

重庆大学网络教育学院

毕业设计（论文）

题目 汽车租赁系统的设计与实现

学生所在校外学习中心 重庆长寿校外学习中心

批次 层次 专业 221批次 专升本 计算机科学与技术

学 号 W22100720

学 生 沈 希

指 导 教 师 杨瑞龙

起 止 日 期 2024-02-21 至 2024-04-16

**摘 要**

随着互联网技术的飞速发展，随着城市化进程的不断加快，人们对于交通方式的需求发生了变化。传统的交通方式已经难以满足现代社会的灵活性和便捷性需求，因此，汽车租赁系统应运而生。本文介绍了一种基于网络平台的汽车租赁系统的设计与实现。随着人们生活水平的提高和交通工具的普及，汽车租赁服务成为了一种便捷的交通方式。然而，传统的租车方式存在着诸多不便之处，如手续繁琐、信息不透明等问题。为了解决这些问题，我们设计并实现了一套汽车租赁系统，旨在提供简单、高效、透明的租车服务。本系统采用了前后端分离的架构，后端采用Spring Boot框架实现业务逻辑，前端采用Vue.js框架进行界面设计。系统包括用户管理、车辆管理、订单管理等模块，用户可以通过系统注册账户、浏览车辆信息、下单租车等操作。同时，系统还支持在线支付、车辆追踪等功能，提升了用户体验。为了保障用户信息安全，系统采用了密码加密、数据传输加密等技术手段，并严格控制了用户权限。在实现过程中，我们充分考虑了系统的可扩展性和稳定性，以应对日益增长的用户量和业务需求。通过本文的研究与实践，我们验证了该汽车租赁系统的可行性和实用性，为用户提供了一种便捷、高效的租车方式，具有一定的市场推广和应用前景。

**关键词：汽车租赁系统 网络平台 前后端分离 用户管理 车辆管理 订单管理 在线支付 安全性 可扩展性 实用性**

目录

[摘 要 2](#_Toc23644)

[1. 引言 3](#_Toc1371)

[1.1研究背景 3](#_Toc3144)

[1.2研究目的与方法 4](#_Toc10612)

[1.3研究内容 5](#_Toc21771)

[2. 系统分析 5](#_Toc15907)

[2.1行业背景 5](#_Toc8116)

[2.2系统功能分析 6](#_Toc17193)

[2.3安全性需求分析 7](#_Toc25665)

[2.4 系统数据流分析 8](#_Toc29378)

[3.汽车租赁系统总体设计 9](#_Toc30032)

[3.1系统架构设计 9](#_Toc6433)

[3.2数据库设计 10](#_Toc18991)

[3.3系统模块设计 19](#_Toc17590)

[3.4技术选型 19](#_Toc17017)

[4.汽车租赁系统关键功能模块设计与实现 20](#_Toc32217)

[4.1用户管理模块 20](#_Toc30636)

[4.2企业管理模块 30](#_Toc26763)

[4.3车辆管理模块 32](#_Toc5209)

[4.4司机管理模块 34](#_Toc2403)

[4.5产品管理模块 36](#_Toc9688)

[4.6订单管理模块 38](#_Toc28401)

[4.7取还车管理模块 44](#_Toc17581)

[5.系统测试与评估 50](#_Toc19219)

[5.1系统测试 50](#_Toc4351)

[5.2性能优化 54](#_Toc9233)

[5.3用户体验持续改进 55](#_Toc11211)

[6.结论与展望 56](#_Toc17276)

[6.1结论总结 56](#_Toc23232)

[6.2系统发展展望 56](#_Toc31009)

# 引言

1.1研究背景

随着社会经济的不断发展和人们生活水平的提高，汽车租赁行业逐渐成为人们出行的重要选择之一。传统的汽车租赁方式存在着诸多问题，例如租车手续繁琐、租金高昂、车辆信息不透明等，这些问题制约了汽车租赁行业的发展。为了解决这些问题，许多汽车租赁企业开始引入信息技术，开发汽车租赁系统，以提升租车体验、提高运营效率。

当前，随着互联网和移动互联网技术的快速发展，汽车租赁系统不仅能够提供在线租车、车辆定位等基本功能，还可以结合大数据分析、智能推荐等技术，为用户提供更加个性化、便捷的租车体验。因此，研究和设计一套高效、智能的汽车租赁系统具有重要的实际意义和应用价值。

本论文旨在通过对汽车租赁系统的设计与实现，探讨如何利用信息技术提升汽车租赁行业的服务水平，优化用户体验，提高企业运营效率，促进汽车租赁行业的健康发展。

1.2研究目的与方法

### 1.2.1研究目的

本论文的研究目的主要包括以下几个方面：

分析汽车租赁行业发展现状：通过对汽车租赁行业的发展现状进行深入分析，了解当前行业存在的问题和挑战，为设计和实现汽车租赁系统提供理论基础。

探讨汽车租赁系统的设计原则：研究汽车租赁系统的设计原则，包括系统架构设计、数据库设计、模块设计等方面，以确保系统具有良好的可扩展性、稳定性和安全性。

实现智能化的汽车租赁服务：结合大数据分析、智能推荐等技术，设计智能化的汽车租赁系统，为用户提供个性化、智能化的租车服务，提升用户体验。

优化汽车租赁系统的运营效率：通过系统测试与性能优化，提高汽车租赁系统的响应速度、稳定性和安全性，降低系统运营成本，提高企业的竞争力。

促进汽车租赁行业的健康发展：通过研究和实践，为汽车租赁企业提供有效的技术支持和管理经验，推动汽车租赁行业向数字化、智能化方向发展，促进行业的健康、可持续发展。

通过以上研究目的的实现，本论文旨在为汽车租赁行业的发展提供理论支持和实践经验，为相关研究和实践工作提供参考和借鉴。

### 1.2.2研究方法

为了实现本论文的研究目的和内容，将采用以下研究方法：

文献综述：对汽车租赁行业的相关文献进行综述，了解行业的发展历程、现状和未来趋势，为后续研究提供理论支持。

案例分析：选取几家代表性的汽车租赁企业作为案例对象，分析它们的运营模式、服务特点和市场表现，从中总结成功经验和问题教训。

需求调研：通过问卷调查、访谈等方式，获取用户对汽车租赁服务的需求和期望，为系统设计提供用户需求的参考依据。

系统设计：基于需求调研和文献综述的结果，设计汽车租赁系统的功能模块、系统架构和数据库设计，确保系统能够满足用户需求。

系统开发：使用相应的开发工具和技术，按照设计方案开发汽车租赁系统的各个功能模块，确保系统的功能完备和稳定性。

系统测试：对系统进行功能测试、性能测试等，发现并解决系统存在的问题，确保系统的质量和稳定性。

系统评估：通过案例分析和用户反馈，评估系统的实际效果和用户满意度，验证系统的可行性和实用性。

通过以上研究方法的综合运用，本论文旨在设计和实现智能化的汽车租赁系统，提升用户体验、优化运营效率，推动汽车租赁行业的数字化、智能化发展。

1.3研究内容

本论文的研究内容主要包括以下几个方面：

汽车租赁行业分析：对汽车租赁行业的发展历程、现状和趋势进行深入分析，探讨行业存在的问题和挑战。

汽车租赁系统设计：基于分析结果，设计汽车租赁系统的功能模块、系统架构、数据库设计等，确保系统具有良好的可扩展性和稳定性。

智能化服务实现：结合大数据分析、智能推荐等技术，实现智能化的汽车租赁服务，包括个性化推荐、智能定价、智能调度等功能。

系统开发与实现：基于设计方案，开发并实现汽车租赁系统的各个功能模块，确保系统能够正常运行并满足用户需求。

系统测试与优化：对系统进行功能测试、性能测试等，发现和解决系统存在的问题，优化系统的响应速度和稳定性。

案例分析与评估：选取实际案例进行系统应用，评估系统的实际效果和用户满意度，验证系统的可行性和实用性。

通过以上研究内容的展开，本论文旨在通过设计和实现智能化的汽车租赁系统，提升用户体验、优化运营效率，推动汽车租赁行业的数字化、智能化发展。

# 系统分析

2.1行业背景

### 2.1.1发展历程

汽车租赁行业起源于20世纪初，最初主要服务于商务出行和旅游租车需求。随着城市化进程加快、交通拥堵问题日益突出，汽车租赁行业得到了快速发展。近年来，共享经济的兴起进一步推动了汽车租赁行业的发展，出现了更多创新型的租车模式。

### 2.1.2现状分析

当前，汽车租赁行业呈现出以下几个特点：

市场规模扩大：随着城市人口增加和出行需求增长，汽车租赁市场规模不断扩大。

服务内容多样化：汽车租赁不仅包括短期租车、长期租车，还涵盖了自驾游、接送机等多种服务形式。

智能化发展：行业开始向智能化方向发展，包括在线预订、智能取还车等服务。

竞争激烈：行业竞争日益激烈，企业需要不断提升服务水平和创新能力。

### 2.1.3发展趋势

未来汽车租赁行业可能呈现以下几个发展趋势：

智能化服务：智能化技术将更多应用于汽车租赁行业，提升服务效率和用户体验。

绿色出行：环保意识增强，汽车租赁行业将向着绿色出行方向发展。

共享经济：共享经济模式将进一步渗透到汽车租赁行业，推动行业创新和发展。

定制化服务：用户需求个性化，汽车租赁企业将提供更多定制化服务。

综上所述，汽车租赁行业正处在快速发展和转型升级的阶段，智能化、绿色化、共享化将成为行业未来的发展趋势。

2.2系统功能分析

汽车租赁系统的功能分析主要包括以下几个方面：

用户管理：包括用户注册、登录、个人信息管理等功能，用户可以通过系统注册账号、登录系统并管理个人信息。

企业管理：平台管理员通过系统查询、新增、编辑、删除等功能，企业主要是用来创建产品、车辆以及司机等功能。

产品管理：企业管理员通过管理后台，查询、新增、修改和删除等功能，产品用于用户租车选择使用。

车辆管理：包括车辆信息录入、查询、修改和删除等功能，管理员可以在系统中录入车辆信息，用户可以通过系统查询并选择租赁车辆。

司机管理：包括司机信息录入、查询、修改和删除等功能，管理员可以在系统中录入司机信息，司机用户派车过程中的接送车以及验收车功能。

订单管理：包括订单生成、支付、取消、修改等功能，用户可以在系统中生成租车订单并进行支付，同时可以取消或修改订单。

评价管理：包括用户对租车服务的评价和反馈功能，用户可以对租车服务进行评价和反馈，帮助系统改进服务质量。

报表统计：包括订单统计、收入统计、车辆利用率统计等功能，系统可以生成各种报表和统计数据，帮助管理员进行数据分析和决策。

消息通知：包括订单状态变更通知、活动提醒等功能，系统可以向用户发送订单状态变更通知和活动提醒等消息。

安全管理：包括数据加密、访问控制、系统漏洞修复等功能，确保系统的安全性和稳定性。

综上所述，汽车租赁系统的功能分析涵盖了用户管理、车辆管理、订单管理、价格管理、评价管理、报表统计、消息通知和安全管理等多个方面，通过这些功能可以实现系统的全面管理和运营。

2.3安全性需求分析

汽车租赁系统的安全分析主要涉及以下几个方面：

数据安全：确保用户的个人信息和交易数据得到保护，防止被未经授权的访问和篡改。可以通过加密技术保护数据传输的安全，限制数据库访问权限，定期备份数据等措施来保障数据安全。

