青岛理工大学

毕 业 设 计（论 文）

**题目：**基于SSM的企业公务用车管理系 统的设计与实现

**院 别：** 信息与控制工程学院

**专业班级：** 软件工程164班

**学生姓名：** 周歆怡

**学 号：** 201607340

**指导教师：** 胡克勇

2020 年 2 月 24 日

摘 要

第一段写研究背景目的和意义4行 第二段写功能 子系统包含模块 稍多

第三段写总结与展望（4行）

社会在慢慢进步，各类企业使用的技术发生了日新月异的变化，计算机技术也相较于之前到达了一定的高度，所以使用计算机系统实现企业公务用车的管理是适应时代发展趋势的一个重要举措。由于各个企业侧重的方向不同，对系统功能的需求也不同，使用计算机系统来高效率地管理企业用车的要求，是推动企业公务用车方面的管理驶向规范化的必由之路。

企业用车管理是一项细致而繁杂的工作，每月每天需要用车的量很大，但买车加上后期养护车非常耗费金钱，假如派人租车，每月需要进行各种车辆记录、用车记录、投诉处理等，就会浪费工作人员许多的时间和精力。

通过企业公务用车管理系统进行用车管理，不但摒弃了人工管理易出错的缺点，保证了数据的准确性，并且因为数据库的事务具有持久性的特点，可以方便地对有关车、人订单等各种信息统筹规划，用于满足用车公司其他方面的需求，比如用车后核算价格和订单处理等。使用该系统能极大地提高企业用车时的效率，方便用车公司和司机的日常工作落实，所以基于SSM框架的企业公务用车管理系统应运而生，本文将详细地介绍设计此系统的详细思路和制作流程，对系统的主要功能和部分重要代码的设计也有较为详尽的说明。

关键词：企业公务用车，信息管理，计算机，SSM

ABSTRACT

The society is slowly progressing.The technologies used by various enterprises have changed with each passing day, and the computer technology has also reached a certain height compared with before. Therefore, the use of computer system to realize the management of corporate official vehicles is an important measure to adapt to the development trend of The Times. As enterprises focus on different directions and have different needs for system functions, the use of computer systems to efficiently manage corporate vehicles is the only way to push the management of corporate official vehicles to standardization. Enterprise car management is a meticulous and multifarious work, the amount that needs car every day every month is very big, but buy a car to add later period to maintain a car to spend money very much, if send a person to rent a car, need to undertake every month all sorts of car record, car record, complain processing, can waste the time and energy of worker a lot of.

Through the enterprise transport business, management system and management, not only abandon the error-prone manual management shortcomings, ensures the accuracy of the data, and because the database transaction has the characteristics of persistence, can easily order to cars, people concerned such as overall planning, all kinds of information to meet the demand of transport company in other ways, such as car after accounting the price and order processing, etc. Using the system can greatly improve the efficiency of the enterprise transport, convenient transport company and the driver of the daily work, implementing so public transport management system based on frame of SSM arises at the historic moment, this article will introduce in detail design of the system thinking and process in detail, on the main functions of the system and the design of some important code also has a more detailed description.

KEY WORDS: official vehicle，information management, computer，SSM

目录

缩进要一样

[第1章 绪论](#第1章)4

[1.1研究背景及意义](#研究背景)4

[1.1.1课题的研究背景](#研究背景)4

[1.1.2课题的研究意义](#研究意义)4

[1.2国内外研究现状](#研究现状)4

[1.3存在的问题](#存在的问题)1

[1.4课题研究内容和目标](#课题研究内容和目标)4

[1.5项目概述](#项目概述)5

[1.5.1产品介绍](#产品介绍)5

[1.5.2用户群体及角色](#用户群体及角色)5

[1.6论文组织结构](#论文组织结构)5

[第2章 课题关键技术介绍](#第2章课题关键技术介绍)6

[2.1 java语言](#Java语言2)6

[2.2 SSM框架介绍](#SSM框架介绍2)6

[2.2.1 Spring框架](#Spring框架2)6

[2.2.2 Spring MVC框架](#SpringMVC框架2)6

[2.2.2 MyBatis & MyBatis-plus框架](#MyBatis2)6

[2.3前端技术介绍](#前端技术介绍2)7

[2.3.1 JSP动态网页技术](#JSP2)7

[2.3.2 Bootstrap前端框架](#Bootstrap2)7

[2.4其他技术](#其他技术2)7

[2.4.1 shiro框架](#shiro2)7

[2.4.2 Lombok插件](#Lombok2)8

[2.5开发工具](#开发工具2)8

[2.5.1 JDK](#JDK2)8

[2.5.2 IDEA](#IDEA2)8

[2.5.3 Tomcat](#Tomcat2)8

[2.5.4 Mysql](#Mysql2)8

[第3章 系统需求分析](#第3章系统需求分析)9

[3.1 设计背景](#设计背景3)9

[3.2 目标设计要求](#目标设计要求3)9

[3.3 用户组织管理及系统用例图](#用户组织管理及系统用例图3)9

[3.4 系统整体数据流程图](#系统整体数据流程图3)10

[3.5系统功能性需求](#系统功能性需求3)11

[3.5.1账号管理模块](#账号管理模块3)11

[3.5.2审核司机模块](#审核司机模块3)12

[3.5.3下单模块](#下单模块3)12

[3.5.4 投诉模块](#投诉模块3)12

[3.6系统非功能性需求](#系统非功能性需求3)13

[3.7系统用户权限需求](#系统用户权限需求3)14

[第4章 系统总体设计](#系统总体设计4)15

[4.1系统总体架构](#系统总体架构4)15

[4.2系统主要功能模块设计](#系统主要功能模块设计5)15

[4.2.1司机管理模块](#司机管理模块4)15

[4.2.2用车公司管理模块](#用车公司管理模块4)16

[4.2.3管理员管理模块](#管理员管理模块4)16

[4.3系统功能流程](#系统功能流程4)17

[4.4系统数据库设计](#系统数据库设计4)17

[4.4.1数据库设计实现](#系统数据库实现4)17

[4.4.2 数据库逻辑设计](#数据库逻辑设计4)21

[4.5本章小结](#本章小结4)21

[第5章 系统详细设计](#第五章系统详细设计)22

[5.1系统总体详细设计](#系统总体详细设计5)22

[5.2系统主要功能模块设计](#系统主要功能模块设计5)23

[5.2.1注册登录模块设计](#注册登录模块设计5)23

[5.2.2审核司机模块设计](#审核司机模块设计5)24

[5.2.3下单模块设计](#下单模块设计5)26

[5.2.4 投诉模块设计](#投诉模块设计5)28

[5.3系统活动图设计](#系统活动图设计5)30

[5.3.1审核信息活动图](#审核信息活动图5)30

[5.3.2企业用车活动图](#企业用车活动图5)30

[5.4本章小结](#本章小结5)31

[第6章 系统实现](#第六章系统实现)32

[6.1系统环境架构](#系统环境架构)32

[6.2系统实现要点](#系统实现要点6)32

[6.2.1配置shiro](#配置shiro6)32

[6.2.2实现shiro](#实现shiro6)33

[6.2.3用户密码机制](#用户密码机制6)33

[6.2.4实体类设计](#实体类设计6)33

[6.2.5 Controller设计](#Controller设计6)34

[6.2.6 Service设计](#Service设计6)34

[6.2.7 Dao层设计](#Dao层设计6)34

[6.3 界面展示及对应部分代码](#界面展示及对应部分代码6)34

[6.3.1登录界面](#登录界面6)34

[6.3.2管理员管理界面](#管理员管理界面6)37

[6.3.3用车公司管理界面](#用车公司管理界面6)43

[6.3.4司机管理界面](#司机管理界面6)46

[6.4 系统总结](#系统总结6)49

[6.5 本章小结](#本章小结6)49

[致谢](#致谢)49

[参考文献](#参考文献)49

1. 绪论

（写的比较少）

1.1研究背景及意义

1.1.1研究背景（2段简化标题）1.5倍行距！！开头空两格字符。

社会经济在不断地进步，企业工资水平也逐渐提高，车辆不再是奢侈品，而成为人们生活的必需品，由于车辆需求与使用人员的不断增多,企业用车的管理问题也层出不穷,因此对司机的管理、对车辆的管理、对订单的管理诸如此类的管理系统应运而生。中小型企业自己购买车辆不仅需要耗费钱财和人力去维护，而且使用频率相对大型企业来说并不高,无形之中增加了运营的压力。

1.1.2研究意义（两段）

在此背景之下，基于SSM框架的企业公务用车管理系统的设计初衷是满足当前企业租赁汽车的需求，它设计的目的是提供广泛的功能，简单易读的界面，舒适的服务，便捷的操作，使得管理人员可以轻松地管理用车信息。

1.2国内外研究现状（多写点 说明优缺点）

文献[1]设计了..系统 有什么优缺点，

XXX系统…

当下公务用车管理系统已经发展到较为完备的程度，公务用车管理系统是一个面向特定用户群体的小型系统。调查研究显示，国内外用车管理公司大部分已经拥有了适合自己的管理系统。利用计算机管理用户的数据不但改变了过去人工管理复杂且易出错的现象，而且代替了过去手工制表填表处理表单的复杂操作，此举提高了数据的独立性和安全性，界面功能强大，显示清晰。

国内外的信息化以网络通信技术和数据库技术为主，把用车公司、司机、管理员使用的所有信息汇聚到数据表里来加以控制和修改，并通过完善系统功能，使得企业用车便利，为推动时代发展提供庞大的技术和理论的支持。

1.3存在的问题（参考了之前的系统 发现的问题 也可以写研究现状后面）

计算机管理系统开发的一个严峻的问题就是如何准确而又系统地管理用车数据，各个企事业单位对车辆的需求不尽相同，司机的安排方面也有所区别，同样管理信息使用的方法也不同，由此系统设计的方向要根据大部分企业用车方面的需求来设计。如何有效和迅速地响应公务车辆管理需求,如何解决目前的困难是现阶段所面临的主要问题。

1.4研究内容和目标

本论文研究的公务用车管理系统采用B/S架构模式,编写代码实现和后期测试维护两个阶段都使用IDEA开发平台。以页面登录权限验证和对企业用车途中各角色管理数据的设计为重点进行深入探究,符合实际用户使用界面的设计要求，模块设计要符合高内聚低耦合的特点。整个系统前台使用了JSP动态网页技术、Javascript、CSS、Bootstrap等技术来实现美观、简洁、便于操作的用户界面。管理员管理数据使用Navicat for MySQL数据库存取数据，而且要统筹规划好各个数据表之间的数据项关联。

1.5项目概述（移到概要设计里）

1.5.1产品介绍

企业公务用车管理系统主要的用途是对企业的用车情况进行合理化管理，包括司机管理（姓名、车牌号、昵称、信用积分、年龄等）、车辆管理（名称、车牌号、颜色、品牌、类型、车型等）、订单管理、账号管理、投诉管理等模块。

还有本系统比较重要的用户登录管理模块，通过判断登录的用户角色（管理员、司机、用车公司）来分别展示不同的列表功能。

1.5.2用户群体及角色

司机：为用车公司提供服务。

用车公司：租赁需要使用的车辆。

管理员：管理用户信息、车辆信息、订单信息等。

1.6论文组织结构

本文共六章。

第一章为绪论部分，主要介绍企业公务用车的相关知识以及该领域在国内外的发展现状，最后对本文的整体组织架构进行说明。

第二章为课题关键技术介绍，主要介绍了撰写本系统所用到的语言、框架、前端技术及开发工具。

第三章为系统需求分析，通过研究系统需求，给出系统设计的整体流程图，使用用例图的形式描述系统功能总体需求和部分模块的需求情况，接着分析系统在非功能需求与功能需求两方面，最后介绍各类用户系统权限的需求。

第四章为系统总体设计，主要通过功能模块图来介绍整个体统具体的小模块以及模块内部的大致功能，接着详细叙述了系统整体的功能流程，最后通过E-R图以及实体属性图来详细介绍数据库表的设计。

第五章为系统详细设计，主要分模块详细介绍了系统是如何设计的，对每个子功能进行详细说明，使用交互图和类图对系统进行详细阐述。

第六章为系统具体实现，对系统的具体模块的代码实现进行详细的叙述，以及系统的实现过程中 SSM 框架的架构和用户登录过程中shiro框架的配置及使用，通过代码和界面的展示来清楚明白地介绍模块内部是如何运作的，最后分析了系统完成后存在的问题。

少弄三级标题

第2章 关键技术简介

2.1 Java语言

Java是一门面向对象编程语言，具有使用便利和理解简单两个主要特征，可以在不同的平台上通过安装JRE来运行。同时它还具有简单性、分布式、健壮性,以及安全性、平台独立与可移植性、多线程、动态性等特点。

2.2 SSM框架

本系统充分考虑用户的需求，服务端采用模型（Model）-视图（View）-控制器（Controller）软件架构模式，简称为MVC，他可以将页面显示、业务逻辑和数据库访问彼此分离，是目前应用很成功的软件架构模式。为了提高应用程序的开发效率，减少系统的复杂度，因此系统采用SSM框架集（Spring+SpringMVC+MyBatis），此框架集由Spring、MyBatis两个开源框架整合而成（SpringMVC是Spring中的部分内容），它常作为数据源较简单的web项目的框架。将整个系统划分为表示层（View层）、控制层（Controller层）、服务层（Service层）、数据库访问层（Dao层）四个大层，下面将详细介绍这三个框架。

