

毕业设计说明书



|  |  |  |
| --- | --- | --- |
| 题目 | : | ­­­­­­基于springboot的微信汽修协作系统-汽修服务平台 |
| 学院 | : | 计算机与软件工程学院 |
| 年级专业 | : | 2014级软件工程 |
| 姓名 | : |  |
| 学号 | : | 3120140905221 |
| 指导教师 | : |  |
| 完成时间 | : | 年 月 日 |

目录

[摘 要 III](#_Toc515092375)

[Abstract IV](#_Toc515092376)

[前 言 1](#_Toc515092377)

[1 可行性分析 2](#_Toc515092378)

[1.1背景知识介绍 2](#_Toc515092379)

[1.2经济性与实用性分析 2](#_Toc515092380)

[1.3技术可行性研究 2](#_Toc515092381)

[1.4 结论 3](#_Toc515092382)

[2 需求分析 4](#_Toc515092383)

[2.1 前景和范围文档 4](#_Toc515092384)

[2.1.1业务需求 4](#_Toc515092385)

[2.1.2解决方案前景 5](#_Toc515092386)

[2.1.3 范围和局限性 6](#_Toc515092387)

[2.1.4 业务上下文 7](#_Toc515092388)

[2.2 用例分析 9](#_Toc515092389)

[2.3系统特性 10](#_Toc515092390)

[2.3.1抢单操作 10](#_Toc515092391)

[2.3.2 租赁操作 12](#_Toc515092392)

[2.3.3签约操作 13](#_Toc515092393)

[2.4外部接口需求 13](#_Toc515092394)

[2.4.1用户界面（User Interfaces, UI） 14](#_Toc515092395)

[2.4.2硬件接口（Hardware Interface, HI） 14](#_Toc515092396)

[2.4.3软件接口（Software Interface, SI） 14](#_Toc515092397)

[2.4.4通信接口（Communication Interface, CI） 14](#_Toc515092398)

[2.5其他非功能性需求 14](#_Toc515092399)

[2.5.1性能(PEformance)需求 14](#_Toc515092400)

[2.5.2防护性（Protection）需求 15](#_Toc515092401)

[2.5.3安全性（SEcurity）需求 15](#_Toc515092402)

[2.5.4软件质量属性 15](#_Toc515092403)

[3 系统设计 16](#_Toc515092404)

[3.1 系统目标 16](#_Toc515092405)

[3.2 系统功能结构 16](#_Toc515092406)

[3.3 系统设计流程 17](#_Toc515092407)

[3.4 编码规则 18](#_Toc515092408)

[3.5 数据库设计 18](#_Toc515092409)

[4 详细设计 27](#_Toc515092410)

[4.1系统后端服务 27](#_Toc515092411)

[4.1.1数据访问层 28](#_Toc515092412)

[4.1.2业务逻辑层 28](#_Toc515092413)

[4.1.3控制层 29](#_Toc515092414)

[4.1.4 统一异常处理 30](#_Toc515092415)

[4.1.5设计思想与算法 31](#_Toc515092416)

[4.1.6关键代码 32](#_Toc515092417)

[4.2 WEB端（汽修服务端）子系统 36](#_Toc515092418)

[4.2.1界面设计 36](#_Toc515092419)

[4.2.2设计思想与算法 38](#_Toc515092420)

[4.2.3关键代码 38](#_Toc515092421)

[4.3 消息推送模块 41](#_Toc515092422)

[4.3.1界面设计 41](#_Toc515092423)

[4.3.2设计思想与算法 41](#_Toc515092424)

[4.3.3关键代码 42](#_Toc515092425)

[5 测试 45](#_Toc515092426)

[5.1 业务接口测试 45](#_Toc515092427)

[5.2 WEB接口测试 46](#_Toc515092428)

[5.3 系统集成测试 47](#_Toc515092429)

[5.4 测试结论 48](#_Toc515092430)

[6 结论 49](#_Toc515092431)

[7 总结与体会 51](#_Toc515092432)

[致 谢 52](#_Toc515092433)

[参考文献 53](#_Toc515092434)

# 摘 要

本设计采用springboot和vue以及微信小程序开发技术 ，按照软件工程的开发流程来设计的汽修协作系统，该设计说明书讨论了开发系统的可行性，提出了本系统的需求并给出了系统的总体框架，该系统选择IntelliJ IDEA作为整套系统的后端服务开发工具，WebStorm作为系统的web端开发工具，微信开发者工具作为微信小程序开发工具，mysql数据库实现数据的存储，共同实现系统的各项功能。该系统主要功能包括汽修工的注册，抢单，签约，租赁；汽修厂的发布需求，处理请求，评价，定位以及移动端与web端的消息通知功能，界面简洁大方，信息查询灵活，方便，快捷，准确，数据存储安全可靠。本系统利于前后端分离，提高了系统的可维护性，可读性。该系统通过测试，能达到预期的效果。

**关键词**：springboot；vue；微信小程序；前后分离

# Abstract

This design uses the springboot and vue and WeChat applet development technology, and the auto repair collaboration system designed according to the software engineering development process. This design specification discusses the feasibility of the development system, and proposes the requirements of the system and gives the system's The overall framework, the system selects IntelliJ IDEA as a back-end service development tool for the entire system, WebStorm as a system-side web-side development tool, WeChat developer tool as a WeChat applet development tool, MySQL database to store data and realize the system together. Various functions. The main functions of the system include registration of auto repairers, rush orders, signing contracts and leases; release requirements of auto repair factories, processing requests, evaluations, positioning, and message notification functions on the mobile and web sides. The interface is concise and generous, and information inquiries are flexible and convenient. , fast, accurate, safe and reliable data storage. The system facilitates the separation of front and rear ends, improving the maintainability and readability of the system. The system passed the test to achieve the desired effect.

**Keywords**:springboot;vue;wechat applet;before and after separation.

# 前 言

现实生活中，有很多的汽修厂以及有能力的技师，在汽修厂和技师之间经常会有一些业务往来，也有一些合作。

对于汽修厂来说，管理自己签约的技师很是繁琐，平时遇到的一些技术瓶颈也是倒是任务进度阻塞的关键元素，自己的闲置工具得不到充分利用。

对于技师来说，自己有能力，有时间，还想利用它们去挣取一些工作之外的薪资，有些工具又不得不去租赁使用。

关于以上两点，我们开发了这款汽修协作系统供汽修厂和技师使用，让双方的工作能更快速，更便捷的，更人性化的进行，同时从中获取自己希望的利益，达到双赢的效果。

# 1 可行性分析

## 1.1 背景知识介绍

整个系统包括两端，WEB端和移动端。WEB端基于B/S架构实现，移动端使用微信小程序[10]开发，两端共用服务器接口，实现了前后分离。服务器开发采用java语言，使用maven+springboot框架来搭建项目，创建RESTful接口以供前端使用；WEB端采用vue.js技术来搭建，同时使用到npm，node.js等前沿技术，微信端则采用微信小程序开发。

Springboot[详情]：基于MVC模式快速搭建自己的微服务[详情]框架。

Vue[详情]：基于MVVM模式，实现数据的双向绑定，使页面的开发更加便捷，且开发的页面显得更加人性化。

介绍开发工具，IntelliJ IDEA作为后端服务器的的开发工具，能使开发变得便捷，其界面显得更加人性化 ；WebStorm则作为PC端页面的开发工具，使用它作为WEB端的开发，其快捷操作会带来更好的便捷性。

## 1.2 经济性与实用性分析

随着时代的发展，在汽修厂处理事务的同时，会产生大量的记录，在管理自己工厂信息的时候往往感觉力不从心，同时一旦出现技术瓶颈，苦于不能及时找到解决方案。另一方面随着生活压力的增大，许多人想着在工作之余挣取一些薪资，许多有能力的技师也可利用本系统去解决一些问题来赚取金钱。

本系统可有效解决当下汽修厂与技师面临的问题，达到双方共赢的局面。

## 1.3 技术可行性研究

通过对用户的需求调查得到用户要求如下：

1. 系统界面看上去要美观；
2. 能够人性化，便捷的使用；
3. 能实现消息的即时推送；
4. 实现基本的业务功能；

采用前端框架可以使界面显得更加美观，操作便捷。

## 1.4 结论

最后，我们通过多方面的考虑，本着给与用户更佳的用户体验的原则，决定采用vue+elementUI作为WEB端的界面的开发，采用当下几乎都在使用的微信作为移动端的载体来开发微信小程序，使用户在日常使用过程想用就用，更加方便，这种方案可以充分实现用户的需求，是完全可行的。

# 2 需求分析

## 2.1 前景和范围文档

### 2.1.1 业务需求

1. **背景、业务机会和客户需求**

当下汽修厂的日常业务日愈繁重，技术瓶颈也经常出现，想要通过当下流行的互联网来解决对汽修厂的管理，发布自己的技术需求，向广大技师租赁一些自己闲置的汽修工具。与此同时，技师也希望通过其他渠道利用自己的才能赚取相应的劳务费。为了使双方解决现存的问题，我们决定开发基于互联网的一套汽修协作系统来帮助解决该问题。

汽修厂的主要功能业务需求的发布，需求的管理，日常数据的统计，工具的租赁，工具的管理，像心仪的用户发送签约邀请，管理历史的合作订单，管理工厂的信息，技师管理，日常的请求处理。需求发布主要包括三大类：招聘，故障处理，其他。数据统计方面包括待处理项，总数据的统计，也可展示最近七天工厂内的各方面数据生成情况。管理工厂可修改工厂的定位信息。日常的请求处理包括用户的订单请求，租赁请求，而订单请求包括订单结束，订单中断两种。管理员可用户的请求进行处理。