访问控制：建立严格的访问控制机制，确保只有经过授权的用户才能访问系统的各项功能和数据。可以采用身份认证、权限管理、多因素认证等技术来加强访问控制。

支付安全：保障用户在系统中进行支付时的安全，防止支付信息被盗取或篡改。可以使用安全的支付网关、加密技术、实时监控支付行为等手段来保障支付安全。

系统漏洞修复：定期对系统进行安全漏洞扫描和漏洞修复，及时更新系统和软件的补丁，以防止黑客利用漏洞进行攻击。

安全监控：建立安全监控机制，对系统的安全状态进行实时监控和日志记录，及时发现异常行为并采取相应的应对措施。

综上所述，汽车租赁系统的安全分析需要综合考虑数据安全、访问控制、支付安全、系统漏洞修复和安全监控等方面，采取相应的安全措施来保障系统的安全性。

2.4 系统数据流分析

汽车租赁系统的数据流分析可以从以下几个方面展开：

用户注册和登录流程：用户注册时，输入个人信息，系统将信息存储到用户数据库中。用户登录时，输入用户名和密码，系统验证用户身份并提供访问权限。

车辆查询和选择流程：用户查询车辆时，系统从车辆数据库中检索相关信息并展示给用户。用户选择车辆并生成订单时，系统将订单信息存储到订单数据库中。

订单支付和确认流程：用户支付订单时，系统接收支付请求，进行支付验证并更新订单状态。系统发送订单确认信息给用户，用户确认订单后，系统标记订单为已确认状态。

订单取消和修改流程：用户取消订单时，系统更新订单状态为取消，并发送通知给用户。用户修改订单时，系统更新订单信息并发送修改确认给用户。

评价和反馈流程：用户对租车服务进行评价时，系统接收评价信息并存储到评价数据库中。管理员可以查看评价信息并作出相应的改进措施。

报表统计和数据分析流程：系统定期生成订单统计、收入统计、车辆利用率统计等报表。管理员可以通过报表数据进行数据分析和决策。

消息通知流程：系统根据订单状态变更、活动提醒等情况发送消息通知给用户。用户可以接收系统发送的消息通知并做出相应的操作。

通过对汽车租赁系统的数据流进行分析，可以清晰地了解系统中不同功能之间的数据流动情况，有助于优化系统设计和提升用户体验。

# 3.汽车租赁系统总体设计

3.1系统架构设计

### 3.1.1架构概述

汽车租赁系统采用典型的三层架构，包括表现层（Presentation Layer）、业务逻辑层（Business Logic Layer）和数据访问层（Data Access Layer）。系统采用分布式架构，前端采用Web应用，后端采用微服务架构，数据库采用关系型数据库。下图3.1为架构图。

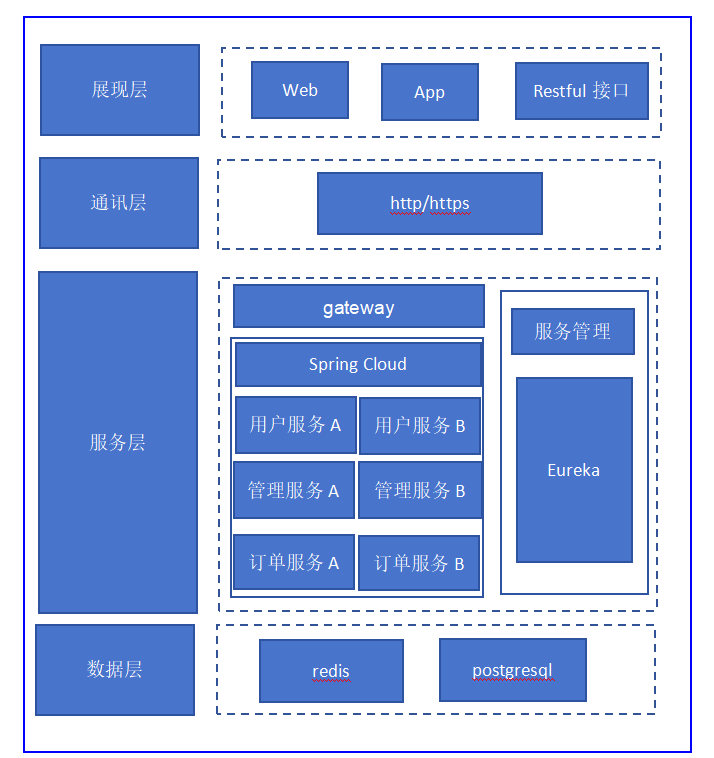


图3.1 系统架构图

### 3.1.2架构组件

Web前端：用户界面采用响应式设计，支持多终端访问，用户友好的交互体验。

移动端应用：提供移动端APP，方便用户随时随地进行车辆租赁操作。

API网关：统一管理和调度各个微服务接口，提供统一的访问入口。

用户服务微服务：负责用户注册、登录、个人信息管理等功能。

车辆服务微服务：负责车辆查询、预订、评价等功能。

订单服务微服务：负责订单生成、支付、取消等功能。

取还车服务微服务：负责取车流程、还车流程、车辆状态监控等功能。

关系型数据库：使用MySQL等关系型数据库存储用户信息、车辆信息、订单信息等数据。

缓存服务：使用Redis等缓存服务提升系统性能，减少数据库访问压力。

日志服务：记录系统运行日志，便于故障排查和性能优化。

### 3.1.3架构特点

微服务化：每个功能模块拆分为独立的微服务，实现高内聚、低耦合，便于扩展和维护。

容器化部署：使用Docker容器技术，实现服务的快速部署和扩展。

负载均衡：使用负载均衡技术，实现流量分发，提高系统的稳定性和性能。

安全性保障：采用HTTPS协议加密数据传输，使用JWT进行用户身份认证，保障系统安全性。

综上所述，汽车租赁系统采用现代化的三层架构，结合微服务、容器化部署等技术，旨在提供稳定、高效、安全的汽车租赁服务。

3.2数据库设计

从上面的需求分析整理出系统需要的信息包括企业信息，用户信息，产品信息，订单信息，车辆信息，派车单信息，取车信息等。

企业：记录企业的信息，包括社会信用代码、公司名称、公司地址、公司法人、联系人电话等信息。

管理员：记录系统管理员信息，包括用户名、密码、电话、状态等。

租车用户：记录用户信息，包括姓名、电话、密码、状态等。

产品：记录企业产品信息，包括企业、车类型、价格、产品描述的信息。

汽车：记录企业下的车辆信息，包括车牌、车类型、车品牌、车出租状态等。

订单：记录租车用户下的订单数据，包括用户、产品、租车开始时间、租车结束时间、租车开始地址、租车结束地址、租车费用等。

派车单：记录整个派车过程的数据，包括订单、派车司机、派车车辆、派车状态、派车开始时间、派车结束时间、租车开始地址、租车结束地址等数据。

取还车：存储整个取还车记录信息，操作时间、验车人、车身图片、备注等。

### 3.1.1数据库概念设计

根据前面的数据流分析，结合系统的功能模块设计，设计出系统的简单ER图如下图3.2。

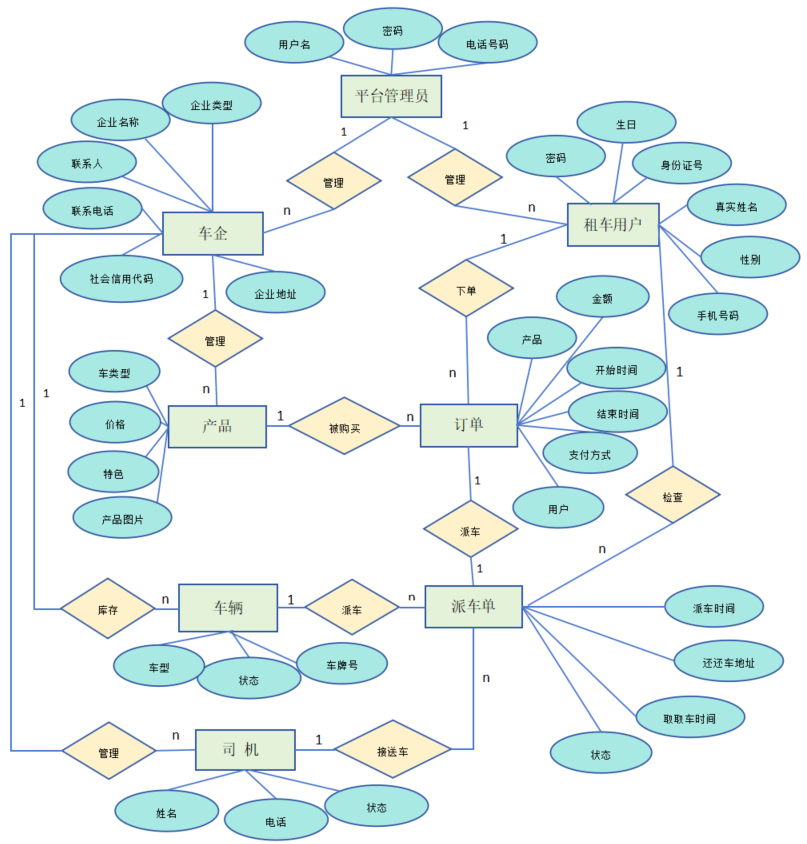


图3.2 系统ER图

### 3.1.2数据库表设计

汽车租赁系统存在的核心表如下：管理员表、用户表、企业表、产品表、车辆表、司机表、订单表、派车单表、取车表、还车表等。

表3.1 管理员表 rent\_v\_manager

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| user\_id | int8 |  | 是 | 用户id |
| user\_name | varchar | 20 | 否 | 用户名 |
| user\_password | varchar | 100 | 否 | 用户密码 |
| user\_mobile | varchar | 15 | 否 | 手机号码 |
| user\_sex | int2 |  | 否 | 性别1:男2：女 |
| enterprise\_id | int8 |  | 否 | 企业ID |
| user\_role | int2 |  | 否 | 用户角色  1:平台管理员; 2:车企管理员 |
| user\_account\_status | int2 |  | 否 | 用户状态  1:启用;0:停用 |
| insert\_timestamp | timestamp(6) |  | 否 | 添加时间 |
| update\_timestamp | timestamp(6) |  | 否 | 修改时间 |

表3.2 企业表rent\_v\_enterprise

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| key\_id | int8 |  | 是 | 企业id |
| ep\_name | varchar | 20 | 否 | 企业名称 |
| credit\_code | varchar | 20 | 否 | 社会信用代码 |
| ep\_type |  | int2 | 否 | 企业类型1:合资 2:独资3:国有4:私营 |
| contacts | varchar | 15 | 否 | 联系人 |
| contacts\_phone | varchar | 15 | 否 | 联系电话 |
| area\_code | varchar | 20 | 否 | 所在地区 |
| address | varchar | 100 | 否 | 详细地址 |
| introduce | varchar | 200 | 否 | 企业简介 |
| ep\_pics | varchar | 200 | 否 | 企业图片,多张逗号隔开 |
| ep\_status | int2 |  | 否 | 状态(0:正常,1:停用) |
| delete\_flag | int2 |  | 否 | 删除标志(0:正常,1:删除) |
| admin\_user\_id | int8 |  | 否 | 系统管理员ID |
| withdraw\_bank\_no | varchar | 30 | 否 | 提现银行卡卡号 |
| withdraw\_bank\_name | varchar | 30 | 否 | 提现银行名称 |
| withdraw\_bank\_user | varchar | 20 | 否 | 提现用户名 |
| insert\_timestamp | timestamp(6) |  | 否 | 添加时间 |
| update\_timestamp | timestamp(6) |  | 否 | 修改时间 |

表3.3产品表 rent\_v\_product

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| key\_id | int8 |  | 是 |  |
| enterprise\_id | int8 |  | 否 | 企业ID |
| car\_model\_id | int8 |  | 否 | 车型ID |
| day\_fee | int4 |  | 否 | 每日费用(元) |
| pics | text |  | 否 | 产品图片多张逗号隔开 |
| config\_params | text |  | 否 | 配置参数(行车记录仪:有) |
| up\_status | int2 |  | 否 | 上架状态(0:待处理,1:已上架,2:已下架) |
| delete\_flag | int2 |  | 否 | 删除标志(0:正常,1:删除) |
| insert\_timestamp | timestamp(6) |  | 否 | 添加时间 |
| update\_timestamp | timestamp(6) |  | 否 | 修改时间 |