2.2.1 Spring框架

Spring框架是Rod Johnson为解决EJB开发存在的一些问题所提出的一种轻量级框架。Spring就像是整个项目中装配bean的大工厂，在配置文件中可以指定使用特定的参数去调用实体类的构造方法来实例化对象。也可以称之为项目中的粘合剂。 Spring的核心思想是IoC（控制反转），即不再需要程序员去显式地new一个对象，而是让Spring框架来完成这一切。

Spring MVC[2]属于SpringFrameWork的后续产品，是Spring框架中用于WEB快速开发的一个模块，分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。SpringMVC在项目中拦截用户请求，它的核心Servlet，即DispatcherServlet，承担中介或是前台这样的职责，将用户请求通过HandlerMapping去匹配Controller，Controller就是具体对应请求所执行的操作。SpringMVC相当于SSH框架中struts。为了简化开发还提供了注解方式, 只需要在java类上写上相应的注解名, SpringMVC就会对其进行扫描并自动注入。

2.2.2 MyBatis & MyBatis-plus框架

MyBatis是一个基于Java的数据持久层框架，包括api接口、数据处理层、基础支持层三部分。区别于Hibernate的全自动化，MyBatis框架通过XML得到sessionfactory，由sessionfactory产生session，在session中对数据进行增删改查。程序员通过xml配置文件来规定SQL语句，便于理想化管理。

Mybatis-Plus（简称MP）是一个 Mybatis 的增强工具，在 Mybatis 的基础上只做增强不做改变，为简化开发、提高效率而生。它已经封装好了一些crud方法，我们不需要再写xml了，直接调用这些方法就行，就类似于JPA。

2.3前端框架

JSP[3]（全称Java Server Pages）是由Sun Microsystems公司主导创建的一种动态网页技术标准。JSP部署于网络服务器上，可以响应客户端发送的请求，并根据请求内容动态地生成HTML、XML或其他格式文档的Web网页，然后返回给请求者。JSP技术以Java语言作为脚本语言，为用户的HTTP请求提供服务，并能与服务器上的其它Java程序共同处理复杂的业务需求。

Bootstrap是美国Twitter公司的设计师Mark Otto和Jacob Thornton合作基于HTML、CSS、JavaScript 开发的简洁、直观、强悍的前端开发框架，使得 Web 开发更加快捷。Bootstrap提供了优雅的HTML和CSS规范，它即是由动态CSS语言Less写成。Bootstrap一经推出后颇受欢迎，一直是GitHub上的热门开源项目，包括NASA的MSNBC（微软全国广播公司）的Breaking News都使用了该项目。国内一些移动开发者较为熟悉的框架，如WeX5前端开源框架等，也是基于Bootstrap源码进行性能优化而来。

2.4 shiro框架

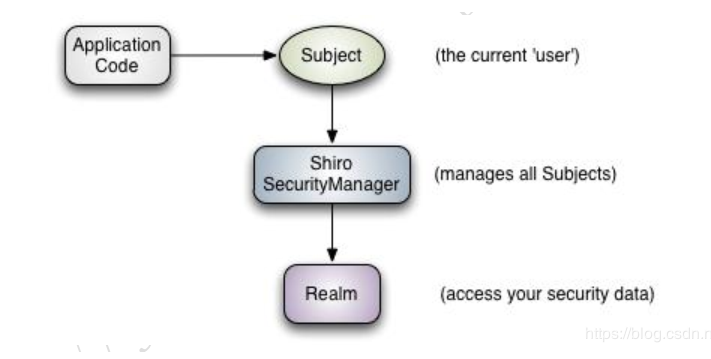


图3.3 shiro框架实现原理

Shiro[4]是Java的一个安全框架，由图3.3可见shiro框架实现原理。目前，使用 Apache Shiro的人越来越多，因为它相 当简单，对比 Spring Security，可能没有 Spring Security 做的功能强大，但是在实际工作时 可能并不需要那么复杂的东西，所以使用小而简单的Shiro 就足够了。

最简单的一个 Shiro 应用：应用代码通过 Subject 来进行认证和授权，而 Subject 又委托给 SecurityManager； 我们需要给 Shiro 的 SecurityManager 注入 Realm，从而让 SecurityManager 能得到合法的用户及其权限进行判断。

2.5 Lombok插件

Lombok项目是一个Java库，它会自动插入编辑器和构建工具中，Lombok提供了一组有用的注释，用来消除Java类中的大量样板代码。仅五个字符(@Data)就可以替换数百行代码从而产生干净，简洁且易于维护的Java类。

2.6开发环境

JDK，全称Java Development Kit,也叫做java开发工具包。它包括了Java编译器、JVM、不计其数的Java工具以及Java基础API里面的Java类库和Java的语言规范。有了JDK再加上一个编辑器(比如Eclipse等等) ，就可以使用java语言写程序。

IDEA 全称IntelliJ IDEA，是java语言开发的集成环境，IntelliJ在业界被公认为最好的java开发工具之一，也是AES算法标准的主要竞争者，其安全性已经在国际密码年会上被证明。

它是一个开源的web应用服务器，本项目中用于运行servlet和jsp页面。

MySQL[5]是一个关系型数据库管理系统，原本是一个开放源码的关系数据库管理系统，在过去有着性能高、成本低、可靠性好的优点，已经成为最流行的开源数据库，因此被广泛地应用在Internet上的中小型网站中。随着MySQL的不断成熟，它也逐渐用于更多大规模网站和应用。

第3章 系统需求分析

3.1设计背景

随着社会上各个公司内部的发展持续性增强，企业的用车需求方面也逐渐增多，通过对各类车辆的监管，企业在用人管理车辆方面也有较大的需求，伴随着车辆的价格越来越可观，企业日常处理一些事务免不了要用车，管理车辆要用人，租用车辆也需要司机，借此管理人事、车辆的效率也在根据需求逐渐提升，因此推动了国内用车管理系统的迅猛发展。

3.2目标设计要求

公务用车系统是一个利用网络通信技术、身份认证登录技术等，提供给有需求的人们的智能处理租赁用车数据的统一平台。相比于常规的信息管理系统，企业公务用车与之有着异曲同工之妙，但也具有其他截然不同的需求，需要在编写代码进程中始终结合企业自己的优缺点，完善在用车过程中所进行的一系列操作管理，并且要与日常信息管理系统区分开。设计系统前查询了网络上的资料[6]，了解了了企业公务用车的实际需求，只有满足了公司用车的实际需求，才能更好地解决企业用车现存的问题。

此系统旨在为用户提供方便查看用车信息的平台，用户大致分为三类：用车公司、司机、管理员。

3.3系统用例图（标题太长56个字即可，图要有引用，对图进行说明，图前面没有缩进的居中）

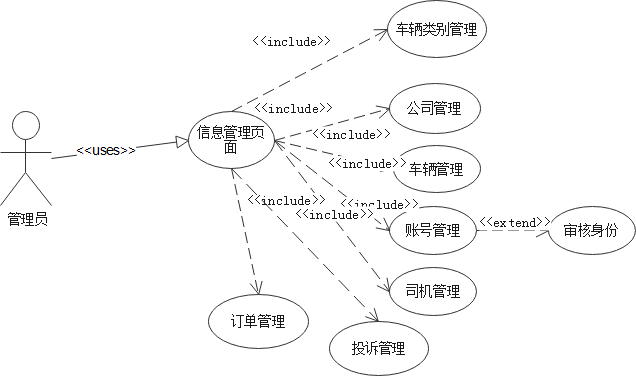


图3.1管理员用例图

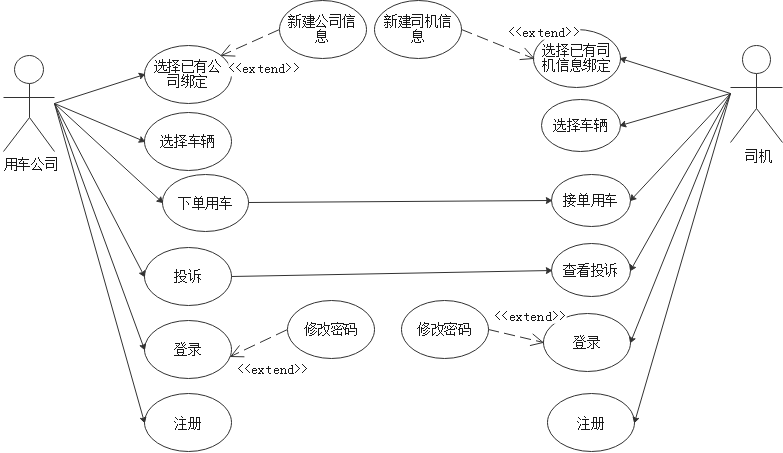


图3.2用车公司和司机用例图

内容有点少，解释一下有哪些车辆类别啥的，解释很多……一两句话描述一下管理员用例图如图3.1和图3.2所示，本系统的主要使用者有三个，分别为用车公司、司机、管理员，管理员负责管理所有用户的信息，包括订单信息投诉信息，个人信息等。用车过程主要是公司和司机的操作，用车结束后管理员根据用车情况处理后续的投诉。

3.4系统数据流程图

企业公务用车系统的总体流程为不同的用户登录系统后，系统会分别展示对应角色的列表，不同的角色有着不同的权限。登录用户的角色为用车公司或司机时，可进行个人信息提交，选择车辆等操作；登录用户为为管理员角色时，则可对后台数据进行相应的增、删、改等操作。如图3.3系统整体数据流程图。

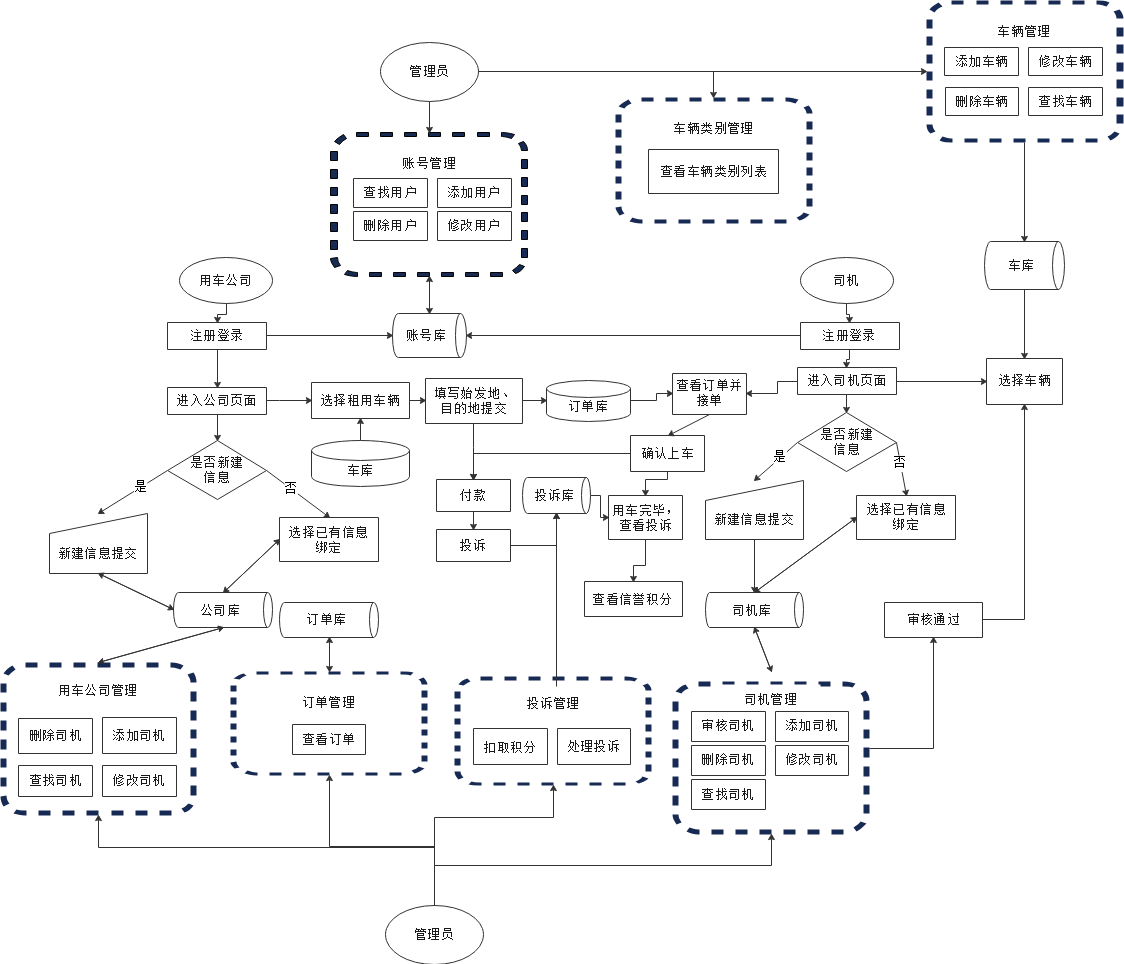


图3.3系统整体数据流程图

3.5系统功能性需求

（从用户的角度自然语言描述大体功能，什么数据结构算法，只谈功能，描述清楚功能就行）

3.5.1账号管理模块

账号管理模块为系统设计的基础部分，用户可以通过注册来选择自己的身份为司机或者是用车公司，然后登陆进入系统，shiro框架设计的部分参考的[9]。管理员可以登陆进入后台维护用户的信息，用户还可以不通过注册直接联系管理员添加新用户，管理员添加后用户可输入用户名和密码直接登录。当用户需要修改角色时也可以联系管理员修改，本系统规定用户自己不可随意更改角色。