用户只包括技师，技师主要业务包括抢单，管理自己的订单，修改个人信息，向工厂租赁汽修工具，管理工厂的签约请求。同时可查询自己的评分，该评分基于汽修厂对历史订单的评价生成。评分高的人在租赁工具时审核更易通过。

汽修厂和用户的协调工作需要消息推送来完成，比如说用户发起租赁请求，汽修厂需要及时获取消息推送来提示管理员处理。

1. **业务目标（Business Objective, BO）和成功标准（Success Criteria，SC）**

通过对业务概况的了解和整理，得出该系统的业务目标如下：

BO-1：为技师提供基本的业务服务，提升技师的收入水平，方便技师与汽修厂进行沟通，在此基础上争取为技师提供更好的服务。

BO-2：规范汽修厂的内部管理，提高工作效率和管理效能。

BO-3：能有效发布汽修厂的技术需求，管理汽修厂的需求以及汽修厂的技师和租赁的工具。

BO-4：能统计汽修厂的业务情况。

BO-5：实现订单的自动生成，能有效实现汽修厂与技师的消息互通。

SC-1：移动端界面简洁大方，操作简便，能有效实现技师的日常业务功能。

SC-2：WEB端需要界面简洁，能使汽修厂管理员快速的发布需求，轻松的查询需要办理的业务，能简单明了的看出汽修厂最近的业务情况。

SC-3：移动端与WEB端的消息传递，需要移动端发起请求，WEB端能够即时的弹出消息框告知管理员前去处理。

1. **业务风险（RIsk）**

RI-1:针对多个即时同时抢单情况，可能会造成数据的写入错误。

RI-2:第三方支付接口存在无法调用的情况。

RI-3:某些业务功能可能存在技术瓶颈，实现可能需要一些新的技术和开发应用等。

RI-4：对现实生活中的汽修厂需求了解不够，造成系统业务功能存在缺失。

### 2.1.2 解决方案前景

1. **前景陈述**

本系统主要针对汽修厂和技师的日常协调工作，一方面既能使汽修厂的日常管理难度得到有效的环节，也能使汽修厂的技术瓶颈得以突破；另一方面，对于技师而言，能在接单过程中获取自己的劳动报酬，还能提升自己的技术，还能向汽修厂租赁自己需要的汽修工具。

由于当前并没有这样一套基于移动端和WEB端双平台的系统，所以本系统的前景可观，它的最大价值在于能使技师和汽修厂双方达到双赢的局面。

1. **主要特性**

FE-1：发布管理需求，汽修厂能在WEB端随时发布自己面临的技术难点即技术需求，同时在技师抢单之前可随时修改需求内容，发布之后可追踪需求的处理情况。 技师在移动端也可即时查看汽修厂发布的需求信息，进行抢单操作，并查看自己正在处理的需求。

FE-2：租赁工具，汽修厂可在系统上添加自己需要租赁的工具，并随时管理，技师也可在移动端查看到这些工具信息，并申请租赁。

FE-3：汽修厂可以管理自己的员工信息，并可以向一些技师发送签约请求，请求一旦获得同意，该技师便成为此汽修厂的员工，汽修厂可查看技师的具体信息。

FE-4：汽修厂管理员可以查看自己的待处理请求数量，查看最近七天汽修厂的业务生成情况。

1. **假设和依赖**

AS-1： 整个系统的操作均属于正常操作范畴，不含恶意攻击后台服务器，且服务器，WEB端项目，微信小程序满足其运行环境。

AS-2： 使用该系统的技师人数规模适中，不会出现特别高的并发量。

AS-3： 基于goeasy的消息即时推送，依赖其服务器的正常运行。

AS-4：基于七牛云的对象存储功能，依赖其服务器的正常运行。

### 2.1.3 范围和局限性

1. **初始版本和后续版本的范围**

初始版本范围：初始版本会以一家汽修厂，多位技师来实现系统的功能，技师只能接受该汽修厂的订单，同时也作为免费系统使用，让技师与该汽修厂获得一定的利益。

后续版本范围：在系统有一定的用户群体，一定的声誉，运行稳定的情况下，会增加更多的汽修厂加盟，使系统的资源更加丰富，让技师有更多的需求可以处理，此时会有更多的用户受益。

1. **局限性和排斥性**

LI-1：因为移动端是以微信小程序的形式搭载在微信上，所以需要微信的正常运行才能实现其正常功能。

LI-2：因为整个项目多处调用了第三方的接口服务，所以部分功能也依赖其公司的正常运行。

LI-3：整个移动端面向的用户是比较单一的用户群体—技师，所以其功能也会显得单一。

LI-4：目前的系统支付功能，由于支付接口的调用需要以公司名义申请，所以尚未进行开发测试。

### 2.1.4 业务上下文

1. **涉众概览**

表2.1 涉众概览

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 说明 | 期望 |
| SH001 | 未注册技师 | 还未在本系统微信小程序注册的技师，无法使用本系统的核心功能。 | 1. 可以在本系统使用微信账号注册账号  2. 可以查看相关的微信小程序界面。 |
| SH002 | 注册技师 | 在本系统已经注册的技师，可以进行抢单，查看，租赁，签约汽修厂等操作 | 1. 可以快速的搜索自己能够处理的需求，进行抢单处理。  2. 可以查看自己的历史订单，历史评分。  3. 可以在汽修厂处租赁自己需要的汽修工具，并查看自己的租赁信息。  4. 可以对汽修厂的签约请求进行处理，达到签约功能。 |
| SH003 | 汽修厂管理员 | 汽修厂的管理者，能够对汽修厂的信息，需求，订单，请求，技师，工具进行全方位的管理，能查看汽修厂的最近业务情况。 | 1.可以快速的进行管理员期望的操作，方便快捷的发布技术需求。  2.能够对本汽修厂的技师进行信息管理，并对最近合作的技师发出签约请求。  3.管理最近完成的订单，可以对该订单的技师进行评分操作。  4.管理汽修厂有意对外开放租赁的汽修工具，并对工具的状态进行追踪。  5.处理用户发起的租赁请求，订单请求。 |

1. **用户概要**

对使用系统的用户进行具体描述，描述包括三个方面：用户概况，特点，用户使用系统的方式。如表2.2

表2.2 用户概要说明表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用户名称 | 用户概况和特点 | 使用系统方式 | 代表涉众 |
| US001 | 技师 | 是现实生活中的所有使用微信的技师人员，主要使用本系统的微信小程序进行自己的一系列抢单操作，在微信上进行抢单，线下进行解决技术问题。 | 1.抢单，查看订单登操作在微信上进行。  2. 解决技术问题线下沟通进行。 | SH001  SH002 |
| US002 | 汽修厂管理员 | 负责对汽修厂的一系列信息进行管理，比如发布需求，查询订单，租赁工具，管理技师等操作。再者对需求的状态进行追踪。汽修厂管理员还需要根据具体情况在线处理用户的订单请求以及租赁请求 | 主要在本系统的WEB端处理，具体查看需求处理情况需要双方线下进行处理需求。 | SH003 |

## 2.2 用例分析

汽修协作系统的用例和主要参与者如表2.3所示：

表2.3 系统用例

|  |  |
| --- | --- |
| 主要参与者 | 用例 |
| 未注册技师 | 1.注册  2.查看汽修厂需求 |
| 注册技师 | 1.登录  2.查看汽修厂需求  3.查看抢到的需求订单  4.提出订单处理结束或者订单中断请求  5.租赁工具  6.查询历史结束订单  7.管理个人信息  8.处理汽修厂的签约请求 |
| 汽修厂管理员 | 1.发布需求  2.管理需求  3.查看订单情况  4.管理带租赁工具  5.查看租赁信息  6.查看汽修厂签约技师  7.对最近合作的技师发送签约请求  8.对完成订单的用户进行评分  9.登录  10.查看汽修厂数据统计  11.对用户的订单请求和租赁请求进行处理  12.管理汽修厂信息  13.管理个人信息 |

## 2.3 系统特性

### 2.3.1 抢单操作

1）描述与优先级

技师在注册本系统账号之后，可在小程序上查看汽修厂发布的需求并进行抢单操作，如果多个技师同时抢单，后进行操作的会被告知需求被受理，无法抢单。用户可以对自己抢到的订单发起订单中断请求或者订单完成请求。

