单位代码：10293 密　级：公开

南京郵電大學

专 业 学 位 硕 士 论 文

论文题目：基于 Spring Boot 的整车出库管理系统设计与开发

学　　号　1215012125

姓　　名　王振宇

导　　师　曹士坷

专业学位类别　工程硕士

类　　型　全　日　制

专业（领域）　电子与通信工程

论文提交日期　二零一八年 六月

Design and Development of a Vehicle Outbounding System Based on Spring Boot

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of Engineering

By

Zhenyu Wang

Supervisor: Prof. Shike Cao

June 2018

南京邮电大学学位论文原创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京邮电大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

本人学位论文及涉及相关资料若有不实，愿意承担一切相关的法律责任。

研究生签名：_____ 日期：_____

南京邮电大学学位论文使用授权声明

本人授权南京邮电大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档；允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索；可以采用影印、缩印或扫描等复制手段保存、汇编本学位论文。本文电子文档的内容和纸质论文的内容相一致。论文的公布（包括刊登）授权南京邮电大学研究生院办理。

涉密学位论文在解密后适用本授权书。

研究生签名：_____ 导师签名：_____ 日期：_____

摘要

随着我国经济的高速增长，人民收入的不断提高，汽车的需求量也越来越高。针对大幅提高的汽车年产销规模，许多汽车厂商配套了信息化的整车仓储物流管理系统，但是其在整车出库部分存在着一个问题：物流公司分派平板运输车司机到仓库取车的过程中，常常会出现某些时间段取车过于集中而造成仓库拥堵。通过上述问题考虑，并充分结合整车出库部分的需求，设计并实现了基于 Spring Boot 的整车出库管理系统。

系统以预约排队为主要业务模式，采用了 B/S 架构，利用 Spring Boot 技术整合系统前后端的代码。系统分为两个子系统，其中，核心业务子系统包含了取车业务、时窗管理和库区管理三个功能模块，主要实现了取车的流程；非核心业务子系统包含了系统管理、用户管理和留言公告三个功能模块，主要实现了用户权限的管理及用户之间的信息交互。

首先对系统进行需求分析，明确系统中的用户角色及系统所要实现的功能；其次对系统进行概要设计，设计系统的功能模块、整体结构及数据库；接着对系统进行详细设计与实现；最后对系统的功能和性能进行测试，测试结果表明：系统在功能和性能上表现良好，基本达到设计目标。

关键词： 仓储物流，整车出库，Spring Boot

Abstract

The people’s demand for automobiles is getting higher and higher with the rapid growth of China's economy and the continuous increase of people's income. Many automobile manufacturers have established the information-based vehicle storage and logistics management systems to cope with the large scale of auto production and marketing. However, there still remains a problem in the vehicle outbounding: when a driver of the platform truck sent by the logistics company picks up cars in the warehouse, congestion often results from concentrated traffic jam at a specific time. To meet the full needs of the vehicle outbounding, taking the above questions into consideration, a vehicle outbounding management system based on Spring Boot is designed and implemented.

The system utilizes an appointment and queue as the main service mode, adopting the B/S architecture, which integrates the front and rear codes of the system by means of Spring Boot technology. The system is divided into two subsystems. The core service subsystem consists of three functional modules: car picking service, time window management and warehouse area management. The whole process of auto picking is thus realized. The non-core service subsystem also includes three functional modules: system management, user management and message announcements, which realizes the management of user authorization and the information interaction between users.

Firstly, the requirement analysis of the system is made, including the definition of the role of an user and functions to be implemented. Secondly, the outline of the system is worked out for the functional modules, the overall structure and corresponding databases. Next the detailed design and implementation will be discussed. Finally, the functionality and performance of the system will be tested, which verifies that the system works very well in actual operation and the design goal is achieved.

Key words: **Warehousing Logistics, Vehicle Outbounding, Spring Boot**

目录

第一章 绪论

1.1 课题的研究背景及意义

随着中国经济的不断成长，汽车已经走进越来越多的家庭。中国的汽车年销量已经连续8年世界第一。目前中国汽车保有量2亿辆，保有量占全球20%，销量占全球30%。尽管如此，从全球市场格局来看，中国汽车市场空间远未饱和；从千人汽车保有量指标来看，中国目前千人保有量水平仅有140辆，与美国的809辆、日本的612辆相比还有极大的差距；从国际经验来看，除了个别人口密度非常高的国家或地区，一般一个国家的千人汽车保有量与人均GDP呈线性相关[1]。随着中国人均GDP水平的不断提高，其千人汽车保有量水平有望进一步提高。中长期中国汽车年产销规模估计在4000万台左右，较2017年的2900万辆还有1100万辆以上的增长空间。

中国的汽车市场仍旧是全球少有的充满活力的汽车市场，集中度偏低，品牌数众多。从现有的状况分析来看，未来中国汽车市场增长的蛋糕将属于少数真正有实力的品牌[2]。然而，汽车销量的增加必然会给整车仓储物流环节带来压力，这就对汽车厂商的整车仓储物流管理提出了非常高的要求：不仅在硬件上要建设完善的配套设施，在软件上也要有一套先进的管理系统来配合管理[3]。国内很多规模较小的汽车厂商受到资金和运输规模的限制，无法配套完善的信息化管理系统，许多管理工作只能依靠人工来进行[4]。信息传递慢，差错率高，管理效率较低，没有完善的信息平台，无法提供高效、优质的服务[5]。

中国的整车仓储物流网络主要分为以下四个部分：

（1）主机厂，即汽车厂商的生产厂，商品车的供应者。

（2）整车分拨中心，指直接接收主机厂下线的商品车，提供仓储服务并承担一次分拨发运职能的整车仓库[6]。

（3）整车仓储中心，指接收由整车分拨中心分拨发运的商品车，提供仓储服务并承担二次分拨发运职能的整车仓库[7]。

（4）经销商，指分布在全国各地的汽车零售企业，它们是商品车的需求者。

从中不难发现，整车仓库是连接汽车厂商和经销商的桥梁，在整个整车仓储物流网络中起着承上启下的枢纽作用，其主要特性的差异可以影响到其他物流设施能力的发挥，直接关系到物流成本、效率及服务水平[8]。整车仓库的管理主要分为入库管理、在库管理和出库管理

三个部分。其中，整车出库部分主要是第三方物流公司派平板运输车司机到整车仓库取车并运送往经销商处的过程[9]。然而在现实中该过程仍然存在着一些问题，最典型的就是部分汽车厂商的整车仓储物流管理系统在整车出库模块没有对司机取车时间进行合理优化，有时会出现某一时间段取车过于集中而导致仓库拥堵等一系列问题。因此，本文主要针对整车出库部分设计了一个信息化管理系统，使其作为整车仓储物流管理系统中的一个小模块而实现对整车出库部分的管理。为了解决前面提到的问题，在系统中使用了预约取车的模式：司机在预约时需要选择取车的时间段，仓库的管理员会根据仓库运作情况设定每个时间段的最大预约人数，达到最大预约人数后该时间段在当天无法继续预约。这样可以在一定程度上对司机取车的时间进行均衡分配，改善仓库拥堵等问题。

1.2 国内外研究现状

1.2.1 国外研究现状

世界上一些发达国家例如美国、日本和德国等汽车工业发展较早，市场规模也十分巨大，所以，这些国家的大多数汽车厂商都已经拥有完善的整车仓储物流管理系统。

日本的丰田是世界上最大的汽车品牌之一。丰田公司除了能够生产出高质量的汽车之外，同时拥有十分先进的整车仓储物流管理系统来支持其庞大的汽车销售网络[10]。丰田公司使用 G-YNS 系统，该系统使丰田公司在信息平台的建立上具有了得天独厚的优势，能够高效管理其复杂的整车仓储物流网络上的每一个环节。G-YNS 系统在整车出库部分主要有以下功能：

（1）仓库管理：对出库车辆停放进行定制管理，实施出库车辆登记，分类统计商品车状态，如仓库各库区的可存放量、库存总量和各库区利用率等[11]。

（2）配车计划管理：对商品车出库进行计划管理，输出运输指令。

（3）交付认证管理：承运商到达后进行运输时间、数量的确认，实施交车确认。

（4）在途监控管理：汽车在移动过程中使用 GPS 进行监控，并将行驶数据保存[12]。

（5）报表处理功能：将系统内的各种整车出库信息自动生成各类报表，方便查阅。

美国通用公司也已经建立了一套较为完善的整车仓储物流体系，而且将整车物流服务供应商的调度管理信息也集成在整个平台上。信息系统对于仓储管理和运输控制等主要职能都有了较强的支撑作用，它所包含的功能子系统有：

（1）整车配送中心和整车区域仓储中心的仓储管理系统。

（2）车辆运输监控系统。

（3）通用物流服务供应商的订单调度系统。

丰田公司和通用公司利用它们强大的整车仓储物流管理系统进行订单的分理分派，在整车的销售物流过程中，完全通过信息技术处理整个运营过程，对第三方物流服务商进行统一管理与考核，物流运作信息的集成管理与共享程度达到了非常高的水准[13]。

1.2.2 国内研究现状

国内许多小型的汽车厂商普遍存在着信息化程度低的问题，它们的整车仓储物流管理基本采用了半信息化平台半手工的模式，这就与国际著名汽车厂商不断追求信息化、科技化的潮流格格不入。没有完善的信息化管理系统的支持，这些小型的汽车厂商想要提高自己的竞争力并做大、做强自己的品牌只能是纸上谈兵。

当然，国内一些大型的汽车厂商由于在发展中积累了一定的经验和资金，也拥有着相对完善的整车仓储物流管理系统。东风汽车在日常的生产、经营和管理活动中，面临的最大挑战是运营管理。这是因为整个管理过程涉及到许多因素，整车仓储物流管理就是其中重要的因素之一。为了更好地进行上述管理活动，东风汽车经过考察比较，最终选用了中软冠群公司的 ES/1 SuperLogistics 系统[14]。在使用该系统进行管理的基础上，东风汽车公司整车仓库的出库流程如下：

（1）经销商在该系统上进行订单的提报并确认采购车辆的型号、数量、颜色和配置等基本信息。

（2）财务部门在该系统上确认经销商账户上是否有足够的金额，确认完毕后系统将该订单分配给相关的物流公司。

（3）物流公司收到分配的订单后根据订单信息生成相应的运单，并安排司机去仓库取车[15]。

（4）司机凭借运单到仓库提车，仓库管理员根据运单进行整车出库作业。

ES/1 SuperLogistics 系统为东风汽车的整车仓储物流管理提供了科学的信息化解决方案，极大地提高了各部门的工作效率。

1.3 课题研究的主要内容

课题研究内容主要是整车仓储物流管理中的一个环节——整车出库管理，并利用 Web 技术将其实现。系统使用了当前最流行的 Web 框架——Spring Boot 框架来整合系统的前后端代码。前端部分使用了 html5、bootstrap、iframe、CSS 和 JavaScript 等技术，后端部分使用了

MVC 模式以及 MyBatis 框架，采用 Java 作为开发语言，MySQL 作为系统的数据库。主要工作如下：

（1）学习各项技术相关理论知识并在公司实习过程中熟练其使用方法。

（2）对系统中相关业务进行调研并对现有业务流程进行分析与改进。

（3）从功能性和非功能性两个方面对系统进行需求分析，并设计系统的需求分析用例图。

（4）对系统进行概要设计，包括功能模块、系统结构和数据库的设计。

（5）对系统进行详细设计，通过设计系统各模块的类图来确定其底层运作流程，并用代码将整个系统实现。

（6）从功能、性能以及安全性三个方面对系统进行测试，分析测试结果。

1.4 论文的组织架构

本文主要分为七个章节，具体每个章节的概要如下：

第一章是绪论部分。首先介绍了课题的背景和意义；然后介绍了本课题国内外的研究现状；接着简要说明了课题主要研究内容及创新点；最后大致概括了整篇论文的组织架构。

第二章对于实现系统所涉及到的关键技术进行简单介绍，包括了 Spring Boot 框架、MVC 模式、数据库相关技术以及一些 Web 前端技术。

第三章是对整车出库管理系统的需求分析。首先对系统的总体需求做概述；其次对系统中的角色进行分析；最后从系统的功能性需求和非功能性需求两方面来对系统需求分析展开具体的说明。

第四章是对整车出库管理系统的概要设计。首先对系统进行功能模块的划分并简要介绍每个模块具体功能及设计细节；随后对于系统中的关键问题及重难点问题提出初步的解决方案；紧接着对系统的结构进行分析；最后详细介绍了系统数据库中的字段信息。

第五章是对整车出库管理系统的详细设计与实现。首先介绍了实现该系统的开发环境与平台；然后分别介绍核心业务子系统和非核心业务子系统的详细设计与实现，设计功能模块的类图并详细说明底层代码的运作流程，展现部分页面的实现效果，对概要设计中提出的关键问题及重难点问题给出详细的实现步骤。

第六章是对整车出库管理系统的测试。主要从系统的功能、性能以及安全性三个方面对系统进行测试，分析测试结果并得出结论。

第七章是总结与展望。主要总结了整车出库管理系统开发的过程；指出了系统的不足之处，并对今后做出展望。

第二章 相关技术概述

2.1 Spring Boot

Web 开发经过多年的技术选型，Spring Boot 渐渐进入开发者的视野，成为目前很多大公司进行 Web 开发的必选技术。

Spring Boot 框架是由 Pivotal 团队所开发的。使用该框架可以迅速构建出一个能够独立运行、准生产级别的基于 Spring 技术的项目[16]。它提供了以下核心功能：

（1）自主运行的项目：Spring Boot 可以将开发完成的程序直接打包成 jar 包来运行，跟平时运行 java 项目没什么区别。

（2）内部提供了 servlet 容器：Spring Boot 框架内部提供了 Tomcat 等 servlet 容器，使项目的部署变得更加方便[17]。

（3）减少配置：Spring Boot 采用“约定大于配置的理念”，开发人员能够尽可能少地编辑配置文件[18]。当然，在一些特殊的开发环境下，有些配置还是需要开发人员手动进行的。

Spring Boot 框架的核心模块是由 Spring 框架提供的，Spring 提供对控制反转、面向切面的编程 AOP、容器管理、MVC 框架、事务管理和异常处理的支持[19]。Spring 框架由核心容器（Core Container）模块、面向切面编程（AOP 和 Aspects）模块、数据访问/整合（Data Access/Integration）模块、Web 模块和测试（Test）模块组成[20]。Spring 框架的组成如图 2.1 所示：

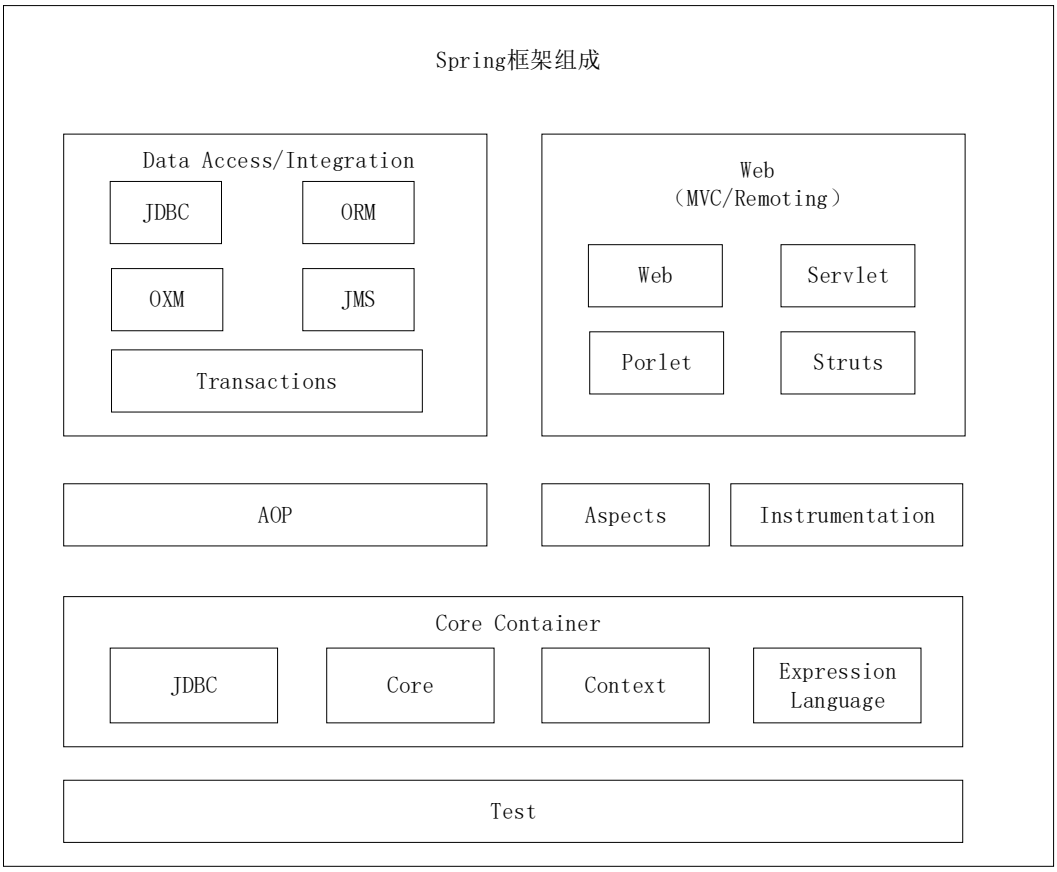


图 2.1 Spring 框架组成图

核心容器模块中，Spring Beans 模块实现 Spring 框架的核心部分，主要包括对控制反转（IOC）和依赖注入（DI）的实现；Spring Core 模块实现 Spring 框架的最基础部分，主要包括服务器的资源访问和一些常见的工具类；Spring Context 模块集成了 Spring Beans 模块的功能并提供对应用资源的绑定、数据验证和国际化等特性的支持；Spring Expression Language 模块提供对表达式语言的支持。

面向切面编程（AOP 和 Aspects）模块提供对面向切面编程的实现，面向切面的编程是指针对项目开发过程中的某一步骤或者阶段进行提取，实现业务处理过程各个阶段之间的低耦合。

数据访问/整合模块中，JDBC 实现一个通用的 Java 数据库连接模板；ORM 模块提供对对象关系模型的支持；OXM 模块支持数种开源对象 XML 模型框架；JMS 模块提供对消息发送和接受服务的支持；事务模块实现 Spring 框架中的事务管理功能。

Web 模块提供了应用开发所需要的基础功能，包括支持多种格式的文件上传、远程过程访问以及相关 Web 服务。

测试模块实现应用程序的测试需求，支持开源测试框架 Junit。

2.2 MVC 模式

MVC 是一种软件的设计规范，用一种将页面、业务逻辑和数据库分离的模式来编写代码[21]。MVC 其实就是 Model、View 与 Controller 三个单词的缩写，分别代表着模型、视图页面与控制器。其中，模型是系统中用于处理应用程序数据逻辑的部分；视图页面是用户在使用中能直观感受到的模块，常用的视图页面包含了 JSP、freeMarker 和 Thymeleaf 等技术；控制器是系统中处理用户交互的部分。图 2.2 展现了 MVC 模式的工作流程：

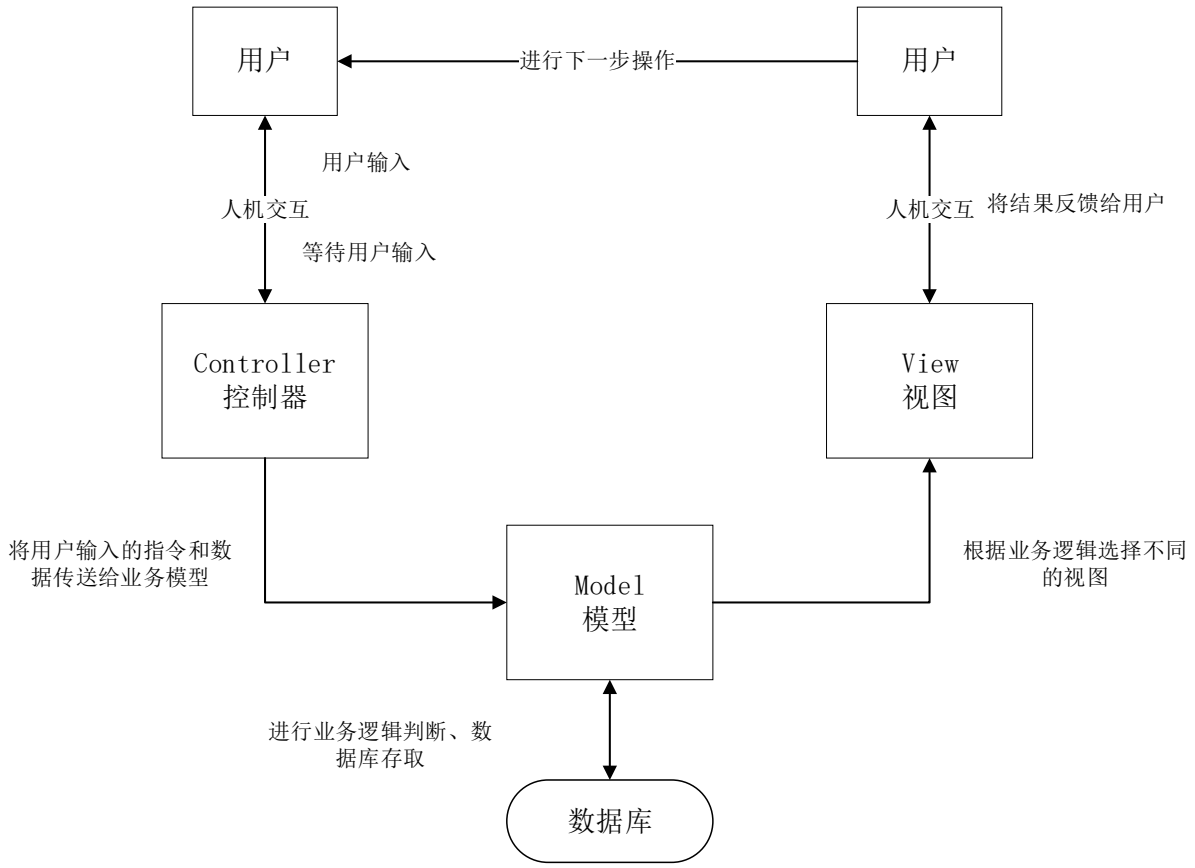


图 2.2 MVC 模式工作流程图

它的工作流程大致是这样的：用户在进行业务操作时会给控制器发出指令，控制器将这些指令以及相关数据传送给模型，模型根据这些指令对业务逻辑进行判断并且与数据库进行交互，完成后根据业务逻辑选择不同的视图页面并发送处理完的数据，视图页面将处理完的结果通过前端技术展现给用户。