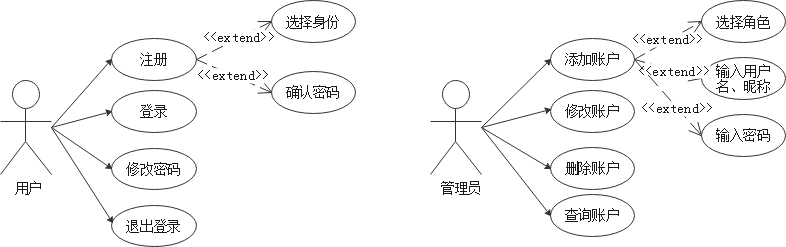


图3.4账号管理模块用例图

3.5.2审核司机模块

审核司机模块为下单模块做铺垫，如果司机无法审核通过就无法拥有车辆，用车公司也就无法选择车辆下单，只有司机提交的信息通过审核才可以正式进入司机库被选择。管理员可以增删改查司机的信息，无论是司机自己提交或者选择已有司机信息绑定都需要通过管理员审核才可以成为正式司机进行选车。

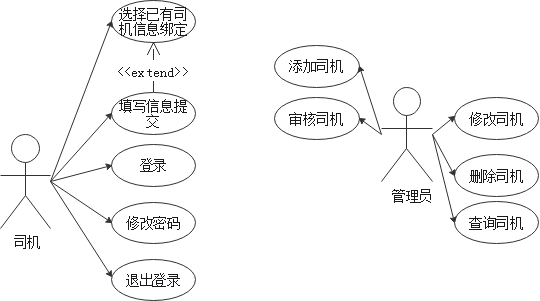


图3.5审核司机模块用例图

3.5.3下单模块

用车公司有意向用车时，下单，司机查看订单去指定地点接人，确认接到人后服务，服务完成后用车公司付款。管理员在用车全程中可查看订单，但无法操作。

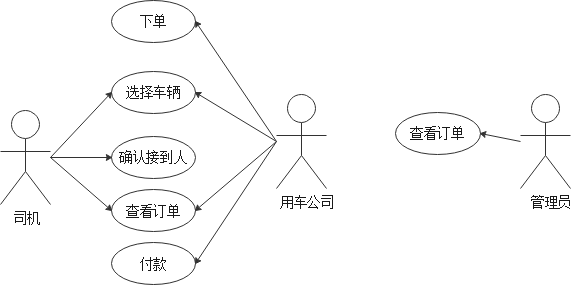


图3.6下单模块用例图

3.5.4投诉模块

用车公司用车完成后，可以根据实际情况投诉司机，投诉完成后可以查看投诉处理情况和投诉信息，管理员查看订单审核完毕后，按情况扣取司机的信誉积。司机可以查看本次订单的投诉情况和自己信誉积分扣除的情况。

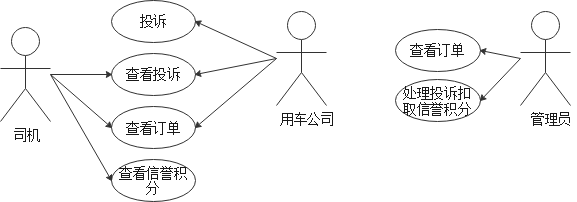


图3.7投诉模块用例图

3.6 系统非功能性需求

作为功能需求的补充，非功能需求是指那些不直接与系统的具体功能相关的一类需求，但它们与系统的总体特性相关，如可靠性、响应时间、存储空间等。非功能需求定义了对系统提供的服务或功能的约束，包括时间约束、空间约束、开发过程约束及应遵循的标准等。增强信息管理的实用性，已成为现阶段做出一个优秀且强大系统的重要目标。

（1）完整性特征

系统的重要部分是各种信息的储存，该管理系统需要随时确认车辆和用户的相关信息，因此在项目开发初期阶段，需要考虑信息的完整性，使用建立系统的自动弹出窗口和警告功能。当使用不合适的数据时，系统会根据所输入的数据规则显示弹出窗口，提示用户哪个数据存在错误。

（2）实用性特征

用户监管服务[7]变成上标X2可以体现企业办公车管理系统的现代化水平和科学化水平，并根据具体用户提供多种系统需求，可以有效实现计算机程序设计替代传统的低效管理模式。该管理系统的管理员是提供租车平台的网络管理员，用户是企业有用车意向的职工以及拥有车辆使用权的司机。为了方便用户使用，功能尽可能言简意赅，而管理人员为了管理方便，功能要求尽量全面。

（3）功能的拓展性

企业公务用车管理系统管理数据的过程非常冗杂[8]，在系统编码的研究开发阶段，要理解现代企业的持续发展会带来许多要求，所以系统功能的可扩展性应该放在任何一个重要位置上。

（4）安全性

从系统中存储的无数个人信息、牌照号码以及其他不能公开阅览的数据观点来看，不同的认证必须为用户的不同设计，不同的使用权必须不同对待，在不同级别的用户可以只授予一个级别的权威，系统的权限管理模块必须包含在设计里面，且为系统设计的重中之重。

（5）便捷性

鉴于人与计算机交互的接口简单适用性，要求必须完成人与计算机的交互设计，并且要满足用户对简单实用操作的要求。

3.7系统用户权限需求

表3.1 角色权限表

|  |  |
| --- | --- |
| 用户 | 操作权限 |
| 司机 | 注册、登录、选择车辆、完善个人信息、改查订单、查看投诉及后续处理 |
| 用车公司 | 注册、登录、选择车辆、增改查订单、完善个人信息、付款、投诉 |
| 管理员 | 登录、查看车辆分类、管理车辆信息、管理公司信息、管理司机信息（包含审核司机）、增删改查账号信息、查看订单、处理投诉 |

第4章 系统总体设计（概要设计）大体用自然语言描述或表怎么算法，模块划分的不太对。每个模块都要有流程图，描述数据结构（用户—用户名、密码啥的）算法结构类，语言描述）数据库表建议用图，图要清楚

4.1系统总体架构（要有大图）

用车公司：通过选择车辆下单用车，完成订单后可以投诉司机。

司机：选择车辆完毕后，等待用车公司用车，完成订单后进行下一单，期间查看之前的投诉情况。

管理员：管理用户信息，车辆信息，订单信息等等，监控每一单用车的完成情况。

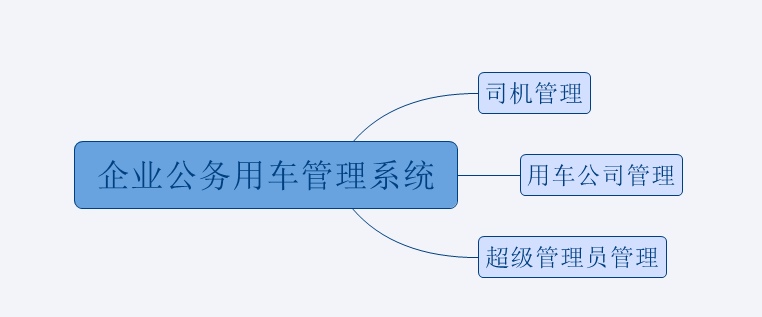


图4.1系统总体结构

4.2 系统主要功能模块设计（画小流程图）

4.2.1司机管理模块

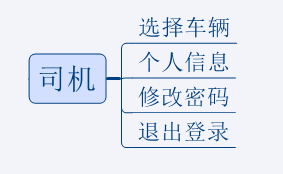


图4.2司机管理模块

司机管理模块主要包含选择车辆、个人信息、修改密码和退出登录，司机提交个人信息管理员审核通过后才可以选择车辆。

（1）选择车辆页面：司机通过注册后（注册时选择自己的角色是司机）选择车辆，若当前无车辆或车辆都已被选中，则超级管理员进入后台管理添加车辆，司机再次进入系统可选择车辆，每个司机只能选择一辆车，选择后不可更改。

（2）个人信息页面：

①注册登录：司机注册后登录进入个人信息界面，系统会根据登录的用户角色不同展示不同的界面，进入后首先需要完善自己的信息，或者从司机列表里选择自己的信息关联，提交后，后台管理员审核完毕才可以成为正式的司机进入企业可选司机列表。

②订单管理：若有企业选中司机拥有的该车并提交订单，司机可以进入系统查看订单并去指定地点接客，接到客人后点击接到客人，状态自动改为已接客。完成订单后，若用车公司投诉，可查看投诉。若管理员处理了投诉信息，可在此界面查看自己信誉积分。

③修改个人信息：司机可以修改自己的信息（除了自己的身份）以及修改密码。若想修改身份，联系管理员修改用户信息即可。

4.2.2用车公司管理模块

用车公司注册登录后可以修改密码，用车时选择服务，服务根据司机的状态分为三个模块，然后可以修改个人信息提交公司信息，提交后管理员才可以看到公司。

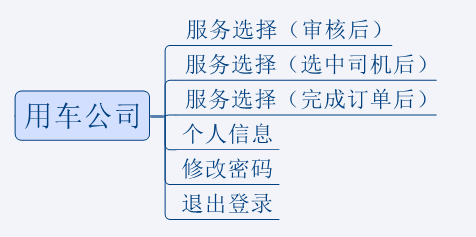


图4.3用车公司管理模块

4.2.3管理员管理模块

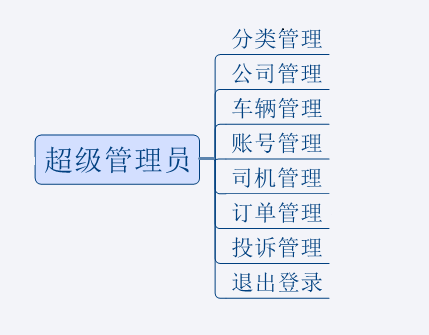


图4.4管理员管理模块

管理员负责统筹管理整个系统的所有信息，包括公司、司机的信息，还有车辆分类和车库信息，用车公司在用车时管理员还可以查看订单信息，用车结束管理员需要处理投诉信息发送给司机和用车公司。

（1）分类管理：此系统预设了5种车的类型，分别为入门级、消费级、普通级、豪华级、奢华级，分别有不同的价值和使用时的价格，可供用车公司比较选择。

（2）公司管理：若公司账号注册登录后未提交公司信息，在此模块无法看到。只有公司提交信息后，管理员方可在此看到此公司的信息，包括选车的状态等等。管理员还可以在此添加删除新公司以及修改用车公司的负责人和邮箱。

（3）车辆管理：此模块展示了所有车辆的各种信息，包括未选中的正在使用的等等，管理员可以增删改查车辆。若车辆过多，可以通过面板右上角的搜索功能来查找需要的车，可供查找的有品牌和颜色。

（4）账号管理：管理员可以增删改查所有用户的信息，每个用户可绑定一个司机或者一个用车公司（绑定后不可修改），也可赋予账号“管理员”的角色，只有admin超级管理员账号信息默认不可修改。

（5）司机管理：与公司管理相差无几，在此不做赘述。

（6）订单管理： 只能查看订单的信息，无法修改添加删除。

（7）投诉管理：查看用车公司投诉司机情况，根据描述管理员需扣取司机的信誉积分，处理完毕后，状态变为已处理。

4.3 系统功能流程

(1) 司机管理：司机进行注册，录入司机的个人信息，根据司机是否有车进行分类，无车司机需要选择系统里提供的车，有车司机则需联系管理员选择车辆信息绑定。提交后，发送至人工审核模块，成为可用状态的司机增加信誉积分栏目，管理员可根据派单模块和投诉模块的反馈对信誉积分进行增减操作。

(2) 人工审核：在审核员确定审核通过后，司机进入可用状态，生成司机编号添加到司机库（添加数据中含有审核员编号、审核时间和车辆层次）并发往司机管理模块。

(3) 用车公司：用车公司进行注册，注册成功后，进入管理界面进行设置其公司员工需要用车的时间、地点、车辆层次及员工的手机号，进行在线缴费并生成交易单号，发至派单模块，如司机有迟到、加价等违规行为则可进入投诉模块。

(4) 派单：针对车辆档次和司机信息发送派单信息，派单成功后，生成订单号发送至司机管理模块和用车公司模块，系统司机的手机号和车牌号码发送至用车人的手机上并将用车人的手机号发送至司机的手机上，司机前往指定地点进行接客，待用车人上车后，司机进行确认上车，订单成为进行状态，送至用车人指定地点后进行订单完成，订单此时成为完成状态，系统根据司机行程的公里数、路上使用的时间和信誉积分计算出司机应得的收益，并添加对应的信誉积分至司机管理模块。