表3.4企业司机表rent\_enterprise\_driver

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| driver\_id | int8 |  | 是 | 司机id |
| enterprise\_id | int8 |  | 否 | 企业id |
| user\_name | varchar | 20 | 否 | 司机姓名 |
| mobile | varchar | 20 | 否 | 司机手机号 |
| driver\_license | varchar | 100 | 否 | 驾照图片 |
| status | int2 |  | 否 | 状态0空闲 1派车中 |
| insert\_timestamp | timestamp(6) |  | 否 | 添加时间 |
| update\_timestamp | timestamp(6) |  | 否 | 修改时间 |

表3.5 用户表rent\_v\_user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| user\_id | int8 |  | 是 | 用户ID |
| user\_name | varchar | 20 | 否 | 用户名 |
| user\_password | varchar | 200 | 否 | 密码 |
| user\_mobile | varchar | 15 | 否 | 手机号 |
| user\_sex | int2 |  | 否 | 男1;女2 |
| user\_nick\_name | varchar | 20 | 否 | 昵称 |
| user\_path | varchar | 100 | 否 | 用户头像 |
| user\_real\_name | varchar | 20 | 否 | 真实姓名 |
| id\_card | varchar | 50 | 否 | 身份证号码 |
| user\_account\_status | int2 |  | 否 | 1:启用;0:停用 |
| insert\_timestamp | timestamp(6) |  | 否 | 添加时间 |
| update\_timestamp | timestamp(6) |  | 否 | 修改时间 |

表3.6 车辆信息表 rent\_enterprise\_car

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| car\_id | int8 |  | 是 | 车ID |
| enterprise\_id | int8 |  | 否 | 企业ID |
| license | varchar | 20 | 否 | 车牌号 |
| car\_model\_id | int8 |  | 否 | 车辆型号ID |
| register\_date | timestamp(6) |  | 否 | 注册日期 |
| delete\_flag | int2 |  | 否 | 删除标志(0:正常,1:删除) |
| rent\_status | int2 |  | 否 | 当前租赁状态0:未租 1:已租 |
| insert\_timestamp | timestamp(6) |  | 否 | 添加时间 |
| update\_timestamp | timestamp(6) |  | 否 | 修改时间 |

表3.7 订单表 rent\_user\_order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| order\_id | int8 |  | 是 | 订单ID |
| user\_id | int8 |  | 否 | 用户ID |
| user\_id\_card | varchar | 50 | 否 | 用户身份证 |
| user\_real\_name | varchar | 20 | 否 | 用户真实姓名 |
| enterprise\_id | int8 |  | 否 | 企业ID |
| product\_id | int8 |  | 否 | 产品ID |
| product\_name | varchar | 50 | 否 | 产品名称 |
| product\_pic | varchar | 100 | 否 | 产品图片 |
| start\_time | timestamp(6) |  | 否 | 开始时间 |
| end\_time | timestamp(6) |  | 否 | 结束时间 |
| start\_address | varchar | 200 | 否 | 取车地址 |
| start\_latitude | float8 |  | 否 | 取车地址纬度 |
| start\_longitude | float8 |  | 否 | 取车地址经度 |
| end\_address | varchar | 200 | 否 | 还车地址 |
| end\_latitude | float8 |  | 否 | 还车地址纬度 |
| end\_longitude | float8 |  | 否 | 还车地址经度 |
| fee | int4 |  | 否 | 费用(元) |
| car\_brand\_id | int8 |  | 否 | 车品牌ID |
| car\_brand | varchar | 20 | 否 | 车品牌 |
| car\_model\_id | int8 |  | 否 | 车型号ID |
| car\_model | varchar | 20 | 否 | 车型号 |
| car\_type\_id | int8 |  | 否 | 车类型ID |
| car\_type | varchar | 10 | 否 | 车类型 |
| product\_detail | text |  | 否 | 产品详情 |
| pay\_way | int2 |  | 否 | 支付方式1:支付宝 2:微信 3:银行 |
| status | int2 |  | 否 | 状态0待支付 1:已支付 2:已取消 |
| settle\_status | int2 |  | 否 | 结算状态0:未结算 1:已结算 |
| settle\_fee | int8 |  | 否 | 结算费用 |
| insert\_timestamp | timestamp(6) |  | 否 | 添加时间 |
| update\_timestamp | timestamp(6) |  | 否 | 修改时间 |

表3.8 派车单表 rent\_dispatch\_car

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| dispatch\_id | int8 |  | 是 | 派车单ID |
| enterprise\_id | int8 |  | 否 | 企业ID |
| user\_id | int8 |  | 否 | 用户ID |
| user\_name | varchar | 20 | 否 | 用户名 |
| order\_id | int8 |  | 否 | 订单ID |
| start\_time | timestamp(6) |  | 否 | 开始时间 |
| end\_time | timestamp(6) |  | 否 | 结束时间 |
| start\_address | varchar | 200 | 否 | 取车地址 |
| start\_latitude | float8 |  | 否 | 取车地址纬度 |
| start\_longitude | float8 |  | 否 | 取车地址经度 |
| end\_address | varchar | 200 | 否 | 还车地址 |
| end\_latitude | float8 |  | 否 | 还车地址纬度 |
| end\_longitude | float8 |  | 否 | 还车地址经度 |
| car\_id | int8 |  | 否 | 车ID |
| license | varchar | 20 | 否 | 车牌号 |
| status | int2 |  | 否 | 状态0:待派车 1:已派车 2:已接车 3:已还车 |
| insert\_timestamp | timestamp(6) |  | 否 | 添加时间 |
| update\_timestamp | timestamp(6) |  | 否 | 修改时间 |

表3.9 接收车表 rent\_user\_pick\_up

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| dispatch\_id | int8 |  | 是 | 派车单id |
| user\_id | int8 |  | 否 | 用户id |
| driver\_id | int8 |  | 否 | 司机id |
| car\_id | int8 |  | 否 | 车id |
| car\_pics | text |  | 否 | 车辆图片信息，多张逗号隔开 |
| remark | varchar | 500 | 否 | 备注 |
| get\_time | timestamp(6) |  | 否 | 取车时间 |

表3.10还车表rent\_user\_car\_back

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否主键 | 描述 |
| dispatch\_id | int8 |  | 是 | 派车单id |
| user\_id | int8 |  | 否 | 用户id |
| driver\_id | int8 |  | 否 | 验车司机id |
| car\_id | int8 |  | 否 | 车id |
| car\_pics | text |  | 否 | 车辆图片信息，多张逗号隔开 |
| remark | varchar | 500 | 否 | 备注 |
| back\_time | timestamp(6) |  | 否 | 归还时间 |

3.3系统模块设计

通过需求分析整理，以及数据库表的分析得出，目前系统支持三种角色的用户，平台管理员、企业管理员、租车用户。针对三种角色设计出3.3系统模块图。

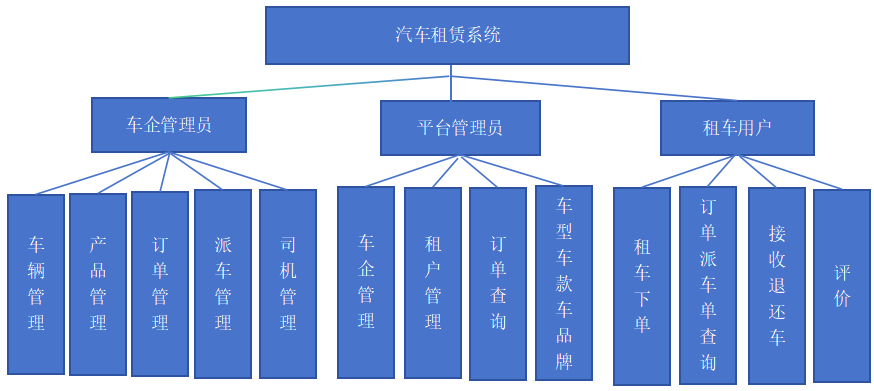


图3.3 系统模块图

3.4技术选型

前端框架：Vue.js，Vue.js是一款流行的前端框架，易学易用，支持组件化开发，有助于提高开发效率和用户体验。

UI框架：Element UI，Element UI是基于Vue.js的UI组件库，提供丰富的组件和样式，能够快速构建美观的前端界面。

后端框架：Spring Boot，Spring Boot是一款轻量级的Java后端框架，简化了Spring应用的开发和部署，适合快速搭建后端服务。

数据库：Postgresql，Postgresql是一款开源的关系型数据库管理系统，稳定可靠，支持高并发和大数据量的处理，适合存储汽车租赁系统的数据。

认证授权：JSON Web Token (JWT), JWT是一种轻量级的身份认证机制，通过Token实现用户身份的验证和授权，适合用于保护系统接口和用户信息安全。

6.4 其他技术: SLF4J + Logback, SLF4J是一个简单的日志门面，Logback是SLF4J的实现，结合使用可以方便地记录系统日志。

接口文档：Swagger, Swagger是一款API文档工具，可以自动生成接口文档，方便前后端协作和接口测试。

云服务器：AWS EC2, AWS EC2提供弹性计算服务，可根据需求动态调整计算资源，适合部署汽车租赁系统的后端服务。

综上所述，汽车租赁系统的技术选型包括Vue.js、Spring Boot、MySQL、JWT、SLF4J + Logback、Swagger和AWS EC2等技术，通过这些技术的组合使用，可以实现系统的高效开发、安全性和稳定性。

# 4.汽车租赁系统关键功能模块设计与实现

该系统的核心流程以及核心实现主要根据以下系统交互图来确定。

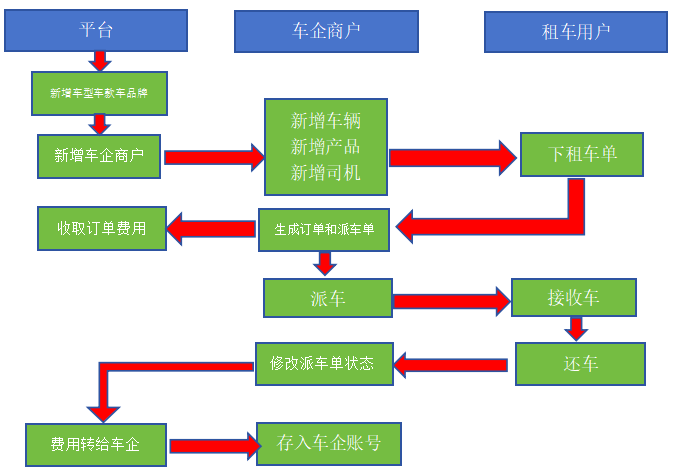


图4.1 系统交互图

4.1用户管理模块

### 4.1.1用户注册

用户注册界面样式展示如下图4.1：



图4.1 用户注册样式图

用户注册流程图4.2：

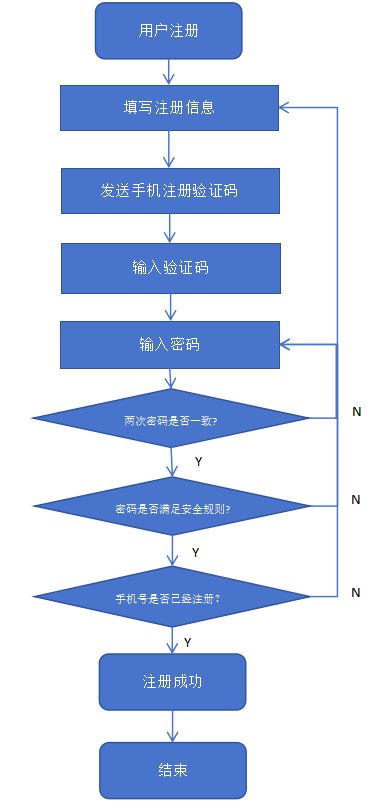


图4.2 注册流程图

用户注册核心代码：

/\*\*

\* 租车用户注册

\* @param reqDto

\* @return

\*/

@Override

public ResUserLoginDto register(ReqRegisterDto reqDto) {

ResUserLoginDto resDto = new ResUserLoginDto();