Spring 提供了一款出色的基于 MVC 模式的框架——SpringMVC，它也是企业开发中使用最多的 MVC 框架[22]。

2.3 数据库相关技术

2.3.1 MySQL

MySQL 已经成为世界上最受欢迎的数据库管理系统之一。无论是用在小型项目开发上，还是用来构建那些声名显赫的网站，MySQL 都已经被证明是一个稳定、可靠、快速和可信的系统，足以满足任何数据存储业务的需求[23]。

MySQL 在世界范围内得到了广泛的使用，它有如下特点：

（1）成本低：MySQL 是开放源代码的，一般可以免费使用，甚至可以免费修改。

（2）性能高：MySQL 的执行速度非常快。

（3）可信赖：目前世界上很多大型的公司或者网站都使用了 MySQL 作为其数据库管理系统，并且用其处理非常重要的数据。

（4）易操作：MySQL 安装和使用非常方便。

2.3.2 MyBatis

MyBatis3 之前的版本叫 ibatis，原本是 Apache 下的一个项目，后来被谷歌公司托管，随即改名为 MyBatis[24]。

MyBatis 是一款非常优秀的持久层框架，所谓持久层框架，就是负责与数据库进行交互的框架。持久层的技术有很多，例如 JDBC、Hibernate 和 MyBatis 等等[25]。下面将各种持久层技术的使用流程进行对比来说明使用 MyBatis 的好处：

（1）JDBC

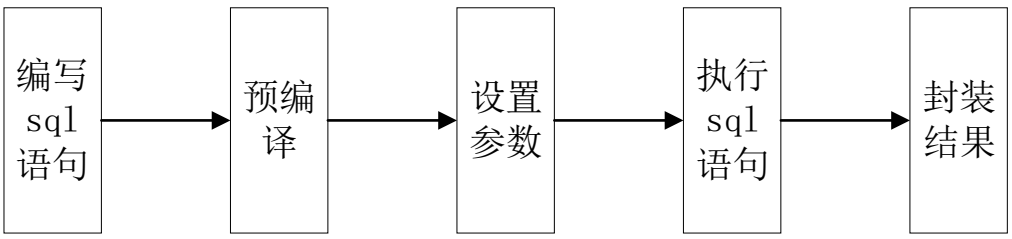


图 2.3 JDBC 使用流程图

如图 2.3 所示，传统的 JDBC 技术使用起来非常复杂，大致要经历编写 sql 语句、预编译、设置参数、执行 sql 语句和封装结果这五个步骤。它可以实现的功能非常少，而且采用了硬编码、高耦合的方式，将 sql 语句编写在 java 代码中，使后期维护变得非常困难。

（2）Hibernate

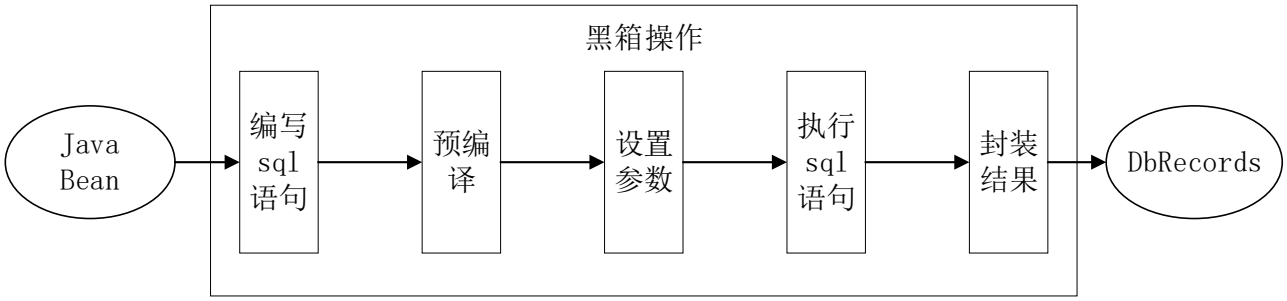


图 2.4 Hibernate 使用流程图

Hibernate 是一个全自动 ORM 框架，ORM 框架将 javaBean 对象映射到数据库中的每一条记录，使它们一一对应。如图 2.4 所示，Hibernate 也需要经历与 JDBC 相同的步骤，不过与 JDBC 不同的是，Hibernate 将这些步骤全部封装起来进行黑箱操作，用户感觉不到经历了这些步骤，所以称之为全自动框架。但是，Hibernate 将编写 sql 语句也封装了起来，由框架自己发送 sql 语句，这样便造成了 sql 语句无法优化的后果。而且，几乎所有的项目都需要使用到一些复杂的定制 sql 语句，要想使用 Hibernate 达到这个目的，则必须要对 Hibernate 提供的 HQL 语句非常精通，这无疑又增加了开发者的学习负担。

（3）MyBatis

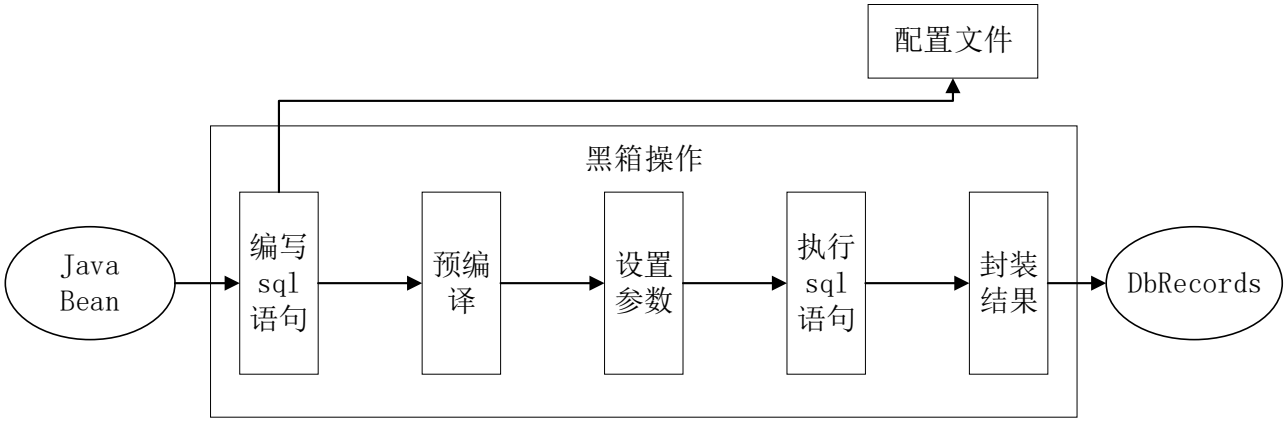


图 2.5 MyBatis 使用流程图

MyBatis 是一个半自动的 ORM 框架。如图 2.5 所示，MyBatis 与 Hibernate 最大的不同就是它将编写 sql 语句抽取出来并以配置文件的形式呈现，即将 sql 语句单独写在配置文件中。这就改变了 Hibernate 中由框架发送 sql 语句的做法，而将这个权力交给了开发者。这样使 sql 语句与 java 编码分离，方便后期维护。使用 MyBatis 只需要掌握 sql 语句的使用，无须学习其他的语句，减轻了开发者的学习负担。

2.4 各种 Web 前端技术

Web 前端就是展现给用户的部分，即客户端或者浏览器端的部分。在整车出库管理系统的开发中主要使用了如下 Web 前端技术：

（1）JavaScript：它是一种直译式的脚本语言，是一种动态类型、弱类型、基于原型的语言，广泛用于客户端的脚本语言中。最早是在 HTML 网页上使用，可以给 HTML 网页增加动态功能。

（2）Ajax：它不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的 Web 应用程序的技术[26]。Ajax 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可使网页从服务器请求少量的信息，而不是整个页面。例如，在网站注册的时候，有时输入用户名后，网页上会提示该用户名已被注册，这种效果就是通过 Ajax 来实现的，全程没有经过页面的跳转。

（3）CSS：它是网页的“装饰工”，通常使用 html 技术编写的网页十分单调，CSS 可以给网页增加样式，增强用户的体验。

（4）html5：它是万维网的核心语言，用来编写 Web 网页。

（5）iframe：它的作用是将整个页面分成不同的框架，每个框架中都可以加载一个网页，实现了网页的嵌入。

（6）bootstrap：它将一些 html5/CSS 进行了简单的封装，使用 bootstrap 后，开发者可以在没有美工的修饰下做出漂亮的网页。

2.5 本章小结

本章主要介绍了开发整车出库管理系统所涉及到的一些相关技术。首先对 Spring Boot 技术进行详细介绍；然后根据流程图简单说明了 MVC 模式的具体工作流程；接下来介绍了数据库相关技术，主要介绍了开发中使用到的 MyBatis 框架，并将该框架与其他 ORM 框架的工作流程进行比较，突出了使用 MyBatis 的好处；最后简单介绍了一些 Web 前端技术。

第三章 整车出库管理系统的需求分析

需求分析是开发人员经过深入细致的调研和分析，准确理解用户和项目的功能、性能以及可靠性等具体需求，将用户非形式的需求表述转化为完整的需求定义，从而确定系统必须做什么的过程。本章主要从系统的功能性需求和非功能性需求两个方面来对系统进行需求分析。

3.1 项目概述

3.1.1 系统实现目标

随着中国经济的不断发展，国内家庭对汽车的需求越来越高，每天都会有很多客户前往汽车经销商（4S 店）购买商品车。

分析客户购买商品车的大致流程：首先客户在经销商处选购商品车，如果经销商没有用户需求的颜色、型号和配置的车辆，则会向汽车厂商提交相应的订单；汽车厂商在接受了订单之后进行商品车的生产，并将下线后的商品车运往整车仓库存放；接下来第三方物流公司会派出平板运输车司机前去仓库提运相应车辆送往经销商处；最后用户去经销商处提取商品车。

系统主要实现上述流程中整车出库部分的信息化管理，即管理商品车从仓库交付给司机的过程。但是，由于司机取车时间的不确定，有的时间段来仓库取车的司机很多，有的时间段几乎没有，这样会造成以下问题：

（1）由于平板运输车的车身长度很长，体积巨大，因此，在去往整车仓库的必经路段会造成该路段的道路拥堵，甚至造成该路段的交通瘫痪；而且，同一时间段过多的平板运输车集中停放在整车仓库，也会造成仓库之间的拥堵。

（2）同一时间过多司机前来仓库取车会降低仓库工作人员的服务效率，仓库服务人员无法满足取车需求，造成司机不必要的等待。

（3）过大的服务压力必然会导致服务时间的缩短以及服务质量的下降，很可能会造成仓库管理人员的工作失误，例如商品车在出库的过程中受到刮损。

本文研究的基于 Spring Boot 的整车出库管理系统是根据汽车厂商在整车出库部分实际的管理需求而设计与开发的，并于开发之前针对上述问题拟定了系统的实现目标：

（1）实现司机取车时间段的合理分配。

（2）实现不同角色的用户共用系统。

（3）实现仓库管理员对库区的管理。

（4）实现司机与仓库工作人员的信息交互。

（5）实现仓库的业务统计。

3.1.2 可行性分析

软件的可行性分析是通过对项目的市场需求、资源供应、建设规模、工艺路线、设备选型、响环境影、资金筹措和盈利能力等方面的研究，本节从经济、技术和操作三个方面对系统进行可行性分析[27]。

（1）经济方面：系统作为汽车厂商配套的整车仓储物流管理系统中的一个模块，与其他模块紧密结合共同对商品车仓储营销网络进行管理，提高工作效率，减轻了汽车厂商在人力雇佣方面的负担；系统设计不以盈利为目的，使用系统的主要是仓库的工作人员以及合作物流公司的运输司机，没有较大的并发性压力，所以对服务器没有太高的要求；系统界面清晰明了，使用流程简明易懂，所以无须对工作人员进行长期的培训与指导；在配套硬件建设上，系统后期需要配套一块电子显示屏用来显示排队情况以及一台二维码取票机供司机进行取票操作，这些对于汽车厂商来说都在合理的开支范围内。综上所述，系统具备相当好的经济性。

（2）技术方面：系统开发中所使用到的所有技术都已经相当成熟，这些技术经过市场多年的验证以及版本的不断更新，已经很少有严重的 bug 存在。本人已对相关技术的理论知识进行学习并且在公司实习时进行过实践，基本掌握了相关技术使用的方法与技巧。另外系统使用的 Spring Boot 框架创建与部署项目相比于 Spring 框架更加简便。综上所述，系统的开发在技术上满足可行性要求。

（3）操作方面：工作人员可以直接输入网址进入系统登录页面或者未来接入到整车仓储物流管理系统后通过其他模块进行跳转。系统的菜单简捷，页面美观，对于一些错误的操作也会有相应的提示。综上所述，系统在操作上满足可行性要求。

3.1.3 系统角色分析

系统的用户分为以下三种角色：司机、大厅管理员和仓库管理员。每种角色可以使用的功能不相同，在登录系统后，不同角色主页面左侧的导航菜单也不相同。另外，为了便于在开发过程中测试系统功能，设置了一个虚拟的超级管理员角色，该角色涵盖了上述三种角色

的大部分功能。系统中各个角色主要功能如表 3.1 所示：

表 3.1 系统各角色主要功能

角色名称	主要功能
司机	预约取车；根据预约号取票；在留言板中留言；查看管理员发布的公告；修改个人密码。
大厅管理员	为司机办理取车业务；管理仓库的时窗，根据仓库实际运作情况对各时窗的最大预约能力进行分配；管理系统中所有用户的个人信息；查看并回复司机的留言；发布相关公告。
仓库管理员	在个人负责的库区进行整车出库作业；管理仓库中各库区的信息；查看并回复司机的留言；发布相关公告；维护库区服务的统计信息。
超级管理员	涵盖了系统的大部分功能，主要用作开发中对于系统的功能进行测试。

3.2 系统功能需求分析

系统设计的目的一定是为了能够实现一些特定的功能，因此功能需求分析对于整个需求分析来说非常重要。在功能需求分析中，最重要的是要确定系统需要“做什么”以及系统各环节如何运作，而不必考虑怎么实现这些功能。在具体分析方法上，本文使用了用例图来形象地表现系统的功能需求[28]。

根据调研的结果以及对系统的功能分析，将整车出库管理系统分为两个子系统，分别为核心业务子系统和非核心业务子系统。

3.2.1 核心业务子系统

核心业务子系统包含了整车出库的核心业务，主要是取车的流程以及对仓库的管理。

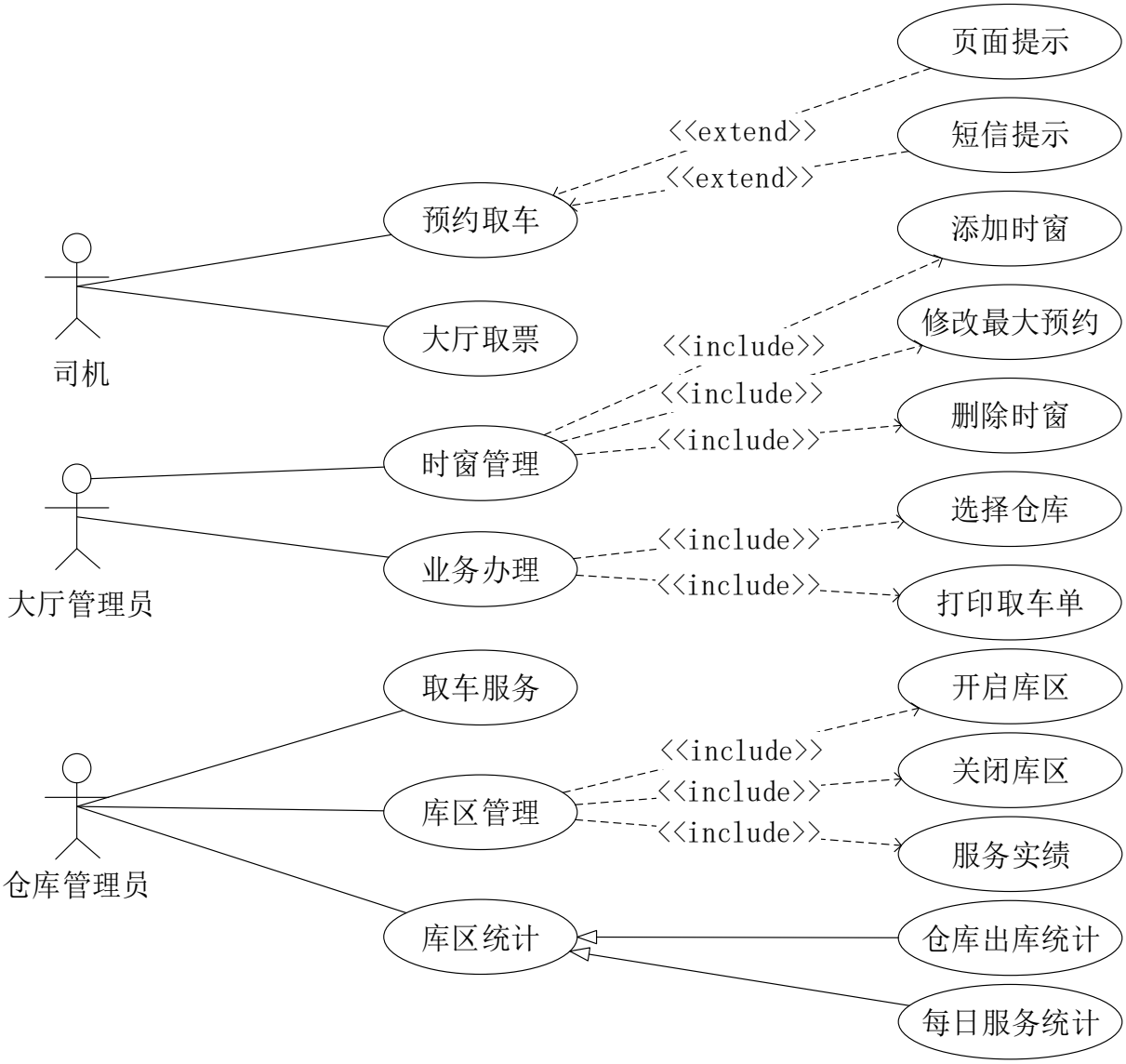


图 3.1 核心业务子系统用例图

核心业务子系统的用例图如图 3.1 所示，该子系统包含了司机、大厅管理员和仓库管理员三种角色。司机主要是取车服务的需求者，在取车前需要进入系统进行预约，预约成功后会有短信提示，在取车日当天根据预约号取票并等待系统叫号；大厅管理员主要负责对正在排队的票号进行业务办理，根据司机的取车信息分配取车的库区并打印生成的取车单，盖章交给司机，司机凭借取车单到相应的地点取车，大厅管理员还需要根据仓库的实际运作情况调整系统中的时窗信息，即改变司机可预约的时间段及每个时间段的最大预约能力；仓库管理员主要负责取车服务以及库区的管理，根据仓库的运作情况开启或关闭库区，另外，仓库管理员可以查看仓库中商品车出库情况以及每日服务情况的统计图。

3.2.2 非核心业务子系统

非核心业务子系统包含了一些其他的功能，这些功能不属于整车出库的主要业务但是又

是系统运作过程中必不可少的。

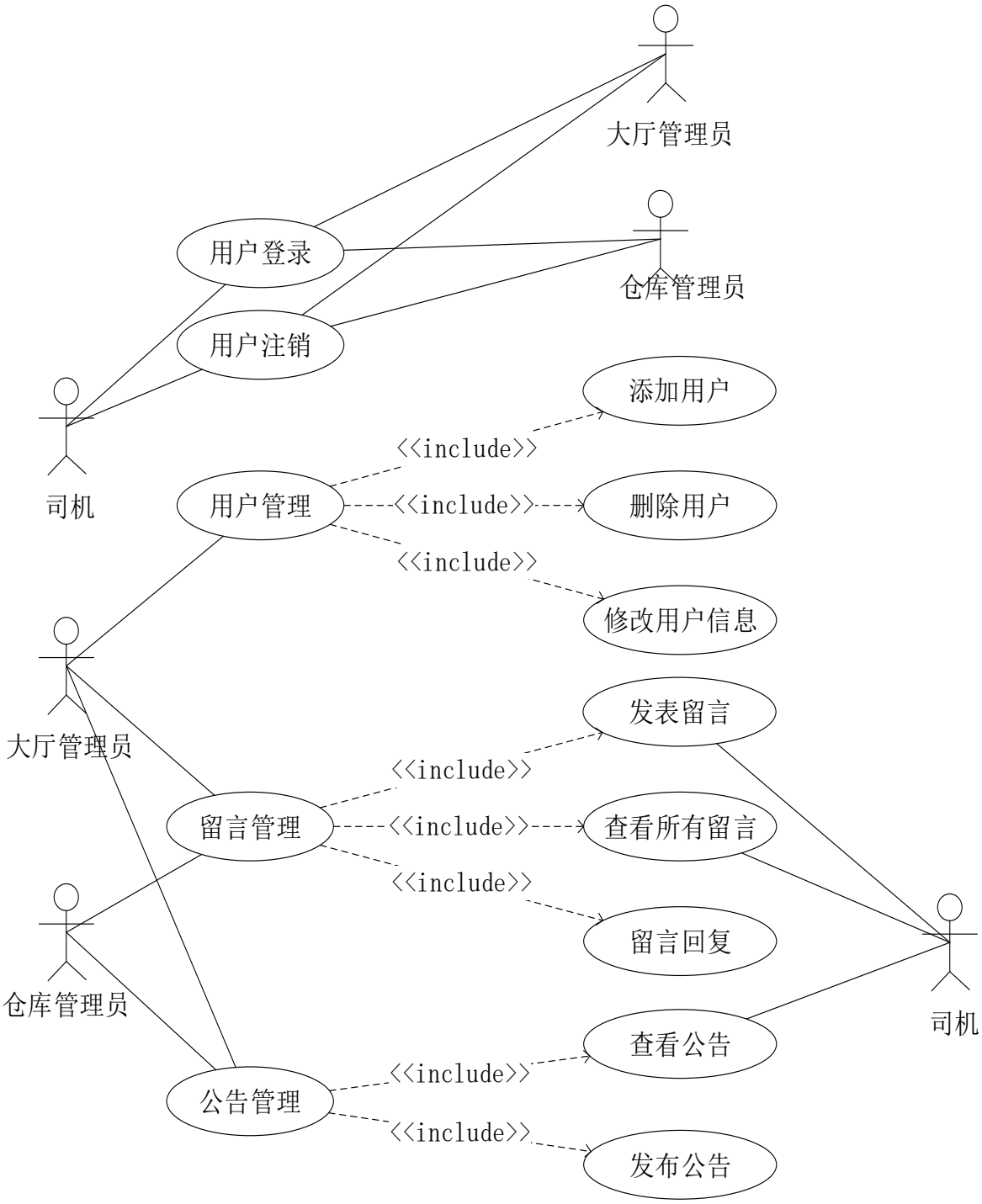


图 3.2 非核心业务子系统用例图

非核心业务子系统的用例图如图 3.2 所示，该子系统也包含司机、大厅管理员和仓库管理员三种角色。三种角色的用户都可以使用账号和密码登录系统，使用完毕后可以注销账号；大厅管理员负责对用户的信息进行管理，修改用户个人信息以及新增用户都需通过大厅管理员完成；司机和管理员之间可以通过留言进行交流；管理员也可以发布公告给司机相关通知。非核心业务子系统中的功能虽然与取车业务并不直接相关，但是却是系统管理中常用的功能，与核心业务子系统共同发挥整车出库管理的作用。