2）刺激/响应序列

刺激：技师对发起的需求进行抢单操作。

响应：系统会自动生成一条订单记录，该订单保存需求信息，用户信息。

刺激：技师对已被受理的需求抢单

响应：系统会告知该需求已被受理中，无法抢单

刺激：技师可对订单执行情况进行相应的订单请求

响应：系统将自动修改需求，订单状态为中断状态，发送一条订单中断请求告知汽修厂管理员处理。

刺激：管理员处理技师的订单中断请求

响应：管理员同意了该请求，则订单会取消，需求将会撤销；如果管理员拒绝该请求，则技师需要继续处理该订单，或者与汽修厂进行协商解决。

刺激：管理员处理技师的订单结束请求

响应：管理员一旦同意，订单状态，需求状态置为结束，否则需要继续处理。3）功能性需求

|  |  |
| --- | --- |
| Order.add | 技师执行抢单操作。 |
| Order.getDemand | 查询需求是否已经被受理。 |
| Order.getDemand.no | 如果没有被受理，则抢单成功，添加一条订单记录，修改需求状态为受理中。 |
| Order.getDemand.yes | 如果需求已经处于受理状态，告知用户需求已经被处理，无法抢单，请用户尝试抢其他的需求。 |
| Order.getOrder | 用户可查看自己生成的订单，对订单进行一系列操作，比如发起订单中断，订单结束的订单请求。 |
| Order. interrupt | 用户对自己的订单发起订单中断请求，将会使这个订单以及对应的需求处于中断状态，直到管理员处理该请求 |
| Order. Interrupt.yes | 管理员同意该订单中断请求，试该订单对应的需求处于撤销状态，该订单处于取消状态。管理员可对需求进行重新提起。 |
| Order. Interrupt.no  Order. Finish.no | 管理员拒绝该订单请求，用户将会受到反馈，且订单，需求重新恢复处理中的状态，知道用户处理结束或者通过协商后再次提起请求待处理 |
| Order. Finish | 用户对自己的订单发起订单结束请求，将会使这个订单以及对应的需求处于中断状态，直到管理员处理该请求 |
| Order. Finish.yes | 管理员同意该请求，将订单状态和需求状态都置于结束状态，表示该需求，该订单正常结束。 |

### 2.3.2 租赁操作

1）描述与优先级

注册的技师可在移动端向汽修厂租赁自己需要的工具，并填写租赁日期一起租赁价格；发起租赁请求待汽修厂管理员处理。

2）刺激/响应序列

刺激：技师对自己需要租赁的工具发起租赁请求。

响应：系统会生成一条请求记录，发送消息推送到WEB端告知管理员处理。

刺激：管理员同意请求，且工具处于闲置中。

响应：系统会生成一条租赁记录，该请求记录会置于已处理状态。

刺激：管理员同意请求，但工具已经被租赁。

响应：告知用户工具已经被租赁，尝试租赁其他工具

刺激：管理员拒绝租赁请求

响应：告知用户请求被拒绝，请协商后再次申请。

3）功能性需求

|  |  |
| --- | --- |
| Lreq.add | 用户发起租赁请求待处理 |
| Lreq.getTool | 查询工具的状态，如果工具处于在租中，告知用户工具被租用，请尝试其他工具。 |
| Lrea.yes | 管理员同意该请求，并将工具状态修改为在租中，生成一条租赁记录。 |
| Lreq.no | 管理员拒绝该请求，告知用户请求被拒绝，可再次提起。 |

### 2.3.3 签约操作

1）描述与优先级

汽修厂在每次与一个用户完成一次成功合作之后，管理员可查询到最近合作且暂时没有公司的用户，管理员可以线下与用户进行交谈，交谈完成之后可发送签约请求，等待技师的处理。技师一旦同意，将会生成签约记录，双方达成签约状态。

2）刺激/响应序列

刺激：汽修厂对最近合作的用户发起签约请求

响应：系统生成一条签约请求记录，等待用户处理

刺激：用户同意该请求

响应：系统生成一条签约记录，修改用户公司，修改请求记录状态

刺激：用户拒绝该请求

响应：告知汽修厂管理员请求被拒绝

3）功能性需求

|  |  |
| --- | --- |
| Sreq.add | 汽修厂管理员发起签约请求 |
| Sreq.getUser | 系统查询该用户，如果该用户已经有公司所属，则告知汽修厂管理员无法提起该申请。 |
| Sreq.yes | 技师对签约申请同意，则添加双方的签约信息 |
| Sreq.no | 技师拒绝该请求，告知管理员请求被拒绝 |

## 2.4 外部接口需求

### 2.4.1 用户界面（User Interfaces, UI）

UI-1:系统应保证界面简洁大方，字体适中，操作快捷简便。

UI-2:整个系统的色彩应保持色彩一致。

UI-3:对于微信端和WEB端都应有用户帮助手册，帮助用户更快的使用系统，另外任何的系统报错都应以用户易懂的方式告知用户异常。

### 2.4.2 硬件接口（Hardware Interface, HI）

后期可能会增加PC端打印机的调用，移动端摄像头的调用。

### 2.4.3 软件接口（Software Interface, SI）

SI-1:数据库的设计以及功能设计应该考虑后续增加复杂业务的接口。

SI-2：系统后期可能有其他的数据统计需求，因此应当适量增加数据统计接口以供后期调用。

SI-3：准备生成验证码，日期转换接口供多次使用，以及需要编写公用对外开放的请求接口以便随时使用。

### 2.4.4 通信接口（Communication Interface, CI）

CI-1：调用GOEASY的消息推送接口来实现消息的即时传递。

## 2.5 其他非功能性需求

### 2.5.1 性能(PEformance)需求

PE-1:系统需要保证在工作期间的工作效率，汽修厂管理员在查询日常处理事务，各种信息，系统响应时间应该在5秒左右。

PE-2:技师在移动端抢单过程中，应保证系统能够承载一定的并发量，且有较快的响应速度。

PE-3:移动端以及WEB端的消息推送，需要保证消息的即时性，其推送响应间隔不能超过5秒。

### 2.5.2 防护性（Protection）需求

防止数据的脏读脏写，需要避免数据库的sql语句注入危险，对外开发的接口需要保证数据安全。

### 2.5.3 安全性（SEcurity）需求

SE-1:每条充值记录都需要有日志文件能够查询。

SE-2:保证抢单以及租赁功能的正常进行，且抢单应当避免出现重复，紊乱，抢单中断造成数据库脏写等现象，租赁应避免工具重复租赁。

### 2.5.4 软件质量属性

Usability(易用性)-1:查询，录入需求信息应当考虑汽修厂管理员使用便捷，操作简单。

Usability(易用性)-2:抢单方式要简洁，快速，一次来提高技师的抢单体验以及抢单速度。

Usability(易用性)-3:汽修厂管理员能够清晰明了的查看厂内的信息情况，以及最近七天的业务情况。

Maintainability(移植性)-1:后期前端页面可能会有很大升级，应当采用前后端分离，一次来减少前后端的耦合程度，增加系统的可维护，可升级性。

Maintainability(移植性)-2:后期可能会有多个汽修厂入驻本系统，需要预留增加汽修厂信息的接口，保证系统易于改进。

Efficiency（有效性）-1:本系统能够有效促进汽修厂和技师的工作，既能改进汽修厂的管理效率，还能使技师发挥自己的长处，赚取自己应得的报酬。

# 3 系统设计

## 3.1 系统目标

本系统为了方便汽修厂与技师的协同工作，基于线上抢单，线下解决的模式，绝大部分操作都是早网上进行。

对于不同的用户对象使用的子系统是不一样的，其中汽修厂管理员主要使用PC的WEB端，技师使用基于微信小程序的移动端，两种用户所需的功能模块虽不相同，缺紧密联系。

我们的目标就是使用本系统来代替以往的汽修厂人为管理厂内信息的工作，能使汽修厂更为方便的解决自己的技术难题，帮助技师拿到额外的薪资。该系统本质上是基于WEB，微信的应用系统，由数据库，应用系统，用户组成。

## 3.2 系统功能结构

由上面的设计过程可知，系统分两个大模块：汽修厂模块，技师模块。

汽修厂模块大致分为8个模块：查看汽修厂业务情况，请求处理，需求管理，技师管理，订单管理，物品租赁，工厂管理，个人管理。其中一些模块包括更具体的子模块，具体看下图。

技师模块分为4个模块：抢单模块，订单管理，个人管理，物品租赁，请求处理。

同时汽修厂模块与技师模块需要紧密联系，包含了一个消息推送模块。

根据上述分析，可得出系统功能结构图，如图3.1所示。

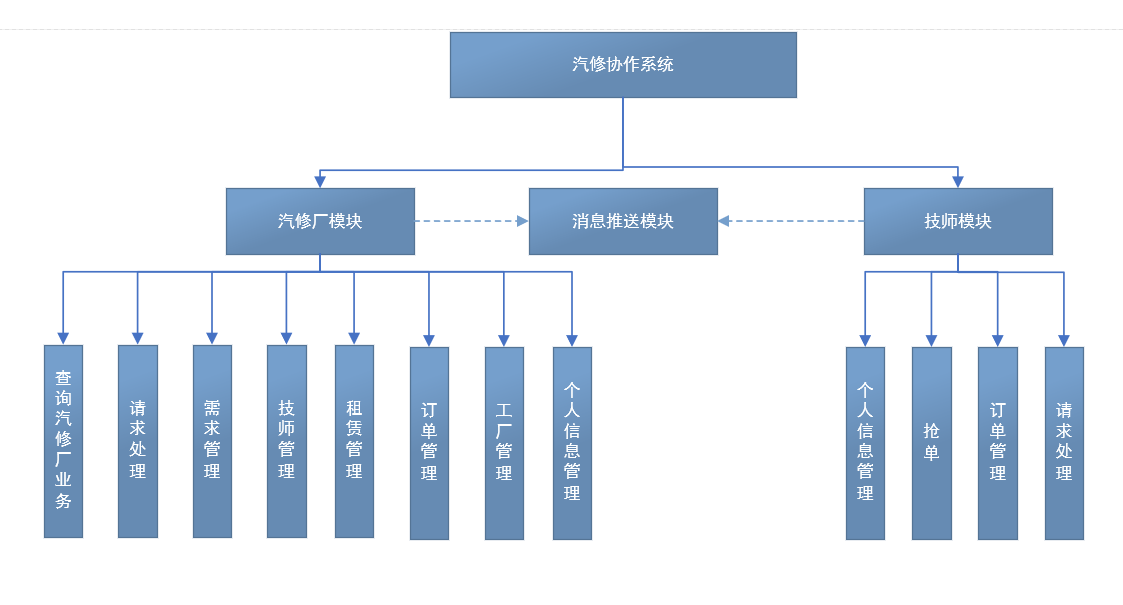


图3.1汽修协作系统功能结构图

## 3.3 系统设计流程

由可行性分析和需求分析可知，汽修协作系统采用前后端分离开发，后端采用springboot开发，前端采用vue和微信小程序开发。

系统的设计过程如下：

1.系统的数据库设计。

2.系统框架的搭建。

3.系统的WEB接口设计以及编写，基于MVC开发模式的各层开发的统ow。

4.WEB端界面设计：登录框的设计，菜单的设计、通知框的设计和数据可视化的设计。汽修厂管理员使用本系统需要先使用账号登录。登录成功后选择自己需要处理的事务进行处理即可。