String mobile = reqDto.getUserMobile();

String code = reqDto.getCode();

String password = reqDto.getPassword();

Assert.isTrue(validatePassword(password), "密码格式不正确");

Assert.isTrue(validateCode(code, String.format("user:code:%d:%s", 1, mobile)), "验证码错误");

RentVUser rentVUser = userMapper.selectOne(Wrappers.<RentVUser>lambdaQuery().eq(RentVUser::getUserMobile, mobile).last(" limit 1"));

Assert.isTrue(rentVUser != null, "无效用户");

rentVUser.setUserPassword(password);

rentVUser.setUpdateTimestamp(new Date());

rentVUser.updateById();

JwtUrsToken jwtUrsToken = new JwtUrsToken();

ResUserInfoDto user = new ResUserInfoDto();

BeanUtils.copyProperties(rentVUser, user);

UserInfo userInfo = new UserInfo();

userInfo.setUserId(user.getUserId());

userInfo.setUserMobile(mobile);

// 有效期暂定10天

userInfo.setExpireTime(System.currentTimeMillis() + 10\*24\*60\*60\*1000);

// 用户类型1:平台|企业 2:租车用户

userInfo.setUserType(2);

String token = jwtUrsToken.genertator(userInfo, null);

resDto.setToken(token);

return resDto;

}

### 4.1.2用户登录

用户登录界面样式展示如下图4.3：



图4.3 登录样式图

用户登录流程图4.4：

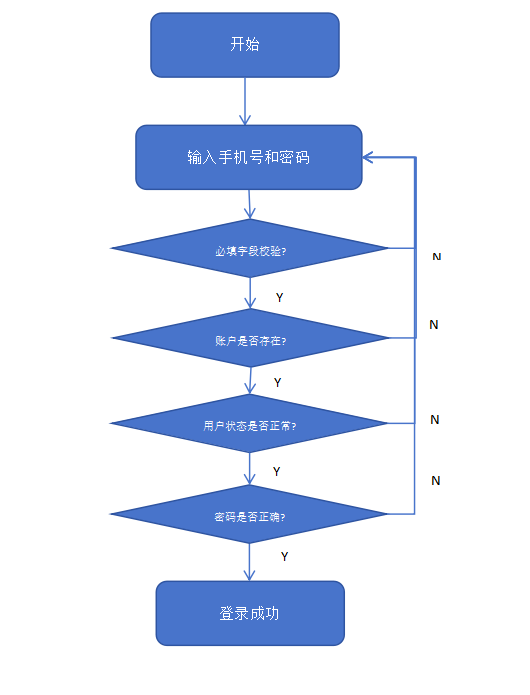


图4.4 登录流程图

用户登录核心代码：

/\*\*

\* 租车用户登录

\* @param reqDto

\* @return

\*/

@Override

public ResUserLoginDto userLogin(ReqLoginDto reqDto) {

ResUserLoginDto resDto = new ResUserLoginDto();

String mobile = reqDto.getUserMobile();

String password = reqDto.getPassword();

RentVUser rentVUser = userMapper.selectOne(Wrappers.<RentVUser>lambdaQuery().eq(RentVUser::getUserMobile, mobile).last(" limit 1"));

Assert.isTrue(rentVUser != null, "用户名或密码错误");

Assert.isTrue(rentVUser.getUserAccountStatus().equals(1), "无效用户");

Assert.isTrue(rentVUser.getUserPassword().equals(password), "用户名或密码错误");

JwtUrsToken jwtUrsToken = new JwtUrsToken();

ResUserInfoDto user = new ResUserInfoDto();

BeanUtils.copyProperties(rentVUser, user);

UserInfo userInfo = new UserInfo();

userInfo.setUserId(user.getUserId());

userInfo.setUserMobile(mobile);

// 有效期暂定10天

userInfo.setExpireTime(System.currentTimeMillis() + 10\*24\*60\*60\*1000);

// 用户类型1:平台|企业 2:租车用户

userInfo.setUserType(2);

String token = jwtUrsToken.genertator(userInfo, null);

resDto.setToken(token);

return resDto;

}

### 4.1.2用户身份验证

用户身份认证界面样式展示如下图4.5：



图4.6 租车用户身份认证样式图

用户身份认证流程图4.6：

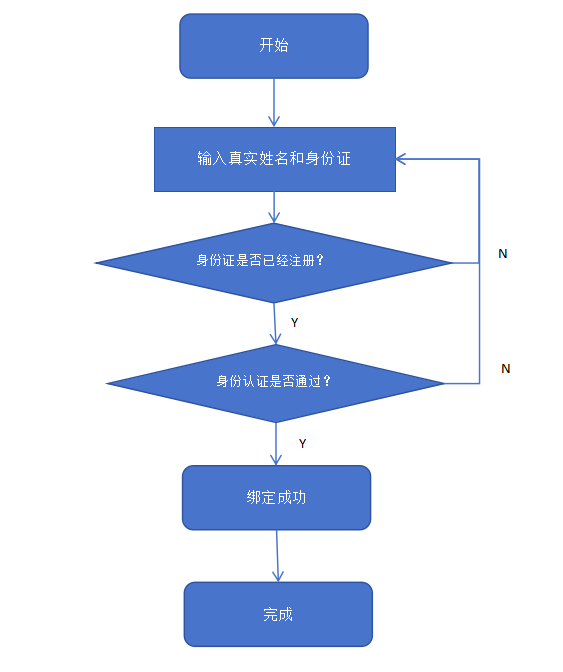


图4.6 租车用户身份认证流程图

用户身份认证核心代码：

/\*\*

\* 用户认证

\* @param reqDto

\* @return

\*/

@Override

public ResAuthenticationDto authentication(ReqAuthenticationDto reqDto) {

ResAuthenticationDto resDto = new ResAuthenticationDto();

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

Long userId = reqDto.getUserId();

String idCard = reqDto.getIdCard();

String realName = reqDto.getRealName();

RentVUser user = rentVUserMapper.selectById(userId);

Assert.isTrue(user != null, "非法访问");

// 调用第三方身份认证接口

Authentication authentication = createAuthentication(idCard, realName);

user.setAuthenticationFlowId(authentication.getFlowId());

user.setIdCard(idCard);

user.setUserRealName(realName);

user.setUpdateTimestamp(new Date());

user.updateById();

// 返回认证支付串

resDto.setAuthenticationStr(authentication.getAuthenticationUrl());

return resDto;

}

/\*\*

\* 认证结果回掉

\* @param reqDto

\*/

@Override

public void authenticationCallback(ReqAuthenticationBackDto reqDto) {

// 认证成功

if(reqDto.getCode().intValue() == 0) {

rentVUserMapper.update(null, Wrappers.<RentVUser>update()

.lambda().set(RentVUser::getAuthenticationStatus, 1)

.set(RentVUser::getAuthenticationFlowId, reqDto.getFlowId()));

} else {

log.info("用户身份认证失败:{}", reqDto);

}

}

4.2企业管理模块

### 4.2.1新增企业

新增企业界面样式展示如下图4.7：

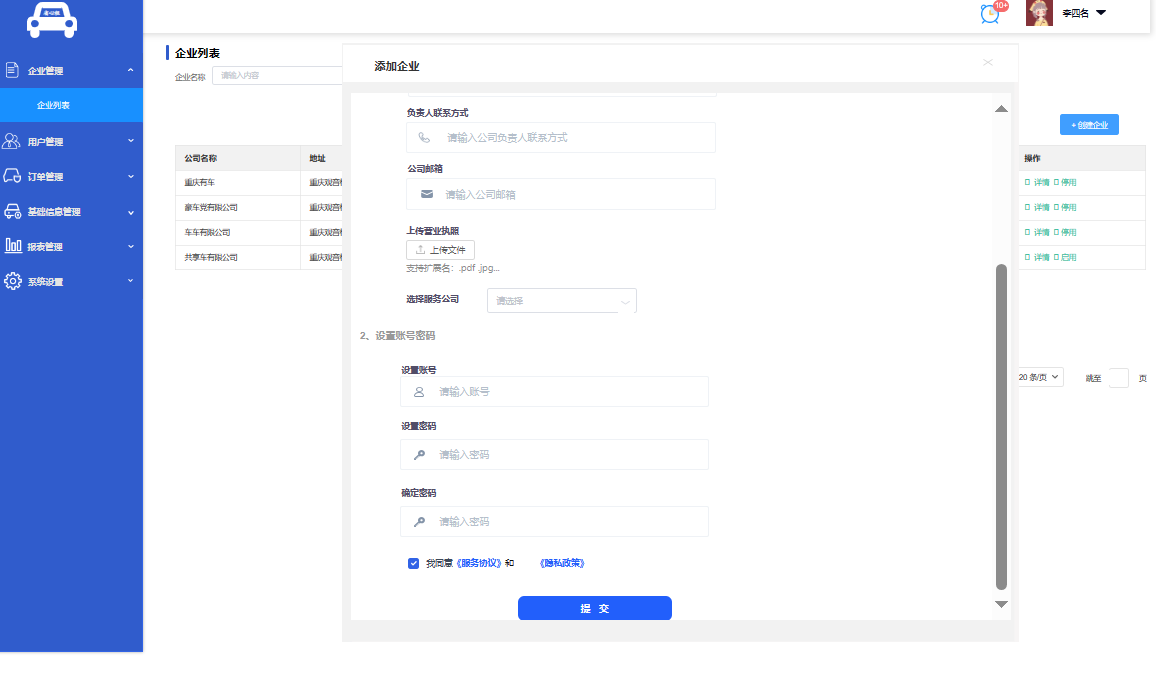


图4.7 添加企业样式图

新增企业流程图4.8：

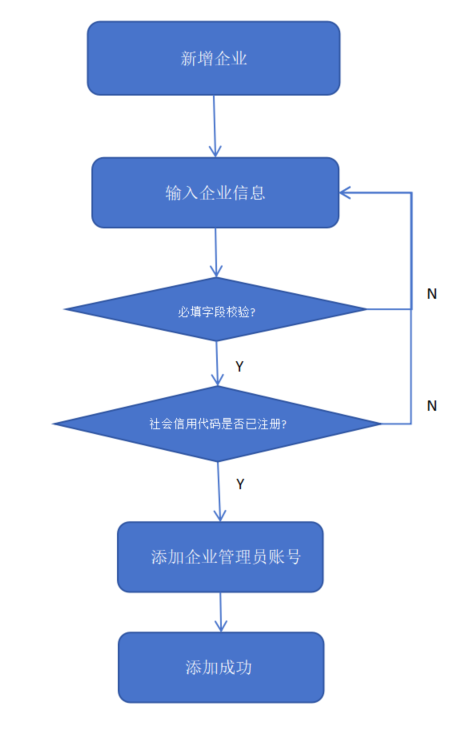


图4.8 添加企业流程图

新增企业核心代码：

/\*\*

\* 新增企业

\* @param reqDto

\*/

@Override

@Transactional

public void addEnterprise(ReqAddEnterpriseDto reqDto) {

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

// 社会信用代码

String creditCode = reqDto.getCreditCode();

int creditCodeNum = enterpriseMapper.selectCount(Wrappers.<RentVEnterprise>lambdaQuery().eq(RentVEnterprise::getCreditCode, creditCode));