3.3 系统非功能需求分析

作为对功能性需求的补充，软件需求分析的内容中还应该包括一些非功能需求。本文主要从安全性、可靠性、界面直观性、和可维护性这四个方面对系统提出相应需求。

3.3.1 安全性需求

安全性需求几乎是每一个系统都要考虑到的，安全性很差的系统是没有实际应用价值的。系统对安全性方面的需求主要体现在以下几个方面：

（1）内部保密性：用户必须凭借特定的账号和密码成功登录系统后才能使用系统的功能。而且，用户账号的授权由管理员来完成，且必须经过严格地审核，只有得到授权的用户才能修改个人信息。

（2）越权拦截：由于系统由多个角色共同使用。为了避免角色之间的信息泄漏，必须要做到访问拦截。例如司机不属于管理人员，当司机通过某种手段获得管理员功能页面的地址并试图访问后，系统需要对其进行拦截。

（3）防范恶意攻击：系统作为一个 Web 项目，必须防范可能会受到的恶意攻击行为，如 SQL 注入攻击等。

3.3.2 可靠性需求

可靠性表示系统运行过程中顺利达到预期功能的能力，它是评价一个系统是否具有高质量的关键指标。服务器性能、网络以及软件结构的稳定性都是影响系统可靠性的相关因素[29]。

（1）服务器承载着整个系统的运行。若系统没有达到一定的规模，通常使用单服务器即可保证系统业务的运行。但是，如果系统涉及到相当重要的数据或者系统的规模过为庞大则应使用多服务器来保障系统的流畅运行，有时还需要在不同服务器上进行数据备份以备不时之需。

（2）网络的影响对于 Web 系统十分关键。网络传输质量的好坏取决于网络带宽和传输信道等相关因素，在项目运行中必须保持网络具有较高的传输质量。

（3）软件结构的稳定性取决于系统在运行过程中的流畅度以及适应力，一个好的系统应该能在不同环境下都具有很高的运行流畅度。

3.3.3 页面直观性需求

页面作为用户与系统交互的直接载体，它的直观性直接影响到用户的体验。页面的设计应该尽量做到简洁，不应该过度渲染导致喧宾夺主。整个系统的所有界面在风格上应该保持统一，在不同功能的界面上又要能够体现其细节的差异性。

页面给用户的视觉呈现受到页面的设计架构、页面的功能元素等多方面的影响。页面的设计架构影响到页面的整体布局，页面的布局应相对合理，各菜单栏、导航按钮以及 logo 图标的位置分配应当经过用户体验后进行不断修改与升级；页面的色彩搭配以及色调选择对于用户的体验感也十分重要，应当避免滥用色彩，导致用户的视觉疲劳；页面的按钮提示应该做到通俗易懂，这对于提高系统的可用性以及降低用户学习难度都至关重要。

3.3.4 可维护性需求

可维护性是指对于软件上线运行时发现的问题进行功能修改和业务拓延的性能。一个项目在上线运行的过程中会不断地进行更新维护，所以系统可维护性的好坏直接决定了后期维护的工作量。如果不注重项目的可维护性需求，后期在维护时会付出巨大的代价。为此，系统的开发应该满足下面几点要求：

（1）在系统的框架选用上应该选用市面上主流的框架。主流的框架具有非常高的成熟度，经历过市场的检验，具有非常好的稳定性；对于框架使用过程中出现的各种问题在网上基本都有解答，这样会使项目在维护上更加方便。

（2）系统在设计的时候应该进行详细的模块划分，并且使用合理的架构来降低各模块之间的耦合度，尽量避免因个别模块的维护而导致其他模块出现问题。

（3）系统在代码编写上应当遵循一定的规范，在变量的命名上应当使用浅显易懂的英文名称。代码的复杂度也是系统可维护性的关键因素，应尽量少使用复杂的算法，并且为代码添加详细的注释。

（4）需要为系统配备完善的开发文档。开发文档可以使程序员在维护时更清楚、更方便地了解系统的开发信息和细节，更加容易地发现系统中的错误。

3.4 本章小结

本章主要对系统的需求进行了详细地分析。首先对系统的实现目标、可行性以及角色进行分析；接着对系统的功能性需求进行分析，主要将系统分为核心业务和非核心业务这两个

子系统，并分别通过用例图来对这两个子系统的功能需求进行说明；最后从系统的安全性、可靠性、页面直观性和可维护性四个方面总结了系统的非功能需求。

第四章 整车出库管理系统概要设计

概要设计就是设计软件的结构，包括组成模块、每个模块的功能以及系统结构等等。同时，还要设计系统总体的数据结构和数据库结构，即数据的类型以及数据间的关系。本章主要从上述方面介绍系统的概要设计部分。

4.1 系统功能模块的设计

根据系统的需求分析，将整车出库管理系统分成两个子系统，分别是核心业务子系统与非核心业务子系统。核心业务子系统中包含了取车业务模块、时窗管理模块和库区管理模块，非核心业务子系统中包含了系统管理模块、用户管理模块与留言公告模块。具体的系统功能结构图如图 4.1 所示。

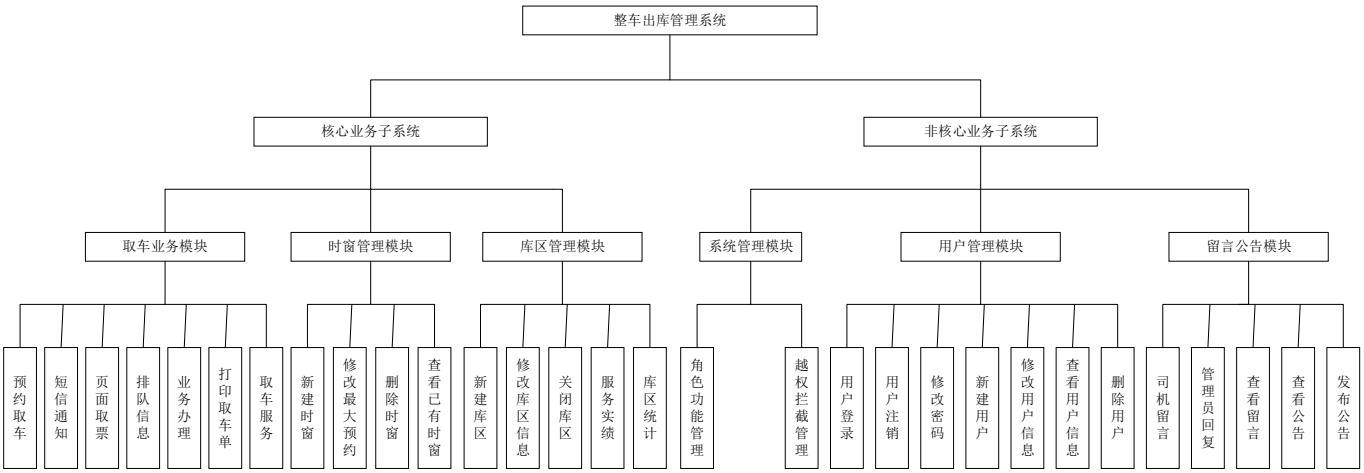


图 4.1 系统功能结构图

4.1.1 核心业务子系统的功能设计

核心业务子系统主要承担着整个系统中主要业务，下面分别对核心业务子系统中的取车业务、时窗管理和库区管理这三个模块的具体功能设计做介绍。

（1）取车业务模块是核心业务子系统中的重点模块，该模块主要实现了司机从预约到取车的所有流程，该流程由司机、大厅管理员和仓库管理员共同完成。取车业务模块具体的功能列表如表 4.1 所示：

表 4.1 取车业务模块功能表

模块名称	模块包含功能
取车业务模块	预约取车
	短信通知
	页面取票
	查看排队信息
	业务受理
	取车服务

预约取车：司机在收到物流公司分派的运输任务后，在预约页面上进行预约。预约时要选择相应的日期以及时窗，并填入需要运送车辆的相关信息以及司机个人的信息。

短信通知：在司机填写正确的预约信息并且提交后，系统会自动向司机填写的手机号码发送一条短信来告诉司机预约已经成功。

页面取票：司机在预约成功后会得到一个预约号，在取车当天司机到达等待大厅凭借预约号在取票页面进行取票。也可通过扫描预约时系统生成的二维码进行取票。

查看排队信息：司机取票后，对应的票号立即进入排队队列。大厅管理员可以查看当前排队的情况，该页面上需要显示正在服务、下一位服务以及正在排队的票号信息。

业务受理：司机在听到大厅广播叫到自己的票号后去往相应的柜台找大厅管理员办理业务。大厅管理员根据司机运单中的信息分配库区以及仓库管理员，生成并打印取车单，司机根据取车单上的地址信息去往对应地点进行取车。值得一提的是，当广播叫号后司机长时间不来柜台办理业务，大厅管理员可以选择将该票号作废。

取车服务：司机来到取车地点后，由大厅管理员分配的仓库管理员对司机进行取车服务，在核对好待取商品车的基本信息以及确认其完好无损后，进行整车出库作业，完成整个取车业务流程。

（2）时窗管理模块的设计主要是为了解决同一时间段预约人数过多导致仓库拥挤的问题。时窗就是司机在预约时选择的时间段，为了方便管理，将每个时窗的时间跨度设置为一个小时，并且需要为每个时窗设置最大预约能力。时窗管理模块具体的功能列表如表 4.2 所示：

表 4.2 时窗管理模块功能表

模块名称	模块包含功能
时窗管理模块	新建时窗
	开启时窗
	删除时窗
	修改最大预约能力
	查看所有时窗

新建时窗：新建一个系统中不存在的时窗，新建时需要设置时窗的起始与终止时间以及最大预约能力。如果选择起始时间与终止时间不为整点，则系统会自动将其变成整点；如果

跨度超过了一个小时，系统会提示错误，需要重新新建。

开启时窗：在新建完一个时窗后，需要将其状态设置为开启，该时窗才能起作用；如果由于特殊情况想暂时关闭某个时窗而又不想将该时窗从系统中删除，则可以将对应时窗的状态修改为关闭。

删除时窗：将系统中已经存在的时窗删除，并且删除该时窗的所有信息。

修改最大预约能力：管理员可以根据仓库的实际情况修改每个时窗的最大预约能力，保证仓库以最大效率运作。

查看所有时窗：大厅管理员和仓库管理员都可以查看每日的时窗信息。

（3）库区管理模块的设计主要是为了有效地进行库区管理。在库区管理模块中还增加了统计的功能，直观展现仓库运作情况。库区管理模块具体的功能列表如表 4.3 所示：

表 4.3 库区管理模块功能表

模块名称	模块包含功能
库区管理模块	新建库区
	修改库区信息
	关闭库区
	服务实绩
	库区统计

新建库区：在现有库区运作负荷较大的情况下，仓库管理员可以用此功能建立新的库区，在新建库区时需要给库区命名并且填写库区的地址。

修改库区信息：该功能用来修改系统中已经存在库区的相关信息。

关闭库区：该功能用来关闭系统中已经存在的库区，并且删除库区相关信息。

服务实绩：仓库管理员使用该功能查看所有已经服务过的信息，即所有商品车出库的信息，包括司机的信息、出库车辆的信息以及对应的库区信息。

库区统计：该部分主要利用图表来统计库区相关的信息，利用折线图统计不同库区的商品车出库情况，利用柱状图统计每天的服务情况。

4.1.2 非核心业务子系统的功能设计

非核心业务子系统在整车出库管理系统中起着辅助管理的作用。根据功能的不同，主要将其分为系统管理、用户管理和留言公告三个模块。

（1）系统管理模块贯穿于整个系统之中，主要用来进行系统中每种角色功能和权限的配置。任何功能模块都必须以系统管理模块作为基础。系统管理模块具体的功能列表如表 4.4 所示：

表 4.4 系统管理模块功能表

模块名称	模块包含功能
系统管理模块	角色功能管理
	越权拦截管理

角色功能管理：由系统的需求分析可知，系统共有三个不同的实际角色，每个角色都分配有不同的功能。系统设计的目标是让不同的角色共同使用一个系统。所以当不同角色使用账号登录系统时，主页面显示的导航菜单会因为角色分配功能的不同而改变，角色功能管理就是起到了这个作用。

越权拦截管理：该功能是为系统的安全需求而设计的。由于系统是一个 Web 项目，每个功能都对应着一个页面，每个页面都对应着一个地址。当某个角色通过某种手段获得了其他角色功能的地址后，在试图访问该地址时，系统就会进行拦截，阻止此类非法访问的行为。

（2）用户管理模块主要实现对系统中所有用户的管理，主要管理用户的登录信息、用户的角色功能以及用户的个人信息。用户管理模块具体的功能列表如表 4.5 所示：

表 4.5 用户管理模块功能表

模块名称	模块包含功能
用户管理模块	用户登录
	用户注销
	修改密码
	新建用户
	修改用户信息
	查看用户信息
	删除用户

用户登录：用户在浏览器中进入登录页面后，输入正确的账号与密码即可成功登录系统，使用系统中的功能。

用户注销：用户在使用完毕后，点击注销便会退出当前已登陆账号，系统自动跳转到登录页面。

修改密码：用户可以对已有账号的密码进行修改，修改时需正确输入原密码。

新建用户：在系统中新建一个用户，新建时需填写相应的登录信息以及用户个人信息，并且选择用户的角色。

修改用户信息：管理员可以修改用户的个人信息以及角色，每次只能修改一个用户的信息，不支持批量修改。

查看用户信息：管理员可以查看系统中所有用户的信息，如果系统中用户较多，支持通过用户的个人信息进行条件查询。

删除用户：管理员可以删除系统中已经存在的用户，当删除用户时，该用户在系统中的

所有使用信息也被删除。

（3）留言公告模块分为留言和公告两个部分。留言部分可以实现司机与管理员之间的交流，公告部分主要由管理员向司机发布仓库的相关通知。留言公告模块具体的功能列表如表4.6 所示：

表 4.6 留言公告模块功能表

模块名称	模块包含功能
留言公告模块	司机留言
	管理员回复
	查看留言
	查看公告
	发布公告

司机留言：司机进入留言页面进行留言，可以对仓库管理员的取车服务进行评价，可以向管理员提出问题，也可以为仓库的管理提出宝贵意见。

管理员回复：管理员可以对未回复的司机留言进行回复。

查看留言：管理员可以查看所有留言信息以及相应的回复信息，也可以查看到留言人的个人信息。查看留言分为查看已回复留言和查看未回复留言两个页面。

查看公告：司机可以通过该功能查看管理员发布的所有公告。

发布公告：管理员进入发布公告页面后输入公告的标题和内容即可发布该公告。

4.2 关键问题及重难点问题初步解决方案

本节主要介绍在各模块功能设计中的一些关键问题和重难点问题，并对于这些问题提出初步的解决方案，具体的设计细节及实现将在下一章中论述。

4.2.1 系统关键问题及其初步解决方案

在系统各个功能模块的设计中，有一些关键性的问题，对于这些关键问题必须在概要设计的时候给出其初步的解决方案，这样才能为之后的详细设计及实现做好铺垫。系统中有如下关键问题：

（1）在系统中有时需要在不进行页面跳转的前提下通过服务器进行数据处理，例如：在预约页面初始化时需要加载系统中的所有时窗；司机在选择完时窗后，系统要根据该时窗是否已经预约满来判断司机所选时窗是否有效。这些情况都需要通过服务器进行验证处理，但是如果每次验证都要等到用户提交信息到服务器之后，则会大大降低系统处理业务的效率。解决这个问题的办法就是采用 Ajax 技术，该技术可以实现在不更新地址的情况下与后台进行

少量的数据交换，增加系统的流畅度。

（2）司机在取完票后，票号便会自动分配给各个大厅管理员来进行业务办理。最好的分配方案如下：将大厅管理员编号，每次系统新增一个票号就按编号的顺序依次分配给每一个大厅管理员，一轮分配完成后，将新的票号再次分配到第一个大厅管理员处排队等待，以此类推。这样可以均衡每个大厅管理员的工作量。为此，需要单独编写一个实现轮流指派的工具类，并在后端代码中调用该工具类中的方法。

（3）越权拦截管理是整个系统中最关键的问题。要实现该功能首先要对用户的请求地址进行拦截，这里可以使用 SpringMVC 中自带的拦截器，但是使用前需要进行相关配置。配置好拦截器后系统对用户的请求地址进行第一轮拦截，该轮拦截主要是针对司机这个角色，因为司机只有查看系统中的用户、时窗、公告和库区的权利而没有对其添加或修改的权利，所以当系统发现角色为司机的用户试图访问含有添加与修改功能的页面时，就要拦截该请求。如果司机没有访问上述含有添加或修改功能的页面或者登录系统中的角色不为司机时，系统需要对用户的所有访问地址进行第二轮拦截。该轮拦截为通用拦截，如果用户访问不属于自己角色功能的地址，就会在这轮被拦截。要实现这种拦截，系统需要判断用户试图访问页面的地址是否包含在该用户所属角色合法访问的范围内，若不包含则将其拦截。

4.2.2 重难点问题及其初步解决方案

系统的设计目标是要让三种角色的用户共同使用。但是三种角色的功能不尽相同，所以在用户使用账号密码登录系统后，主页面的导航菜单就会不同。为实现此效果，下面两种方案相对比较容易想到：

（1）为每种角色设计一个主页面，并为每种角色的主页面设计不同的导航菜单。因为系统中只有三个实际角色，所以该方案可以被采用，只是要多编写两个主页面，但如果需要设计一个很多种角色用户共同使用的系统，则该方案会使工作量明显增加。而且采用这种方案一旦需要修改或者增加某种角色的功能，必须改动页面的代码，这也给后期维护带来了很大的困难。

（2）把系统中所有角色的功能都放置在一个主页面上的导航菜单中，用户如果点击不属于自己角色功能的导航菜单，越权拦截管理功能便会起作用，拦截该访问地址。这种方案看似很简单，但是也有一个致命的缺陷：由于将所有角色的功能菜单全部显示在页面上，用户使用时触发到越权拦截机制的机会就大大增加；而且用户的访问如果经常被拦截，也会影响用户体验。

在对上述两个方案的优势与缺陷进行分析的基础上，本人总结出了第三种方案：只为该系统编写一个主页面，将系统中所有角色功能的页面地址存放到数据库，在页面上动态分配每种角色主页面的导航菜单。这样可以达到每种角色主页面的导航菜单只包含该角色所有功能的效果，减少开发的工作量，而且在后期维护时，可以根据业务的需要在不修改页面代码的前提下为每种角色动态分配其他的功能，实现了对前两种方案的扬长避短。第三种方案的具体设计细节将在系统详细设计与实现部分陈述。

4.3 系统结构

在系统的概要设计阶段，需要确定系统的结构。确定一个系统的结构需要从多个层面入手。本节将从系统的设计架构、系统分层以及系统的包结构三方面来介绍。

系统在设计架构上采用了 B/S 架构，即浏览器/服务器架构。在这种架构下，用户通过浏览器输入地址来进入系统，大部分的业务处理任务放在服务器端，浏览器端只进行少部分的业务处理[30]。使用 B/S 架构可以有以下好处：

（1）可以省去客户端的开发，减少了开发系统的工作量；在后期维护中，由于所有的客户端均为浏览器，所以只需要对服务器进行维护，降低了维护的难度。

（2）B/S 架构中的各部分组件相对独立，因此采用该架构的系统具有非常好的重用性。

（3）B/S 架构建立在广域网上，不需要使用特定的硬件来提供专门的网络环境。

在对系统架构设计完成的基础上，使用 MVC 模式对系统进行分层，将整个系统分成表现层、业务逻辑层和数据访问层。系统各层之间相互独立，在对于各层的设计中可以不用考虑其他层。每层的分工十分明确：表现层要求将页面设计得美观、简洁，符合用户需求，增强用户体验；业务层主要负责对业务进行处理，实现相应的功能；数据访问层不包含任何对业务的处理，只负责与数据库中的实体进行交互，即对数据库中的表进行增删改查。