(5) 投诉：根据生成的订单号查询司机，如有迟到、加价或车型不符等行为，则扣除司机的信誉积分，并发送警告通知给司机。

4.4数据库设计

4.4.1数据库设计实现

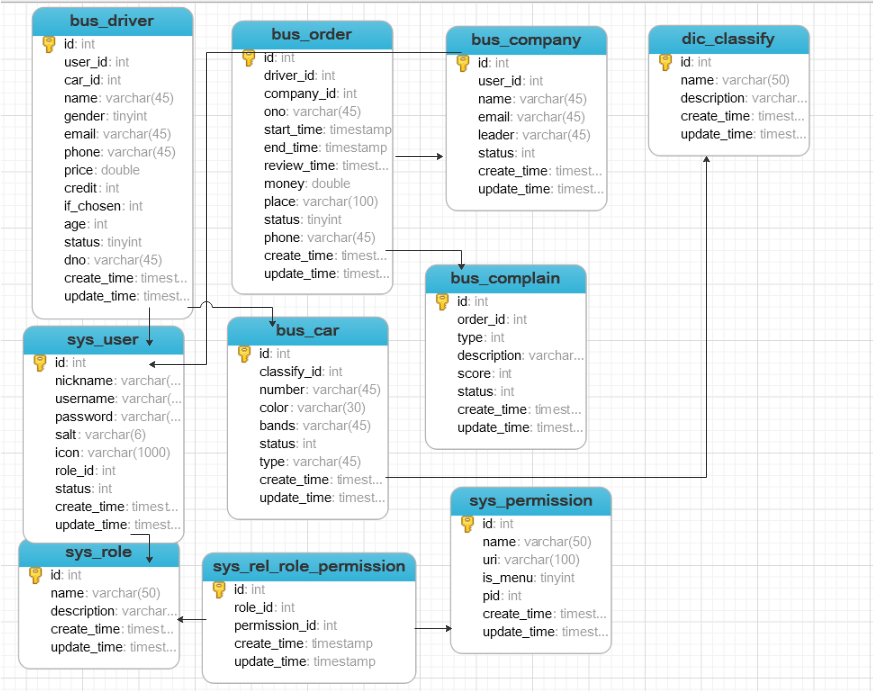


图4.5数据库整体设计

本系统针对每个角色模块功能的独特性, 为系统的每个角色的模块进行数据库的设计。如图3.2公务用车管理模块数据模型所示, 该模块中有10张表，彼此相互依存，为的是降低系统开发的复杂程度, 方便后期数据库的维护，实现了数据的持久化。它们分别存储企业在用车过程中涉及到的信息，下面会通过实体属性图详细介绍。

（1）sys\_user 用户表，存储了用户的id，用户名username，昵称nickname，对应的角色id，加密后的密码password，随机6位字符salt，头像路径icon，状态status等。

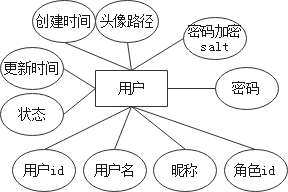
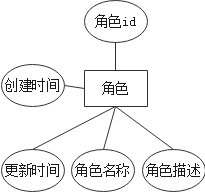


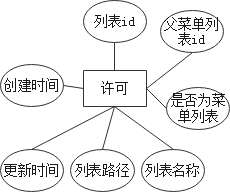
图4.6用户属性图

（2）sys\_role 角色表，分别存了三种用户的id、名称name、描述description等。



4.7角色属性图

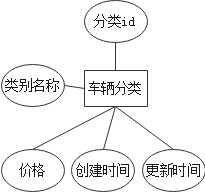
（3）sys\_permission许可表，存储了列表id，列表名称name，对应路径uri，是否为主菜单列表is\_menu，父菜单列表pid（若列表是主菜单为0）等。



4.7许可属性图

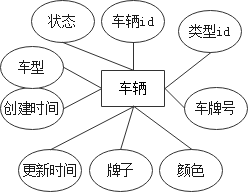
（4）sys\_rel\_role\_permission角色许可表，存储了角色许可id，角色id，对应每个角色对应许可表里列表pemission\_id等。

（5）dic\_classify车辆类型表（共5种类型），存储了类型id，类型名name，类型价格描述description等。



4.8车辆分类属性图

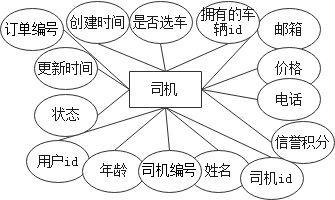
（6）bus\_car公务车辆表，存储了车辆id，类型classify\_id，车牌号number，颜色color，牌子bands，状态status，车型type等。



4.9车辆属性表

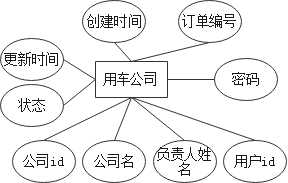
（7）bus\_driver司机表，存储了司机id，对应的用户id，车辆id，名字name，价格price，

信誉积分credit，是否被选中if\_chosen，状态status，编号dno等。



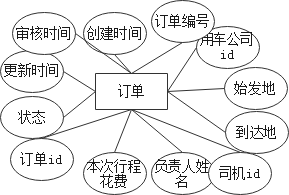
4.10司机属性表

（8）bus\_company用车公司表，存储了公司id，对应用户id，公司名name，负责人姓名leader，用车状态status，订单编号ono等。



4.11用车公司属性表

（9）bus\_order订单表，存储了订单id，司机id，用车公司id，订单号ono，开始用车时间start\_time，结束用车时间end\_time，审核时间review\_time，本次行程花费money，始发地phone，到达地place，订单状态status等。



4.12订单属性表

4.4.2 逻辑机构设计（实体关系+实体属性）

物理结构设计（每个表的设计，每个字段解释说明）

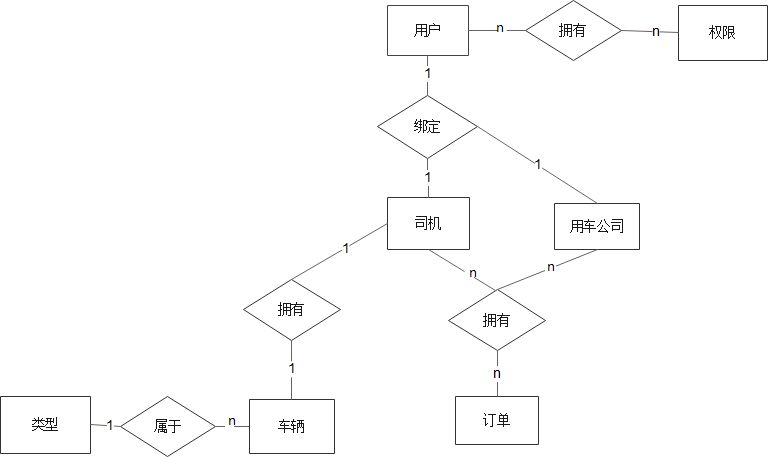


图4.6系统E-R图

4.5本章小结

本章从系统的整体设计开始，通过介绍主要模块来详细介绍系统的设计理念，最后通过数据库每个表间的联系和实体间的联系来建立整个系统数据的架构。

第5章 系统详细设计

（分功能模块）

5.1系统总体详细设计

本系统将充分考虑用户的需求, 并结合时下流行的SSM框架，(Spring MVC、Spring、Mybatis) 、JSP动态网页技术等对系统架构进行良好的设计、功能进行合理划分并实现。

系统主要涉及到用车公司、司机、超级管理员三个用户，针对每一个用户设计了不同的页面列表，这个页面根据登录用户的不同展示不同的列表，用车公司和司机在登录页面点击注册进入注册页面，注册完毕后可登陆进入系统，系统会根据登录的角色来展示不同的页面，同时页面也会有不同的功能供用户使用，整个系统的架构由图5.1系统的总体设计图所示。其中超级管理员的功能最为强大，它可以任意增删改查司机、用车公司、用户和车辆的信息，还可以查看用车过程中订单的状态、完成订单后的投诉情况等等，下面会一一讲解。

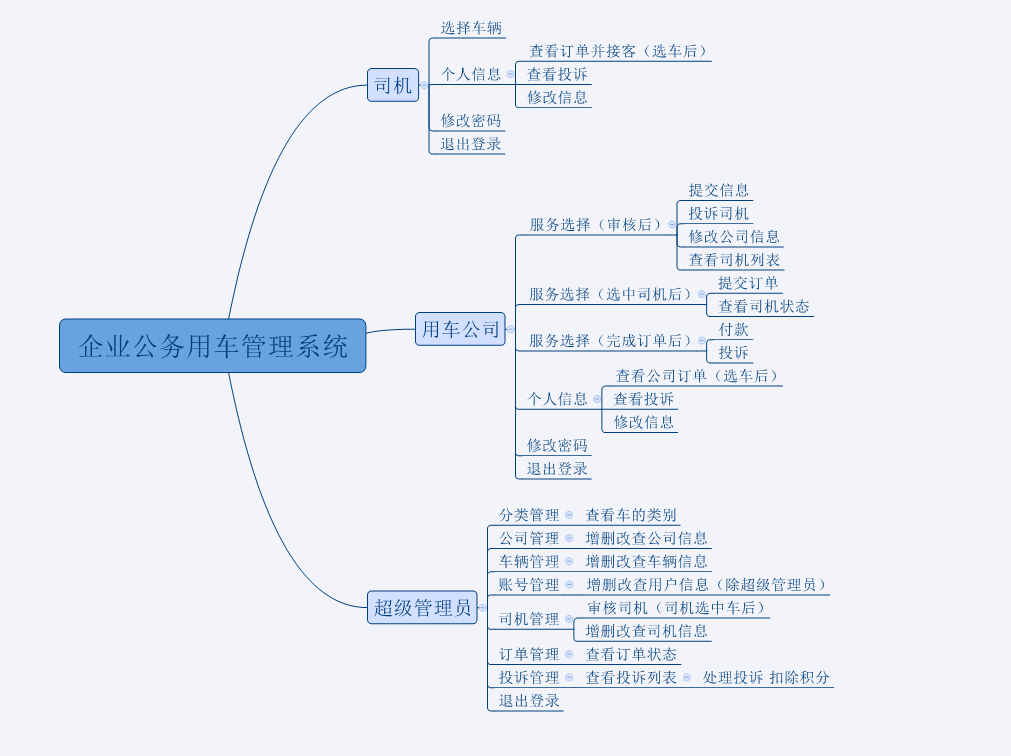


图5.1系统总体详细设计图

5.2系统主要功能模块设计

5.2.1注册登录模块设计（算法用伪代码形式表示，别怕重复）

注册登录模块的功能是用户通过注册录入信息，转而进行登录进入系统。注册登录主要的核心类有：AuthController类、UserServiceImpl类、UserService接口、Service接口、ServiceImpl类、UserMapper接口、User类、Role类。注册登录模块核心类的类图如图5.2所示。

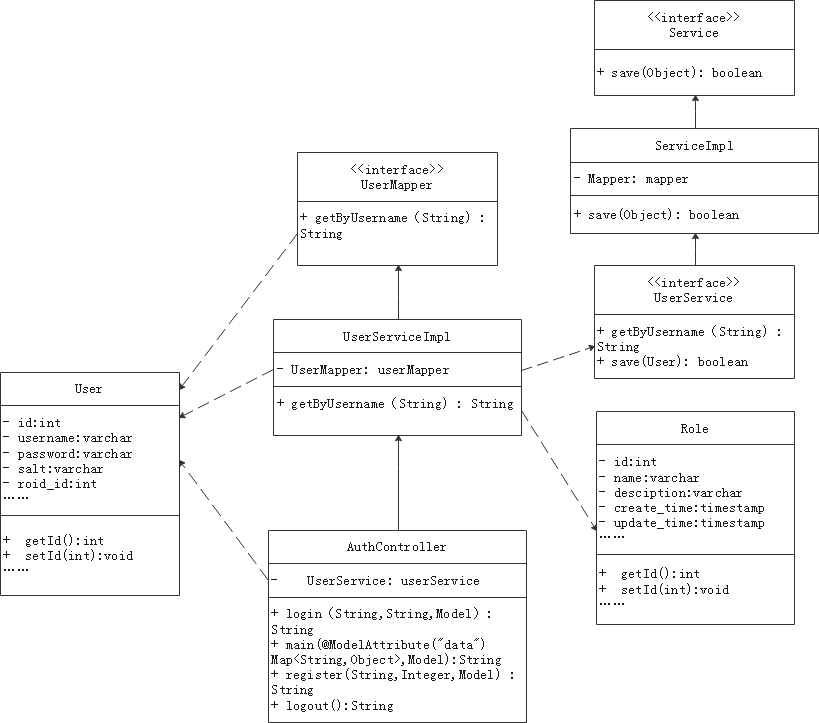


图5.2注册登录模块类图

AuthController类是属于控制层的类，用来处理登录注册进行查询添加的 各类请求，通过调用 服务层处理具体的业务逻辑，Service内部调用Dao层根据mapper配置文件从数据库内读取数据，最后将处理结果返回给浏览器的前端jsp页面，页面通过JavaBean等进行展示。类中的主要方法有： 判断用户名密码是否正确login()，跳转到首页main()，注册完毕后保存数据register()，退出登录logout()。

UserServiceImpl 类是属于业务逻辑层层的类，用来处理和登录注册有关的增删改查业务。类中的主要方法有：根据输入的用户名获取所有用户信息getByUsername，保存所有注册完毕的用户数据save()。Service和ServiceImpl均为Mybatis-plus提供的类，可直接继承使用它里面的方法。

UserMapper 接口用来与数据库进行交互。接口中的方法与Service相同。

User类、Role类属于实体类，可以在数据库实体属性图中查看各个属性，在此不做赘述。

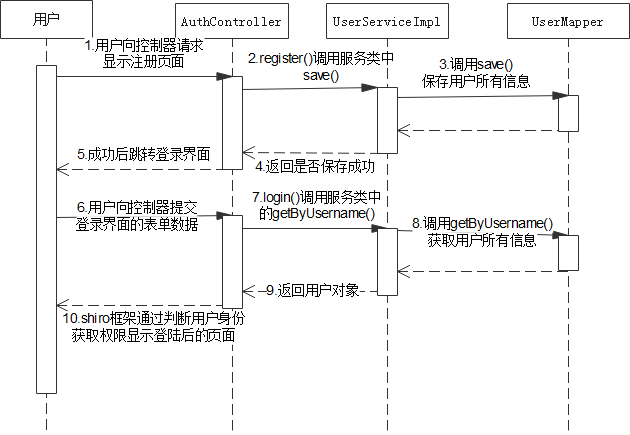


图5.3登录注册模块时序图

5.2.2审核司机模块设计

审核司机模块的功能是用户通过录入个人信息提交，然后管理员登录进行审核。审核司机模块主要的核心类有：JumpController类、BusDriverController类、UserController类、UserServiceImpl类、BusDriverServiceImpl类、BusDriverMapper接口、User类、BusDriver类。审核司机模块核心类的类图如图5.4所示。