Assert.isTrue(creditCodeNum <= 0, "社会信用代码重复");

int managerNum = managerMapper.selectCount(Wrappers.<RentVManager>lambdaQuery().eq(RentVManager::getUserMobile, reqDto.getContactsPhone()));

Assert.isTrue(managerNum <= 0, "电话号码重复");

/\*\* 创建企业管端账号 \*\*/

String password = reqDto.getPassword();

password = StringUtils.isBlank(password) ? new Sha256Hash("password123").toHex() : password;

Long enterpriseId = IdWorker.getId();

Long managerId = IdWorker.getId();

RentVManager manager = new RentVManager();

manager.setEnterpriseId(enterpriseId);

manager.setCreatorId(reqDto.getUserId());

manager.setUserMobile(reqDto.getContactsPhone());

manager.setUserPassword(password);

manager.setUserName(reqDto.getContactsPhone());

manager.setUserId(managerId);

manager.insert();

/\*\* 创建企业 \*\*/

RentVEnterprise enterprise = new RentVEnterprise();

enterprise.setKeyId(enterpriseId);

BeanUtils.copyProperties(reqDto, enterprise);

enterprise.insert();

}

4.3车辆管理模块

### 4.3.1新增车辆

新增车辆界面样式展示如下图4.9：

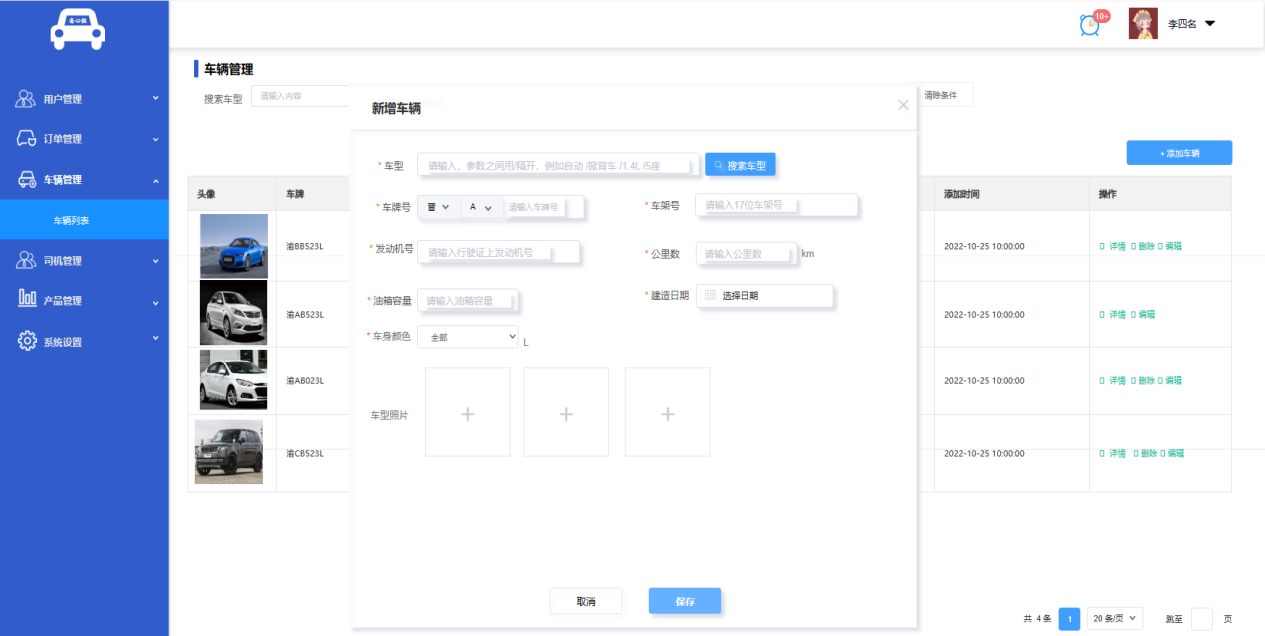


图4.9 新增车辆样式图

新增车辆流程图4.10：

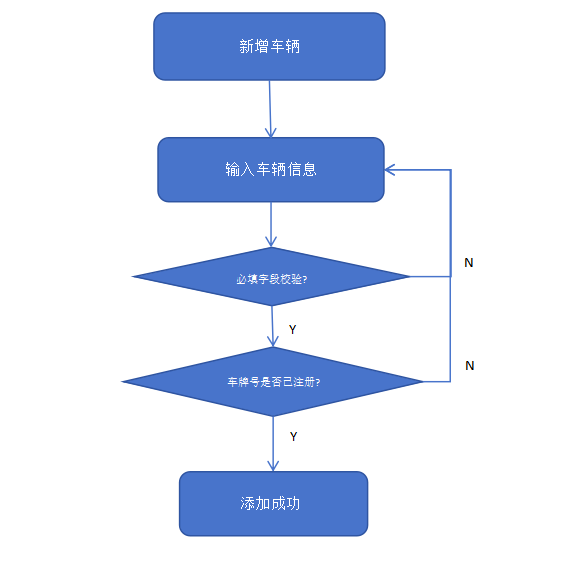


图4.10 新增车辆流程图

新增车辆核心代码：

/\*\*

\* 添加车辆

\* @param reqDto

\*/

@Override

public void addCar(ReqAddCarDto reqDto) {

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

// 效验车牌号

String license = reqDto.getLicense();

int carNum = carMapper.selectCount(Wrappers.<RentEnterpriseCar>lambdaQuery().eq(RentEnterpriseCar::getLicense, license).eq(RentEnterpriseCar::getDeleteFlag, 0));

Assert.isTrue(carNum <= 0, "该车牌已经存在");

/\*\* 创建司机 \*\*/

RentEnterpriseCar car = new RentEnterpriseCar();

car.setCarId(IdWorker.getId());

car.setEnterpriseId(reqDto.getEnterpriseId());

car.setLicense(reqDto.getLicense());

car.setCarModelId(reqDto.getCarModelId());

car.insert();

}

4.4司机管理模块

### 4.4.1新增司机

新增司机界面样式展示如下图4.11：

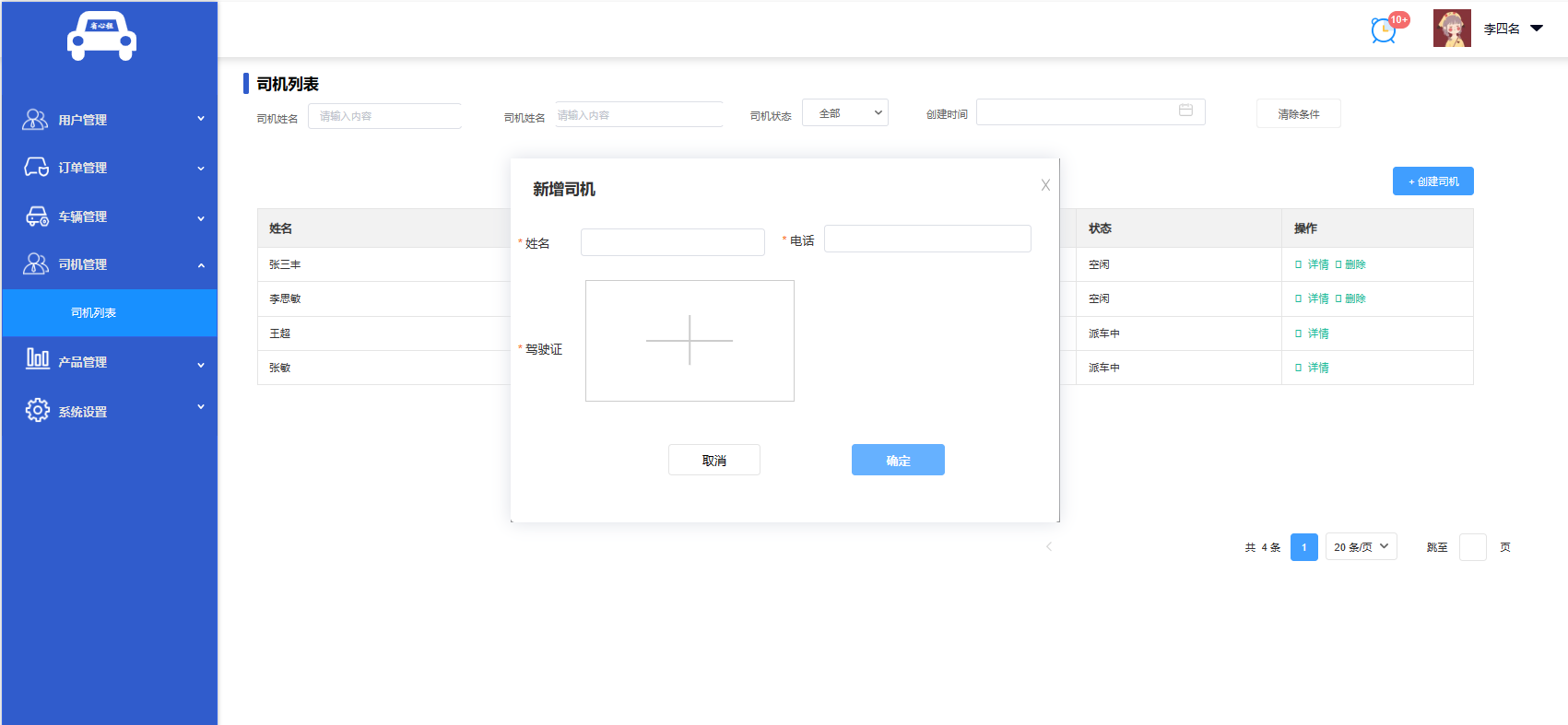


图4.11 新增司机样式图

新增司机流程图4.12：

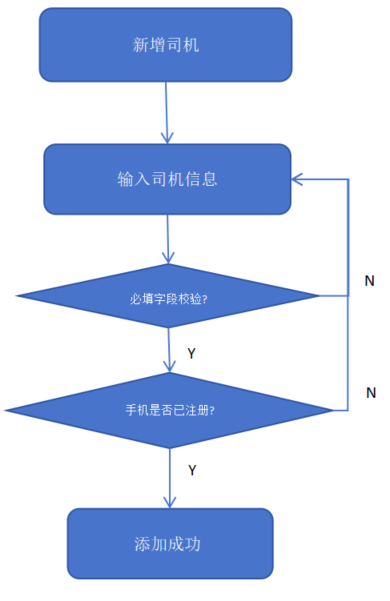


图4.12 新增司机流程图

新增司机核心代码：

/\*\*

\* 新增司机

\* @param reqDto

\*/

@Override

public void addDriver(ReqAddDriverDto reqDto) {

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

// 效验电话号码

String mobile = reqDto.getMobile();

int driverNum = driverMapper.selectCount(Wrappers.<RentEnterpriseDriver>lambdaQuery().eq(RentEnterpriseDriver::getMobile, mobile).eq(RentEnterpriseDriver::getIsDel, 0));

Assert.isTrue(driverNum <= 0, "该司机已经存在");

/\*\* 创建司机 \*\*/

RentEnterpriseDriver driver = new RentEnterpriseDriver();

driver.setDriverId(IdWorker.getId());

driver.setEnterpriseId(reqDto.getEnterpriseId());

driver.setUserName(reqDto.getUserName());

driver.setMobile(reqDto.getMobile());

driver.setDriverLicense(reqDto.getDriverLicense());

driver.insert();