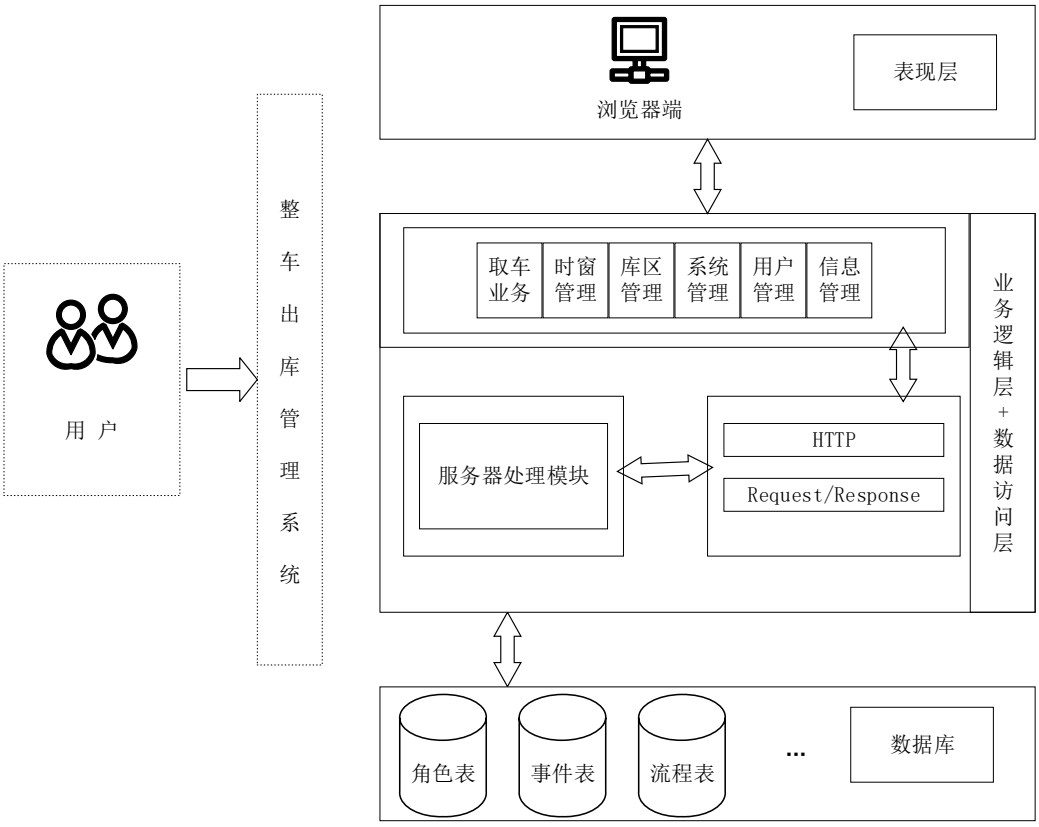


图 4.2 系统层次结构图

系统层次结构如图 4.2 所示：最顶层的是表现层，表现层向用户展现了系统中的页面，最初展现在用户面前的是登录页面，当用户登录后便会进入包含着许多业务的主页面；当用户操作某个业务时，系统就会向服务器发出请求，服务器便会调用业务逻辑层来对该请求所代表的业务进行处理，处理完成后服务器将处理的结果返回给表现层中的页面；在业务逻辑层进行处理的过程中，常常需要底层数据库的信息，这时数据访问层便会从数据库获取相应的信息并将这些信息传送给业务逻辑层使用。

系统的包结构依据系统的层次结构而设计。其中 pojo 包主要存放系统所有的实体类，这些类与数据库中的表一一对应；controller 包主要存放系统中的控制器；service 包主要存放业务逻辑的接口与其实现类；dao 包主要存放数据访问层的相关接口，由于使用了 Mybatis 作为系统的 ORM 框架，所以 DAO 层中接口的实现类由该框架根据 Mapper 包中的配置文件自动生成；前端的页面、js 代码以及 CSS 样式等放在系统根目录下的 resource 包中。系统中各种包之间的调用关系如图 4.3 所示：

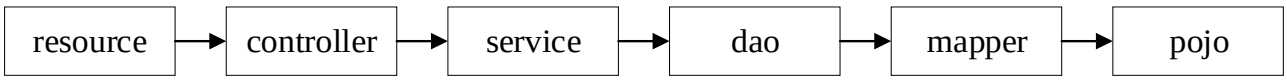


图 4.3 系统包调用图

4.4 系统数据库的设计

在明确了系统的功能模块以及整体结构之后，就需要对系统的数据库进行设计。数据库是整个系统的根基，数据库的设计最少需要占用整个系统开发的三成以上的时间。若在确定系统需求后就想急功近利，直接进入编码阶段，最后只能是得不偿失：轻则需要对数据库进行大量修改，重则导致项目开发无法进行。在设计数据库时要遵守“三少”的原则，即在实现系统设计目标的前提下使数据库中的表、每张表中的主键数量以及每张表中的字段数量尽可能地少。这样可以避免在系统编码阶段中因数据库的修改而导致开发过程的中断，使系统的开发更加流畅。

4.4.1 数据库 E-R 图的设计

数据库的 E-R 图即实体-联系图，E-R 图能够很好地反映出数据库中的实体、实体的属性和各实体之间的联系，是对系统数据库的总体概括[31]。E-R 图有实体、属性和关系这三个要素，分别用矩形、椭圆形和菱形来表示，图 4.4 即为系统数据库的整体 E-R 图，由于实体中的属性数量较多，因此仅画出实体中的部分属性：

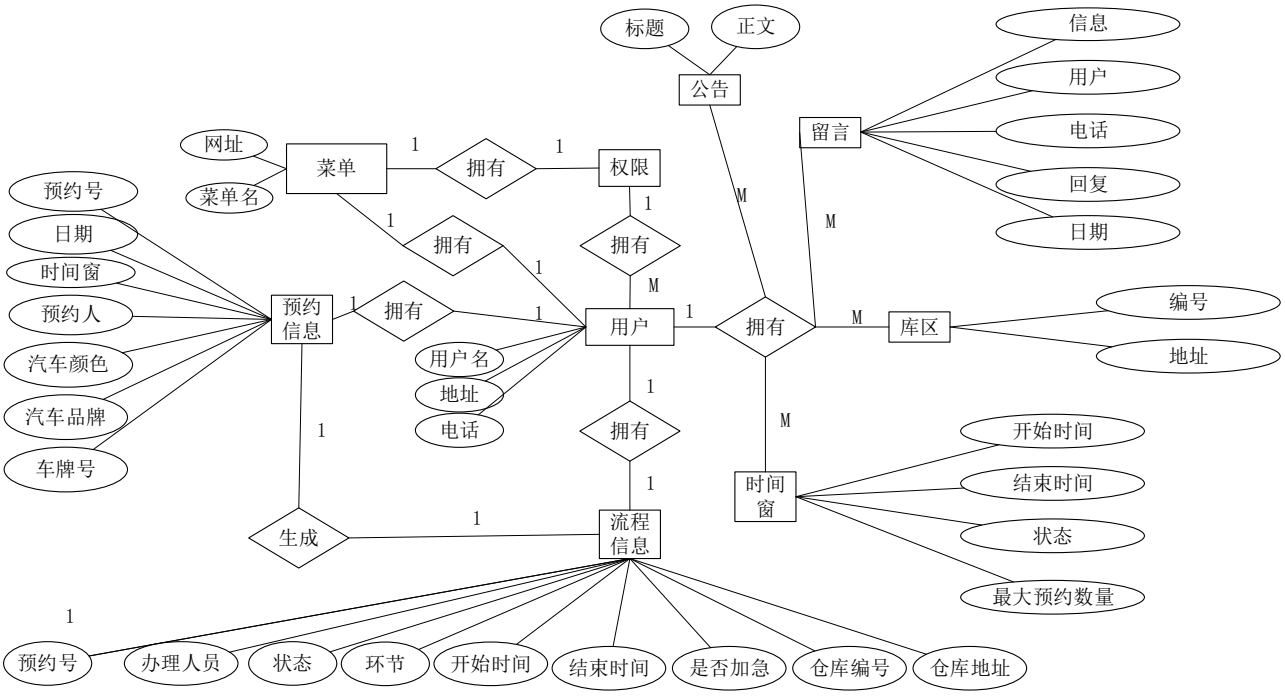


图 4.4 数据库整体 E-R 图

由 E-R 图可以看出，系统包含了 9 个实体，分别为：菜单、权限、预约信息、用户、流程信息、公告、留言、库区和时间窗。其中，用户与公告、留言、库区和时间窗这四个实体间都是是一对多的关系，即一个用户可以对多个公告、留言、库区和时间窗进行管理；用户

与预约信息、流程信息和菜单是一对一的关系；用户与权限是多对一的关系，即每种权限对应着多个用户；预约信息与流程信息也是一对一的关系，即通过一个预约信息只能生成一个流程信息。

4.4.2 数据表的设计

根据系统功能模块的设计以及数据库的 E-R 图可以设计出详细的数据表，下面分别介绍每张表中具体字段的信息以及每张表的作用。

（1）库区表（demo_depot）

该表主要存储库区相关的信息，主要用于库区管理模块。库区表中所有字段信息如表 4.7 所示：

表 4.7 库区表

字段名	数据类型	数据长度	允许空值	约束类型	备注
id	int	11	否	主键	库区 id
number	int	11	是		库区编号
address	varchar	255	否		库区地址
brand	varchar	255	是		商品车品牌
color	varchar	255	是		商品车颜色
type	varchar	255	是		商品车类型
model	varchar	255	是		商品车型号

（2）留言表（demo_message）

留言表主要存储用户留言信息，留言功能需要显示发表该留言的用户的信息以及该留言是否被回复。留言表中所有字段信息如表 4.8 所示：

表 4.8 留言表

字段名	数据类型	数据长度	允许空值	约束类型	备注
id	int	20	否	主键	留言 id
message	varchar	255	是		留言
username	varchar	255	是		留言用户
isreply	int	1	是		是否回复
phone	varchar	255	是		电话
reply_message	varchar	255	是		回复信息
createtime	Date		是		留言日期

（3）预约表（demo_event）

预约表存放了司机预约时所填写的信息，其中 timesid 为时窗号，它是时窗表的主键，在司机选择时窗时通过该字段与预约表联结。预约表中所有字段信息如表 4.9 所示：

表 4.9 预约表

字段名	数据类型	数据长度	允许空值	约束类型	备注
event_id	int	20	否	主键	预约 id
startdate	date		是		预约日期
username	varchar	255	是		预约人
timesid	int	11	是	外键	时窗 id
phone	varchar	255	是		电话
coler	varchar	255	是		颜色
brand	varchar	255	是		牌子
carbrand	varchar	255	是		车牌号

（4）公告表（demo_notice）

公告表存放与公告功能有关的信息，因为公告内容比一般的字段要长，所以应该设置更长的数据长度。公告表中所有字段信息如表 4.10 所示：

表 4.10 公告表

字段名	数据类型	数据长度	允许空值	约束类型	备注
id	int	20	否	主键	公告 id
title	varchar	30	是		标题
noticeinfo	varchar	400	是		公告信息

（5）流程表（demo_process）

流程表是比较重要的一张表，该表存放着与取车业务流程相关的信息。通过 event_id 与 userid 分别和预约表、用户表联结；state 为票号的状态，1 代表受理，2 代表作废，3 代表结束；link 为票号正在经历的环节，1 代表预约环节，2 代表取票环节，3 代表取车服务环节，4 代表业务结束环节。流程表中所有字段信息如表 4.11 所示：

表 4.11 流程表

字段名	数据类型	数据长度	允许空值	约束类型	备注
id	varchar	32	否	主键	流程 id
event_id	int	11	否	外键	预约 id
userid	int	11	否	外键	收件人 id
username	varchar	255	否		收件人
state	int	11	是		状态
link	int	11	是		环节
createtime	date		是		创建时间
endtime	date		是		结束时间
isvip	int	11	是		是否加急
address	varchar	255	是		库区地址
number	varchar	255	是		库区

（6）权限表（demo_role）

权限表中的字段虽然较少，但是该表是非常重要的一张数据表，通过 role 字段和 treeid

字段分别与用户表和功能树表相关联。通过该表来实现对系统中每种角色功能的动态分配。权限表中所有字段信息如表 4.12 所示：

表 4.12 权限表

字段名	数据类型	数据长度	允许空值	约束类型	备注
role	int	11	否	外键	权限
treeid	int	11	否	外键	树 id

（7）时窗表（demo_times）

时窗表中存储着与时窗管理相关的信息。其中 starttime 和 endtime 分别表示时窗中的开始时间与结束时间，在系统中对开始时间和结束时间做了处理，保证开始时间与结束时间都为整点，并且相隔一个小时；state 字段中 1 代表时窗开启，表示该时窗有效；2 代表时窗关闭，表示禁用该时窗，但不会从系统中删除该时窗的信息。时窗表中所有字段信息如表 4.13 所示：

表 4.13 时窗表

字段名	数据类型	数据长度	允许空值	约束类型	备注
id	int	20	否	主键	时窗 id
startTime	time		是		开始时间
endTime	time		是		结束时间
state	int	1	是		状态
maxnumber	double	11	是		最大预约能力

（8）功能树表（demo_tree）

功能树表中存储着系统中每种功能的相关信息。其中 treeurl 表示每种功能的页面地址；ulid 表示该功能菜单的代号，liid 表示该功能菜单的上级菜单代号。state 表示状态，1 表示该功能菜单属于一级菜单，2 表示该功能菜单属于二级菜单；style 表示菜单旁边的样式图案。功能树表中所有字段信息如表 4.14 所示：

表 4.14 功能树表

字段名	数据类型	数据长度	允许空值	约束类型	备注
treeid	int	11	否	主键	流程 id
treename	varchar	255	是		目录名称
treeurl	varchar	255	是		目录 url
ulid	int	11	是		菜单 id
liid	int	11	是		上级菜单 id
style	varchar	255	是		样式
state	int	11	是		状态
number	varchar	255	是		库区

（9）用户表(demo_user)

用户表中存放着用户管理模块的相关信息。其中，role 代表用户的角色代号，1 表示超级管理员；2 表示大厅管理员；3 表示司机；4 表示仓库管理员。用户表中所有字段信息如表 4.15 所示：

表 4.15 用户表

字段名	数据类型	数据长度	允许空值	约束类型	备注
userid	int	11	否	主键	用户 id
username	varchar	255	是		用户名
password	varchar	255	是		密码
phone	varchar	255	是		电话
address	varchar	255	是		地址
rolename	varchar	255	是		角色名
role	int	11	否	外键	角色

4.5 本章小结

本章主要对整车出库管理系统的概要设计进行介绍。首先介绍了系统的功能模块设计，将系统分成核心业务子系统与非核心业务子系统，核心业务子系统包含了取车业务、时窗管理和库区管理三个模块，非核心业务子系统包含了系统管理、用户管理和留言公告三个模块；然后就系统功能模块设计中的关键问题和重难点问题提出初步解决方案；接下来对系统的结构设计做出详细介绍；最后列出所有数据表并对相关字段做出解释。

第五章 整车出库管理系统的详细设计与实现

系统的详细设计需要明确系统中每个模块底层的结构和运作流程。本章在概要设计的基础上，通过类图的方式设计每个功能模块的底层结构，对部分具体功能的底层实现进行说明，并展示页面效果。

5.1 系统开发准备

本节主要介绍系统开发前所需要做的一些准备工作，包括开发工具的选择以及开发环境的搭建。

首先介绍系统开发工具。

（1）系统主要使用 sts（Spring Tool Suite）来开发，这款开发工具是在 Eclipse 的基础上，将 Spring 技术集成在其中，使用 sts 工具可以很大程度地减少项目创建的工作量[32]。

（2）系统开发中使用谷歌公司的 Chrome 浏览器，该浏览器具有简洁、快速和不易崩溃等特点[33]。

（3）系统使用 navicat 作为数据库处理工具[34]。

系统的开发环境如表 5.1 所示。

表 5.1 系统开发环境

名称	描述
数据库	MySQL
操作系统	Windows10
开发语言	Java
Web 服务器	Tomcat 8.0

5.2 核心业务子系统的详细设计与实现

核心业务子系统主要包括了取车业务、时窗管理和库区管理三个模块。本节分别对这三个模块中的部分具体功能的详细设计与实现进行介绍。

5.2.1 取车业务的详细设计与实现

取车业务模块实现了司机从预约到取车成功之间的所有环节。该模块是整个系统中关键模块之一，其大致流程如图 5.1 所示：司机首先在预约页面上预约；取车当天在大厅凭借预

约号或者二维码取票；取票成功后票号加入排队队列，并根据是否加急确定排队顺序；接着系统将正在排队的票号依次分配给所有在线的大厅管理员，每个管理员在为一位司机办理完业务后就叫号提示下一位司机前来办理业务；司机听到叫号后前去柜台办理业务，大厅管理员根据司机的运单分配取车的仓库和仓库管理员，并将取车单打印后交给司机；司机根据取车单上的地址到相应的库区取车。

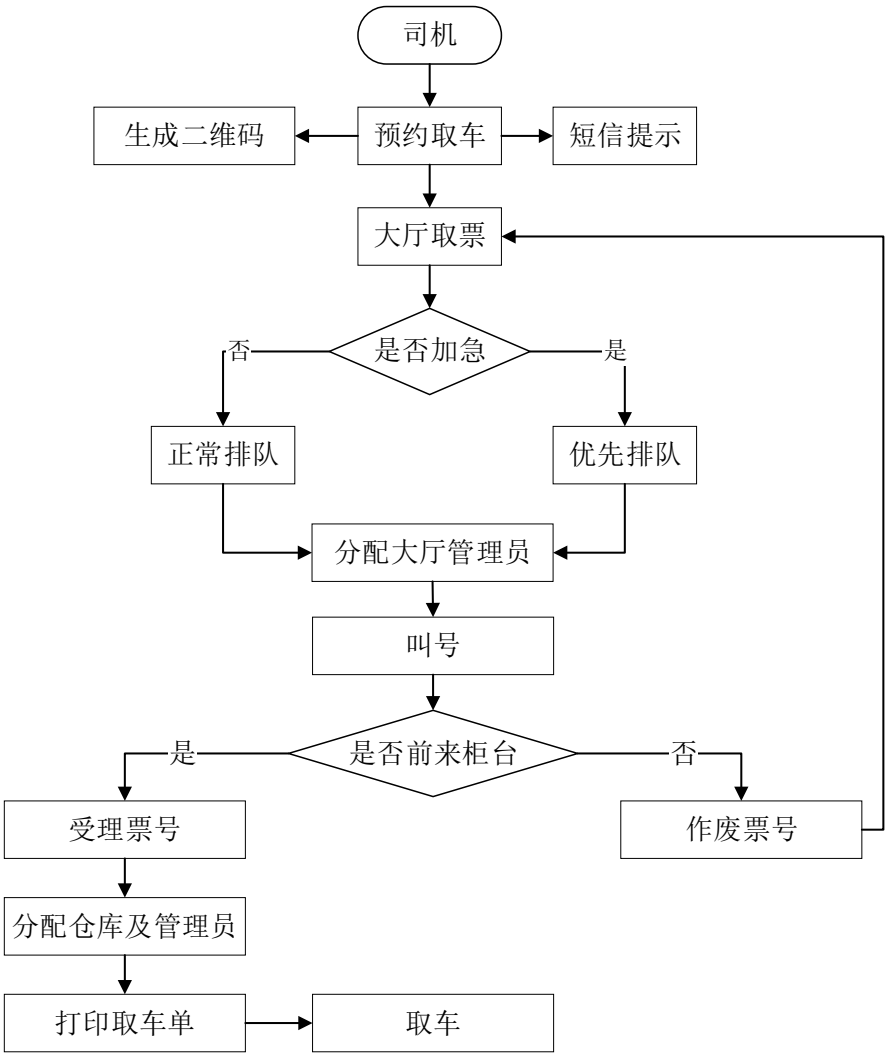


图 5.1 取车业务流程图

开发中将 MVC 模式与系统的三层结构结合起来，在三层结构的基础上融入 MVC 模式中的控制器，即多加入一个控制层[35]。控制层的作用不是处理具体的业务逻辑，而更像是一个“连接器”，将系统的表现层与业务逻辑层和数据访问层连接起来：表现层要实现某种业务时会先向控制层发送请求，SpringMVC 框架会根据请求的地址找到控制层中与该地址相映射的方法，控制层中的方法并不进行业务的处理，而是通过在其中调用业务逻辑层的方法来完成具体的业务逻辑，之后会将所有页面需要的数据信息进行“加工”和“打包”并传回页面使用。业务逻辑层要处理具体的业务则需要对数据库进行访问，该层也不直接与数据库交互，而是通过调用数据访问层（或 DAO 层）接口中的具体方法，由于系统使用了 Mybatis 技术，

将 sql 语句单独写在了 xml 文件中，所以系统会自动根据调用的 DAO 层接口的方法名寻找相应的 sql 语句实现对数据库的访问[36]。

下面依次介绍取车业务模块中所有功能的详细设计以及实现细节。

（1）预约取车

该功能类图的设计如图 5.2 所示，主要实现过程分为页面初始化和页面提交两个部分。

首先初始化二维码和预约号。利用 jQuery 中的插件 qrcode 来生成一个宽度和高度均为 200 的二维码，同时用预约当天的日期加上 4 位随机数作为预约号自动生成在表单中[37]。

接着初始化时窗下拉菜单，这里采用 Ajax 技术：在页面的 js 文件中向后台发送 Ajax 请求；通过请求地址找到控制层中 Reservations_carController 类的 init()方法[38]；接着调用业务逻辑层 Reservations_carService 接口的 init()抽象方法，Reservations_carServiceImpl 类中实现了该抽象方法并完成了相关业务，即找到系统中所有状态为开启的时窗信息；由于时窗信息存储在数据库中，于是在该方法中调用了 DAO 层 demo_timesMapper 接口中的 selectIsState()方法，Mybatis 根据方法名在 mapper 包里的 demo_timesMapper.xml 文件中找到相应的 sql 语句。至此，从页面顶层到数据库底层的信息调用完成，接下来系统要做的就是将从底层数据库获得的数据逐层“加工”并返回，直到控制层将所有有用的数据传回页面，页面通过相关前端技术将所有时窗信息加载到下拉菜单中。使用 Ajax 技术后使得上述过程不需要经过页面的跳转，页面采用异步传输的方式获取信息。