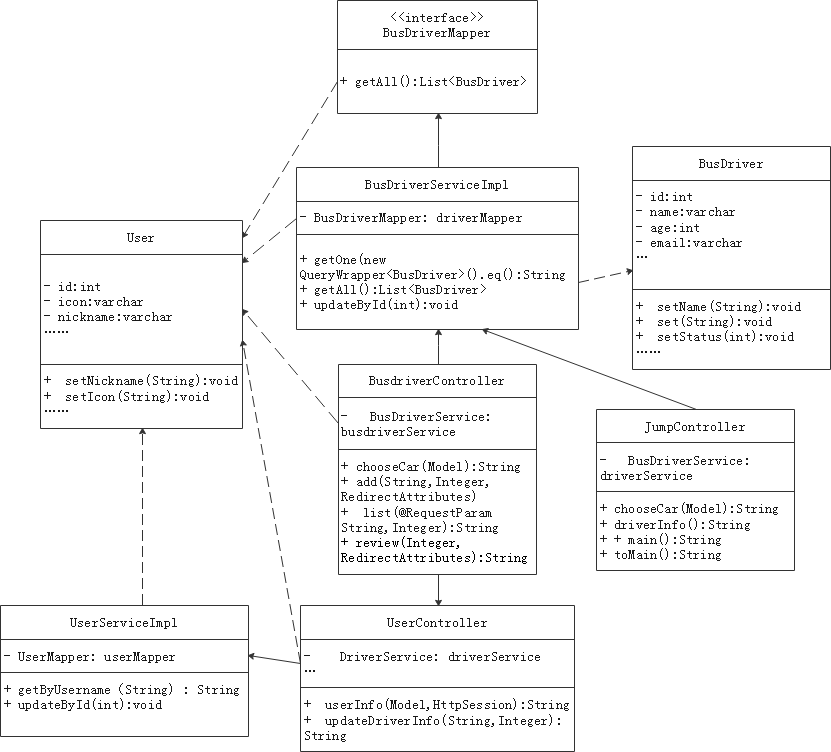


图5.4审核司机模块类图

JumpController类是属于控制层的类，用来处理登录后跳转到对应页面的请求，类中的主要方法有：跳转到选择车辆页面chooseCar、跳转到司机主页面driverInfo()。UserController类封装所有用户进行的操作，主要方法有：展示司机的操作列表userInfo。BusdriverController类主要用来管理员和司机更新司机不同阶段的信息，用到的方法有：选择车辆chooseCar、司机新建自己的信息add、展示司机列表给管理员list、审核司机review。

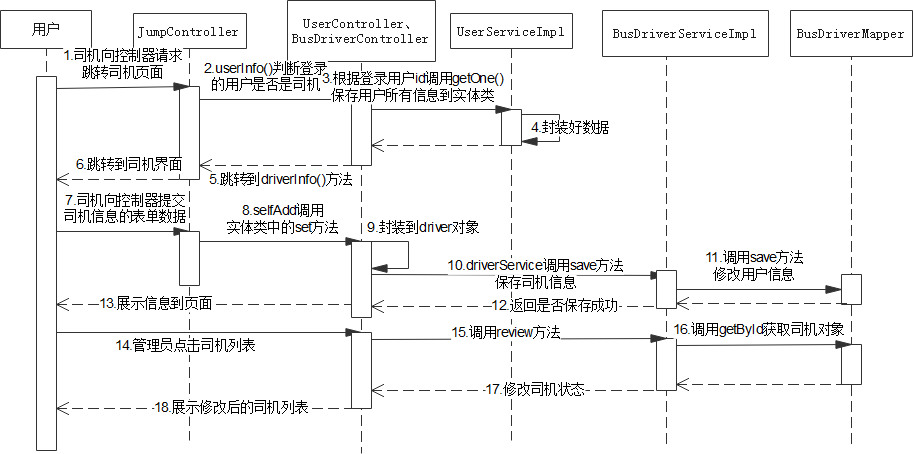
BusDriverServiceImpl 类中的主要方法有： 根据登录用户的id获取司机对象getOne、获取所有司机的信息getAll()、审核后更新司机id updateById。

图5.5审核司机时序图

5.2.3下单模块设计

下单模块的功能是用车公司通过录入用车提交，然后司机进行接客人等操作完成订单。下单模块主要的核心类有：JumpController类、BusDriverController类、UserController类、GoodsController类、BusOrderServiceImpl类、BusDriverServiceImpl类、BusDriverMapper接口、BusOrderMapper接口、User类、BusDriver类、BusOrder类。审核司机模块核心类的类图如图5.6所示。

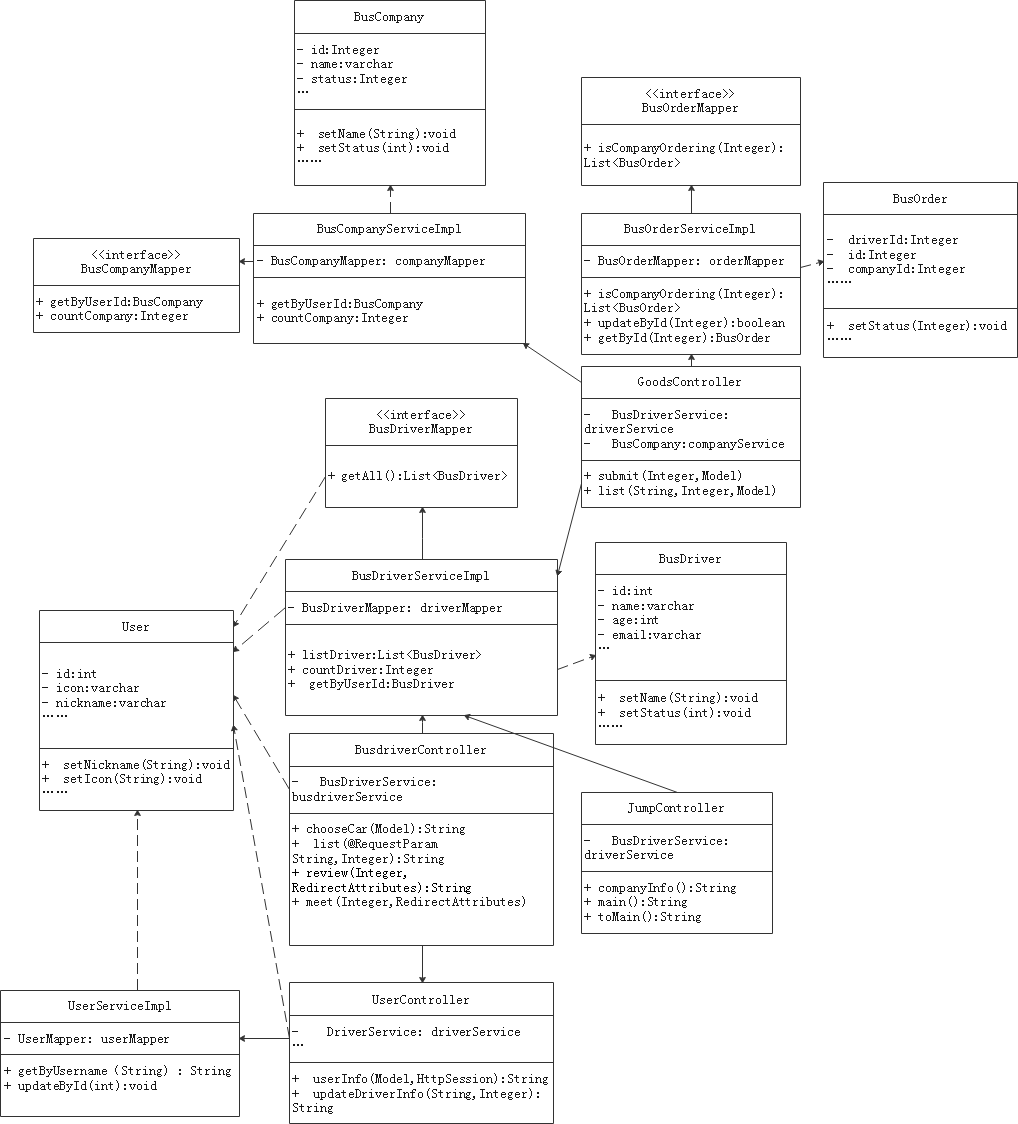


图5.6下单模块类图

JumpController类中的主要方法有：跳转到用车公司主页面companyInfo()、跳到管理员主页面main()、根据登录用户跳转到主页面tomain()。UserController类封装所有用户进行的操作，主要方法有：展示公司的操作列表userInfo、更新司机信息updateDriverInfo。BusdriverController类主要用来管理员和司机更新司机不同阶段的信息，用到的方法有： 展示司机信息列表list、更新用车状态meet。

BusDriverServiceImpl、BusOrderServiceImpl、BusCompanyServiceImpl类是属于 Service 层（业务逻辑层）的类，用来处理下单信息相关的业务逻辑。类中的主要方法有：展示列表listDriver、统计数量countDriver有订单则返回订单列表isCompanyOrdering、根据用户id更新order对象getByid。

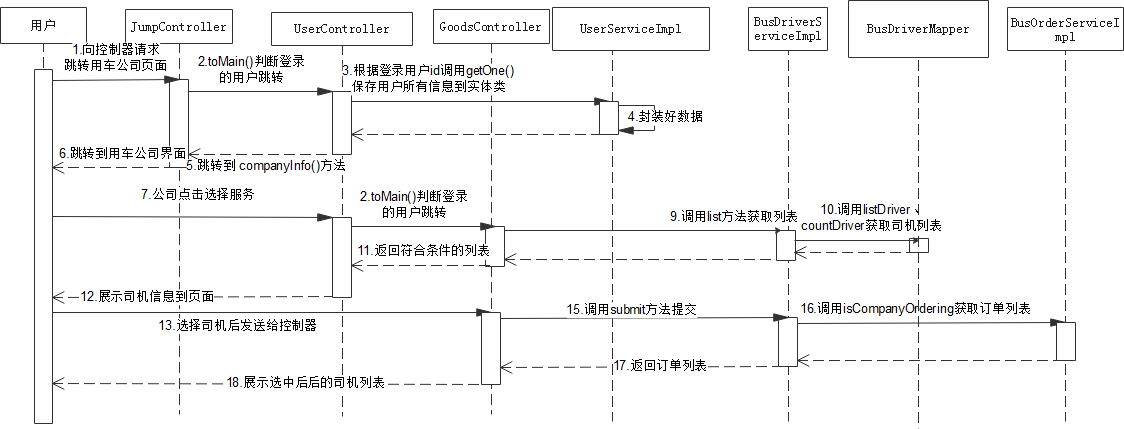


图5.7下单前选车时序图

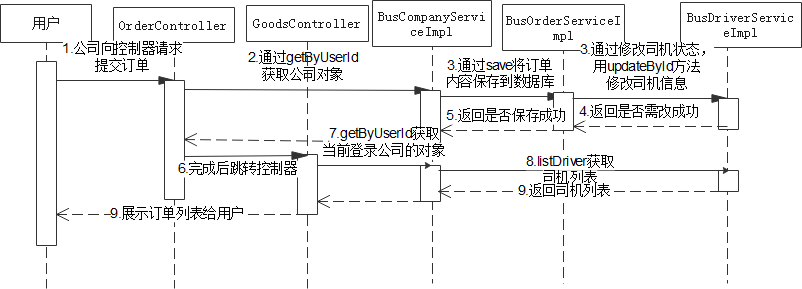


图5.8提交订单时序图

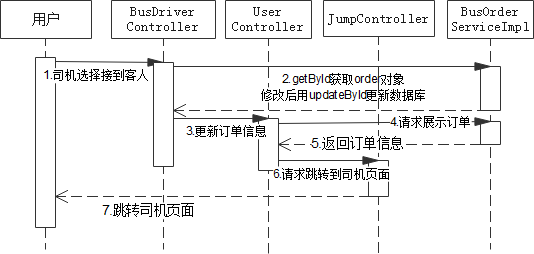


图5.9司机接人时序图

5.2.4投诉模块设计

投诉模块的功能是用车公司用车结束后投诉，然后管理员处理投诉。投诉模块主要的核心类有：BusDriverController类、UserController类、ComplainController、BusOrderServiceImpl类、BusDriverServiceImpl类、BusDriverMapper接口、BusOrderMapper接口、BusComplainMapperUser类、BusCompany类、BusOrder类。审核司机模块核心类的类图如图5.6所示。