}

4.5产品管理模块

### 4.5.1新增产品

新增产品界面样式展示如下图4.13：

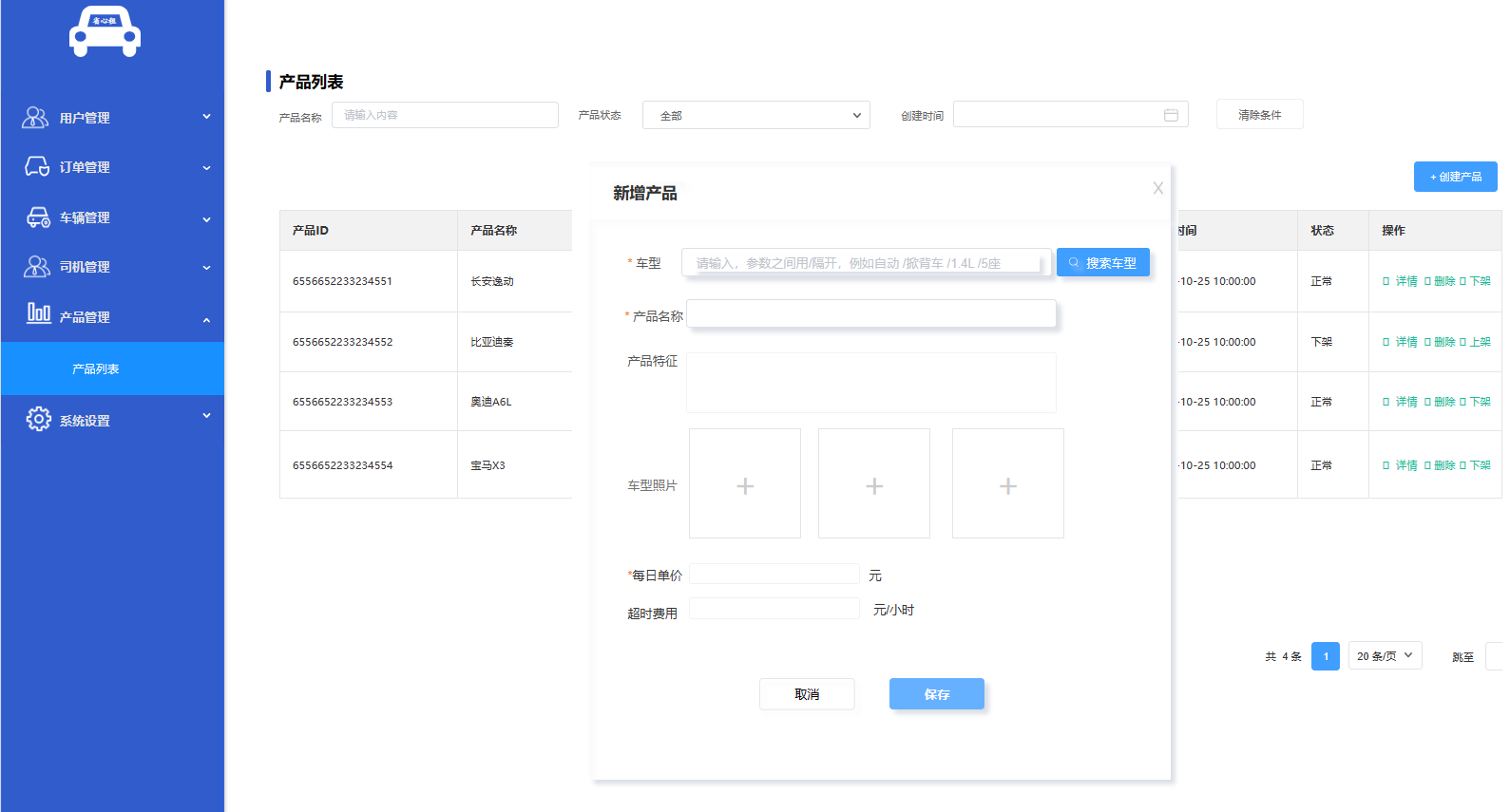


图4.13 新增产品样式图

新增产品流程图4.14：

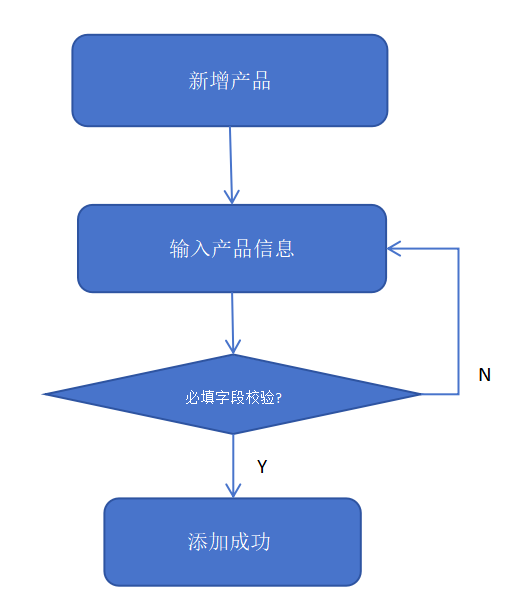


图4.14 新增产品流程图

新增产品核心代码：

/\*\*

\* 新增产品

\* @param reqDto

\*/

@Override

public void addProduct(ReqAddProductDto reqDto) {

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

/\*\* 创建产品 \*\*/

RentVProduct product = new RentVProduct();

product.setKeyId(IdWorker.getId());

product.setPic(reqDto.getPic());

product.setConfigParams(reqDto.getConfigParams());

product.setDayFee(reqDto.getDayFee());

product.setEnterpriseId(reqDto.getEnterpriseId());

product.setCarModelId(reqDto.getCarModelId());

product.setRemark(reqDto.getRemark());

product.insert();

}

4.6订单管理模块

### 4.6.1用户下单

用户下单界面样式展示如下图4.15：

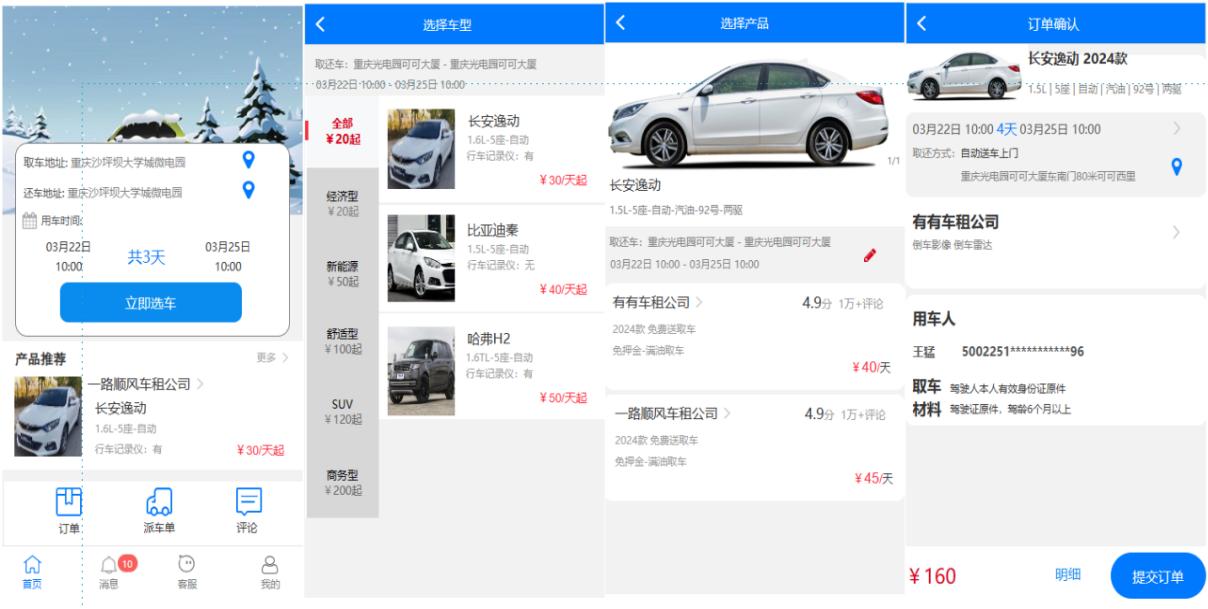


图4.15 下单样式图

用户下单流程图4.16：

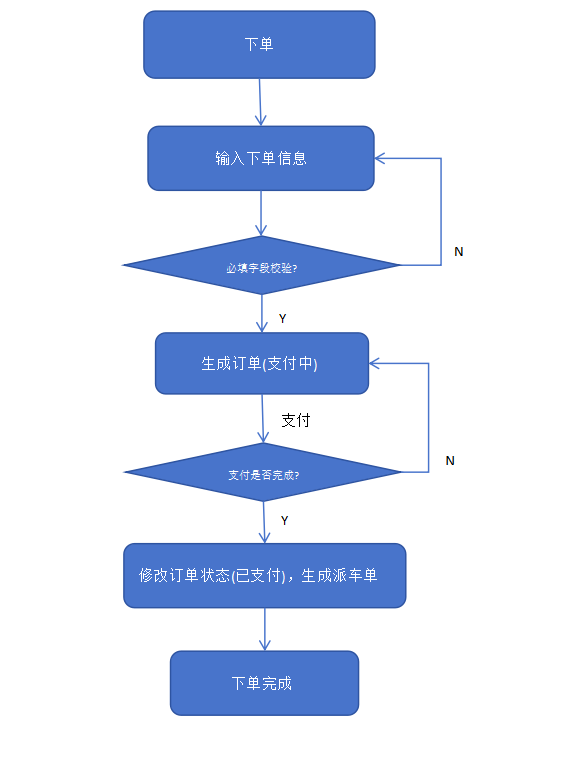


图4.16 下单流程图

用户下单核心代码：

/\*\*

\* 下单

\* @param reqDto

\* @return

\*/

@Override

@Transactional

public ResCreateOrderDto createOrder(ReqCreateOrderDto reqDto) {

ResCreateOrderDto resDto = new ResCreateOrderDto();

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

Long productId = reqDto.getProductId();

// 获取产品详情

RentVProduct product = productMapper.selectById(productId);

Assert.isTrue(product != null && product.getDeleteFlag().intValue() == 0, "产品不存在");

Assert.isTrue(product.getUpStatus().intValue() == 1, "产品已下架");

// 获取车型

RentCarModel carModel = carModelMapper.selectById(product.getCarModelId());

// 获取租赁天数

int days = days(reqDto.getStartTime(), reqDto.getEndTime());

Long fee = Long.valueOf(days\*product.getDayFee());

RentUserOrder order = new RentUserOrder();

BeanUtils.copyProperties(reqDto, order);

order.setOrderId(IdWorker.getId());

order.setFee(fee);

order.setProductName(product.getProductName());

order.setProductPic(product.getPic());

order.setCarBrandId(carModel.getCarBrandId());

order.setCarTypeId(carModel.getCarTypeId());

order.setProductDetail(createProductDetail(product));

order.insert();

String payStr = createPayStr(reqDto.getPayWay(), order);

Assert.isTrue(!StringUtils.isBlank(payStr), "生成字符串异常");

resDto.setPayStr(payStr);

return resDto;

}

/\*\*

\* 支付完成回调

\* @param reqDto

\*/

@Override

@Transactional

public void payFinishBack(ReqOrderPayFinishDto reqDto) {

Long orderId = reqDto.getOrderId();

RentUserOrder order = orderMapper.selectById(orderId);

Assert.isTrue(order != null, "订单不存在");

order.setStatus(1); // 状态0:待支付 1:已支付 2:已取消

order.updateById();

/\*\* 创建待派车派车单 \*\*/

RentDispatchCar dispatchCar = new RentDispatchCar();

dispatchCar.setOrderId(orderId);

dispatchCar.setEnterpriseId(order.getEnterpriseId());

dispatchCar.setStartTime(order.getStartTime());

dispatchCar.setEndTime(order.getEndTime());

dispatchCar.setStartAddress(order.getStartAddress());

dispatchCar.setEndAddress(order.getEndAddress());

dispatchCar.setStartLatitude(order.getStartLatitude());

dispatchCar.setStartLongitude(order.getStartLongitude());

dispatchCar.setEndLatitude(order.getEndLatitude());

dispatchCar.setEndLongitude(order.getEndLongitude());

dispatchCar.setUserId(order.getUserId());

dispatchCar.insert();