在司机选择具体时窗时，系统需要判断司机所选时窗是否已经预约满。同样使用 Ajax 技术获取系统中已经预约该时窗的人数和该时窗的最大预约能力并将两者进行比较，若发现该时窗已经预约满，则弹出警告框提示司机重新选择时窗。

还需要注意的一点是，由于预约成功后系统会发短信提醒，所以在司机输入手机号码后，需要对该号码做验证，具体方法是：使用正则表达式表示出所有合法的手机号，再将司机输入的手机号与之对比并做出判断。

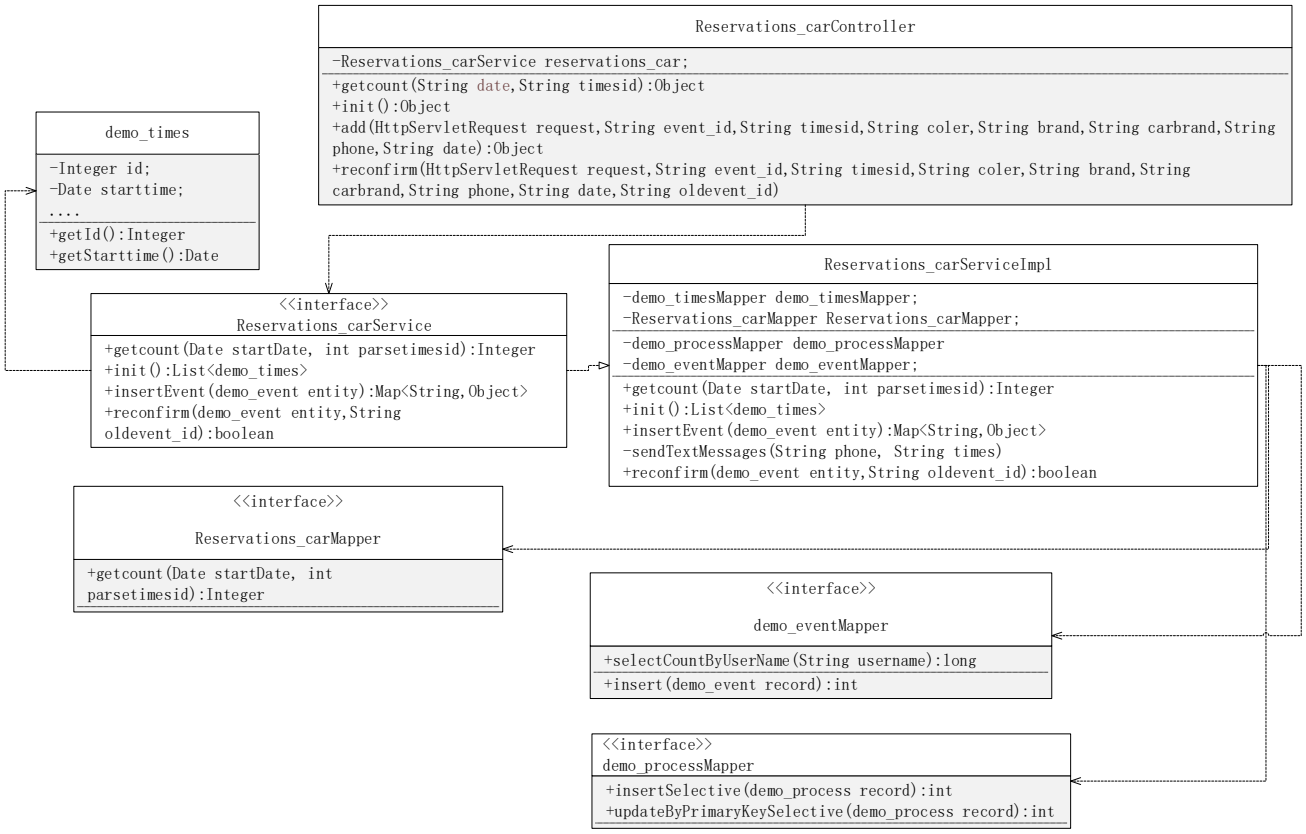


图 5.2 预约取车类图

司机点击“立即提交”按钮后，系统会调用控制层 Reservations_carController 类中的 add()方法，该方法作用是将用户填写的预约信息存入预约表中，并且将当前时间作为该预约信息的创建时间存入 startdate 字段。为了防止司机多次预约，系统还需要在司机提交预约信息时判断该司机是否已经预约过，通过调用 reconfirmed()方法来判断司机提交的预约信息是否在预约表中有与之相同的记录，如果有则弹出警告“您已经预约，是否重新预约”。预约页面的实现效果如图 5.3 所示：

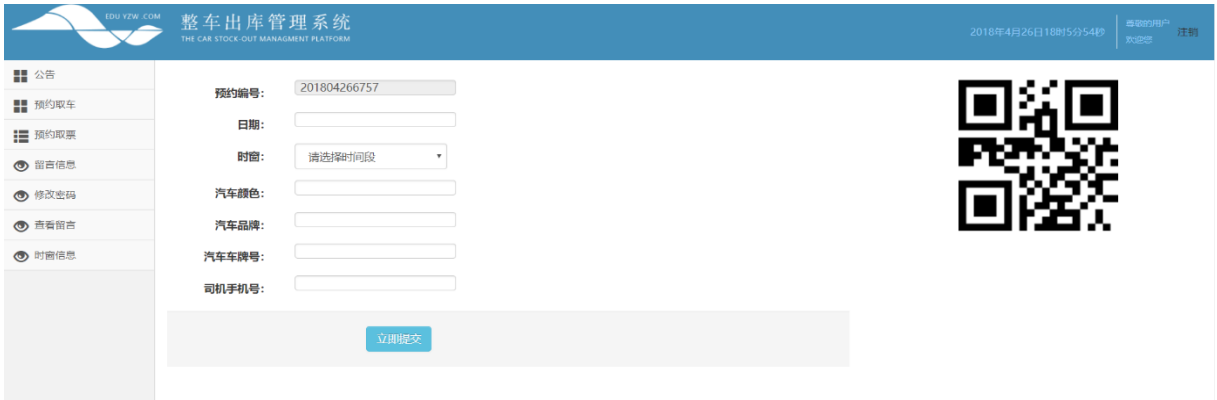


图 5.3 预约页面

（2）短信提醒

该部分使用了阿里云的短信服务[39]。首先需要在阿里云上开通短信服务；然后将阿里云提供的工具类 SmsDemo 下载到项目中，并按要求进行相关配置；最后在业务逻辑层中找到

Reservations_carServiceImpl 类中的 add()方法并在其中调用 sendTextMessages(phone,times)方法即可， phone 表示要发送的手机号，times 表示司机预约的日期[40]。

（3）页面取票

页面取票功能的类图如图 5.4 所示，需要实现的效果是当司机在取票页面输入预约号时，页面上的其他预约信息便会自动填入。为此，当司机输入预约号后，页面立即发送请求到控制层找到 Reservations_carController 类中的 getEvent()方法，该方法通过调用业务逻辑层获取与当前预约号相关的预约信息并返回给页面。页面将其中的汽车颜色、汽车品牌、司机车牌号和司机手机号等信息通过前端技术写入对应的表单中。

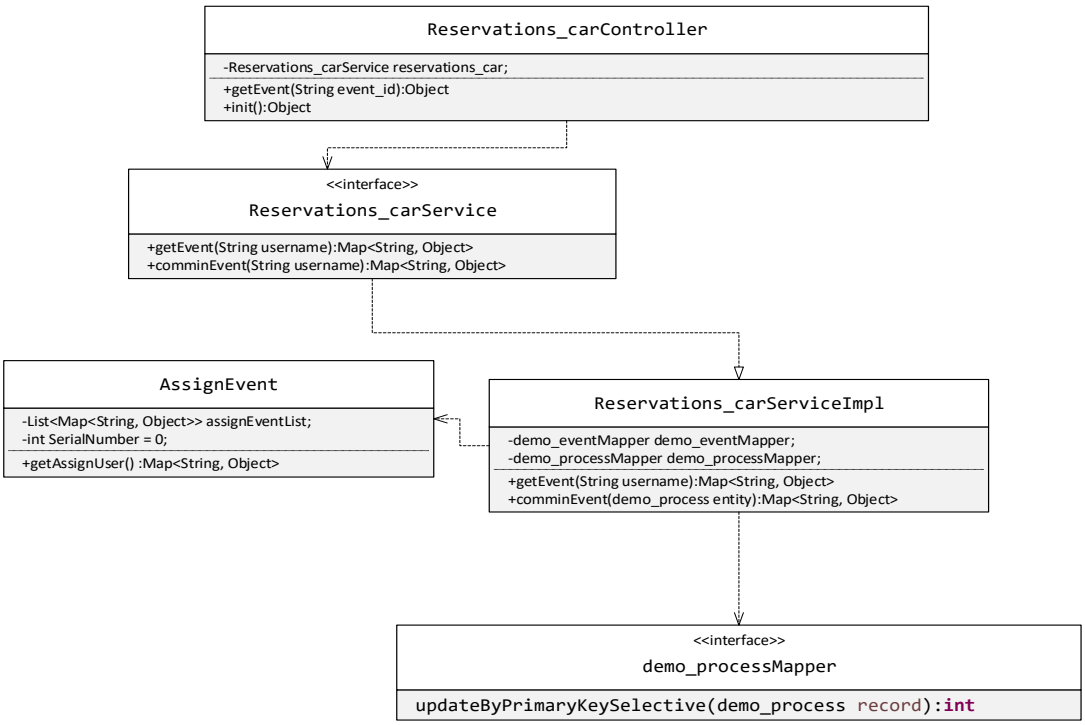


图 5.4 页面取票类图

为了使取车业务更加灵活，在取票时提供了加急的功能，司机只需要在取票时勾选加急单选框，系统就可以对该票号进行加急处理，将其移动到当前排队队列的第一个，如果队列中不止一个需要加急的票号，则判断这些加急票号的取票时间，取票时间越早票号排队位置越靠前。

司机点击“取票”后，页面向后端发送请求，通过请求地址找到 Reservations_carController 类中的 comminEvent()方法，在该方法中调用了业务逻辑层 Reservations_carService 接口的 comminEvent()方法来完成票号分配的业务。这里需要写一个 AssignEvent 类作为工具类，通过该类的 getAssignUser()方法实现概要设计中提到的分配方案。该方法的设计思路：定义 SerialNumber 和 assignEventList 这两个变量，其中 SerialNumber 表示系统中正在排队票号的序号，assignEventList 表示服务人员，当排队票号

的序号大于或等于服务人员的数量时，将序号置零。举个例子：假设系统中有三个大厅管理员提供业务办理，序号分别为 0-2，现在系统中共有 4 位司机排队，序号分别为 0-3，系统先将序号为 0-2 的三位司机分别分配给序号为 0-2 的大厅管理员，当系统对序号为 3 的司机进行分配时，由于其序号满足了大于或等于大厅管理员数量的条件，所以将该票号的序号置零，即将序号为 3 的司机分配给序号为 0 的大厅管理员，以此类推。该方法的具体代码如表 5.2 所示：

表 5.2 票号分配方法代码表

```
public Map<String, Object> getAssignUser() {
        Map<String, Object> map = new HashMap<>();
        if (assignEventList == null||assignEventList.size()==0||assignEventList.isEmpty()) {
            userService.getUserData(2);
            if (SerialNumber >= assignEventList.size()) {
                SerialNumber = 0;
                map = assignEventList.get(SerialNumber);
                SerialNumber = SerialNumber + 1;
            } else {
                map = assignEventList.get(SerialNumber);
                SerialNumber = SerialNumber + 1;
            }
        } else {
            if (SerialNumber >= assignEventList.size()) {
                SerialNumber = 0;
                map = assignEventList.get(SerialNumber);
                SerialNumber = SerialNumber + 1;
            } else {
                map = assignEventList.get(SerialNumber);
                SerialNumber = SerialNumber + 1;
            }
        }
        return map;
    }
```

在对系统中排队的票号进行分配后，系统便将取票的信息添加到流程表中，并将流程表中的 link 字段置为 2，代表流程进行到取票环节。此时每个大厅管理员都可以看到需要服务票号的排队情况，具体效果如图 5.5 所示：

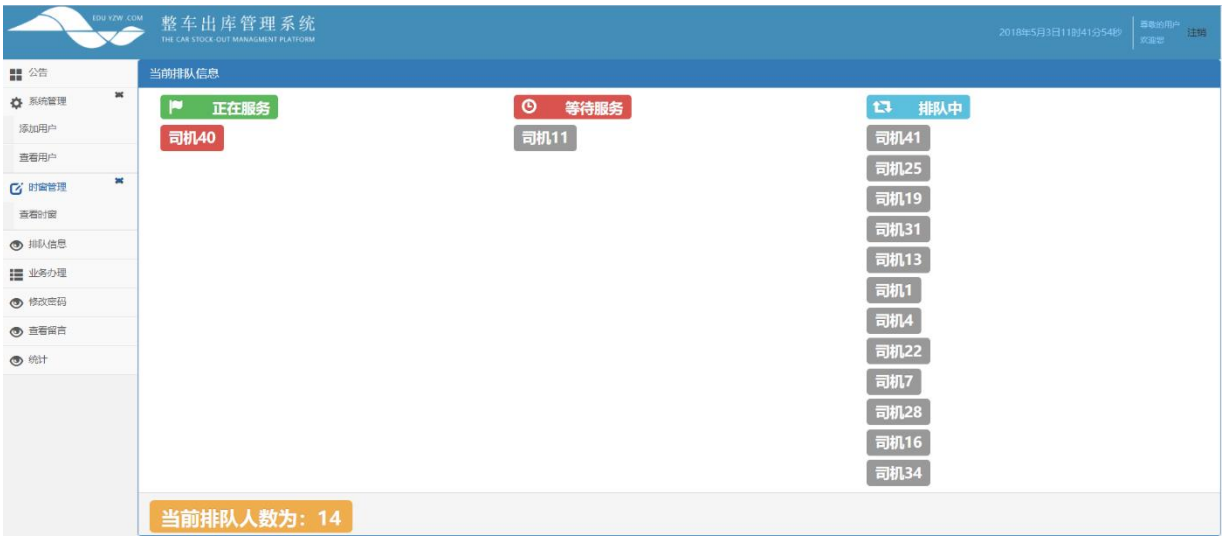


图 5.5 排队信息页面

（4）业务办理

大厅管理员进入业务办理页面后，待办理的票号会全部出现在页面上，管理员点击“受理”按钮时，业务办理页面通过 js 文件中的代码控制其跳转到业务信息页面上；管理员点击“作废”按钮时，系统根据请求地址找到控制层 QueuingController 类中的 Tovoid()方法，该方法中调用了业务逻辑层的 Reservations_carService 接口，然后找到该接口实现类中的 Tovoid()方法将流程表中当前票号记录的 state 字段设置为 2，代表该票号处于作废状态，并将 link 字段置为 4，代表该票号结束所有流程。此时票号作废，如果需要办理业务则需重新取票。

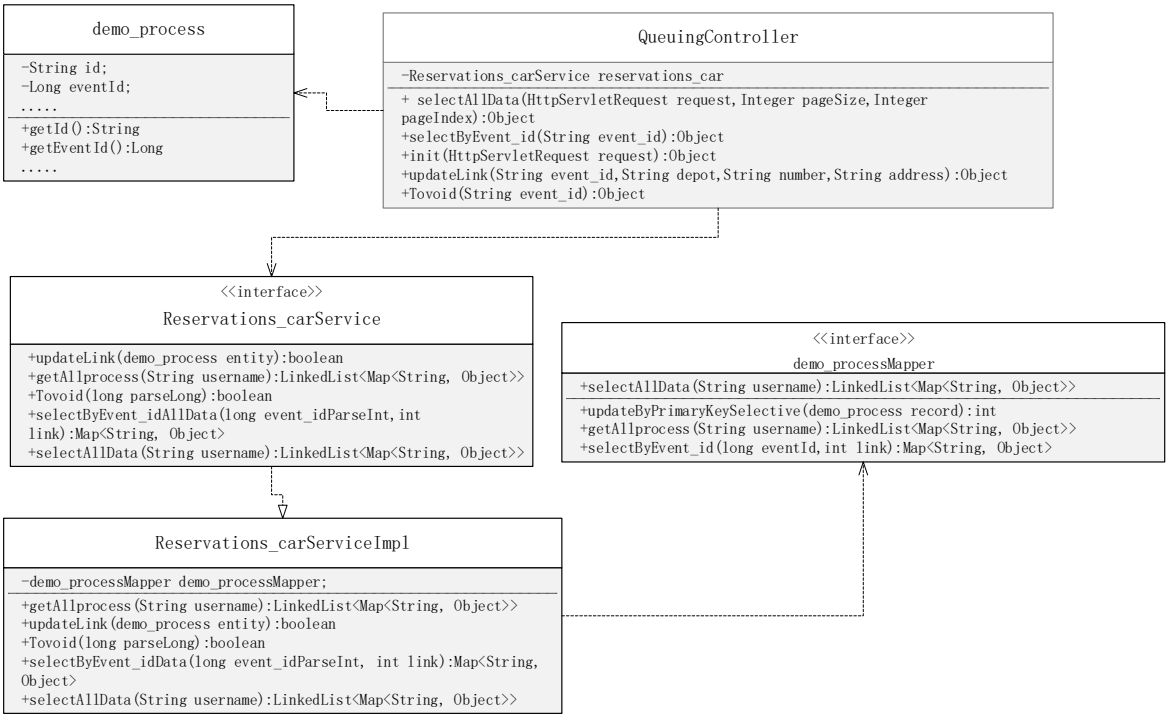


图 5.6 业务办理类图

业务办理功能的类图如图 5.6 所示，当大厅管理员选择受理票号时，系统跳转到业务信息页面，该页面初始化时通过调用 QueuingController 类中的 selectByEvent_id（String event_id）方法来获取当前受理票号的预约信息，并通过前端技术让相关信息显示在页面表单中；接下来系统会调用 QueuingController 类中的 updateLink()方法将流程表中对应记录的 link 字段置为 3，代表进入取车服务的环节，并将分配的仓库管理员以及库区信息更新到流程表对应记录的相应字段中。业务办理与业务信息的页面效果分别如图 5.7 和图 5.8 所示：

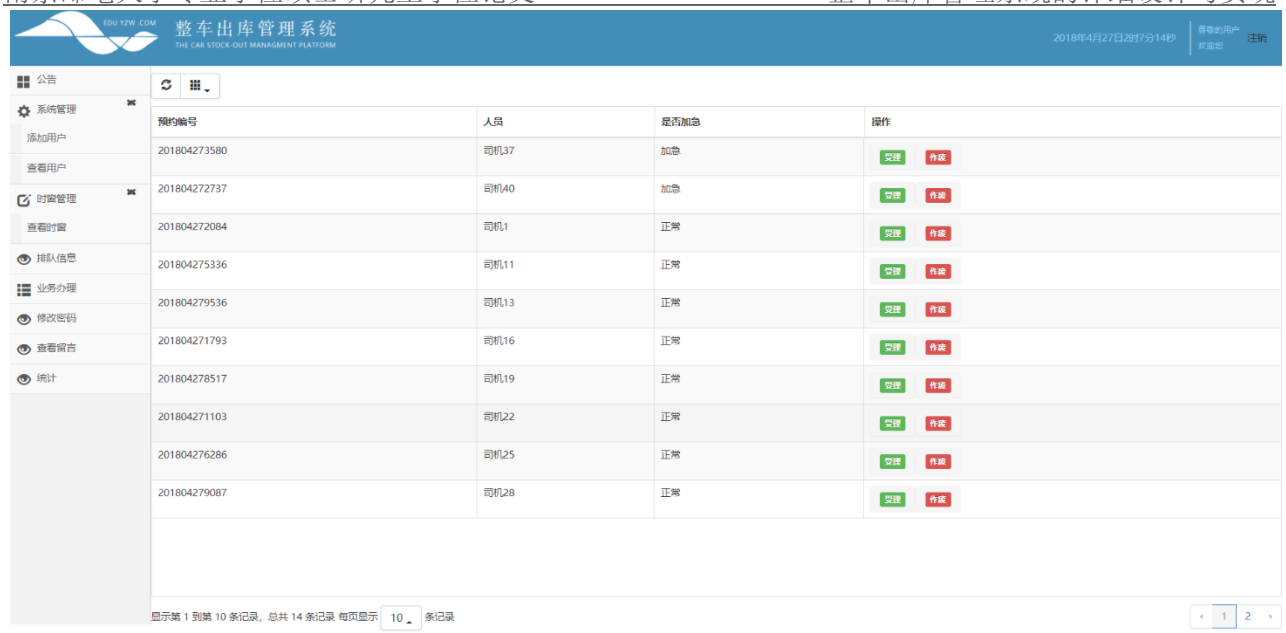


图 5.8 业务办理页面

图 5.9 业务信息页面

（5）打印取车单

当大厅管理员在业务信息页面点击“发送”时，系统会自动跳转到打印取车单的页面。因为打印的内容需要包含流程表、预约表以及库区表中的一些信息， 所以在初始化打印页面时会向控制层发送请求，根据请求地址找到 QueuingController 类中的 selectByEvent_idData（String event_id）方法，参数 event_id 是从前端获取的当前正在办理业务的司机预约号。在该方法中调用业务逻辑层中 Reservations_carService 接口的 selectByEvent_idData(long event_idParseInt,int link)方法，并将预约号 event_id 转换成长整数后传入 event_idParseInt 参数中，将整数 3 传入 link 参数中，则该方法会通过 DAO 层的 sql 语句获取预约号为 event_id 并且处于取车环节的票号信息。查询到这些信息后，控制层将这些信息放入 map 集合中传回页