5.微信端的界面设计：菜单的设计，按钮的设计，风格的设计。

6.WEB端与服务端，微信端和服务端的接口请求，获取数据填充到界面。

7.集成测试。

## 3.4 编码规则

后端服务API开发返回采用json类型，统一返回固定的格式。其所有的开发编码均采用UTF-8编码。所有的开发都要符合开发规范，易于后期维护代码。

## 3.5 数据库设计

数据库设计包括数据字典、数据流程、ER图及数据表设计：

1. **数据字典**

表3.1 公司表（Company）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Company\_id | Int(11) |  | no | yes | 公司ID |
| Company\_name | Varchar(50) |  | no |  | 公司名称 |
| Company\_address | Varchar(100) |  | no |  | 公司地址 |
| Company\_score | Float(2) | 5.0 | no |  | 公司评分 |
| Company\_location | Varchar(30) |  | yes |  | 公司地址 |
| Company\_phone | Varchar(20) |  | yes |  | 公司电话 |
| Company\_email | Varchar(20) |  | yes |  | 公司邮箱 |

表3.2 角色表（Role）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| id | Int(11) |  | no |  | ID |
| name | Varchar(50) |  | no |  | 角色名称 |

表3.3 技师表（User）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| User\_id | Int(11) |  | no | yes | 用户ID |
| User\_name | Varchar(20) |  | no |  | 账号 |
| User\_pwd | Varchar(20) |  | no |  | 密码 |
| User\_mc | Varchar(20) |  | yes |  | 用户姓名 |
| User\_sex | Varchar(5) |  | yes |  | 用户性别 |
| User\_phone | Varchar(20) |  | yes |  | 用户电话 |
| User\_email | Varchar(20) |  | yes |  | 用户邮箱 |
| User\_address | Varchar(20) |  | yes |  | 用户地址 |
| User\_score | Float(2) | 5.0 | yes |  | 用户得分 |
| User\_open\_id | Varchar(50) |  | yes |  | 微信开放ID |
| Company\_id | Int(11) |  | yes |  | 公司ID |

表3.4 管理员表（Manage）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Manage\_id | Int(11) |  | no | yes | ID |
| Manage\_name | Varchar(20) |  | no |  | 登录名 |
| Manage\_mc | Varchar(20) |  | yes |  | 姓名 |
| Manage\_pwd | Varchar(20) |  | no |  | 密码 |
| Manage\_phone | Varchar(20) |  | yes |  | 电话 |
| Manage\_email | Varchar(20) |  | yes |  | 邮箱 |
| Manage\_role | Int(11) |  | no |  | 角色 |
| Create\_time | Varchar(20) |  | no |  | 创建时间 |
| Last\_login\_time | Varchar(20) |  | yes |  | 上次登录 |
| Company\_id | Int(11) |  | yes |  | 公司ID |

表3.5 需求表（Demand）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Demand\_id | Varchar(50) |  | no |  | ID |
| Demand\_type | Int(11) |  | no |  | 需求类型ID |
| Demand\_title | Varchar(100) |  | no |  | 标题 |
| Demand\_content | Varchar(500) |  | no |  | 内容 |
| Demand\_remark | Varchar(100) |  | yes |  | 备注 |
| Demand\_repay | Float(8) |  | yes |  | 报酬 |
| Demand\_state | Varchar(1) | 1 | no |  | 状态 |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Create\_time | Varchar(20) |  | no |  | 创建时间 |
| Last\_update\_time | Varchar(20) |  | no |  | 上次更新 |

表3.6 需求类型表（Dtype）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| id | Int(11) |  | no |  | ID |
| name | Varchar(50) |  | no |  | 名称 |

表3.7 工具表（Tool）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Tool\_id | Int(11) |  | no |  | ID |
| Tool\_type | Int(11) |  | no |  | 工具类型ID |
| Tool\_des | Varchar(200) |  | no |  | 描述 |
| Tool\_img | Varchar(100) |  | yes |  | 图片 |
| Tool\_state | Varchar(1) | 1 | no |  | 状态 |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Tool\_name | Varchar(50) |  | no |  | 工具名称 |

表3.8 工具类型表（Ttype）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| id | Int(11) |  | no |  | ID |
| name | Varchar(50) |  | no |  | 名称 |

表3.9 订单表（Orders）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Order\_id | Varchar(50) |  | no |  | ID |
| User\_id | Int(11) |  | no |  | 用户ID |
| Demand\_id | Varchar(50) |  | no |  | 需求ID |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Order\_title | Varchar(100) |  | no |  | 订单标题 |
| Order\_state | Varchar(1) |  | no |  | 订单状态 |
| Order\_score | Float(2) |  | yes |  | 订单评分 |
| Create\_time | Varchar(20) |  | no |  | 创建时间 |
| Last\_update\_time | Varchar(20) |  | no |  | 更新时间 |

表3.10 订单请求表（Oreq）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Oreq\_id | Int(11) |  | no | yes | ID |
| User\_id | Int(11) |  | no |  | 用户ID |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Order\_id | Varchar(50) |  | no |  | 订单ID |
| Oreq\_type | Varchar(1) |  | no |  | 请求类型 |
| Oreq\_content | Varchar(200) |  | no |  | 请求内容 |
| Oreq\_result | Varchar(1) |  | yes |  | 请求结果 |
| Oreq\_create\_time | Varchar(20) |  | no |  | 创建时间 |
| Oreq\_handle\_time | Varchar(20) |  | yes |  | 处理时间 |

表3.11 租赁表（Lease）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Lease\_id | Varchar（20） |  | no |  | ID |
| User\_id | Int(11) |  | no |  | 用户ID |
| Tool\_id | Int(11) |  | no |  | 工具ID |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Lease\_mes | Varchar(200) |  | yes |  | 租赁描述 |
| Lease\_repay | Float(5) |  | no |  | 租赁金额 |
| Lease\_term | Varchar(5) |  | no |  | 租赁天数 |
| Lease\_state | Varchar(1) |  | no |  | 租赁状态 |
| Total\_money | Float(5) |  | yes |  | 总金额 |
| Create\_time | Varchar(20) |  | no |  | 创建时间 |
| End\_time | Varchar(20) |  | yes |  | 结束时间 |

表3.12 租赁请求表（Lreq）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| L\_req\_id | Int(11) |  | no | yes | ID |
| User\_id | Int(11) |  | no |  | 用户ID |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Tool\_id | Int(11) |  | no |  | 工具ID |
| L\_req\_money | Float(5) | 0.0 | yes |  | 请求价格 |
| L\_req\_term | Varchar(5) |  | yes |  | 请求天数 |
| L\_req\_content | Varchar(200) |  | no |  | 请求状态 |
| L\_req\_result | Varchar(1) | 0 | yes |  | 请求结果 |
| L\_req\_create\_time | Varchar(20) |  | no |  | 创建时间 |
| L\_req\_handle\_time | Varchar(20) |  | yes |  | 处理时间 |

表3.13 签约表（Sign）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| User\_id | Int(11) |  | no |  | 用户ID |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Sign\_time | Varchar(5) |  | no |  | 签约年限 |
| Sign\_date | Varchar(20) |  | no |  | 签约日期 |
| End\_date | Varchar(20) |  | yes |  | 到期日期 |

表3.14 签约请求表（Sreq）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| sr\_id | Int(11) |  | no | yes | ID |
| User\_id | Int(11) |  | no |  | 用户ID |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Sr\_time | Int(11) |  | no |  | 请求年限 |
| Sr\_state | Int(11) | 0 | yes |  | 请求状态 |
| Create\_time | Varchar(20) |  | no |  | 创建时间 |
| Handle\_time | Varchar(20) |  | yes |  | 处理时间 |

表3.15招聘信息表（Recruit）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Recruit\_id | Int(11) |  | no | yes | ID |
| Recruit\_number | Int(11) |  | no |  | 招聘数量 |
| Create\_time | Varchar(20) |  | no |  | 创建时间 |
| Company\_id | Int(11) |  | no |  | 公司ID |
| End\_time | Varchar(20) |  | no |  | 结束时间 |
| Recruit\_type | Int(11) |  | no |  | 招聘职位 |
| Recruit\_remark | Varchar(20) |  | yes |  | 招聘备注 |