}

### 4.6.2企业派车

企业派车界面样式展示如下图4.17：

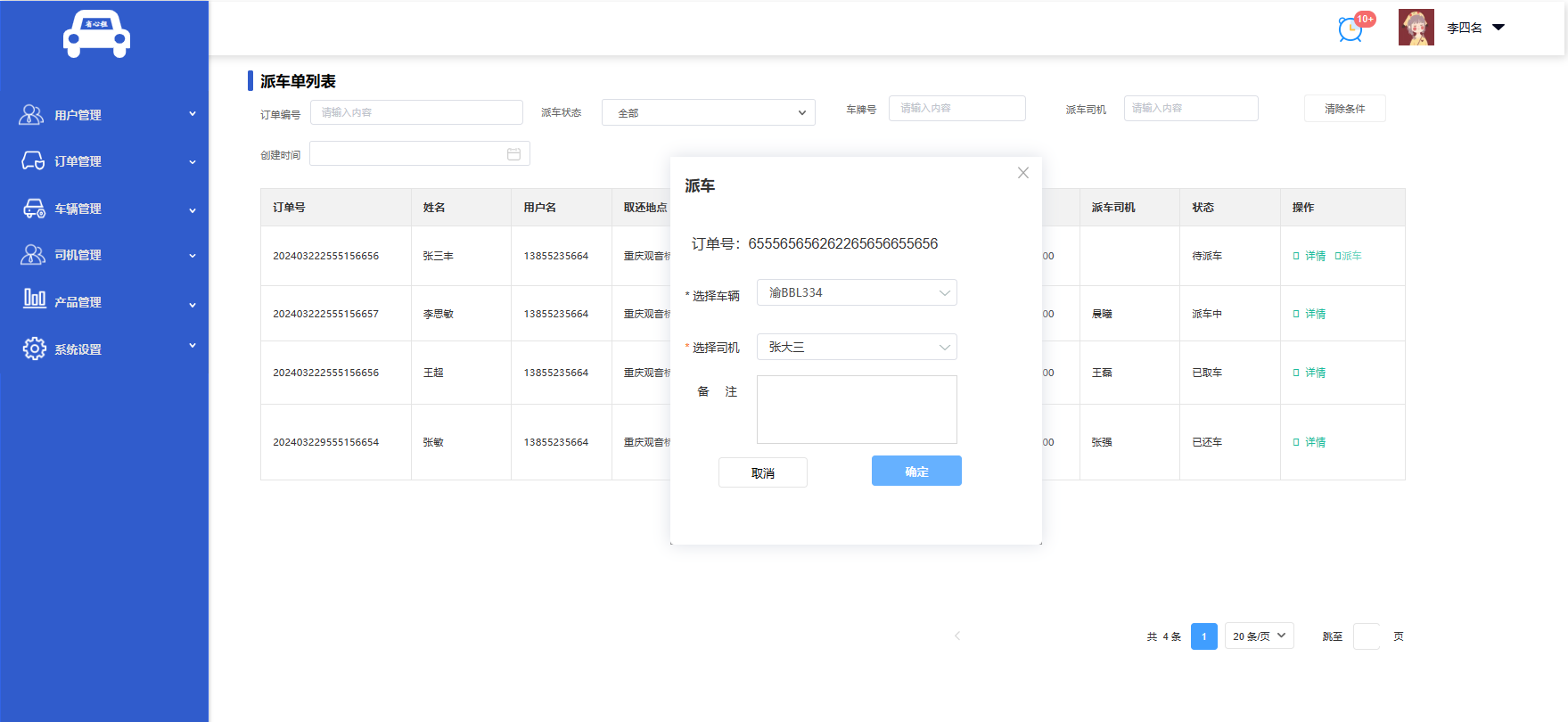


图4.17 派车样式图

企业派车流程图4.18：

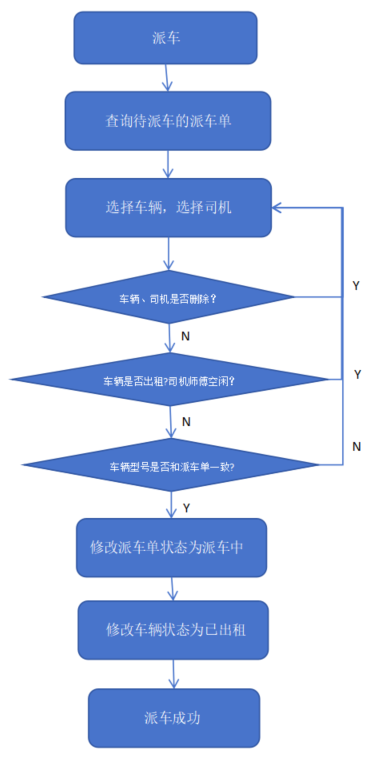


图4.18 派车流程图

企业派车核心代码：

/\*\*

\* 派车

\* @param reqDto

\*/

@Override

@Transactional

public void dispatchCar(ReqDispatchCarDto reqDto) {

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

Long dispatchId = reqDto.getDispatchId();

Long driverId = reqDto.getDriverId();

Long carId = reqDto.getCarId();

RentDispatchCar dispatchCar = dispatchCarMapper.selectById(dispatchId);

Assert.isTrue(dispatchCar.getStatus().intValue() == 0, "派车单状态异常");

RentEnterpriseDriver driver = driverMapper.selectById(driverId);

Assert.isTrue(driver.getStatus().intValue() == 0, "司机目前已处于派车中");

RentEnterpriseCar car = carMapper.selectById(carId);

Assert.isTrue(car.getRentStatus().intValue() == 0, "车辆处于租赁中");

/\*\* 更改司机状态 \*\*/

// 状态0空闲 1派车中

driver.setStatus(1);

/\*\* 更改车辆状态 \*\*/

// 当前租赁状态0:未租 1:已租

car.setRentStatus(1);

car.updateById();

/\*\* 更改派车单信息 \*\*/

dispatchCar.setDriverId(driverId);

dispatchCar.setCarId(carId);

dispatchCar.setLicense(car.getLicense());

// 状态0:待派车 1:已派车 2:已接车 3:已还车

dispatchCar.setStatus(1);

dispatchCar.setUpdateTimestamp(new Date());

dispatchCar.insert();

/\*\* 给司机推送派车信息 \*\*/

sendMsgToDriver(driverId, String.format("%你好,你有派车单任务,派车地点:{} 到达派车点时间:{}", driver.getUserName(), dispatchCar.getStartAddress(), dispatchCar.getStartTime()));

}

4.7取还车管理模块

### 4.7.1用户取车

用户取车界面样式展示如下图4.19：



图4.19 取车样式图

用户取车流程图4.20：

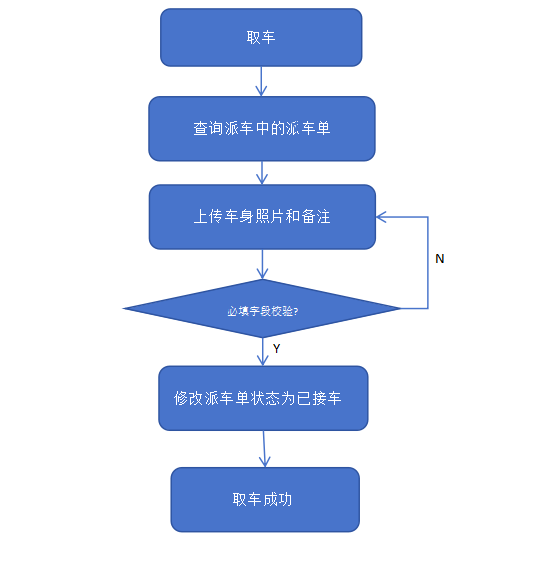


图4.20 取车流程

用户取车核心代码：

/\*\*

\* 取车

\* @param reqDto

\*/

@Override

@Transactional

public void pickUpCar(ReqPickCarDto reqDto) {

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

Long userId = reqDto.getUserId();

Long dispatchId = reqDto.getDispatchId();

Long driverId = reqDto.getDriverId();

RentDispatchCar dispatchCar = dispatchCarMapper.selectById(dispatchId);

Assert.isTrue(dispatchCar != null, "派车单不存在");

Assert.isTrue(dispatchCar.getUserId().equals(userId), "该派车单不属于你");

RentEnterpriseDriver driver = driverMapper.selectById(driverId);

Assert.isTrue(driver != null, "司机不存在");

/\*\* 设置派车单状态 \*\*/

// 状态0:待派车 1:已派车 2:已接车 3:已还车

dispatchCar.setStatus(2);

dispatchCar.setUpdateTimestamp(new Date());

dispatchCar.updateById();

/\*\* 新增取车记录 \*\*/

RentUserPickUp pickUp = new RentUserPickUp();

pickUp.setCarId(dispatchCar.getCarId());

pickUp.setUserId(userId);

pickUp.setDriverId(driverId);

pickUp.setCarPics(reqDto.getPics());

pickUp.setGetTime(new Date());

pickUp.setRemark(reqDto.getRemark());

pickUp.insert();

}

### 4.8.2用户还车

用户还车界面样式展示如下图4.21：

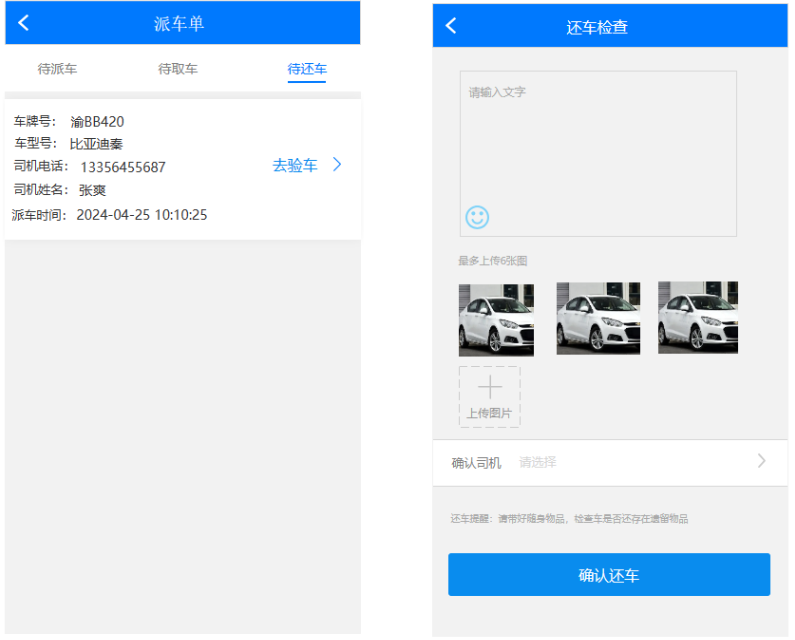


图4.21 还车样式图

用户还车流程图4.22：

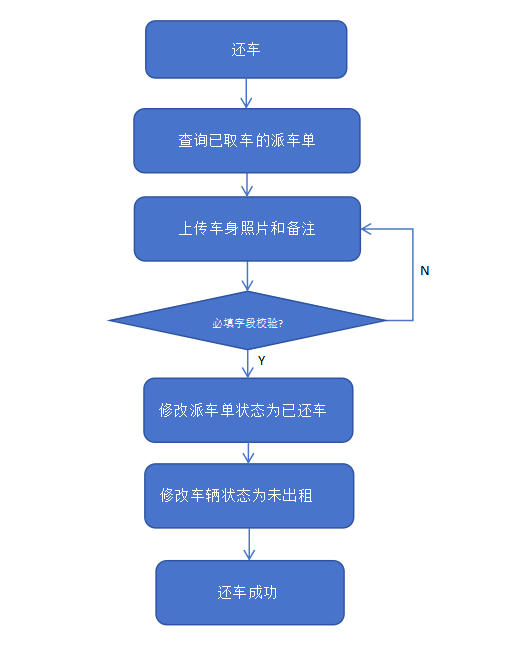


图4.22 还车流程图

用户还车核心代码：

/\*\*

\* 还车

\* @param reqDto

\*/

@Override

public void backCar(ReqBackCarDto reqDto) {

/\*\* 参数校验 \*\*/

Assert.isTrue(validateParams(reqDto).getCode() == 0, validateParams(reqDto).getMsg());