面。页面在 js 文件中用这些获取到的信息生成需要打印的内容，并调用 windows.print()方法打印页面。打印取车单的页面效果如图 5.10 所示：

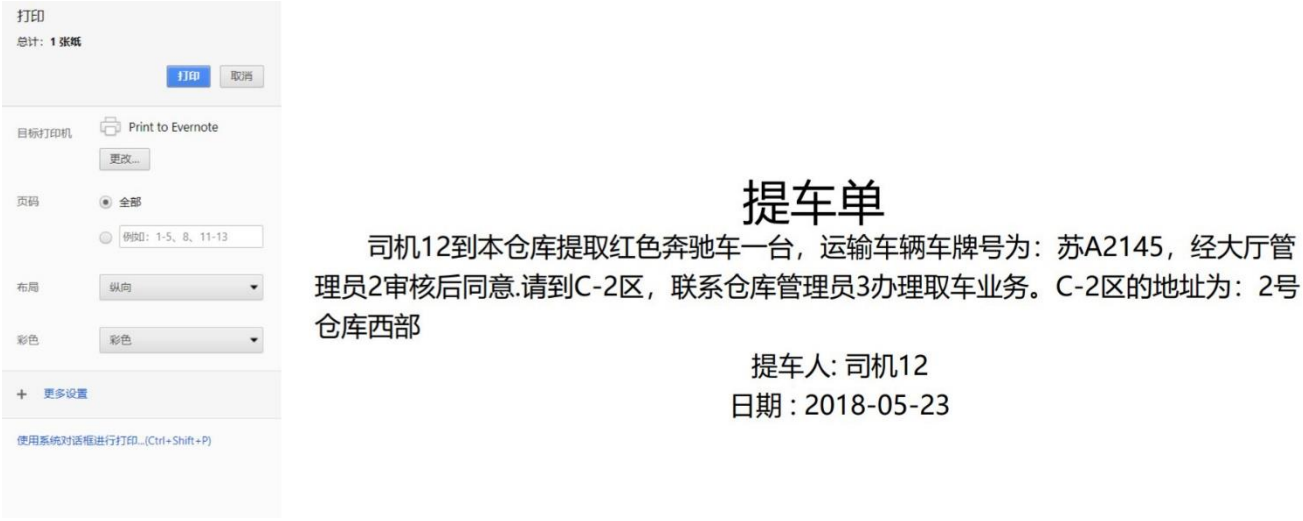


图 5.10 打印取车单页面

司机在拿到取车单后，按照上面提示的信息找到相应的库区即可进行取车服务。取车完成后，仓库管理员点击取车服务页面相应票号记录中的“取车”按钮，系统将该票号流程表中的 link 字段置为 4，代表该票号结束所有流程。

5.2.2 时窗管理的详细设计与实现

时窗管理模块主要功能有新建时窗、修改最大预约、删除时窗和查看已有时窗。本节仅对新建时窗功能的详细设计与实现中两个关键问题的解决方案做介绍。

（1）新建时窗时需要填入时窗的开始时间和结束时间，如果让管理员人工填入时间，则可能会因为输入格式错误而导致系统无法识别。因此在系统中使用了 laydate 时间日历插件，这样管理员可以方便地选择时间，使系统设计更加人性化。具体方法是在页面上引入 laydate.js 文件，接着在添加时窗页面的 js 代码中调用 render()方法，并对该方法中的元素进行配置：type 代表数据类型，设置为 time；istime 代表是否必须填入时间，设置为 true；min 和 max 分别代表设置时间的最小、最大值；btns 代表需要显示的其他功能按钮，例如 clear——清空按钮、confirm——确认按钮。该插件的效果图如图 5.11 所示。

（2）为了管理的方便，需要对管理员选择的时间进行处理。总共需要进行三次处理：第一次判断管理员选择的结束时间是否小于开始时间，如果小于则弹出警告框提醒“开始时间不能大于结束时间”；第二次对时间进行取整处理，在系统的概要设计中规定所选时间必须为整点，而管理员选择的时间是精确到秒的，所以首先判断该时间的分钟数是否超过 30，若超过则将该时间的小时数加 1 并将分钟和秒用“:00:00”替换，如果分钟数没有超过 30 则直接替换；

由于系统概要设计中还规定了时窗的长度只能为一小时，所以第三次处理需要判断用户选择的结束时间与开始时间的小时数相差是否等于 1（这里用户选择的时间是已经经过两次处理的时间），如果不等于 1 则弹出警告框提醒“所选时间跨度过大”。

图 5.11 laydate 插件效果图

5.2.3 库区管理的详细设计与实现

库区管理主要功能有新建库区、关闭库区、修改库区信息、查看库区与库区统计。下面对部分功能的详细设计与实现进行说明。

新建库区、关闭库区、修改库区和查看库区这四个功能都是对库区信息的管理，其类图如图 5.12 所示：

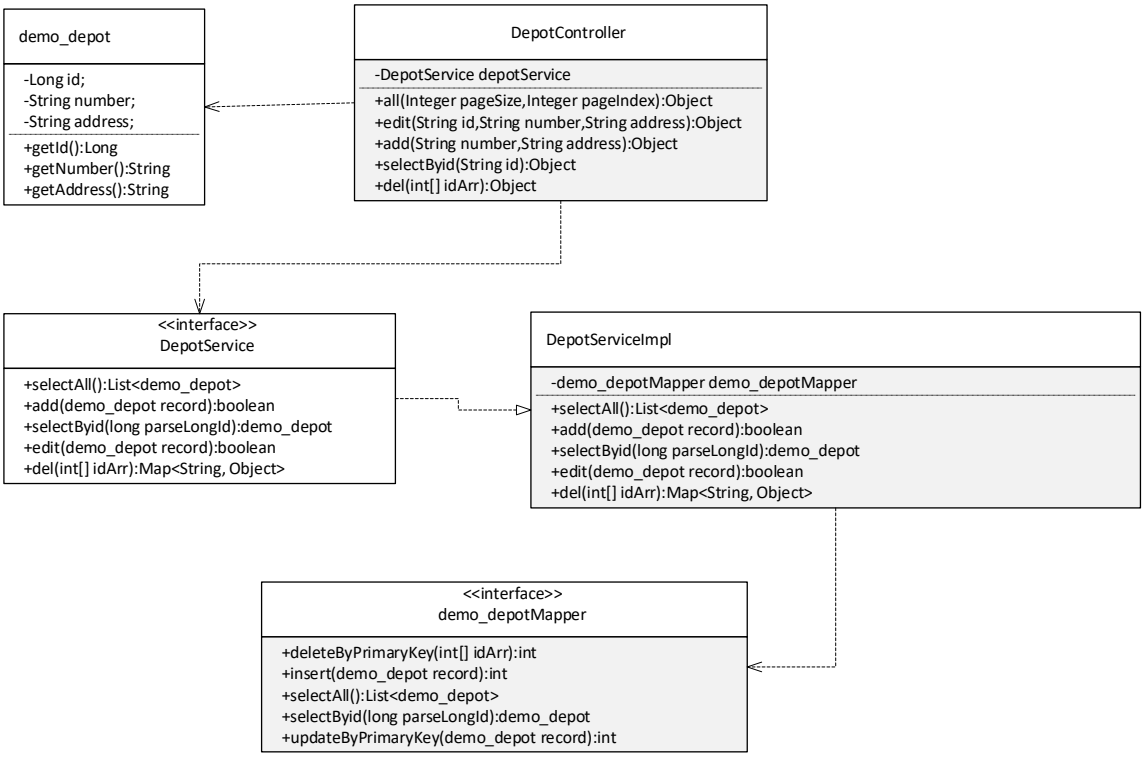


图 5.12 库区信息管理类图

由于各个类中方法过多，所以图中没有将各个类的全部方法都列出。下面分别介绍各个类中方法的作用：

（1）demo_depot 为库区的实体类，getId()、getNumber()和 getAddress()分别是库区 id、库区编号和库区地址这三个属性的 getter 方法，用于获取这三个属性。

（2）DepotController 为控制层的类，其中方法名中含有 all、edit、add、selectById 和 del 分别代表查找所有库区信息、修改库区信息、新增库区、查找对应 id 号的库区信息以及删除库区。

（3）DepotService 是业务逻辑层的接口，其方法名与 DepotController 类大致相同，但是方法中传入的参数不同。

（4）DepotServiceImpl 类是 DepotService 接口的实现类，在该类中实现了 DepotService 接口中所有抽象方法。

（5）demo_depotMapper 是 DAO 层的接口，其中 deleteByPrimaryKey(int[] idArr)方法作用是根据库区的属性删除该库区的所有信息；insert(demo_depot record)方法作用是向库区表中插入一条库区的记录；selectAll()方法作用是查找所有库区信息；selectByid(long parseLongId)方法作用是查找对应 id 的库区信息；updateByPrimaryKey(demo_depot record)作用是根据库区的某个属性来更新该库区的信息，在修改库区信息时使用。

由于新建库区、关闭库区、修改库区和查看库区这四个功能实现的方法大同小异，所以本文只对查看库区的具体实现细节做介绍，其步骤如下：

（1）首先，在前端部分使用了表格插件 BootstrapTable，该插件具有非常多的优点，其中最为突出的是它强大的分页功能[41]。因为查询到的库区信息可能会很多，所以必须分页显示。具体方法是在页面 js 文件中调用 bootstrapTable()函数，并为 pageSize 和 pageList 这两个参数配值，pageSize 代表初始每页显示的记录数量，pageList 代表每页记录显示数量的所有可选值，系统中将 pageSize 设置为 5，将 pageList 设置为[5，10，20，30]，即每页初始显示 5 条记录，用户可以选择每页显示 5、10、20 或者 30 条记录。

（2）bootstrapTable()函数中的 url 参数为查询库区信息的请求地址，系统通过该地址找到 DepotController 类中的 all(Integer pageSize,Integer pageIndex)方法，其中 pageSize 参数的意义在（1）中提到过，pageIndex 参数代表需要查找信息的页数。设置这两个参数可以控制系统筛选出与页码相对应的记录。例如，现每页显示 5 条记录，需要查询第 2 页的信息，则该方法便会从查询的所有记录中筛选出第 6 至 10 条记录并发送给页面。

（3）接下来就要获取库区表中的所有库区记录。在（2）中提到的方法中调用业务逻辑层 DepotService 接口的 selectAll()方法，然后通过 DAO 层的 sql 语句查找库区表中的所有信息并

逐层返回到控制层中，控制层筛选出页面需要显示的信息后将其传回页面。

查找库区的页面效果如图 5.13 所示：

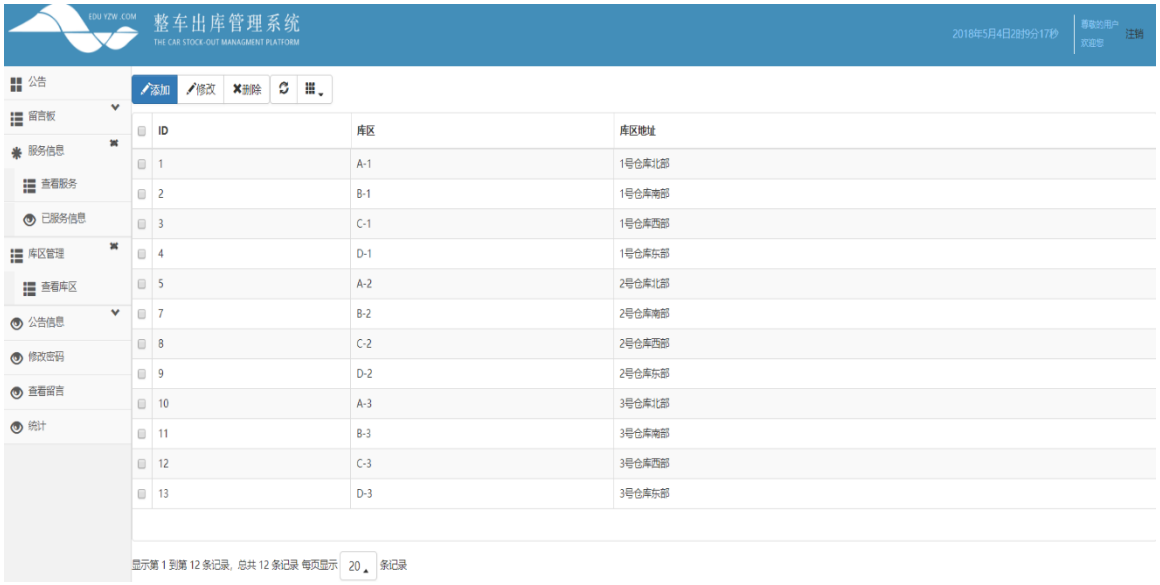


图 5.13 查看库区页面

为了直观地反映仓库的商品车出库情况，系统用柱状图和折线图分别对本月的服务情况以及各库区商品车出库情况做统计。

该功能在开发中使用了 echarts.js,这是一款由百度公司开发的前端数据可视化框架[42]。它与其他框架相比具有如下优势：

（1）包含非常详细的官方 API 文档，并且在文档中对于框架使用的各种案例均有演示，便于学习和使用[43]。

（2）在其官网中可以在线构建项目，对需要使用到的模块进行勾选，从而减小了文件的体积，不用将使用不到的模块加载到项目中[44]。

（3）该框架完全开源，所有人都可以免费使用，而且框架的更新也非常及时[45]。

使用该框架前先要对其进行配置。echarts 框架需要配置的内容非常多，下面列举其中的部分配置：title 表示图表的标题；legend 表示图表的形状；grid 表示图表在页面中的布局，即设置图表在页面中的具体摆放位置；xAxis 中是对 X 轴的相关配置，yAxis 中是对 Y 轴的相关配置；series 设置图表类型，可以是折线图、柱状图和饼状图等等。

统计功能的类图如图 5.14 所示，由于类中的方法过多，图中只列出部分关键方法，下面分别介绍各个类及类中方法的作用：

（1）StatisticsController 类中的 init()方法用来向底层获取初始化图表所需要的信息并将这些信息放入 map 集合中传回页面。

（2）StatisticsService 是业务逻辑层的接口，StatisticsServiceImpl 类是该接口的实现类。其中，init()方法用来获取创建图表所需的信息，在该方法中调用了 getLineData()和

getBarData()这两个方法，分别用来获取折线图和柱状图中的信息。

（3）LinePojo 和 BarPojo 分别为折线图和柱状图的实体类。

（4）demo_depotMapper 为 DAO 层的接口，主要与库区表进行数据交互。类中方法具体作用在库区信息管理部分已做介绍，故不重复赘述。

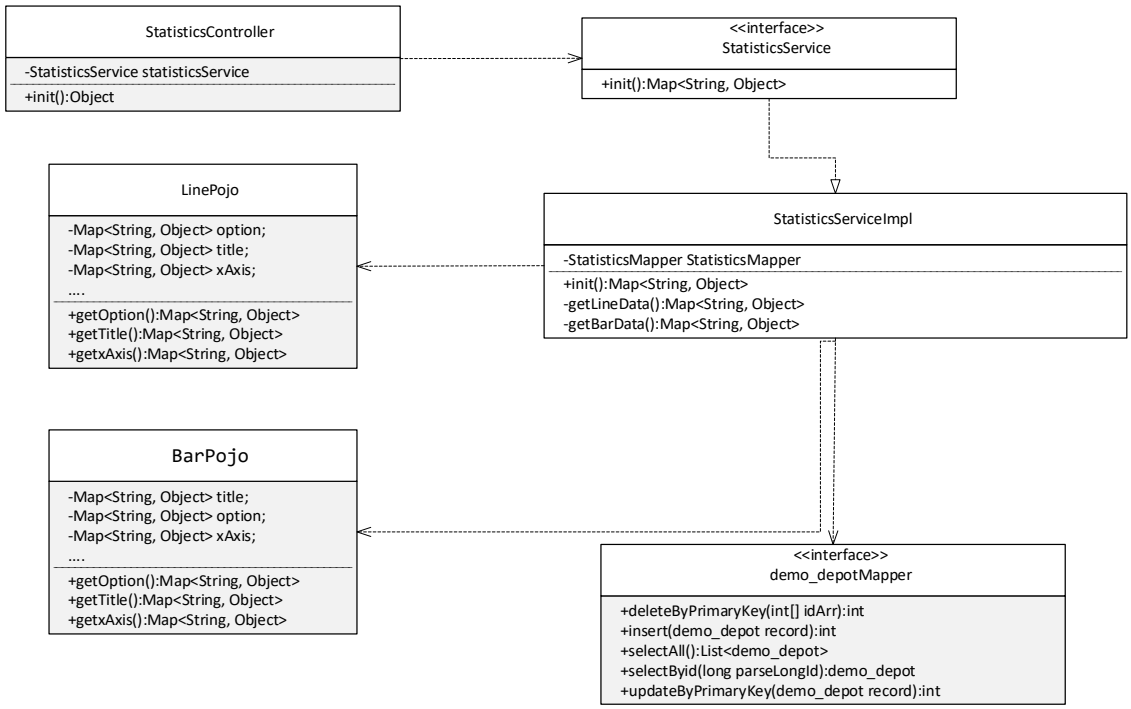


图 5.14 统计功能类图

下面仅对各库区商品车出库情况折线统计图的实现做介绍，其步骤大致如下：

（1）在统计页面文件 View_statistics.html 中调用 echarts.js 框架，并对其做好相关配置。

（2）在统计页面的 View_statistics.js 文件中向控制层发送初始化页面的请求，系统根据请求地址找到 StatisticsController 类中的 init()方法。

（3）在（2）中的控制层方法中调用了业务逻辑层 StatisticsService 接口，该接口的实现类 StatisticsServiceImpl 中的 init()方法作用是获取折线图中需要的信息，其中 x 轴需要获取所有库区名，y 轴需要获取 x 轴中每个库区商品车出库的数量，通过 DAO 层的 sql 语句便可以轻松完成这些信息的获取。

统计页面效果如图 5.15 所示：

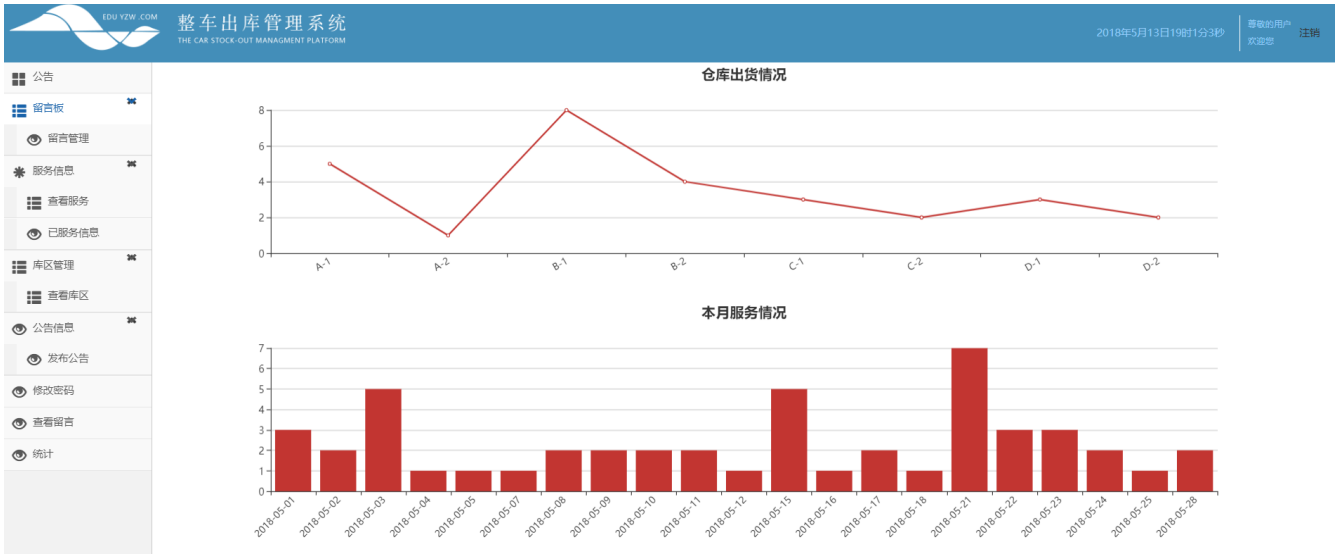


图 5.15 统计页面

5.3 非核心业务子系统的详细设计与实现

非核心业务子系统主要包括了系统管理、用户管理和留言公告三个模块。本节分别对这三个模块中各个功能的详细设计与实现进行介绍。

5.3.1 系统管理的详细设计与实现

系统管理模块主要包含了角色功能管理和越权拦截管理。在系统概要设计中已经给出了这两个功能的初步解决方案，现对其中的设计细节以及具体实现方法做详细介绍。

角色功能管理是该系统设计与实现中的重难点问题。其设计目标是对于不同的角色显示不同的导航菜单，而且为了便于后期维护，导航菜单需要能够随着功能分配的变化而动态改变，因此仅仅在前端进行设计不能满足要求，必须建立相应的数据表并通过前后端结合来实现该功能。

具体设计思路如下：

（1）由于需要为每个角色动态分配功能，所以需要设计一张用于功能配置的数据表，即 demo_role 表。开发者通过该表将系统中已经实现的功能分配给不同的角色，同一种功能可以分配给不同的角色，比如系统中修改密码的功能是司机、大厅管理员和仓库管理员三种角色都具有的。表 5.3 是 demo_role 表中的部分数据：

表 5.3 demo_role 表部分数据

role	treeid
3	1
3	9
3	10
3	18
3	23
3	24
3	26

其中，role 字段代表角色，将 role 字段设置为 3 代表角色为司机，treeid 可以理解为功能的代号，每个功能用一个数字表示。有了这张表，在开发或者维护时只需要将所有功能实现然后对每个角色的功能进行动态增减，例如：在系统使用过程中如果需要为司机新增一个查看库区地图的功能，假设该功能实现后其 treeid 值为 30，则只需要在 demo_role 表中增加一条 role 和 treeid 分别为 3 和 30 的记录即可。