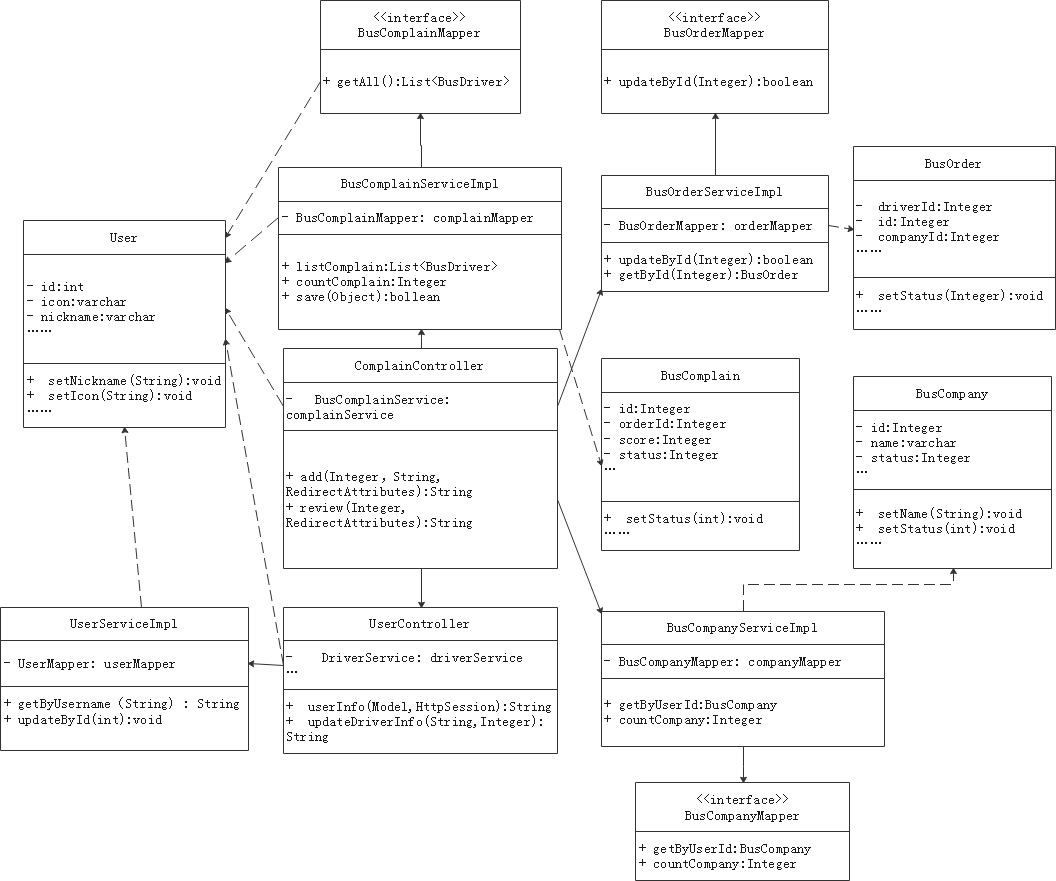
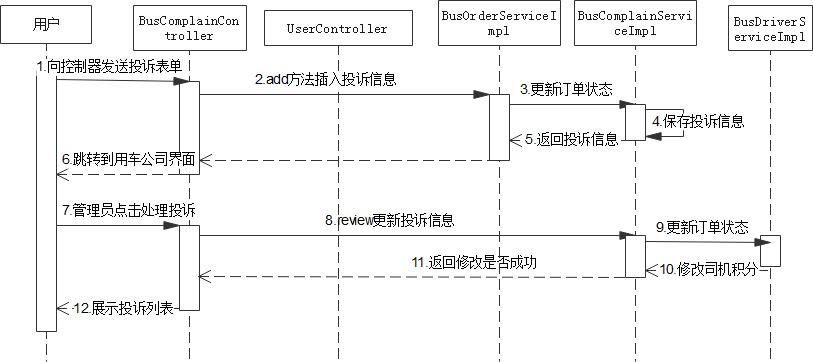


图5.10投诉模块类图

BusComplainController类中的主要方法有：添加投诉信息add、提交投诉信息review。

BusDriverServiceImpl、BusOrderServiceImpl、BusCompanyServiceImpl用来处理投诉信息相关的业务逻辑，类中的主要方法之前相似。

5.11投诉模块时序图

5.3系统活动图设计

5.3.1审核信息活动图

审核信息是企业公务用车管理系统的重要组成部分，这部分要着重设计对个人信息的管理，特别是对司机信息的管理，应该由管理者来运营。审核模块作为整个系统的出发点同时也是比较重要的功能，因为审核通过之后两位用户才可以依次进行后面的操作。活动图见图 3.4和图3.5。

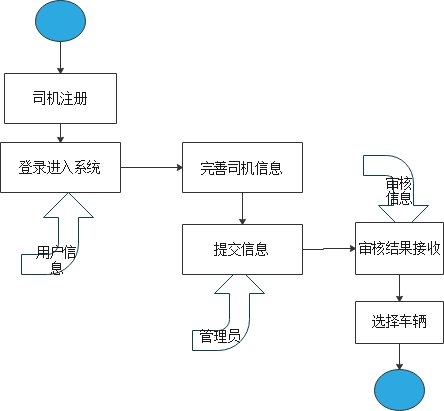


图5.12审核司机信息活动图

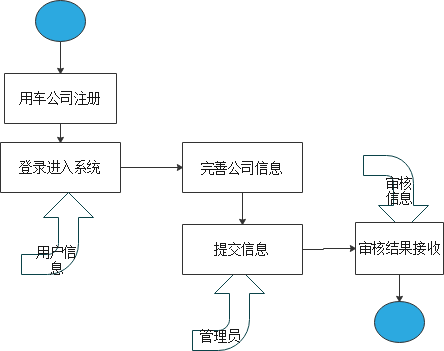


图5.13用车公司信息活动图

5.3.2企业用车活动图

用车是企业的商务车管理系统的中心功能，整个系统的其他功能都是为了服务于用车功能，包括前期对车辆的管理、对车类的管理等等。通过三种用户的共享管理数据模式，完成无缝数据传输功能，增加了系统的实用性。如图3.6和图3.7。

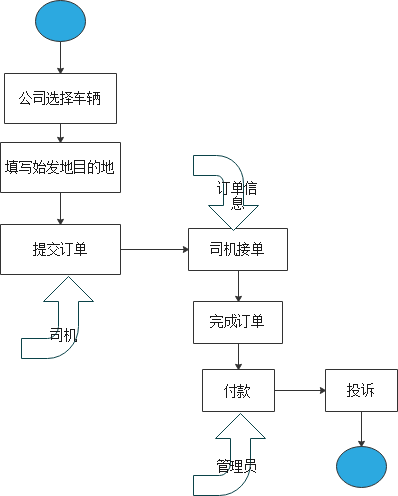


图5.14用车公司用车活动图

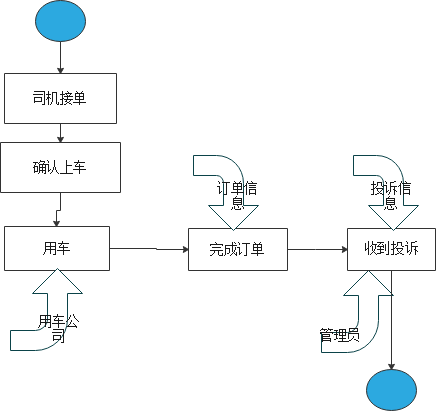


图5.15司机接单活动图

5.4本章小结

本章分模块详细介绍了系统的具体实现，通过类图、时序图和活动图从各个方面讲解了每个模块实现的过程，为接下来完善整个系统做铺垫。

第6章 系统实现

6.1系统环境架构

本系统主要使用了SSM框架架构进行设计，登录权限控制使用了java提供的shiro框架，前端使用了JSP动态网页技术、Javascript、CSS等技术来实现的，数据库使用Mysql数据库, WEB应用服务器使用Tomcat 9.0，如图6.1、图6.2。



图6.1 系统技术架构图

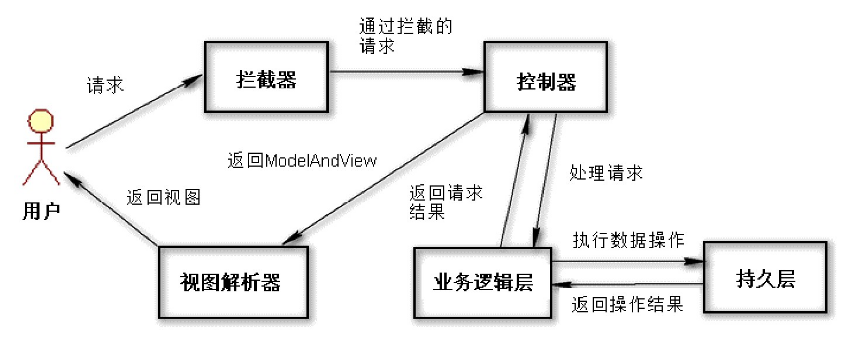


图6.2本系统的SpringMVC工作流程图

6.2系统实现要点

6.2.1配置shiro

由于企业公务用车系统使用的 Spring 框架，要将 Shiro 集成到Web 应用中，新建一个Shiro配置类，通过利用@Configuration注解在配置文件里，springAOP切面编程动态代理实现（在配置类上添加@EnableAspectJAutoProxy注解，开启注解版的AOP功能），配置Shiro最为核心的安全管理器SecurityManager和注册bean。再配置Shiro的过滤器工厂类，将上一步配置的安全管理器注入，并配置相应的过滤规则。

接下来创建最简单的shiro应用：应用代码通过 Subject 来进行认证和授权，而 Subject 又委托给 SecurityManager； 我们需要给 Shiro 的 SecurityManager 注入 Realm，从而让 SecurityManager 能得到合法 的用户及其权限进行判断。

6.2.2实现shiro

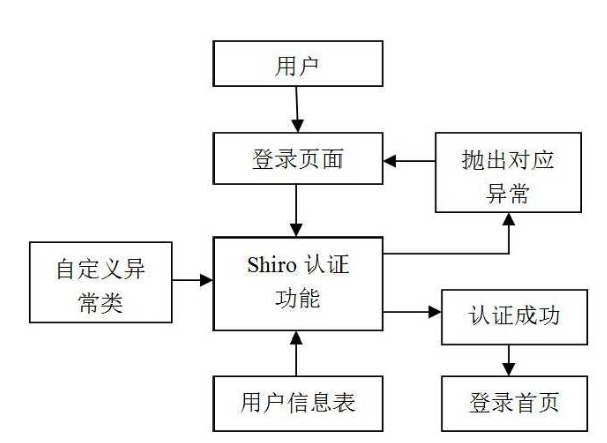


图6.3登录认证模块设计图

参考[10]，首先定义一个自定义标签menu，通过继承SimpleTagSupport类来重写doTag()方法，通过SecurityUtils.getSubject().getPrincipal()抓取登录的用户，根据角色许可表来获取这个角色拥有的列表路径，用getJspContext().getOut().write方法拼接出前台可以识别的列表代码。

6.2.3用户密码机制

用户名密码采用DigestUtils实现md5码加密：这是一个非常好用的使用MD5+salt加密的工具类。Salt通过自己设置的随机字符类，通过产生随机数字，从数组里随机取字符来组成6位字符串。使用这个工具类，从前台拿到密码password，直接MD5Util.getEncryptedPwd(passwd)就可以返回一个长度为56的字符串，可以用来保存到数据库中，相反，登录的时候，因为MD5加密是不可逆的运算，只能拿用户输入的密码走一遍MD5+salt加密之后，跟数据库中password比较，看是否一致，一致时密码相同，登录成功。

6.2.4实体类设计

@Data:注解在类上；提供类所有属性的 get和set方法，此外还提供了equals、hashCode、toString 方法.

@TableName（数据库名），声明此对象映射到数据库的数据表，通过它可以为实体指定表@TableId(type = IdType.AUTO)数据库字段注解，非数据库字段注解 @TableField(exist = false)

6.2.5 Controller设计

每个控制器内的方法最初都要检验一下当前用户的角色是否拥有访问权限。

JumpController：通过@GetMapping注解到判断用户角色和提示需要完善信息的方法，大部分操作都通过这个控制器来跳转到相对应页面。

AuthController：封装用户登录注册退出时的方法，通过用前端输入的用户名和密码与数据库里用户表存的密码和salt同时加密后进行比对，结果一致后将用户的所有权限写成列表输出到页面上。

UserController：封装所有用户关于账户信息的操作，注解@Transactional(rollbackFor = RuntimeException.class)来控制出现异常时的数据回滚。

BusCompanyController：封装用车公司可进行的大部分操作。

BusDriverController：封装司机可进行的所大部分操作。

GoodsController、OrderController、ComplainController、DicClassifyController、BusCarController、OrderController：封装对他们各自的操作。

6.2.6 Service设计

设置接口Service，通过继承mybatis-plus提供的IService<T>接口来获取许多便利的方法，然后设置ServiceImpl实现类，继承 ServiceImpl<Mapper, T>方法来具体实现。

6.2.7 Dao层设计

设置接口Mapper，通过继承BaseMapper<T>类提供的方法可以很方便地封装实体对象，方法与Service几乎一样。

6.3界面展示及对应部分代码

(登录注册少贴，贴少量主要的代码，贴主要控制流程就行)

6.3.1登录界面

图6.4登录界面



图6.5注册界面

1. 前台

login.jsp登录页面

<form action="${pageContext.request.contextPath}/auth/login" method="post"></form>

Register.jsp注册界面

<form action="/auth/register" method="post"></form>

（2）控制器

JumpController.java

@RequestMapping("/auth/")

@GetMapping("/login")

public String toLogin(){

User sUser = (User) SecurityUtils.getSubject().getPrincipal();

if (sUser != null){

return "redirect:/main";//跳转到欢迎登录界面

}

return "auth/login"; }

@GetMapping("/main")

public String toMain(){//跳转到管理员管理的主页面

SecurityUtils.getSubject().checkPermission("/main");

User user = (User) SecurityUtils.getSubject().getPrincipal();

BusCompany company = companyService.getByUserId(user.getId());