Long userId = reqDto.getUserId();

Long dispatchId = reqDto.getDispatchId();

Long driverId = reqDto.getDriverId();

RentDispatchCar dispatchCar = dispatchCarMapper.selectById(dispatchId);

Assert.isTrue(dispatchCar != null, "派车单不存在");

Assert.isTrue(dispatchCar.getUserId().equals(userId), "该派车单不属于你");

RentEnterpriseDriver driver = driverMapper.selectById(driverId);

Assert.isTrue(driver != null, "司机不存在");

RentEnterpriseCar car = carMapper.selectById(dispatchCar.getCarId());

/\*\* 设置派车单状态 \*\*/

// 状态0:待派车 1:已派车 2:已接车 3:已还车

dispatchCar.setStatus(3);

dispatchCar.setUpdateTimestamp(new Date());

dispatchCar.updateById();

/\*\* 更改司机状态 \*\*/

// 状态0空闲 1派车中

driver.setStatus(0);

driver.setUpdateTimestamp(new Date());

driver.updateById();

/\*\* 更改车辆状态 \*\*/

// 当前租赁状态0:未租 1:已租

car.setRentStatus(0);

car.setUpdateTimestamp(new Date());

/\*\* 新增取车记录 \*\*/

RentUserCarBack carBack = new RentUserCarBack();

carBack.setCarId(dispatchCar.getCarId());

carBack.setUserId(userId);

carBack.setDriverId(driverId);

carBack.setCarPics(reqDto.getPics());

carBack.setBackTime(new Date());

carBack.setRemark(reqDto.getRemark());

carBack.insert();

/\*\* 整个订单结束,平台将费用转给企业,设置结算费用和结算状态 \*\*/

RentUserOrder order = orderMapper.selectById(dispatchCar.getOrderId());

RentWithdrawRule rule = ruleMapper.selectById(order.getEnterpriseId());

// 提现比例,计算转账金额

Long ratio = rule != null ? rule.getRatio() : 95;

Long transFee = order.getFee() \* (ratio/100);

order.setSettleFee(order.getFee());

boolean trans = transferToEnterprise(order.getEnterpriseId(), transFee);

if(trans) {

//结算状态0:未结算 1:已结算

order.setSettleStatus(1);

}

order.setUpdateTimestamp(new Date());

order.updateById();

}

# 5.系统测试与评估

5.1系统测试

表5.1 用户名登录功能测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于登录流程的测试 | |
| 测试目的 | 检测登录过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 填写不存在用户 | 提示用户不存在 | 与预期一致 |
| 填写错误密码 | 提示密码错误 | 与预期一致 |

表5.2 用户注册功能测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于租车用户注册流程的测试 | |
| 测试目的 | 检测租车用户注册过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 填写错误手机号 | 提示用户不存在 | 与预期一致 |
| 填写错误验证码 | 提示验证码错误 | 与预期一致 |
| 添加成功后平台用户列表可查询 | 列表可查询到 | 与预期一致 |

表5.3 创建企业功能测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于创建企业功能的测试 | |
| 测试目的 | 检测创建企业功能过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 填写已存在用户名 | 提示用户已经存在 | 与预期一致 |
| 填写已存在电话 | 提示手机号已存在 | 与预期一致 |
| 填写社会信用代码 | 填写社会信用代码重复 | 与预期一致 |
| 添加成功后查询列表 | 新增数据展示出来了 | 与预期一致 |

表5.4 创建、删除产品测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于创建、删除产品的测试 | |
| 测试目的 | 检测文章在增删改查过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 选择删除的产品 | 提示删除成功 | 与预期一致 |
| 产品列表改变 | 产品列表正确改变 | 与预期一致 |
| 产品添加成功并上架后，用户可查询 | 用户查询可查询 | 与预期一致 |

表5.5 创建车辆测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于创建车辆相关功能的测试 | |
| 测试目的 | 检测创建车辆过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 车牌号重复 | 提示车牌号重复 | 与预期一致 |
| 车辆列表改变 | 车辆列表正确改变 | 与预期一致 |
| 派车单可查询到该车 | 可查询该车 | 与预期一致 |

表5.6 创建司机测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于创建司机相关功能的测试 | |
| 测试目的 | 检测创建司机过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 手机号重复 | 提示手机号重复 | 与预期一致 |
| 司机列表改变 | 司机列表改变正确 | 与预期一致 |

表5.7 用户下单测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于用户下单相关功能的测试 | |
| 测试目的 | 检测用户下单过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 商家订单列表新增订单 | 出现新订单 | 与预期一致 |
| 用户订单列表可查看 | 查看到该订单 | 与预期一致 |
| 新增待派车订单 | 可查询到待派车单 | 与预期一致 |

表5.8 派车测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于派车相关功能的测试 | |
| 测试目的 | 检测派车过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 派车完成派车单状态改为派车中 | 状态修改 | 与预期一致 |
| 用户查询到待取车 | 可查询到 | 与预期一致 |
| 车辆状态修改为已出租 | 状态修改 | 与预期一致 |
| 车辆状态修改为派车中 | 状态修改 | 与预期一致 |

表5.9 租车用户取车测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于租车用户取车相关功能的测试 | |
| 测试目的 | 检测租车用户取车过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 派车单状态变化为已取车 | 状态变化 | 与预期一致 |
| 用户可查询到待还车记录 | 可查询到 | 与预期一致 |

表5.10 租车用户还车测试用例

|  |  |  |
| --- | --- | --- |
| 功能描述 | 用于租车用户还车相关功能的测试 | |
| 测试目的 | 检测租车用户还车过程中可能出现的问题 | |
| 测试数据及操作 | 预期结果 | 实际结果 |
| 必填项合法输入 | 无错误提示 | 与预期一致 |
| 必填项非法法输入 | 有错误提示 | 与预期一致 |
| 派车单状态变为已还车 | 状态正确 | 与预期一致 |
| 车辆状态变化为未出租 | 状态正确 | 与预期一致 |
| 司机状态变化为空闲中 | 状态正确 | 与预期一致 |

5.2性能优化

### 5.2.1前端性能优化

压缩资源：使用Webpack对前端资源进行打包压缩，减小文件体积，提高加载速度。

懒加载：对页面中的图片和组件进行懒加载，减少首次加载时的资源消耗。

减少HTTP请求：合并CSS和JS文件，减少页面请求次数，提升页面加载速度。

CDN加速：使用CDN加速静态资源的访问速度，提高页面加载效率。

### 5.2.2后端性能优化

数据库索引：对频繁查询的字段建立索引，加快数据库查询速度。

缓存机制：使用Redis等缓存工具缓存热点数据，减少数据库查询次数，提高系统响应速度。

并发控制：使用线程池和连接池控制系统的并发访问数量，避免系统过载。

代码优化：对系统中的瓶颈代码进行优化，减少不必要的计算和IO操作。

### 5.2.3服务端性能监控

监控工具：使用Prometheus + Grafana对服务端性能进行监控，实时监控系统的运行状态。

日志分析：使用ELK Stack对系统日志进行收集和分析，及时发现问题并进行处理。

性能测试：使用JMeter等性能测试工具对系统进行压力测试，评估系统的性能指标。

### 5.2.4网络性能优化

网络请求合并：合并多个小请求为一个大请求，减少网络传输开销。

减少重定向：减少页面重定向次数，提高页面加载速度。

使用HTTP/2：使用HTTP/2协议，支持多路复用和头部压缩，提高网络传输效率。

通过以上性能优化措施，可以提升汽车租赁系统的整体性能和用户体验，保证系统的稳定性和高效运行。

5.3用户体验持续改进

用户体验持续改进是一个持续优化和提升产品或服务用户体验的过程。以下是一些详细的步骤和方法：

收集用户反馈：定期进行用户调研、用户访谈、用户反馈收集等活动，了解用户需求和痛点。

数据分析：通过用户行为数据、用户体验数据等进行分析，找出用户体验中存在的问题和瓶颈。

制定改进计划：根据用户反馈和数据分析结果，制定用户体验改进的具体计划和目标。

优化设计：对产品或服务的界面设计、交互流程等进行优化，以提升用户体验。

原型测试：在推出新功能或改进之前，进行原型测试，收集用户反馈，及时调整和优化。

持续迭代：采用敏捷开发等方法，持续对产品或服务进行改进和优化，不断提升用户体。

# 6.结论与展望

6.1结论总结

通过对汽车租赁系统用户体验持续改进的深入研究，我们取得了显著的成果。首先，通过现有用户体验分析，我们全面了解了当前系统存在的问题和用户反馈。其次，通过精心设计的用户调研和反馈收集，我们获取了宝贵的用户意见，为制定改进策略提供了有力支持。在用户体验改进策略的实施过程中，我们成功优化了系统界面、调整了功能流程，并引入了新的特性，有效提升了用户满意度。

总体而言，用户体验持续改进的策略在提高系统易用性、用户满意度和品牌形象方面取得了良好的效果。用户对系统的反馈更加积极，租车流程变得更加流畅，系统的整体性能也得到了提升。

6.2系统发展展望

尽管我们在用户体验持续改进方面取得了显著的进展，但这仅仅是一个阶段。未来，我们将继续努力，不断改进系统以适应不断变化的用户需求和市场趋势。

技术创新与智能化：随着技术的不断发展，我们将探索更先进的技术，如人工智能、机器学习等，以提高系统的智能化水平，为用户提供更个性化、智能化的服务体验。

跨平台优化：随着移动设备的多样化，我们将致力于跨平台优化，确保在不同设备上都能提供一致且优质的用户体验，包括手机、平板和笔记本等设备。

生态系统建设：建立租车生态系统，与相关行业合作，提供更多增值服务，如导航、油料充值等，以进一步满足用户的多样化需求。

用户参与与反馈机制：强化用户参与感，建立更便捷的反馈机制，不断听取用户建议，及时调整和改进系统，使用户成为系统的重要参与者。

通过这些持续改进的措施，我们有信心将汽车租赁系统打造成为用户首选的便捷、高效、智能的租车平台，满足用户不断升级的期望和需求。

**参考文献：**

[1]张海潘.软件工程导论（第六版），北京：清华大学出版社，2013.8

[2]沈择刚.Java web编程技术（第三版），北京：清华大学出版社，2019.4

[3]史济民.软件工程--原理、方法及应用（第三版），北京：高等教育出版社，2009

[4]何玉洁.数据库基础与实践技术，北京：机械工业出版社，2020.4

[5]赵韶平.PowerDesign系统分析与建模（第2版），北京：清华大学出版社，2010.1

[6]周立明.深入浅出MySQL：数据库开发、优化与管理维护（第2版），北京：电子工业出版社，2012.1

[7]王福强.Spring Boot编程思想（第2版），北京：电子工业出版社，2020.5

[8]杨开振.MyBatis技术内幕（第一版），北京：电子工业出版社，2012.12

[9]黄健宏.Redis设计与实现（第2版），北京：电子工业出版社，2018.3