（2）在对每种角色进行功能分配后，还需要将这些功能变成导航菜单显示在主页面上。因此需要完善每个功能的信息，即为 demo_tree 表中设计字段，表 5.4 为 demo_tree 表中的部分数据。demo_tree 表中储存了每条记录的相关信息：首先需要为每个功能设置不同的 treeid；其次每个功能都包含一个页面，每个页面都有相对应的地址，用 treeurl 字段来表示；接下来需要考虑各个功能之间的关系，即有些功能对应的导航菜单是可以展开的一级菜单，而展开后附属在该功能下的菜单为二级菜单，用户在页面点击一级菜单后会展开相应的二级菜单，所以需要设置字段来明确各功能级别，在该系统中利用 ulid 和 liid 这两个字段来实现，如表 5.4 所示，查看服务和已服务信息这两个功能的 liid 字段均为 14，这里的 14 代表服务信息 ulid 的值，表示查看服务与已服务信息这两个菜单是服务信息菜单的二级菜单。

表 5.4 demo_tree 表部分数据

treeid	treename	treeurl	ulid	liid	style	state
14	服务信息	/home/History_records.html	14	14	glyphicon glyphicon-asterisk	1
15	查看服务	/home/View_Service.html	15	14	glyphicon glyphicon-th-list	2
25	已服务信息	/home/History_Service.html	25	14	glyphicon glyphicon-eye-open	2

（3）主页面在 js 文件中向控制层发送请求，系统根据请求地址找到 TreeController 类中的 init(String role)方法，该方法根据前端传入的角色信息将属于该角色的所有功能放入 List 集合中传回页面，页面通过 js 语句将这些信息变成用户可见的导航菜单。

越权拦截管理也是系统中非常关键的一个功能，它保障了系统的安全。在系统的概要设计中介绍了越权拦截功能共分为两轮拦截：第一轮是专门针对角色为司机的用户；第二轮为通用拦截。越权管理的实现主要依赖于 WebSecurityConfig 类中的方法，下面介绍越权拦截管理的实现步骤：

（1）首先通过 InterceptorRegistry 中的 addInterceptor(SecurityInterceptor s)方法向 Spring Boot 框架中添加拦截器；然后对拦截器进行排除配置：将“/error.html”和“/login.html”这两个地址排除拦截，前者为错误页面，后者为登录页面，当用户进入这两个页面时不做任何拦截；接着对拦截器进行拦截配置：因为系统中的页面都放在 home 包下，所以将符合“/home/*.html”格式的地址全部拦截。

（2）查询当前登录用户的角色信息，如果系统发现当前登录的用户角色为司机，当该用户访问页面时，系统会将该页面拦截并对页面的地址做判断：若地址中包含了“edit”或者“add”字符串，则跳转到错误页面；反之，放开对该地址的拦截。

（3）若用户角色不为司机或者页面的地址中没有包含 “edit”、“add”字符串，系统依然会将用户访问页面的地址拦截并进行第二次判断：由于一个页面对应着一个功能，若该功能不属于当前用户，则系统会跳转到错误页面；反之放开对该地址的拦截。完成这次判断最简单的方案就是每次都去数据库中查询，但是这样做会大大降低系统的性能。因此，开发中采用了更好的方案：在系统启动时将所有页面信息加载到内存中，这样每次只需要从这些存储在内存中的页面信息中筛选出属于当前角色的信息并判断被拦截的地址是否存在于其中即可，具体判断方法如表 5.5 中的代码所示。访问内存的速度要比访问数据库快很多，所以对系统的性能没有太大的影响[46]。

表 5.4 判断地址是否存在的关键代码

```
public boolean isExit(List<demo_tree> list, String pathInfo) {
        if (list == null) {
            return false;
        }
        boolean flag = false;
        for (demo_tree entity : list) {
            String treeurl = entity.getTreeurl();
            if (pathInfo.indexOf(treeurl) != -1) {
                flag = true;
                break;
            } else {
                List<demo_tree> children = entity.getChildren();
                flag = isExit(children, pathInfo);
            }
        }
        return flag;
    }
```

该方法的设计思路是遍历获取到的所有属于当前角色的地址，若发现被拦截的地址存在于其中则返回 true，系统取消对该地址的拦截，否则返回 false，系统自动跳转到错误页面。但是页面信息加载到内存中是以多叉树形式的 JSON 数据储存的，所以需要采用递归的方式来遍历这些数据[47]。

5.3.2 用户管理的详细设计与实现

用户管理模块主要包含用户登录、用户注销、修改密码、新建用户、修改用户个人信息、查看用户和删除用户。用户管理模块的类图如图 5.16 所示：

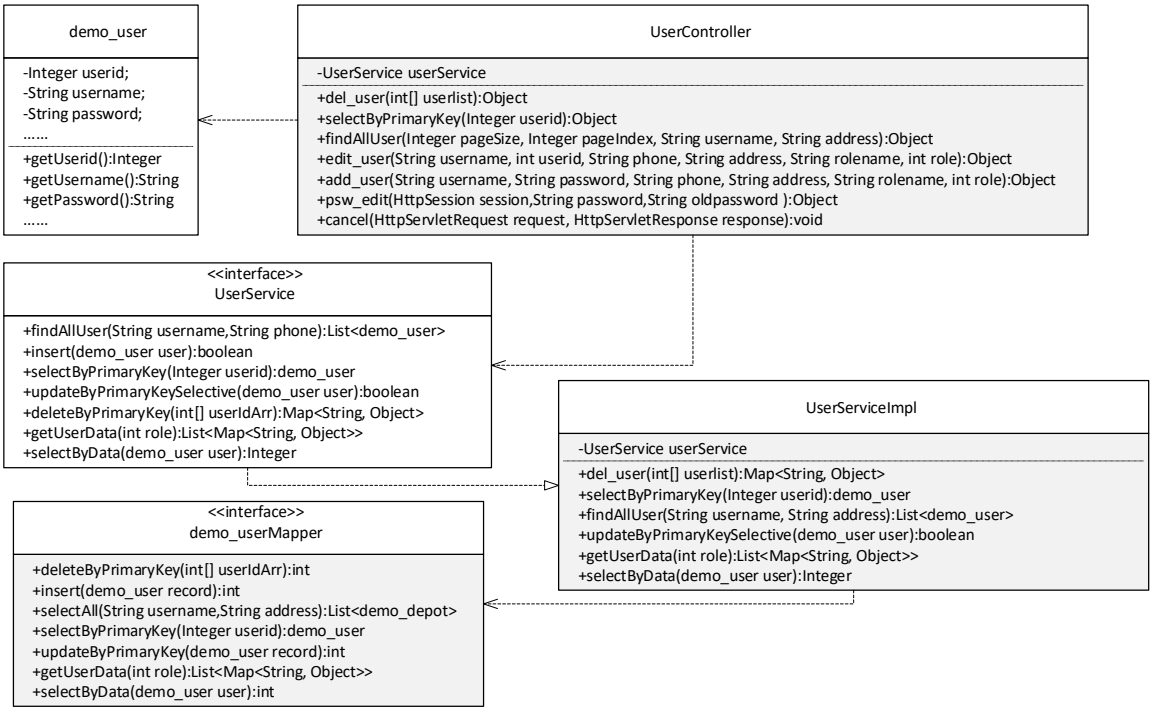


图 5.16 用户模块类图

下面对用户模块类图中的每个类及其方法的作用进行说明：

（1）demo_user 类是用户的实体类，该类中定义了用户的一些属性以及这些属性的 getter 方法，该类与数据表 demo_user 相对应。

（2）UserController 类属于用户管理模块的控制层。其中名为 del_user、selectByPrimaryKey、findAllUser、edit_user、add_user、psw_edit 和 cancel 的方法作用分别是删除用户、通过给定条件来查询用户、查询所有用户、修改用户个人信息、新建用户、修改密码和用户注销。

（3）UserService 接口属于用户管理模块的业务逻辑层，UserServiceImpl 为该接口的实现类，方法名及其作用与 UserController 类大致相同。

（4）demo_userMapper 接口属于用户管理模块的 DAO 层。其方法同样与 UserController 类

中的方法相对应，区别在于 DAO 层直接与数据库进行交互。

下面对用户管理模块部分功能的实现做介绍并展示页面效果。

用户登录功能的实现非常简单，在登录前系统会调用拦截器为当前会话设置生存时间，即登录后到了设置的时间该登录信息便会被清空，需要用户重新登录；点击“登录”按钮后，系统后端会去 demo_user 表中查找输入的账号和密码是否存在，若存在则跳转到该用户所属角色的主页面。登录页面如图 5.17 所示：

图 5.17 登录页面

查看用户功能的设计目标是要能够在页面显示出所有用户的个人信息并可以通过用户名或者用户地址进行条件查询。其实现步骤如下：

（1）在查询用户页面的 js 文件中向控制层发送请求，系统根据请求地址找到控制层中 UserController 类的 findAllUser 方法，该方法首先进行分页设置，然后调用业务逻辑层的方法获取系统中所有用户信息，最后将这些信息以 JSON 格式传回页面。

（2）在（1）中提到的业务逻辑层的方法即为 UserServiceImpl 类的 findAllUser(String username, String address)方法，这里传入的 username 和 address 代表用户的用户名和地址信息，使用这两个参数的作用是实现对用户的条件查询。

（3）在业务逻辑层的方法中调用了 DAO 层 UserService 接口的 selectAll(String username, String address)方法，Mabatis 会根据该方法名到 demo_userMapper.xml 文件中寻找相应的 sql 语句。值得一提的是，这里使用了 MyBatis 的动态 SQL 功能,在 sql 语句中加入了<choose>和<when>标签，使该 sql 语句达到如下效果：当 username 和 address 为空时查找所有用户信息；当 username 或 address 不为空时，以传入的 username 或 address 作为条件进行用户信息的查询。这样大厅管理员在刚进入用户查询页面时由于没有在姓名和地址表单中输入信息，因此 username 和 address 参数均为空，则系统便会将所有用户信息显示在页面上；当大厅管

理员在姓名或地址表单中输入了信息后，username 或 address 参数被赋值，则系统会将条件查询后的用户信息显示在页面上。

查看用户的页面效果如图 5.18 所示：

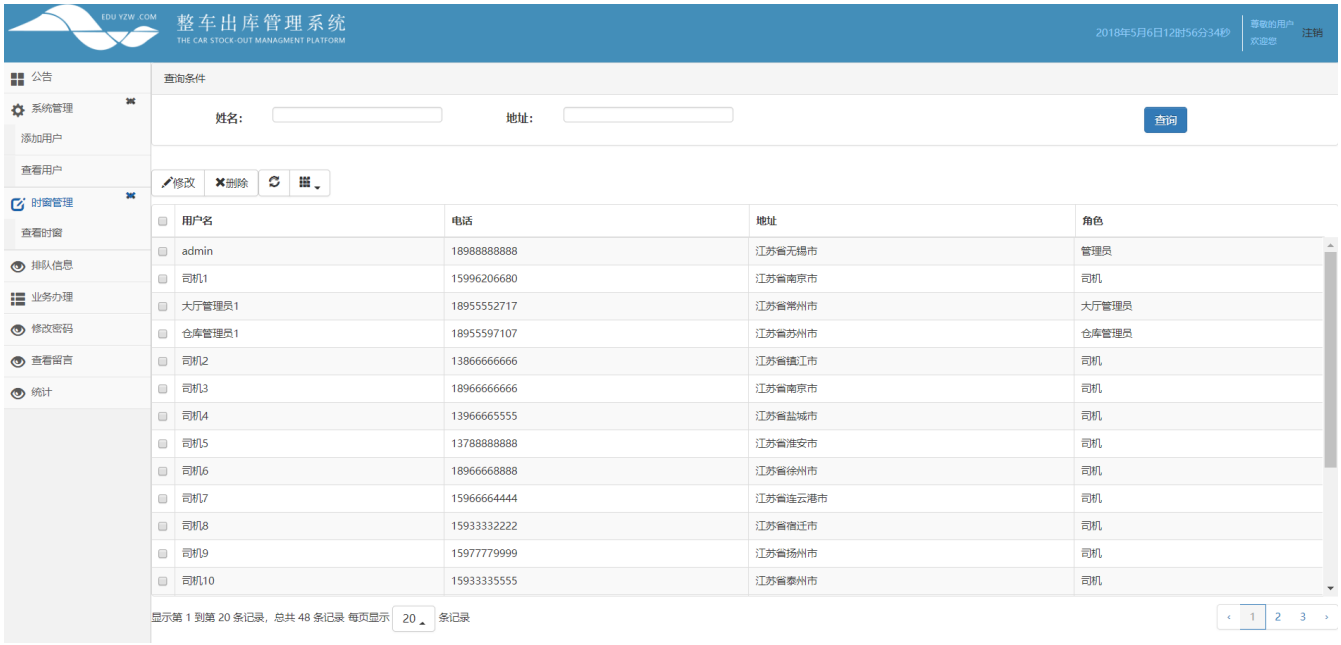


图 5.18 查看用户页面

5.3.3 留言公告的详细设计与实现

留言公告模块设计的目的是为了让司机与管理员之间进行信息交流。留言模块主要分为司机留言、管理员回复与查看留言三个功能，其中管理员可以分别查看已回复的留言和未回复的留言；公告模块主要分为管理员发布公告与司机查看公告两个功能。本节仅介绍管理员查看留言的详细设计与实现。

留言模块的类图如图 5.19 所示：

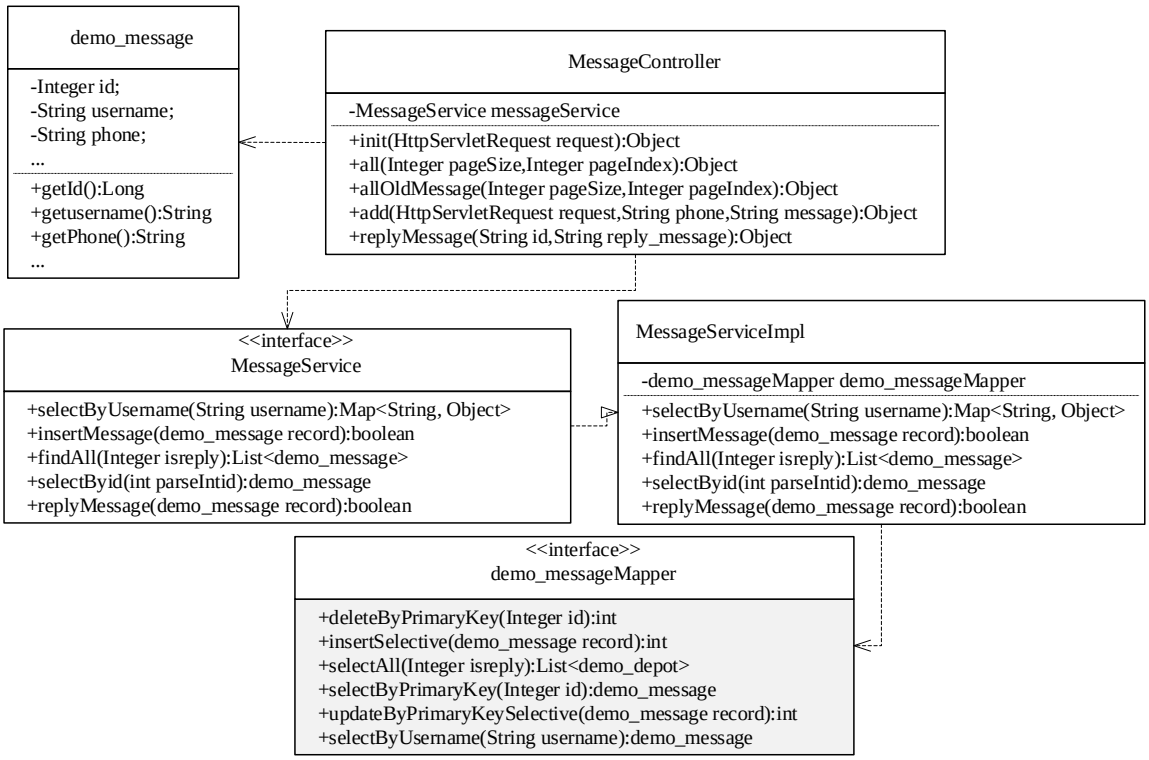


图 5.19 留言模块类图

下面对留言模块类图中的每个类及其方法的作用进行说明：

（1）demo_message 类为留言的实体类，与数据库中的 demo_message 表相对应，并提供各属性的 getter 方法。

（2）MessageController 类属于留言模块的控制层，其中方法名名为 init、all、allOldMessage、add 和 replyMessage 的方法作用分别为初始化留言人信息、查找所有未回复的司机留言、查找所有已回复的司机留言、添加留言和回复留言。

（3）MessageService 是业务逻辑层的接口，MessageServiceImpl 是该接口的实现类。其中方法名为 selectByUsername、insertMessage、findAll、selectByid 和 replyMessage 的方法作用分别为查询指定用户的留言信息、添加留言、查找所有留言、查询指定 id 的留言信息和判断留言是否回复。

（4）demo_messageMapper 为 DAO 层的接口，其方法作用与业务逻辑层大致相同，区别是 DAO 层的方法直接对数据库进行操作。

管理员进入查看未回复留言页面时，页面加载留言信息需要向控制层发送请求，系统根据请求的地址找到 MessageController 类中的 all(Integer pageSize,Integer pageIndex)方法，其中参数 pageSize 代表每页显示的数据量，pageIndex 代表当前显示的页数。该方法调用业务逻辑层的 findAll(Integer isreply)方法，并将参数 isreply 设置为 1，其中 isreply 代表留言是否回复：1 表示未回复，2 表示已回复。业务逻辑层的方法又通过 DAO 层从数据库中获取系统中所有未回复的司机留言信息并逐层传给控制层，控制层接收后将这些信息放入 Map 集

合中以 JSON 格式传回给页面，页面通过前端技术将这些信息展现给用户。管理员查看未回复留言页面的效果如图 5.20 所示：

图 5.20 管理员查看未回复留言页面

5.4 本章小结

本章主要介绍了整车出库管理系统的详细设计与实现。首先介绍了系统开发的准备，包括开发工具和开发环境；其次分别对核心业务子系统和非核心业务子系统中各个模块具体功能的详细设计与实现做介绍，主要通过类图说明每个功能底层类的结构，并对类的实现过程做了详细的说明，展示了部分页面的效果。

第六章 整车出库管理系统的测试

每个系统在开发完成后都需要做全面的测试以保证上线后能够正常地运行。本章从功能、性能和安全性三个方面对整车出库管理系统进行测试，并分析测试结果，得出测试结论。

6.1 系统测试流程

整车出库管理系统的测试流程如下：

（1）强化需求分析：进一步了解系统的需求分析，对每个业务分析需求的要点。

（2）拟定测试计划：确定系统测试的具体内容，拟定测试的先后顺序，做好测试环境以及测试工具的选择。

（3）设计测试方案：编写功能测试用例，测试用例需要包括操作步骤、预期目标以及测试结果，其中操作步骤和预期目标都需要详细编写；确定性能测试中的并发数量；确定系统安全性测试的内容以及步骤[48]。

（4）测试实施阶段：执行功能测试用例，使用测试工具进行多用户并发性测试，并记录测试的结果。

6.2 系统功能测试

系统的功能测试是测试中最主要的部分，系统的开发都是为了实现某些特定的功能。本节采用测试用例表的形式对系统的功能测试进行描述。系统的测试用例表如表 6.1 所示：

表 6.1 系统测试用例表

序号	具体功能名称	测试操作	预期目标	测试结果
1	登录系统	用户输入用户名和密码	登录成功，进入系统	成功
2	司机预约	选择预约日期及时窗并填入相关信息	生成预约号及二维码，预约表中增加相应记录	成功
3	短信提醒	预约时输入司机手机号码	该手机号码收到短信	成功
4	司机取票	司机在取票页面输入预约时系统生成的预约码点击取票按钮	弹出取票成功的提示并将票号信息存入流程表	成功
5	司机留言	输入留言内容及个人信息点击提交按钮	留言成功并且管理员可以看到司机的留言	成功
6	查看公告	点击公告的标题	页面显示公告的具体内容	成功
7	修改密码	输入原密码并输入两次新密码	用户密码改变	成功

8	排队信息	大厅管理员点击排队信息菜单	页面显示正在服务、等待服务、正在排队的票号信息	成功
9	业务办理	管理员对下一个等待服务的票号点击受理按钮	票号进入正在服务队列，司机去柜台办理业务	成功
10	票号作废	管理员对下一个等待服务的票号点击作废按钮	票号移除排队队列且所有流程清空	成功
11	打印取车单	大厅管理员在生成的打印页面点击打印按钮	取车单被打印	成功
12	添加用户	管理员点击添加用户按钮并输入用户的相关信息	用户数据存入用户表中	成功
13	修改用户个人信息	管理员选中用户点击修改按钮并修改相关用户信息	更新用户表中相应记录	成功
14	删除用户	管理员选中用户并点击删除按钮	数据表中相应记录被删除	成功
15	查看所有用户	管理员点击查看用户菜单	页面显示系统中所有用户的个人信息	成功
16	发布公告	管理员填写公告标题和公告内容	公告发布在公告栏中	成功
17	查看未回复留言	管理员点击留言管理菜单	所有未回复的留言及留言人信息显示在页面上	成功
18	查看已回复留言	管理员点击查看留言菜单	所有已回复的留言、留言人信息和回复信息显示在页面上	成功
19	留言回复	点击回复按钮，输入回复的内容	司机能看到管理员的回复信息	成功
20	查看服务	仓库管理员点击查看服务	所有在大厅管理员办理过业务的票号信息被显示在页面上	成功
21	取车服务	仓库管理员点击取车按钮	进行整车出库作业，相应票号走完所有取车流程	成功
22	查看已服务信息	仓库管理员点击已服务信息菜单	在页面上显示所有接收过取车服务的司机信息	成功
23	库区统计	管理员点击统计菜单	页面上显示各库区整车出库情况的折线图以及本月每天业务量的柱状图	成功
24	新增库区	仓库管理员在库区页面点击添加按钮并输入相应的库区信息后提交	系统增加一个库区。库区表中增加一条记录	成功
25	删除库区	仓库管理员选中相应的库区并点击删除按钮	选中的库区从系统中删除，库区表中减少相应记录	成功
26	修改库区信息	仓库管理员选中单个库区点击修改按钮，并修改相应库区信息	相应库区信息被修改，库区表中相应字段被更新	成功
27	查看库区信息	仓库管理员点击查看库区菜单	页面显示系统中所有库区的信息	成功
28	添加时窗	大厅管理员在时窗页面点击添加按钮并填入相应时窗信息	时窗被添加到系统中，时窗表中新增一条记录	成功
29	修改最大预约能力	大厅管理员选中某个时窗点击修改按钮并修改最大预约	时窗信息被修改，时窗表中相应字段被更新	成功

