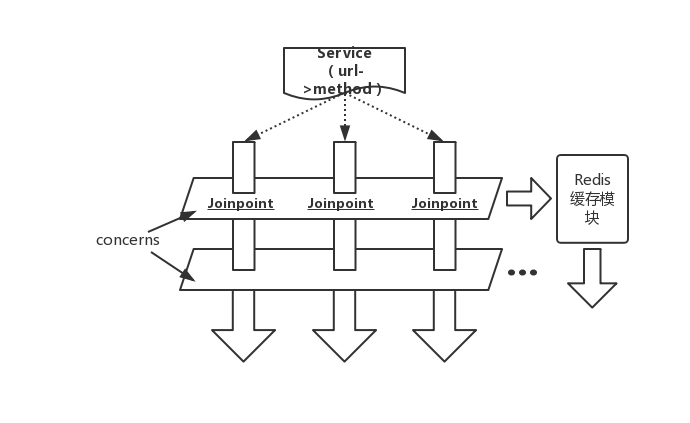
 系统中AOP模式下的Redis缓存架构设计如图5-2所示：

图5-2 AOP模式的Redis架构图

### 5.3.4 缓存设计的不足

在这样的缓存系统下，能够有效的解决热点数据查询效率的问题，但是这种Redis缓存架构设计依然有不足之处。缓存雪崩是由于原有缓存失效（过期），新缓存未到期间。所有请求都去查询数据库，而对数据库 CPU 和内存造成巨大压力，严重的会造成数据库宕机。从而形成一系列连锁反应，造成整个系统崩溃。所以，在设计缓存时需要考虑多方面因素，比如说：缓存失效、缓存节点挂掉等问题。这些都是需要很多实践场景一步一步去完善起来的。

# 6 系统主要功能

## 6.1 系统主要功能特征

主要功能特性是介绍基于Redis缓存的生活服务平台管理系统所含有的功能及其特点。

### 6.1.1 用户角色权限管理

实现了用户与角色一对多的映射关系，一个用户可以被授予多个角色。比如说，整个系统分为仓库管理员、门店管理员、订单管理员、超级管理员等几个角色，公司运营部门或者市场部门需要进行相应的细化管理，给不同的后台用户授予不同的角色权限，然后开发人员就需要进行相应的超级管理员权限，对后台所有功能进行测试开发。角色和权限是一对多的映射关系，一个角色可以关联多个菜单，形成它所拥有的菜单列表。

### 6.1.2 平台基础功能管理

生活服务平台主要就是给各个本地线下商店或超市提供一个可靠的销售平台。基本功能有门店管理、商品管理、订单管理、活动版块管理、销售统计、会员管理等模块。每一个管理模块都涵盖看基本的增加、删除、变更等操作。

## 6.2 系统特色与指标

主要介绍基于Redis缓存的生活服务平台管理系统的所含有的系统特色和指标。

### 6.2.1 前后端分离

系统以经典的SSM框架为基础搭建，在controller层进行良好的返回数据封装设计，具有统一的返回特征码和JSON数据显示。后端暴露API接口，利用swagger自动生成API文档，方便前端工程师编码设计。

前后端分离设计图如图6-1所示：

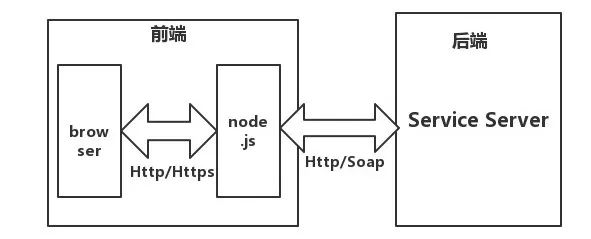


图6-1 前后端设计图

### 6.2.2 AOP日志统计

采用Spring框架中的代理模式，将所有请求拦截（拦截所有的controller方法），对方法名、方法参数列表、返回数据进行获取，封装成日志数据统一持久化。

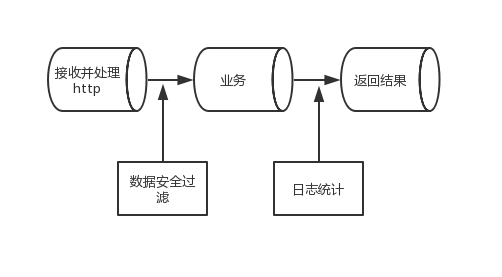
 日志统计结构图如图6-2所示：

图6-2 日志统计结构图

### 6.2.3 服务动态负载均衡实现弹性计算

服务能力保障对于关键业务来说至关重要。保障服务高可用往往需要较多的IT资源配给。现实中对服务的需求在时间上也存在高峰和低谷的波动。服务动态负载均衡提供弹性的计算资源配给能力，高效满足不同时间对服务的不同负载需求。从而服务提供者能够经济的提供高可用服务。无论业务高峰还是低谷时刻，动态负载均衡技术都能根据用户的预期自动调整计算资源配给，以极少的人工干预和资源投入获得最大化的服务能力。