表3.16应聘表（Rreq）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许为空 | 自动递增 | 备注 |
| Rreq\_id | Int(11) |  | no | yes | ID |
| Recruit\_id | Int(11) |  | no |  | 招聘信息ID |
| User\_id | Int(11) |  | no |  | 用户Id |
| Company\_id | Int(11) |  | no |  | 公司ID |
| Rreq\_state | Int(11) |  | no |  | 应聘状态 |
| Rreq\_content | Varchar(500) |  | yes |  | 内容 |
| Create\_time | Varchar(20) |  | no |  | 应聘时间 |

1. **数据流程图**

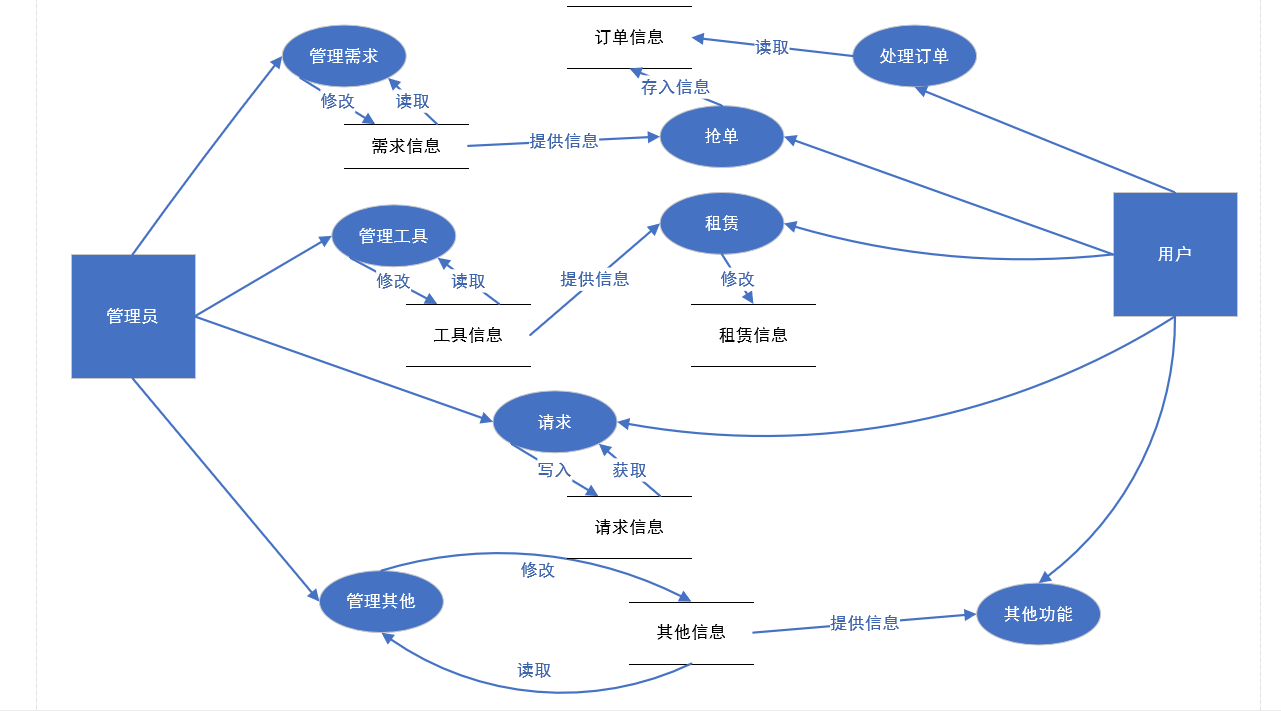


图3.2数据流程图

1. **ER图**

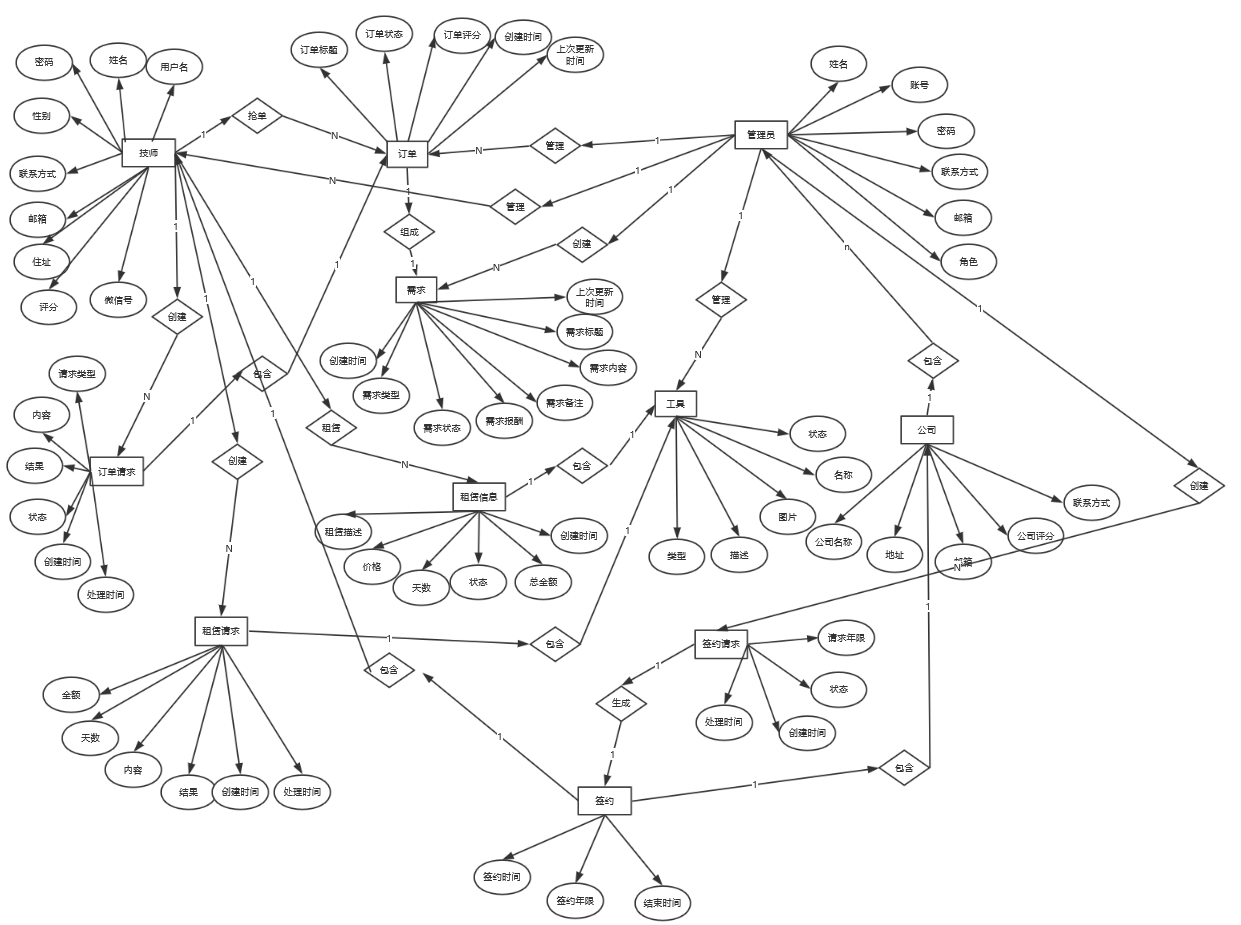


图3.3 ER图

1. **数据表设计**

1． 概念视图



图3.4概念视图

2． 逻辑视图



图3.5逻辑视图

3． 物理视图



图3.6物理视图

# 4 详细设计

由于本系统采用前后端分离开发，使用maven建立一个springboot项目工程[6]，名为CSSR，并在框架中选择WEB，mybatis框架，作为后端服务。使用vue-cli建立一个名为CSSR-WEB的项目作为项目的WEB端。

## 4.1 系统后端服务

系统后端服务作为本系统的一个最重要的子系统，主要为其他子系统提供数据访问的API，本系统主要实现对数据库的读取，写入，封装web访问接口，以及封装请求返回的数据，作为一个核心子系统，需要实现代码的规范，易于代码重构，代码修改，本系统采用MVC的开发模式[5][7][8]开发。主要分为数据访问层，业务逻辑层，控制层的接口和实现类的设计。

下图为后端服务器的开发整体架构：

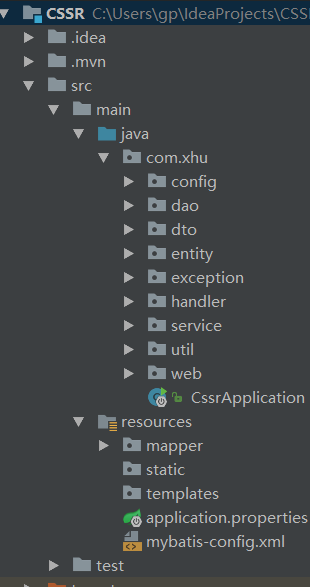


图4.1 服务器目录分层

### 4.1.1 数据访问层

数据访问层作为直接访问数据库的模块，在sql语句的编写上应当严谨，防止数据的写入错误，数据访问层主要采用mybatis框架实现对mysql数据库的访问，数据访问层的编写主要包括接口编写，实现类编写，mybatis的mapper文件编写，sql语句的编写。文件目录如下：