		能力		
30	删除时窗	大厅管理员选中某个时窗点击删除按钮	相应时窗信息从系统中删除，数据库减少相应记录	成功
31	角色功能管理	不同角色的用户登录系统	不同角色的用户在主页面看到不同的导航菜单	成功

由测试用例表可知该系统基本满足功能设计的要求。

6.3 系统性能测试

性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。对于一个 Web 系统来说，性能的好坏很多情况下取决于多用户并发访问时请求的响应时间。本节对整车出库管理系统进行压力测试，给出测试结果并得出测试结论。

6.3.1 测试工具和测试步骤

测试借助性能测试工具 Apache JMeter 来实现[49]。该工具最初就是被设计用于 Web 应用测试，所以其在 Web 测试方面具有很强大的性能，不仅可以用于对静态和动态的资源（文件、Servlet、Perl 脚本、java 对象、数据库和查询和 FTP 服务器等等）性能进行测试，也可以用于对服务器、网络或对象模拟繁重的负载测试它们的抗压性能[50]。

在测试前需要录制测试的脚本，使用 BadBoy 可以实现。它是一款免费的 Web 自动化测试工具，支持对录制出来的脚本进行调试，很重要的一点是，它也支持将脚本导出为 JMeter 脚本[51]。

具体测试步骤如下：

（1）用 BadBoy 测试工具录制单用户访问时的测试脚本。

（2）分析压力测试的具体压力数值。由于系统的使用群体较少，故不需给予特别大的压力数值，将压力数值拟定为 500，即 500 个用户同时并发访问系统。

（3）对（1）中录制好的脚本进行适当修改后在 Apache JMeter 中进行测试，测试前需要向数据库添加足够量的数据并且修改相关测试配置。

（4）运行测试程序并得出测试数据。

6.3.2 测试结果及分析

对系统进行 500 用户并发访问压力测试后得到的结果如图 6.1 和图 6.2 所示：

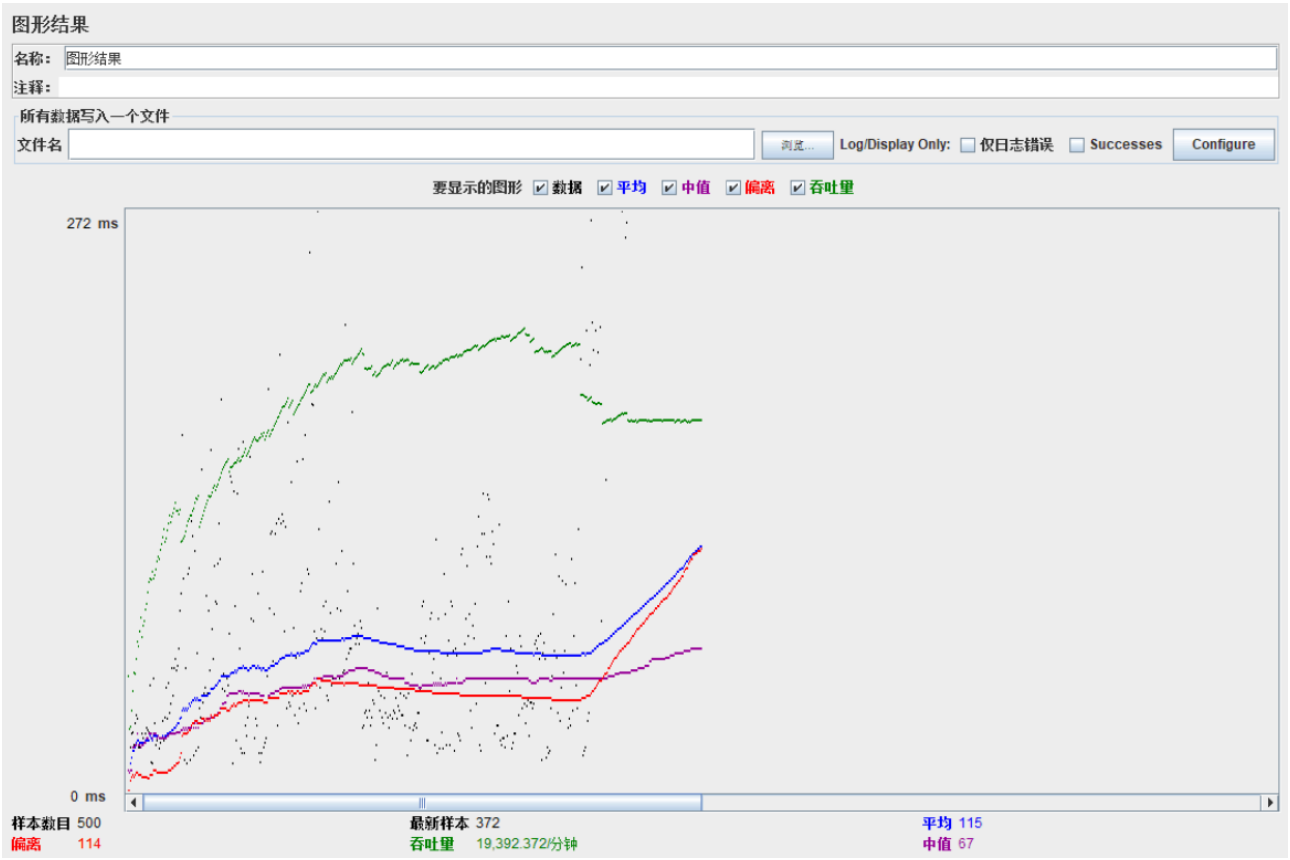
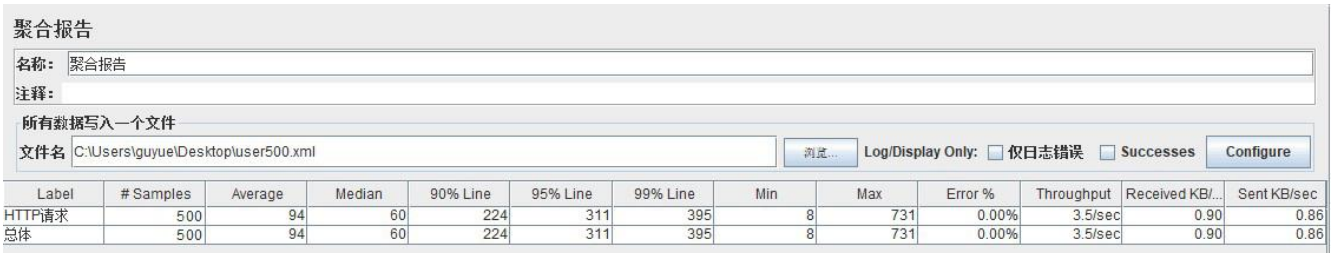


图 6.1 压力测试图形结果图



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/...	Sent KB/sec
HTTP请求	500	94	60	224	311	395	8	731	0.00%	3.5/sec	0.90	0.86
总体	500	94	60	224	311	395	8	731	0.00%	3.5/sec	0.90	0.86

图 6.2 压力测试聚合报告图

其中，两张结果图分别由 JMeter 中的两个监听器得到，两个监听器间相互独立，得到的两张结果图部分数据大致相同。

由图 6.1 可以看出，随着系统中并发用户数量的增加，请求响应时间也随之增加。请求的平均响应时间约为 115ms，表现正常。

由图 6.2 得到请求响应时间的峰值为 731ms，考虑到测试时网络环境因素，属于可接受范围。

由于设定的压力值 500 已经足够模拟系统运作时的最大负荷，在这种情况下用户访问系统的响应时间无论从平均值还是峰值来看均对用户体验没有太大影响，所以该系统具有很强的抗压能力，性能优良。

6.4 系统安全测试

由于该系统是一个 Web 项目，而且系统采用了 B/S 架构，只需要在浏览器中输入相应的地址便可以使用，所以对于系统进行相关安全测试十分必要。系统在设计时加入了越权拦截机制来保障系统的安全，本节主要对系统的越权拦截机制做测试。

具体测试方法及测试结果如下：

（1）用司机的账号登录系统，假设该司机通过某种方法获得了大厅管理员添加用户页面的地址：http://127.0.0.1:8080/demo/home/user_add.html。在地址栏中输入该地址，系统自动跳转到错误页面，该地址成功被拦截。

（2）用大厅管理员的账号登录系统，同样在地址栏中输入（1）中的添加用户页面的地址，浏览器成功跳转到添加用户的页面。这是由于添加用户的功能属于大厅管理员，系统的越权管理机制在拦截到该地址后经过判断取消对其拦截。

（3）前两种方法是在用户登录后测试的，现不用任何账号登录系统，直接在浏览器的地址栏中输入用户主页面的地址：http://127.0.0.1:8080/demo/index.html。测试结果：系统自动跳转到登录页面，即在不登陆的情况下无法访问系统。

综上所述，系统具有很好的安全性。

6.5 本章小结

本章主要对整车出库管理系统进行测试。首先对系统进行功能测试，通过测试用例图展现测试的过程及结果；其次对系统的性能进行测试，主要测试系统的多用户并发访问的性能；最后对系统进行安全性测试。

第七章 总结与展望

7.1 论文总结

随着互联网技术的不断积累和迅速发展，国内一些小型汽车厂商的半信息化平台半手工的整车仓储物流管理方式已经难以适应当今信息化时代，本文设计的整车出库管理系统作为整车仓储物流管理系统中的一个模块而实现了对于整车出库过程的管理。系统分为两个子系统，一方面实现了整车出库的业务流程，另一方面也实现了仓库的管理以及用户之间的信息交流。

随着 web 开发的周期的不断缩短，项目难度不断提高，传统的 Spring 框架由于其复杂的配置和部署过程在应对大型的 web 项目时显得捉襟见肘，而 Spring Boot 技术由于其简单快捷的开发过程越来越受到开发者的青睐。本人在南京握手信息科技有限公司实习的期间，有幸参与了基于 Spring Boot 的项目开发，让我对 Spring Boot 在开发中的优异表现有了更为直观的感受。

本文的工作总结如下：

（1）对整车出库管理系统进行需求分析，确定了系统的实现目标、系统角色和系统功能。系统需要实现对整车出库流程的管理并均衡分配司机取车的时间；系统中的角色主要有司机、大厅管理员和仓库管理员；从功能上将系统划分为两个子系统，每个子系统又包含三个功能模块，它们分别是取车业务模块、时窗管理模块、库区管理模块、系统管理模块、用户管理模块和留言公告模块。

（2）对整车出库管理系统进行设计，分为概要设计和详细设计。在概要设计中设计了系统功能的具体细节、系统的结构和系统的数据库，并就系统设计中的关键问题以及重难点问题提出初步的解决方案；在详细设计中确定了系统的底层类结构和运作流程，并用类图的方式呈现。

（3）实现整车出库管理系统，并对概要设计中提出的关键问题和重难点问题给出详细的解决方案和实现细节，展现部分页面的实现效果。

（4）从功能、性能和安全性三个方面对整车出库管理系统做测试并分析测试结果。

7.2 下一步研究工作

受限于本人的开发能力以及开发经验，该系统中仍然有很多不足之处需要改进：

（1）在用户登录时没有加入微信等当今流行的登录方式。

（2）没有很好地为整车仓储物流管理系统的其他部分预留相关接口。

（3）系统中业务设计有所缺陷，因将物流支付加入该系统的业务中。

（4）页面的渲染需要改善。

参考文献

[1] 尹绪松. 中国第一汽车集团公司东南亚市场的营销网络体系研究[D]. 长春: 吉林大学, 2008.
[2] 卢锋. 中国汽车市场点爆式增长规律研究[D]. 南京: 南京航空航天大学, 2011.
[3] 龙伟, 郭凯明. 数据采集系统在汽车仓储管理中的应用[J]. 物流技术与应用, 2005, 10(6):88-90.
[4] 张小平. 浅谈汽车运输分公司绩效考核管理的建立与完善[J]. 中国管理信息化, 2015, 18(15):127-128.
[5] 李歆玥. 基于第三方物流的大众一汽平台物流系统优化的研究[D]. 长春: 吉林大学, 2011.
[6] 姜彦宁. 资源共享模式下的整车物流路径优化[J]. 公路交通科技, 2017, 34(6):114-121.
[7] 孟曦. 中国汽车物流的现状及问题研究[J]. 物流工程与管理, 2009, 31(3):1-3.
[8] 周文军, 赵辉. 汽车行业第三方物流管理供货模式[J]. 物流技术与应用, 2003, 8(6):50-53.
[9] 陈兆俊, 王丽娜. 汽车备件管理[M]. 北京: 北京理工大学出版社, 2015:14-20.
[10] Iijima M 1. Logistics innovation for Toyota's world car strategy[J]. International Journal of Integrated Supply Management, 2005, 1(4):478-489.
[11] 刘美侠. 出入库与订单管理系统的设计[J]. 齐鲁工业大学学报, 2011, 25(1):38-40.
[12] 王鹏辉, 陈光, 鲍萍萍. GPS物流车辆行驶数据采集与存储系统的开发[J]. 电子设计工程, 2017, 25(11):9-13.
[13] 黄伟东. 广汽丰田汽车有限公司物流管理优化研究[D]. 长沙: 湖南大学, 2015.
[14] 李玉华. 东风汽车公司改善物流管理的实践[J]. 物流技术与应用, 2005, 10(5):118-119.
[15] 张俊伟, 王勃, 马范援. 多仓库多配送点的物流配送算法[J]. 计算机工程, 2005, 31(21):192-194.
[16] 席晓峰, 吕良双, 逯鹏. 使用J2EE框架技术构建可重用的Web应用[J]. 计算机工程与应用, 2005, 41(29):208-210.
[17] 汪云飞. JavaEE开发的颠覆者:Spring Boot实战[M]. 北京: 电子工业出版社, 2016: 65-67.
[18] Meng Q. Web Service Based on Spring JavaBean[J]. Computer & Modernization, 2012, 1(1):194-195.
[19] Daszykowski M, Serneels S, Kaczmarek K, et al. TOMCAT: A MATLAB toolbox for multivariate calibration techniques[J]. Chemometrics & Intelligent Laboratory Systems, 2007, 85(2):269-277.
[20] 王永和. Spring Boot研究和应用[J]. 信息通信, 2016, 20(10):91-94.
[21] 张宇, 王映辉, 张翔南. 基于Spring的MVC框架设计与实现[J]. 计算机工程, 2010, 36(4):59-62.
[22] Qin-Ping O U. The Design and Implementation of the Bioinformatics Database Based on Spring MVC and iBATIS[J]. Journal of Southwest University, 2008, 30(11):142-145.
[23] Cattell R. Scalable SQL and NoSQL data stores[J]. Acm Sigmod Record, 2011, 39(4):12-27.
[24] 阳小兰, 罗明. 基于Spring+SpringMVC+MyBatis网上论坛的设计与实现[J]. 科学技术创新, 2016, 27(36):279-280.
[25] 刘淑英, 曹悦, 吕利娜. 基于Spring+MyBatis的高校工资信息管理系统的设计与实现[J]. 数字技术与应用, 2017, 27(9):161-162.
[26] Mesbah A, Deursen A V, Lenselink S. Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes[J]. Acm Transactions on the Web, 2012, 6(1):1-30.
[27] 周喜, 蔡明杰. 基于MS Excel平台开发专用型会计软件的可行性分析[J]. 湖南工业职业技术学院学报, 2011, 11(4):18-20.
[28] 张志勇, 田启家, 史忠植. 一种基于工作流的软件需求分析方法[J]. 计算机工程与应用, 2002, 38(17):106-109.
[29] 陆文, 徐锋, 吕建. 一种开放环境下的软件可靠性评估方法[J]. 计算机学报, 2010, 33(3):452-462.
[30] 吴大刚, 肖荣荣. C/S结构与B/S结构的信息系统比较分析[J]. 情报科学, 2003, 21(3):313-315.
[31] 李威, 高锦涛, 高腾. 基于E-R图的关系数据库关键字查询[J]. 计算机系统应用, 2012, 21(9):156-161.
[32] Frost R. Jazz and the Eclipse Way of Collaboration[J]. IEEE Software, 2007, 24(6):114-117.
[33] 陈萧宇, 黄震, 刘謴哲,等. Scratch:一个基于Chrome浏览器的用户操作捕捉与回放工具[J]. 计算机科学, 2014, 41(11):112-117.
[34] 赵光亮, 舒小松. Navicat for MySQL平台中的SQL语言分析与应用[J]. 无线互联科技, 2017, 17(19):74-75.
[35] 任中方, 张华, 闫明松,等. MVC模式研究的综述[J]. 计算机应用研究, 2004, 21(10):1-4.
[36] 陈欣. 基于java三层构架的管理信息系统中DAO层的构建探索[J]. 科技资讯, 2015, 13(11):26-27.
[37] 李志秀, 张军, 陈光,等. JQuery Ajax异步处理JSON数据在项目管理系统中的应用[J]. 云南大学学报

(自然科学版), 2011, 33(2):247-250.
[38] 谭力, 杨宗源, 谢瑾奎. Ajax技术的数据响应优化[J]. 计算机工程, 2010, 36(7):52-54.
[39] 钟啸灵. 阿里云的电子商务云服务[J]. IT经理世界, 2010, 12(17):92-93.
[40] 牛禄青. 阿里云:创新云计算[J]. 新经济导刊, 2013, 15(3):66-68.
[41] 董天奇. 保定市大气污染城市智能管理平台设计[D]. 杭州: 浙江理工大学, 2017.
[42] 王子毅, 张春海. 基于ECharts的数据可视化分析组件设计实现[J]. 微型机与应用, 2016, 35(14):46-48.
[43] 冀潇, 李杨. 采用ECharts可视化技术实现的数据体系监控系统[J]. 计算机系统应用, 2017, 26(6):72-76.
[44] 刘志远, 杨琭, 阎光伟. 基于ECharts-X的安全事故数据三维可视化系统[J]. 中国科技信息, 2015, 19(23):34-35.
[45] 郑幸源, 洪亲, 蔡坚勇,等. 基于AJAX异步传输技术与Echarts3技术的动态数据绘图实现[J]. 软件导刊, 2017, 16(3):143-145.
[46] 周忠丽, 张建伟, 陈鹏. 采用内存数据访问对象提高数据库访问速度[J]. 四川大学学报:自然科学版, 2002, 39(3):435-438.
[47] 袁栋梁, 孙忠林, 田刚,等. 基于JSON格式的信息资源共享技术的应用研究[J]. 计算机与现代化, 2010, 18(9):175-178.
[48] 刘星妍. 有效的功能测试用例设计[J]. 信息技术, 2010, 15(1):142-144.
[49] 佘青. 利用Apache Jmeter进行Web性能测试的研究[J]. 智能计算机与应用, 2012, 02(2):55-57.
[50] 吴志刚. 使用JMeter插件提高性能测试效率[J]. 软件导刊, 2010, 09(4):33-35.
[51] 冯国军, 熊冬青, 吴宏杰,等. 大规模并发入侵检测实验系统研究[J]. 信息安全与通信保密, 2012, 12(7):74-76.

致谢

转眼间已经来到了 2018 年的春天，快要到了毕业的时间。两年半前那个秋天刚进入校园的场景依然历历在目，仿佛就在昨天发生。细数这两年半的研究生生活，点点滴滴都使我不断成长。在学习方面，我提高了自己的独立自主学习的能力，在学习之中遇到困惑的时候我便会一个人静静地去图书馆寻找答案；在工作方面，第一次去公司实习，让我学到了在学校中学不到的知识，让我感受到了团队的力量；在生活方面，朝夕相处的室友以及师门兄弟都让我学会了如何与他人和睦相处。所以，在即将离别校园之际，我发自内心地感谢在研究生期间给过我帮助的所有人。

首先，我要衷心感谢我研究生期间的导师曹士坷教授。曹老师和蔼亲切，为人光明磊落，做学问态度严谨、实事求是，工作上孜孜不倦、一丝不苟，是我学习的榜样。不仅在学业上给我提供了巨大的帮助，在生活上也对我关怀备至。在毕业设计期间，曹老师无论从开题到中期还是到最后的论文定稿，都一直耐心认真地给予我指导，使我的毕业设计能够顺利地完成。我能够完成我研究生的学业离不开曹老师的辛劳。

其次，我要感谢给我提供实习机会的南京握手信息科技有限公司。在公司实习期间，我将书本上理论性的知识转变成了实际应用的经验，学会了软件开发的流程，提高了我的编程能力。同时，与同事一起工作也让我体会到了团结的力量，非常感谢公司给我提供了这次锻炼的机会。

另外我还要感谢仝野、邵敏、刘振国、胡浩和茆玉庭，他们是跟我同师门的兄弟。在我进行毕业设计期间，他们都给予了我很大的帮助，帮我解决了不少困难，让我少走了很多弯路。

然后我还要感谢我的父母，他们辛苦养育了我二十六年，也是他们的鼓励使我在知识的道路上不断前进，最后进入研究生的殿堂。

最后，我还要感谢一下为我审阅论文及参加答辩的老师们，感谢你们指出我论文中做的不够好的地方，我一定会尽我最大的努力去完善论文，改进其中不足的地方，做一名合格的研究生！