### 6.2.4 服务性能统计分析支持基于数据的服务运维

过服务性能统计分析仪表盘可以对运行中的服务调用做近实时监控。统计数据可以按照租户，用户，发布者，时间，聚合时间等多维度查询，并以折线图的形式展现。通过这些维度可以聚焦特定服务运行状态，提升系统运维效率。

### 6.2.5 集群节点动态伸缩提升基础设施弹性

在实际应用中，也会有集群所有的节点计算资源耗尽的情况发生，此时无法提供新服务，旧有服务也出现响应时间变长的现象。对于这种情况，只需要把新节点加入集群就可以快速解决问题。注意：增加新计算资源的过程可能会导致已经运行的服务被重新分配资源并调度的状况发生。但是服务中断时间会是秒级，对于大多数情况来说是可以满足要求的。

# 7 程序测试与评价

## 7.1 程序调试

程序设计过程中，由于对业务理解的欠缺，对编码过程中使用的工具灵活性没有全面的考虑，存在一些错误性的编码是很正常的事情。在开发过程中，特别是针对开发人员来说（测试人员更偏向于功能性的测试，黑盒测试），使用断点调试时普遍的事情了；在自己认为比较容易出错或者说是把我不准确的代码段打上相应的断点符号，然后启动程序（启动服务器），在相应请求下，程序运行到断点位置时，程序便会中断，然后会显示相应的各个变量此时的赋值情况，然后逐条语句调试。对于此类程序调试，虽然精准，但是会比较耗时费力的。

## 7.2 程序的测试

### 7.2.1 测试的重要性及目的

1. **测试的重要性**

在软件生命周期中，从从程序的需求开始到程序的开发过程，再到交接整个过程中，软件测试这一环节起到至关重要的一环，这一环决定了整个项目能不能上线，线上反馈的程序BUG反馈率。在软件工程这门学科中，提到的瀑布模型中，软件测试所安排的时间段是在开发完成之后，运行维护之前。测试这一个环节，是软件产品完工后交付使用之前用以保证软件质量的。但是，目前大部分公司所施行的方式并不是传统的软件测试方案，而是采用开发测试交替进行的。比方说，对于开发人员在进行开发一个模块时，他开发完成后，自己也是一位测试人员，自己写完的代码必须得自己确保自己代码的准确性和健壮性，其实这就是白盒测试，也就是单元测试。

对于软件来讲，事实上，无论采用任何方法或者任何技术，系统中仍然会出现错误。采用新的开发方式、先进的语言、完善的开发过程，可以最大限度的降低错误的引进，但还是不能完全的杜绝软件中有可能出现的错误，这些引入的错误需要通过测试找出来，系统中错误的密度也需要通过测试来进行估量。测试是所有学科的工程的最基本的组成单元，是软件开发的一个很重要的部分。自有设计程序的那天起测试就一直伴随着。统计表明，在典型的软件开发项目中，软件测试工作量往往占软件开发总工作量的40％以上。而在软件开发的总成本中，用在测试上的开销要占30％到50％。如果把维护阶段也考虑在内，讨论整个软件生存期时，测试的成本比例也许会有所降低，但实际上维护程序的工作相当于二次开发，甚至是多次开发，其中一定又包含了很多测试的工作。

在实践中，测试软件非常困难，从而常常使我们敷衍了事或望而却步，这是因为对测试还是存在了一些错误的看法和不正确的态度，这包括:

1. 难以给测试的工作人员某种成就感; 认为测试工作并不像编码和设计那样容易进行发展
2. 在测试的过程中发现错位其实是一种对责任者工作的否定; 测试是以发现软件的错误为目标的，这是非建设性的，甚至还是具有破坏性的，
3. 测试的工作过程非常枯燥无味，因此不足以引起我们的兴趣和爱好;
4. 测试工作需要是细致而艰苦的进行;
5. 一般来说，我们都对自己编写出来的程序比较盲目的自信，在发现到错误的时候，可能会顾虑别人对我们自己开发的能力的看法。

以上这几个观点对测试系统的工作进行是尤为不利的，我们必须要端正态度、澄清认识，才可能将系统产品的质量加以提高。

1. **测试的目的**

如果测试程序的目的是为了尽可能全面地找出系统中的错误，那么测试就必须直观的针对软件较为复杂的部分或者是前期出现错误比较多的位置。

1. 系统的测试是为了检查出错误而执行程序的运行过程;
2. 测试则是为了证明程序中有错误，而不是用来证明程序没有错误;
3. 每一个好的测试的用例是因为它能检查到至今未出现的错误;
4. 一个很成功的测试就是检查出了至今未能发现的错误的测试。