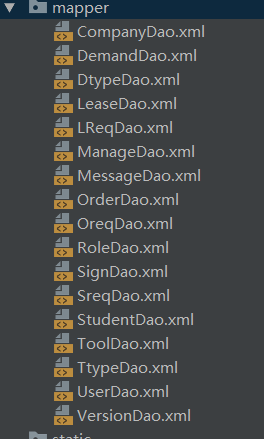
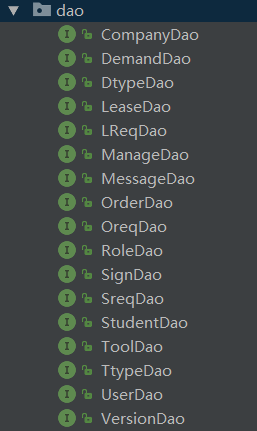


图4.2 服务器Dao层

### 4.1.2 业务逻辑层

业务层主要对数据访问层得到的数据进行封装传至控制层，对控制层传来的数据进行处理传至数据访问层，起到一个中间层的作用，其主要作用就是根据需求转化数据。对于本系统的复杂业务逻辑，同时操纵多张表时应当在该层同意访问，处理，为了防止数据的脏读脏写，应当在该层进行相应的事务控制。其目录结构如下：

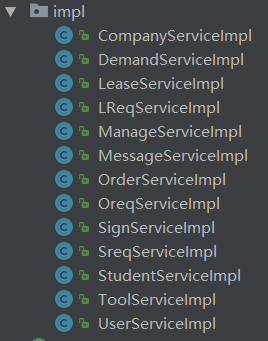
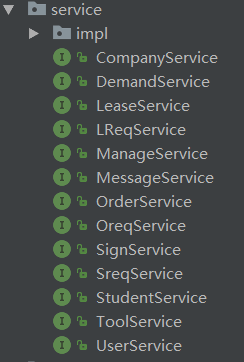


图4.3 服务器Service层

本系统的事务控制采用spring框架的声明式事务，利用注解@Transactional声明方法需要事务控制的效果。

### 4.1.3 控制层

控制层主要为了接受网络请求，根据请求方式以及请求参数访问业务层进而取到需要的数据或者实现其他子系统需要的功能。该层需要编写请求的URL以及访问方式（POST，GET，DELETE等），API编写采用Restful风格编写，易于其他子系统调用。其目录结构如下：



图4.4 服务器Controller层

### 4.1.4 统一异常处理

此外，根据spring事务对运行时异常进行事务回滚机制，系统的服务器采用spring统一异常处理，包括异常类，异常处理器（handler），异常枚举的编写[9]。

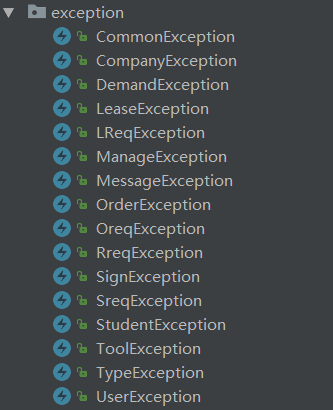
 

图4.5 异常类和异常枚举

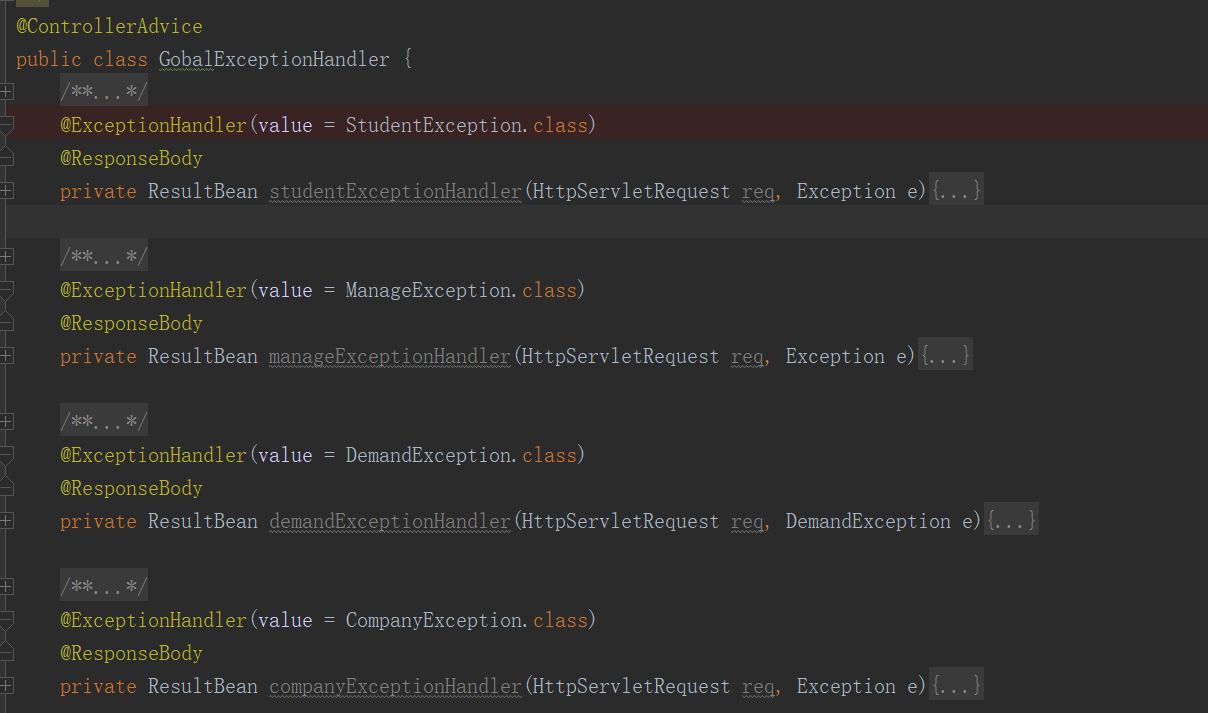


图4.6 异常处理器

### 4.1.5 设计思想与算法

后端服务器主要基于当前流行的微服务架构设计，具体架构描述如下：

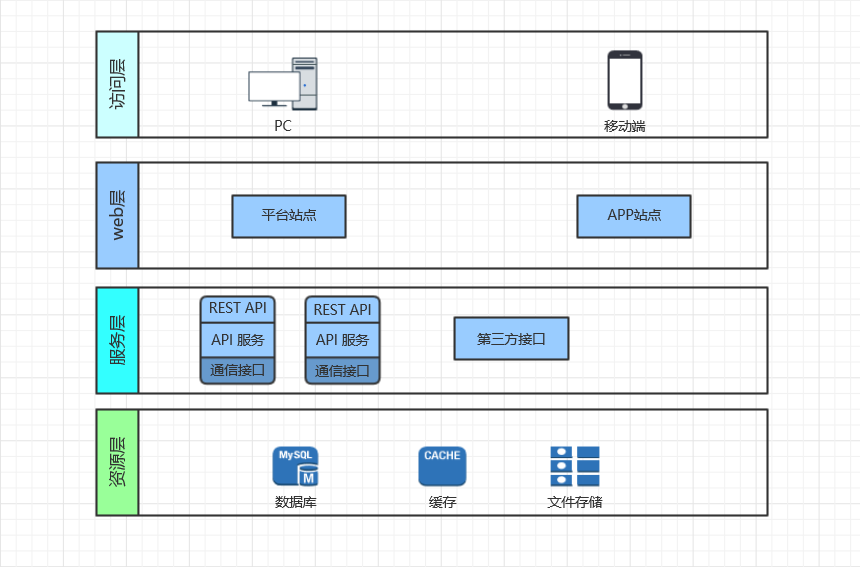


图4.7 系统大体架构

为了防止用户过多，操作过于频繁对数据库造成太大压力，本系统采用缓存技术来储存常用数据，防止过多的访问数据库，从而减轻数据库的压力，提高服务器的响应速度，也优化了用户体验。

### 4.1.6 关键代码

本模块由于采用springboot框架搭建，springboot两大重要文件，一个是springboot的程序启动类，本服务需要运行，只需启动此类，具体代码如下：

@SpringBootApplication

public class CssrApplication {

public static void main(String[] args) {

SpringApplication.run(CssrApplication.class, args);

}

}

另一个项目的配置文件，包括项目的运行端口，数据库设计mybatis配置[详情]，redis设计，配置代码如下：

1.总配置文件

server.port=8888

## 数据库配置

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/cssr?useUnicode=true&characterEncoding=UTF8&useSSL=false

jdbc.username=root

jdbc.password=123456

## Mybatis配置

mybatis\_config\_file=mybatis-config.xml

mapper\_path = /mapper/\*\*.xml

entity\_package = com.xhu.entity

##redis配置

spring.redis.host=localhost

spring.redis.port=6379

2.配置DataSource

@Configuration

//配置mybatis的mapper的扫描路径

@MapperScan("com.xhu.dao")

public class DataSourceConfiguration {

@Value("${jdbc.driver}")

private String jdbcDriver;

@Value("${jdbc.url}")

private String jdbcUrl;

@Value("${jdbc.username}")

private String jdbcUsername;

@Value("${jdbc.password}")

private String jdbcPassword;

@Bean(name = "dataSource")

public ComboPooledDataSource createDataSource() throws PropertyVetoException {

ComboPooledDataSource dataSource = new ComboPooledDataSource();

dataSource.setDriverClass(jdbcDriver);

dataSource.setJdbcUrl(jdbcUrl);

dataSource.setUser(jdbcUsername);

dataSource.setPassword(jdbcPassword);

//关闭连接后不自动提交

dataSource.setAutoCommitOnClose(false);

return dataSource;

}

}

3.配置系统的事务管理器：

@Configuration

public class SessionFactoryConfiguration {

@Value("${mybatis\_config\_file}")

private String mybatisConfigFilePath;