BusDriver driver = driverService.getByUserId(user.getId());

if (company != null || driver != null){//否则就是管理员登录

return "redirect:/user/info";

}

return "main/index";

}

AuthController.java

@PostMapping("login")//验证用户名密码是否正确

public String login(String username, String password, Model model){

//省略部分代码

User user = userService.getByUsername(username);

password = DigestUtils.md5DigestAsHex((password+user.getSalt()).getBytes());

UsernamePasswordToken token = new UsernamePasswordToken(username,password);

Subject subject = SecurityUtils.getSubject();

try {

subject.login(token);//验证是否能登录

return "redirect:/main";

}catch (AuthenticationException ae){

model.addAttribute("msg","密码错误");

return "redirect:/login";

}

}

@PostMapping("register")

public String register(String nickname,String username, String password,String rePassword,Integer roleId,Model model){

//省略部分代码

User one = userService.getByUsername(username);

String salt = RandomUtil.getRandomString(6);//获取随机6位字符

User user = new User();

user.setSalt(salt);

user.setPassword(MD5Utils.getMD5(password+salt));//加密密码

boolean res = userService.save(user);//利用类中对象的属性信息来修改数据库

return "redirect:/login";

}

（3）服务层

UserServiceImpl.java

@Service

public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements UserService {

@Autowired

private UserMapper userMapper;

@Override

public User getByUsername(String username) {

return userMapper.getByUsername(username);

}

（4）Dao层配置文件

User.xml

<mapper namespace="xyz.jfshare.car.system.dao.UserMapper">

<select id="getByUsername" resultType="xyz.jfshare.car.system.entity.User">

select \* from sys\_user where username = #{username}

</select>

</mapper>

6.3.2管理员管理界面



图6.6管理员界面

1. 前台页面

<aside class="main-sidebar">

<section class="sidebar">

<ul class="sidebar-menu" data-widget="tree">

<mytag:menu/>//展示侧拉列表的目录

</ul>

</section>

</aside>

1. 控制器

JumpController.java

@GetMapping("/")

public String main(){

return "redirect:/main";

}

@GetMapping("/main")

public String toMain(){

SecurityUtils.getSubject().checkPermission("/main");

User user = (User) SecurityUtils.getSubject().getPrincipal();

BusCompany company = companyService.getByUserId(user.getId());

BusDriver driver = driverService.getByUserId(user.getId());

if (company != null || driver != null){//否则就是管理员

return "redirect:/user/info";

}

return "main/index";

}

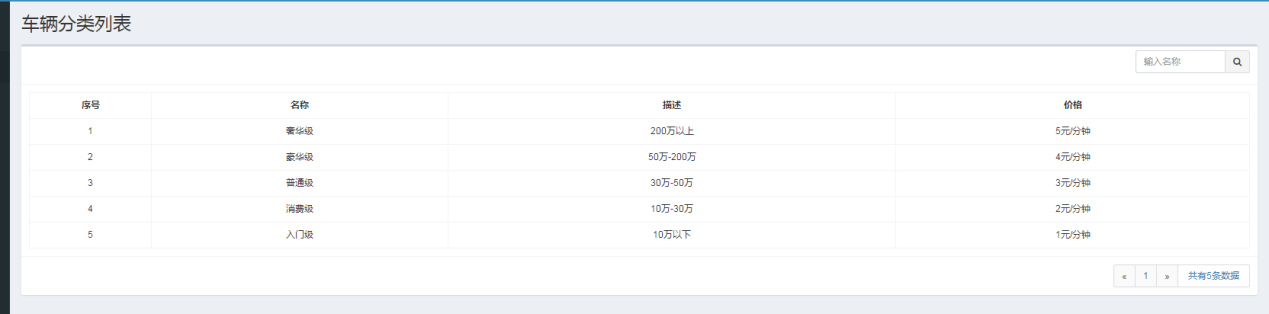


图6.7车辆管理界面

DicClassifyController.java

@RequestMapping("/classify/")

public class DicClassifyController {

@RequestMapping("list")//展示分类列表

public String list(@RequestParam(required = false,defaultValue = "")String name,

@RequestParam(required = false,defaultValue = "")String description,

@RequestParam(required = false,defaultValue = "1") Integer page,

@RequestParam(required = false,defaultValue = "10")Integer pageSize, Model model){

SecurityUtils.getSubject().checkPermission("/classify/list");//检查权限

SecurityUtils.getSubject().checkRole("admin");

// 筛选条件

DicClassify classify = new DicClassify();

if (StringUtils.isNotBlank(name)){

classify.setName(name);

}

if (StringUtils.isNotBlank(description)){

classify.setDescription(description);

}

if(!Objects.equals(0,pageSize)){

classify.setSkip((page-1)\*pageSize);

classify.setSize(pageSize);

}

List<DicClassify> list = dicClassifyService.listClassify(classify);

Integer total = dicClassifyService.countClassify(classify);

int pages = total/pageSize + (total % pageSize==0?0:1);

model.addAttribute("dataList",list);

model.addAttribute("total",total);

model.addAttribute("pages",pages);

return "classify/classifyList";

}



图6.8账号管理界面

UserController.java

@RequestMapping("list")

public String list(@RequestParam(required = false,defaultValue = "")String nickname,

@RequestParam(required = false,defaultValue = "1") Integer page,

@RequestParam(required = false,defaultValue = "10")Integer pageSize, Model model){

SecurityUtils.getSubject().checkPermission("/user/list");

SecurityUtils.getSubject().checkRole("admin");

// 筛选条件

User user = new User();

if (StringUtils.isNotBlank(nickname)) {

user.setNickname(nickname);

}

if(!Objects.equals(0,pageSize)){

user.setSkip((page-1)\*pageSize);

user.setSize(pageSize);

}

List<User> userList = userService.listUser(user);

Integer total = userService.countUser(user);

List<Role> roleList = roleService.list();

int pages = total/pageSize + (total % pageSize==0?0:1);

model.addAttribute("dataList",userList);

model.addAttribute("roleList",roleList);

model.addAttribute("total",total);

model.addAttribute("pages",pages);

return "user/userList";

@RequestMapping("update")//更新账户信息

public String update(Integer id,String nickname,String username,String password,Integer roleId,Model model){

SecurityUtils.getSubject().checkPermission("/user/update");

SecurityUtils.getSubject().checkRole("admin");

User one = userService.getOne(

new QueryWrapper<User>()

.eq("username",username)

.ne("id",id)//条件构造器：eq判断是否相等，ne判断不相等

);

//省略判断合理性代码

User user = userService.getById(id);

user.setNickname(nickname);

user.setUsername(username);

String salt = RandomUtil.getRandomString(6);

String md5 = DigestUtils.md5DigestAsHex((password + salt).getBytes());

user.setSalt(salt);

user.setPassword(md5);

user.setRoleId(roleId);

boolean res = userService.updateById(user);

model.addAttribute("msg","修改成功");

return "redirect:list";

}



图6.9用车公司管理

UserController.java

@RequestMapping("add")

public String add(String name, Integer userId, String leader, String email, RedirectAttributes attributes){

SecurityUtils.getSubject().checkPermission("/company/add");

BusCompany company = new BusCompany();

company.setName(name);

company.setLeader(leader);

company.setEmail(email);

boolean res = companyService.save(company);

if (!res){

attributes.addAttribute("msg","添加失败,请联系管理员");

return "redirect:list";

}

attributes.addAttribute("msg","添加成功");

return "redirect:list";

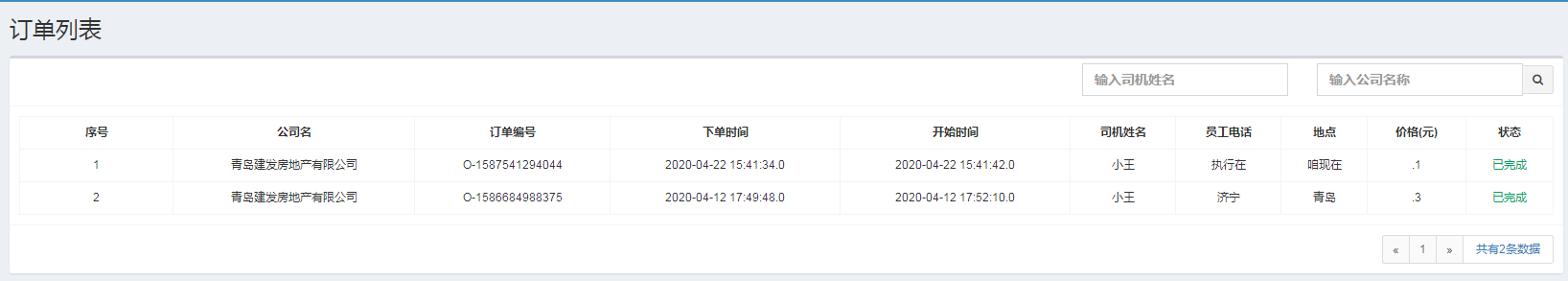
}



图6.10车辆管理



图6.11司机管理



6.12订单管理



6.13投诉管理

BusComplainController.java

@RequestMapping("/complain/")

@RequestMapping("review")//提交投诉

@Transactional(rollbackFor = RuntimeException.class)

public String review(Integer complainId,Integer score, RedirectAttributes attributes){

SecurityUtils.getSubject().checkPermission("/complain/review");

SecurityUtils.getSubject().checkRole("admin");

BusComplain complain = complainService.getById(complainId);

//set方法

boolean res = complainService.updateById(complain);

if (!res){

throw new RuntimeException("更新投诉信息失败");

}

BusOrder order = orderService.getById(complain.getOrderId());

BusDriver driver = driverService.getById(order.getDriverId());

if (driver.getCredit() < score){

attributes.addAttribute("msg","扣除积分已经大于当前积分");

return "redirect:list";

}

driver.setCredit(driver.getCredit() - score);//管理员传来扣除的信誉积分score，用司机现有的减去管理员扣除的

driver.setUpdateTime(null);

boolean b = driverService.updateById(driver);

if(!b){

throw new RuntimeException("更新司机信息失败");

}

attributes.addAttribute("msg","处理成功");

return "redirect:list";//跳到展示列表方法

}

}

1. Service层

大多是展示各个列表信息。

public interface BusDriverService extends IService<BusDriver> {

List<BusDriver> listDriver(BusDriver driver);

Integer countDriver(BusDriver driver);

List<BusDriver> getAll();

BusDriver getByUserId(Integer userId);

}

1. Dao层配置文件

BusDriverMapper.xml

<mapper namespace="xyz.jfshare.car.system.dao.BusDriverMapper">

<resultMap id="BaseResultMap" type="xyz.jfshare.car.system.entity.BusDriver">

<id column="id" jdbcType="INTEGER" property="id" />

//此处省略和实体类的对应

</resultMap>

<sql id="Base\_Column\_List">

id, user\_id, car\_id, `name`, gender, email, phone, credit, age, `status`, dno, create\_time, update\_time

</sql>

<select id="listDriver" parameterType="xyz.jfshare.car.system.entity.BusDriver" resultType="xyz.jfshare.car.system.entity.BusDriver">

SELECT

//此处省略表内各列

FROM bus\_driver bd

left join sys\_user su on bd.user\_id = su.id

left join bus\_car bc on bd.car\_id = bc.id

where bd.id != 0

<if test="name != null">

and bd.name LIKE CONCAT('%',#{name},'%')

</if>

<if test="nickname != null">

and su.nickname LIKE CONCAT('%',#{nickname},'%')

</if>

ORDER BY bd.id DESC

<if test="size != null">

LIMIT #{skip},#{size}

</if>

</select>

<select id="countDriver" parameterType="xyz.jfshare.car.system.entity.BusDriv er" resultType="java.lang.Integer">

SELECT COUNT(\*)

FROM bus\_driver bd

left join sys\_user su on bd.user\_id = su.id

left join bus\_car bc on bd.car\_id = bc.id

where bd.id != 0

<if test="name != null">

and bd.name LIKE CONCAT('%',#{name},'%')

</if>

<if test="nickname != null">

and su.nickname LIKE CONCAT('%',#{nickname},'%')

</if>

</select>

<select id="getAll" parameterType="xyz.jfshare.car.system.entity.BusDriver" resultType="xyz.jfshare.car.system.entity.BusDriver">

SELECT \* //此处省略表各列

FROM bus\_driver bd

left join sys\_user su on bd.user\_id = su.id

left join bus\_car bc on bd.car\_id = bc.id

ORDER BY bd.id DESC

</select>

</mapper>

6.3.3用车公司管理界面



图6.14选择已有信息或新建公司信息

UserController.java

@PostMapping("updateDriverInfo")

@Transactional(rollbackFor = RuntimeException.class)

public String updateDriverInfo(String nickname, String icon,String name,Integer gender,String age,String phone,String email,HttpSession session,RedirectAttributes attributes){

SecurityUtils.getSubject().checkRole("driver");

Subject subject = SecurityUtils.getSubject();

User user = (User) subject.getPrincipal();

BusDriver driver = (BusDriver) session.getAttribute("driver");

user.setNickname(nickname);

user.setIcon(icon);

driver.setName(name);

driver.setGender(gender);

int intAge;

try {

intAge = Integer.parseInt(age);

}catch (NumberFormatException e){

attributes.addAttribute("msg","年龄应是整数");

return "redirect:/driverInfo";

}

if (intAge <= 0 || intAge > 100){

attributes.addAttribute("msg","年龄应大于0小于100");

return "redirect:/driverInfo";

}

//set方法

boolean res1 = userService.updateById(user);

PrincipalCollection principals = subject.getPrincipals();