这种观点可以提醒人们测试要以查找错误为中心，而不是为了演示软件的正确功能。但是仅凭字面意思理解这一观点可能会产生误导，认为发现错误是软件测试的唯一目，查找不出错误的测试就是没有价值的，事实并非如此。

首先，测试并不仅仅是为了要找出错误。通过分析错误产生的原因和错误的分布特征，可以帮助项目管理者发现当前所采用的软件过程的缺陷，以便改进。同时，这种分析也能帮助我们设计出有针对性地检测方法，改善测试的有效性。其次，没有检查出错误的测试也有它的价值，完整的测试是测试质量评定的一种方式。

### 7.2.2 测试的步骤

### 7.2.3 测试的主要内容

### 7.2.4 测试用例

## 7.3 系统的特点和优点

## 7.4 系统存在的不足和改进方案

## 7.5 设计收获和心得

# 8 结论

~~大型网站技术架构的核心价值并不是从无到有搭建一个大型网站，而是能够伴随小型网站业务的逐步发展，慢慢地演化成一个大型网站。一方面随着互联网的高速发展，越来越多新的软件技术和产品从互联网公司诞生，挑战传统软件巨头的江湖地位，另一方面却是中小网站十几年如一日地使用LAMP技术（Linux+Apache+MySQL+PHP）开发自己的网站，因为LAMP既便宜又简单，而且对付一个中小型网站绰绰有余。因此，一味追随大公司，大型网站的解决方案，为了技术而技术以及企图用技术解决所有问题都是网站架构设计的误区，我们都需要在网站架构中去避免，去解决。~~

~~另一方面，大型网站的架构演化方案已经非常成熟，各种技术方案也逐步产品化，许多小型网站已经慢慢不需要再经历大型网站经历过的架构演化之路就可以逐步发展壮大，因为现在越来越多的网站从建立之初就是搭建在大型网站提供的云计算服务基础之上，所需要的一切技术资源：计算、存储、网络都可以按需购买，线性伸缩，不需要自己一点一点地拼凑各种资源，综合使用各种技术方案去完善自己的网站架构了。所以，这也是本人选题的真正原因，能亲身经历一个网站从小到大的架构演化对于未来软件开发之路肯定是大有裨益的。~~

# 参考文献

[1]许晓斌，Maven实战[M].北京：机械工业出版社，2011.

[2]李智慧． 大型网站技术架构：核心原理与案例分析[J]．电子工业出版社，2013-9-1，218

[3]李平．大型网站系统架构的演化． http://www.cnblogs.com/leefreeman/p/3993449.html 2014-09-26．

[4]陶辉，深入理解Nginx[M].北京：机械工业出版社，2014.

[5]郭峰，Spring从入门到精通[M].北京：清华大学出版社，2006.

[6]Charles A Bell，Expert MySQL[M].北京：人民邮电出版社，2010.

[7] Bruce Eckel著，Thinking In Java[M].北京：机械工业出版社，2006.

[8]Michael Kofler，MySQL 5权威指南（第三版） [M].北京：人民邮电出版社，2006.

[9]蒋文旭，辛阳（北京邮电大学计算机学院信息安全中心，北京 100876）大型高并发web应用系统架构分析与设计 http://www.doc88.com/p-4179020971453.html

[10] 杨光辉．互联网型系统架构的演进． http://www.csdn.net/article/2013-08-27/2816716 2013-08-29．

# 致谢

~~经过一段时间的努力，本次论文终于圆满完成，平时在工作中与同事交流，无意中受到启发，才选定了本次的课题。本次课题，对于我这个刚入门互联网企业的初级程序员来说，是一个较大的挑战，而且主要以理论为主，并不是之前所做的偏应用型的增删改善项目，在定题之后，阅读并尝试学习了大量技术，初步对架构有了一定的认识，也明白了架构是随业务而发展的，并不能一味追求技术的狂热。~~

~~其次，我要感谢导师邓泓副教授的帮助，开题之初就给了我很多的宝贵意见，让我能顺利的完成此次论文的撰写，另外也要感谢同学，同事们的帮助和指导，是你们的耐心讲解与分析才让我更好的完成了论文。~~

~~最后，也要感谢自己，因为此次课题的研究历经磨难和考验，很多次都想过放弃而去随意完成一些简单的系统，但是为了能亲身经历一次大型网站技术架构的演化过程，并且自己搭建各种软件环境，为自己的坚持而感到骄傲，希望在未来的软件开发道路上能继续坚持下去，努力地成为一名优秀的互联网开发人员。~~