@Value("${mapper\_path}")

private String mapperPath;

@Value("${entity\_package}")

private String entityPackage;

@Autowired

@Qualifier("dataSource")

private DataSource dataSource;

@Bean(name = "sqlSessionFactory")

public SqlSessionFactoryBean createSqlSessionFactoryBean() throws IOException {

SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();

sqlSessionFactoryBean.setConfigLocation(new ClassPathResource(mybatisConfigFilePath));

PathMatchingResourcePatternResolver resolver = new PathMatchingResourcePatternResolver();

String packageSearch = PathMatchingResourcePatternResolver.CLASSPATH\_ALL\_URL\_PREFIX + mapperPath;

//设置数据源dataSource

sqlSessionFactoryBean.setDataSource(dataSource);

//设置mapper路径

sqlSessionFactoryBean.setMapperLocations(resolver.getResources(packageSearch));

//设置实体类的包

sqlSessionFactoryBean.setTypeAliasesPackage(entityPackage);

return sqlSessionFactoryBean;

}

}

4.配置mybatis

<configuration>

<!-- 配置全局属性 -->

<settings>

<!-- 使用jdbc的getGeneratedkeys获取数据库自增主键值 -->

<setting name="useGeneratedKeys" value="true"/>

<!-- 使用列标签替换列别名 默认:true -->

<setting name="useColumnLabel" value="true"/>

<!-- 开启驼峰命名转换 表中user\_id 映射实体类 userId-->

<setting name="mapUnderscoreToCamelCase" value="true"/>

<setting name="logImpl" value="STDOUT\_LOGGING" />

</settings>

</configuration>

@Configuration

@EnableTransactionManagement

public class TransactionManagerConfiguration implements TransactionManagementConfigurer{

@Autowired

private DataSource dataSource;

@Override

public PlatformTransactionManager annotationDrivenTransactionManager() {

return new DataSourceTransactionManager(dataSource);

}

}

具体详细代码请参考源代码。

## 4.2 WEB端（汽修服务端）子系统

WEB端主要功能是使汽修厂的管理员在上面处理汽修厂的业务，实现其业务功能。底层实现需要WEB端向服务端根据用户的选择发送请求取得数据，进而拿到数据像用户展示，WEB端需要达到页面风格一致，操作简便。本系统的WEB端采用vue.js搭建，利用vue实现页面与数据的双向绑定，提升用户的使用体验，使页面的数据随着用户的操作即使变化。此外采用elementUI框架设计WEB端的总体界面，使界面更加优美。

### 4.2.1 界面设计

登陆页面作为PC端的直接入口，其界面如下：



图4.8 WEB端登录界面

为了增加用户的体验，增加即时的提示来提示用户输入项的要求。

此外，WEB端用户使用最多的应该是菜单项，对于菜单项，应当易于切换，反应迅速，其大致菜单界面如下：

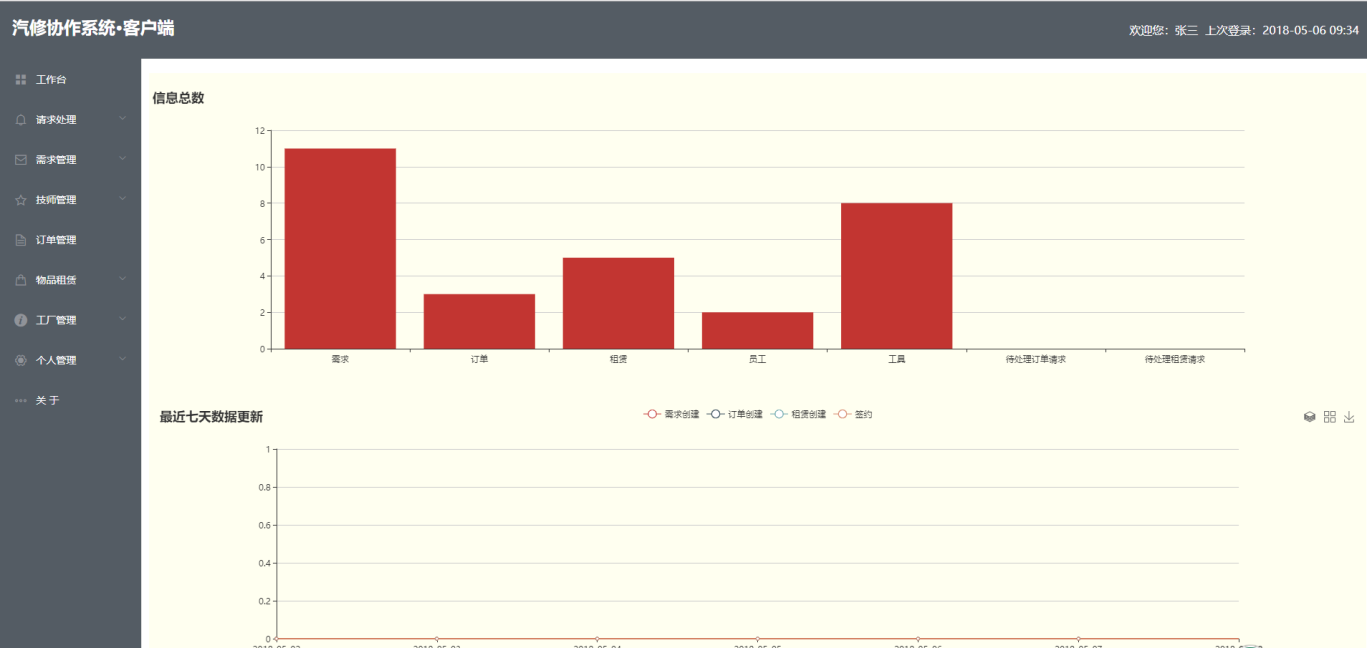


图4.9 WEB端主页

项目具体细节请参考项目本身。

### 4.2.2 设计思想与算法

WEB端的设计主要基于vue的路由跳转，elementUI，以及vue的数据双向绑定，设计的目的注重提升系统的操作流畅度，使用户的体验感得以提升。WEB设计大体如下：

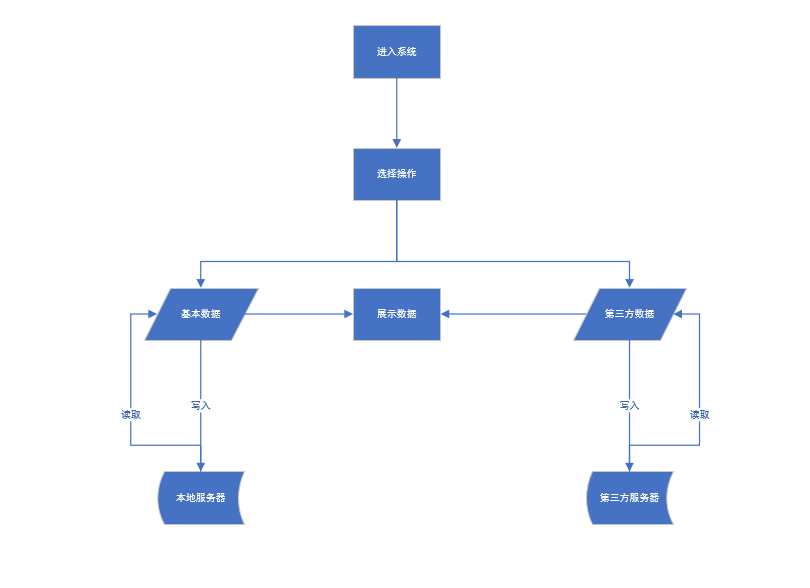


图4.10 WEB端设计思想

### 4.2.3 关键代码

WEB的业务关键，实现菜单的路由跳转，数据的请求，数据的呈现三个方面，路由的跳转依赖路由的配置，包括路由设置，路由拦截，其配置如下：

const router = new Router({

mode:'history',

routes: [

{

path: '/',

name: 'login',

redirect:"/login"

},{

path:'/home',

name:'home',

component:Home,

redirect:"/home/my-work",

children:[

{

path:'sign-request',

name:'sign-request',

component:SignRequest

},{

path:'history-order',

name:'history-order',

component:HistoryOrder

}

]

},

{

path:'/login',

name:'login',

component:Login

}

]

})

router.beforeEach((to, from, next) => {

if(to.path == "/login" || to.path == "/" ){

next();

}else if(sessionStorage.getItem('manageName') == undefined){

next("/login");

}else{

next();

}

})

此处只展示部分路由配置，其余都大同小异，具体代码请参考源代码；

另外数据的请求依赖于跨域请求，所以需要设置系统的服务器代理，解决跨域请求无法请求的问题，代码如下：

proxyTable: {

'/api/': {

target:'http://127.0.0.1:8888/api',

changeOrigin: true,

pathRewrite: {

'^/api':''

}

}

}

WEB端像服务器请求数据都是通过http异步请求拿到数据，具体请求方式包括get，post，delete，请求代码如下：

this.$http.post(url, sendData).then((res)=> {

//解析res回调参数

});

Get方式与delete方式与此格式一样。请求结束之后通过解析res的内容，通过服务器传回的状态码向用户展示不同的内容。

由于数据的呈现代码过多，具体请参考源代码。

## 4.3 消息推送模块

为了使移动端与PC端能够实现消息的即时推送例如技师在移动端发送一个租赁请求，PC端应该即时弹出提示框告知管理员前去处理。为了提升响应速度，本处我们需要调用第三方接口goeasy达到即时消息推送的效果。