String realmName = principals.getRealmNames().iterator().next();

PrincipalCollection principalCollection

= new SimplePrincipalCollection(user,realmName);

subject.runAs(principalCollection);

BusCar car = carService.getById(driver.getCarId());

session.setAttribute("car", car);

session.setAttribute("driver", driver);

attributes.addAttribute("msg","修改成功");

return "redirect:/driverInfo";

}



图6.15用车公司主页面

UserController.java

@RequestMapping("info")

public String userInfo(Model model,HttpSession session){

User user = (User) SecurityUtils.getSubject().getPrincipal();

if(user.getRoleId() == 2) {//使用者为用车公司时

BusCompany company = companyService.getByUserId(user.getId());

if (company != null){

List<BusOrder> orderList = orderService.getByCompanyId(company.getId());

List<BusComplain> complainList = complainService.listByCompanyId(company.getId());

session.setAttribute("company", company);

session.setAttribute("complainList", complainList);

session.setAttribute("orderList", orderList);

return "redirect:/companyInfo";

}else {

List<BusCompany> companyList = companyService.getAll();

model.addAttribute("companyList",companyList);

return "company/companyChoose";

}

} else if(user.getRoleId() == 3) {//使用者为司机时

BusDriver driver = driverService.getOne(

new QueryWrapper<BusDriver>()

.eq("user\_id",user.getId())

);

if (driver != null){

BusCar car = carService.getById(driver.getCarId());

List<BusOrder> orderList = orderService.getByDriverId(driver.getId());

List<BusComplain> complainList = complainService.listByDriverId(driver.getId());

session.setAttribute("car", car);

session.setAttribute("driver", driver);

session.setAttribute("complainList", complainList);

session.setAttribute("orderList", orderList);

return "redirect:/driverInfo";

}else {

List<BusDriver> driverList = driverService.getAll();

model.addAttribute("driverList",driverList);

return "driver/driverChoose";

}

}else {

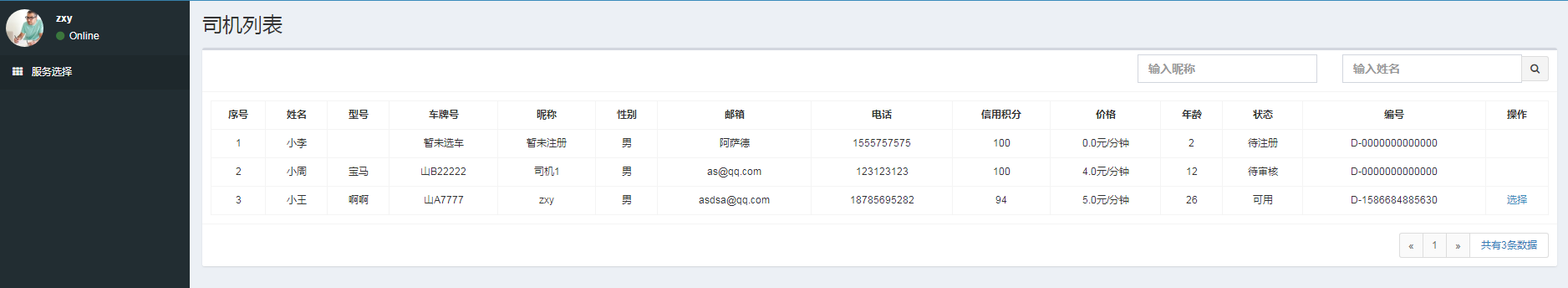
return "redirect:/main";

}

}



6.16我的投诉板块



6.17服务选择页面

6.3.4司机管理界面

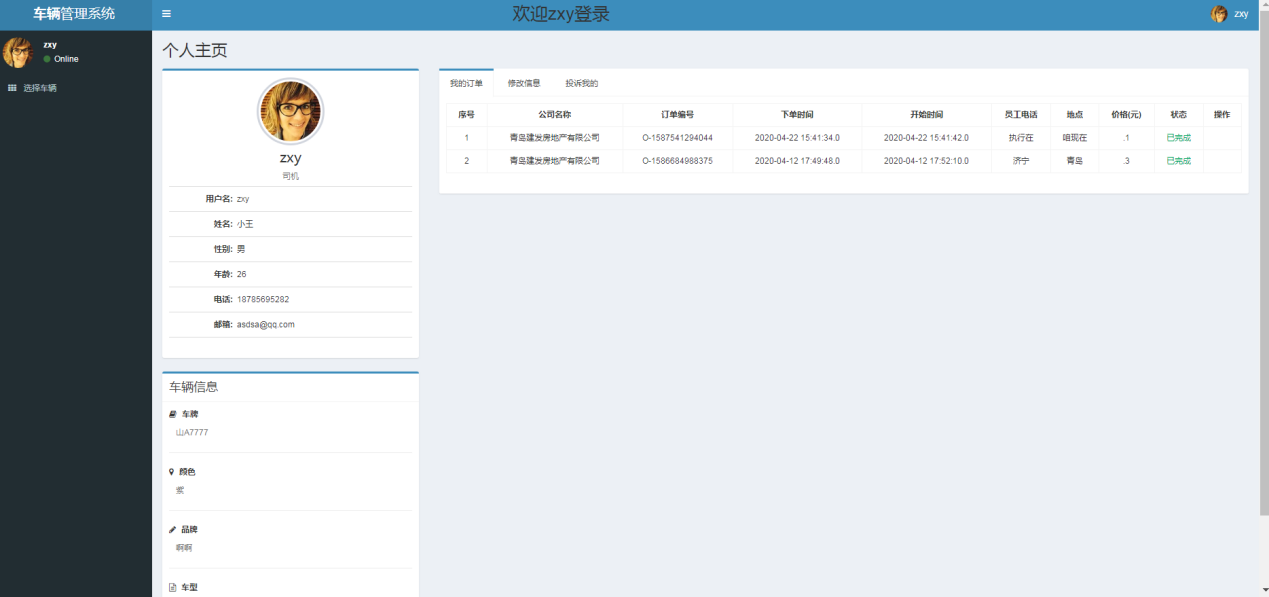


图6.18司机主页面

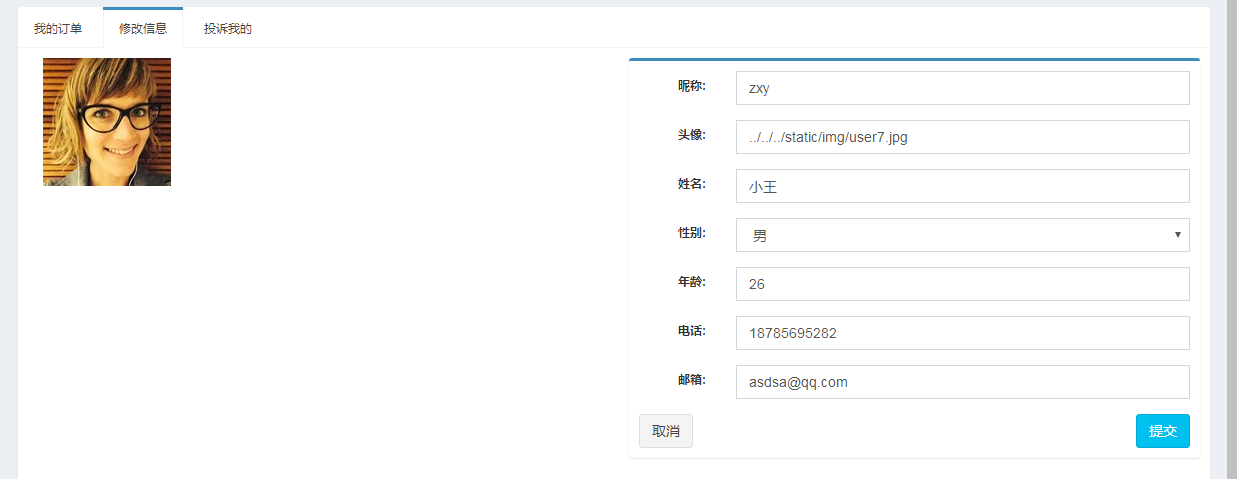


图6.19修改个人信息



图6.20选择车辆

BusDriverController.java

@RequestMapping("chooseCar")

@Transactional(rollbackFor = RuntimeException.class)

public String chooseCar(Integer carId, RedirectAttributes attributes) {

SecurityUtils.getSubject().checkPermission("/chooseCar/list");

SecurityUtils.getSubject().checkRole("driver");

BusCar car = carService.getById(carId);

//省略判断数据代码

car.setStatus(Config.CAR\_USING);

User user = (User) SecurityUtils.getSubject().getPrincipal();

BusDriver driver = driverService.getOne(

new QueryWrapper<BusDriver>()

.eq("user\_id", user.getId())

);

DicClassify classify = classifyService.getById(car.getClassifyId());

driver.setPrice((double)classify.getId());

driver.setStatus(Config.DRIVER\_NOT\_REVIEW);

driver.setCarId(carId);

boolean res = driverService.updateById(driver);

if (!res) {

attributes.addAttribute("msg", "更新失败,请联系管理员");

} else {

attributes.addAttribute("msg", "更新成功");

}

return "redirect:/chooseCar/list";

}



6.21投诉我的列表

BusDriverController.java

@RequestMapping("arrive")//用车完毕后，根据用车时间修改订单信息

@Transactional(rollbackFor = RuntimeException.class)

public String arrive(Integer orderId,RedirectAttributes attributes){

SecurityUtils.getSubject().checkPermission("/driver/arrive");

SecurityUtils.getSubject().checkRole("driver");

User user = (User) SecurityUtils.getSubject().getPrincipal();

BusDriver driver = driverService.getByUserId(user.getId());

BusOrder order = orderService.getById(orderId);

order.setStatus(Config.ORDER\_NOT\_PAY);//修改订单状态

order.setEndTime(new Timestamp(System.currentTimeMillis()));

long time = order.getEndTime().getTime() - order.getReviewTime().getTime();

double money = driver.getPrice() \* (time / 60000.0) \* driver.getCredit() / 100;//根据用车时间计算相对应的价格

order.setMoney(money);

order.setUpdateTime(null);//用车完毕要清零

boolean res1 = orderService.updateById(order);

if (!res1){

throw new RuntimeException("更新订单数据失败,请联系管理员");

}

driver.setStatus(Config.DRIVER\_NORMAL);

driver.setUpdateTime(null);

boolean res2 = driverService.updateById(driver);

if (!res2){

throw new RuntimeException("更新司机数据失败,请联系管理员");

}

attributes.addAttribute("msg", "更新成功");

return "redirect:/user/info";//根据登录用户，跳转到对应主页面

}

6.4系统总结

目前本系统应用的技术还达不到三个用户的页面同时使用，导致用车时候需要来回切换用户，而且同时使用对于修改数据库也存在着许多问题，若采用分布式系统则可以解决同时使用的问题，但对于数据的并发控制的部分还需要进一步完善。

用户每次登陆后若没有点击退出登录而是后退到上一个页面，则无需输入密码即可登陆，虽然能给人带来许多便利，但是在用户忘记退出登录时，系统的安全性无法保证。

6.5本章小结

本章主要展示了界面环境设计、登录权限设计和整个系统的界面设计的代码，可以清楚地看见每个模块对应的代码是如何设计的，由于篇幅不够只展示了部分关键代码。

系统测试 测试主要模块 根据模块测试用例！表！

致谢

首先，我要衷心感谢一直以来给予我无私帮助和关爱的老师们，特别是我的导师胡克勇老师，以及班主任安昭老师、李建萍老师等，谢谢您们这四年以来对我的关心和照顾，从您们身上，我学会了如何学习，如何工作，如何做人。其次，我还要认真地谢谢我身边所有的朋友和同学。最后，我要感谢我的父母及家人，没有人比你们更爱我，你们对我的关爱让我深深感受到了生活的美好，谢谢你们一直以来给予我的理解、鼓励和支持，你们的鼓励是我不断取得进步的永恒动力。

参考文献（15篇以上）

[1]李洋.SSM框架在Web应用开发中的设计与实现[J].计算机技术与发展,2016,26(12):190-194.(要缩进，上网自己查)

[2] 陈辉.软件工程中的解耦核心技术MVC和三层架构比较[J].廊坊师范学院学报 (自然科学版) .

[3]李春燕;李根.基于JSP技术的网络信息教学平台设计[J/OL].电子技术与软件工程，2017,（20）：9（2017-10-26）。

[4] 丁洁;.基于Shiro的Web应用安全框架设计研究[J].信息与电脑(理论版),2018 ,No.407(13):38-39.

[5] 韦美雁;段华斌;周新林;.大数据环境下的MySQL优化技术探讨[J].现代计算机(专业版),2018,No.630(30):68-72.

[6] 叶明.网络约租车监管系统技术方案分析[J].交通与港航,2016,3(04):43-45+76.DOI:10.16 487/j.cnki.issn2095-7491.2016.04.010

[7] 于明鹭,刘南杰,赵海涛,彭江琴.基于车联网的智能打车系统[J].计算机技术与发展,2016,26(02):118-123.

[8] 李营.汽车租赁信息系统的设计与实现[J].数字技术与应用,2014,(12):146.DOI:10.19695/ j.cnki.cn12-1369.2014.12.096

[9] 倪晓锋.基于SSM和Shiro的火电SIS系统的设计与实现[J].工业控制计算机,2019,32(09):131-132+134.

[10] 易文康,程骅,程耕国.Shiro框架在Web系统安全性上的改进与应用[J].计算机工程,2018,44(11):135-139.DOI:10.19678/j.issn.1000-3428.0048417