### 4.3.1 界面设计

界面设计需要弹出一个消息通知框，具体如下：



图4.11消息通知提示框

### 4.3.2 设计思想与算法

消息推送模块的设计思想需要调用中间件，具体实现业务流程如下：



图4.12消息推送模块设计思想

### 4.3.3 关键代码

本模块的关键代码在于maven添加第三方库，然后调用第三方接口发送消息推送之后PC端进行解析展示内容：

添加三方库

<dependency>

<groupId>io.goeasy</groupId>

<artifactId>goeasy-sdk</artifactId>

<version>0.3.8</version>

</dependency>

服务器业务层发送消息核心方法代码如下：

private static final String REGIONHOST = "https://rest-hangzhou.goeasy.io";

private static final String APPKEY = "BC-b904a79209154acf92c19edfcd2df323";

public void sendMessage(String mes){

try {

GoEasy goEasy = new GoEasy(REGIONHOST,APPKEY);

goEasy.publish(CHNANEL, mes);

}catch (Exception e){

throw new CommonException("消息推送异常");

}

}

PC端解析推送内容:

mounted(){

var goEasy = new GoEasy({ appkey: "BC-b904a79209154acf92c19edfcd2df323"});

goEasy.subscribe({

channel: "cssr",

onMessage: function (message) {

if(message.content == "402"){

Notification.info({

title: '提示',

dangerouslyUseHTMLString: true,

offset: 50,

message: '<span>您有新的订单请求，请前去处理！</span>',

onClick : function () {

Notification.close();

},

duration:30000

});

}

}

});

}

PC端解析需要先订阅goeasy上创建的应用，才能接收goeasy主动发送的消息。

# 5 测试

系统的测试分为三个阶段，业务接口测试，WEB接口测试，系统集成测试。其中业务接口测试采用junit单元测试，WEB接口采用postman软件模拟http请求测试，系统的继承测试需要服务器，PC端，移动端三端进行所有的业务功能测试。测试过程中应保证测试用例严谨，全面，以此保证测试的可靠性。

## 5.1 业务接口测试

此处展示分页查询单个测试方法，更多测试详情参考源代码：

@Test

public void list() throws Exception {

Demand demand1 = new Demand();

List<Demand> list = demandDao.list(0,5,demand1);

for (Demand demand:list) {

System.out.print(demand);

}

assertEquals(5,list.size());

}

该方法中测试用例如下：当前页0，查询数量5，查询条件无，预期查询数量5

运行该方法，得到测试结果：

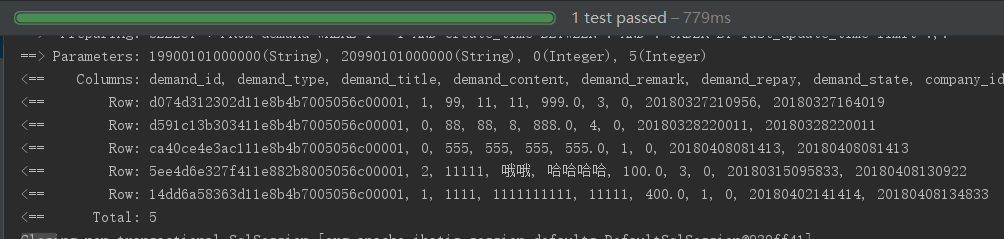


图5.1 单元测试需求分页查询

根据测试结果得知，测试通过，表示分页功能实现。

## 5.2 WEB接口测试

WEB接口测试采用postman模拟http请求，由于本系统的数据交换格式统一采用json格式，部分测试过程如下：

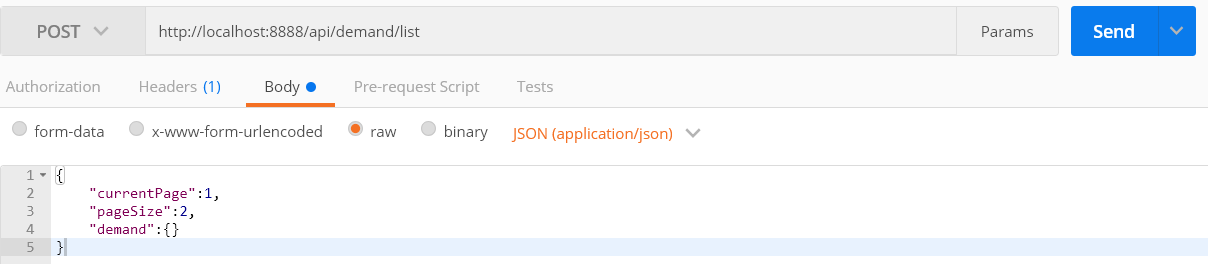


图5.2 WEB接口测试请求

请求结束后获得结果：

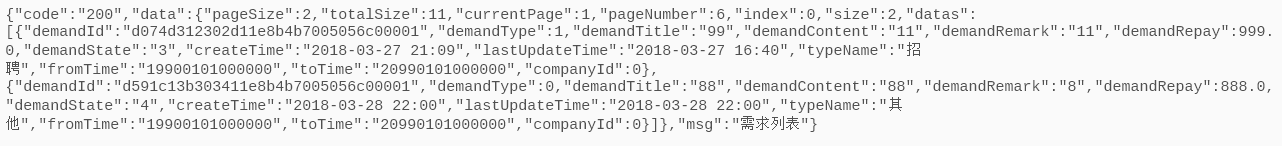


图5.3 WEB接口测试请求结果

抢单测试：

1.测试需求被受理仍创建订单

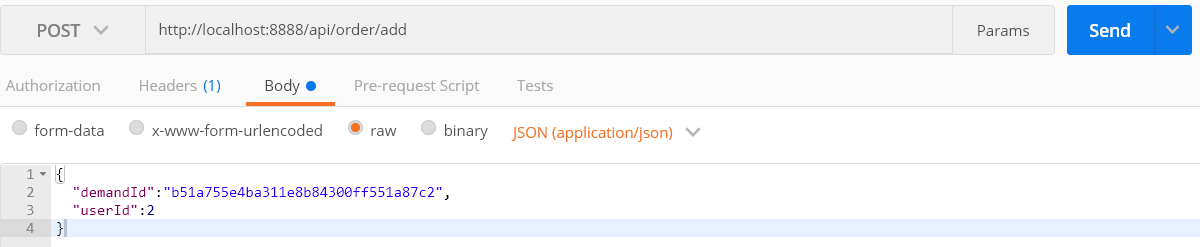


图5.4 抢单WEB接口测试

访问结果：

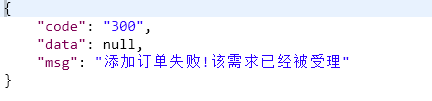


图5.5 重复抢单测试结果

2.正确抢单

修改需求id为”f22abbb643a511e8b84300ff551a87c2”再次请求，请求结果如下：

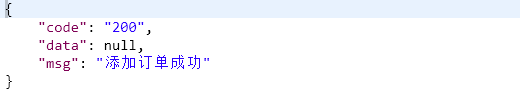


图5.6 抢单成功测试

此时会自动创建一条订单信息：

C:\Users\gp\AppData\Roaming\Tencent\Users\1101494689\TIM\WinTemp\RichOle\{2_TY3$MPSNC$]M24W3SIDI.png

图5.7 抢单生成的订单信息

以上为抢单操作的测试，测试结果表示成功。

由于篇幅有限，更多测试不便列出，详情参考源代码和集成测试。

## 5.3 系统集成测试

系统的集成测试本处只展示服务器与WEB端的测试,主要测试三个方面：登陆测试，信息查询测试，消息推送测试，测试过程如下:

1.查询测试





图5.8 PC端查询需求

2.消息推送测试

此处测试用户订单请求，使用postman模拟用户发送，测试PC端的消息推送，测试过程如下：

模拟请求：

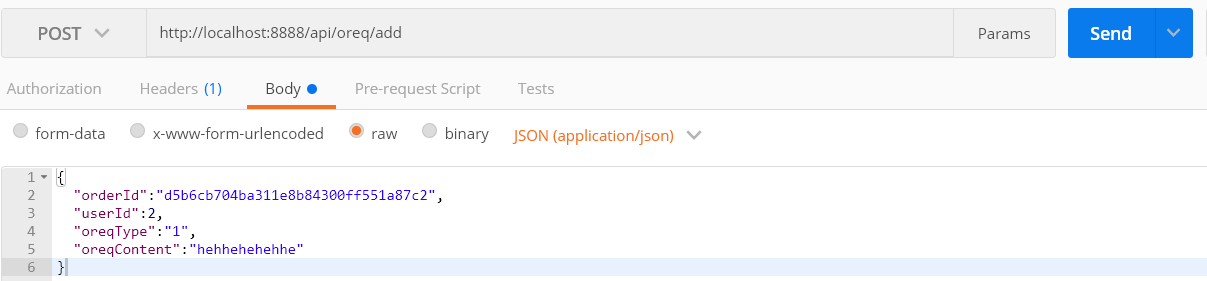


图5.9 模拟抢单请求

请求结果：

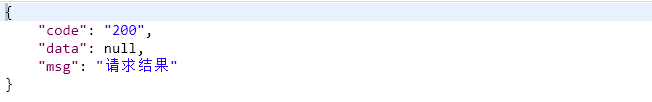


图5.10 抢单结果

此时查看PC端右上角的消息推送效果：



图5.11 PC端实时消息通知

PC端查看订单请求：

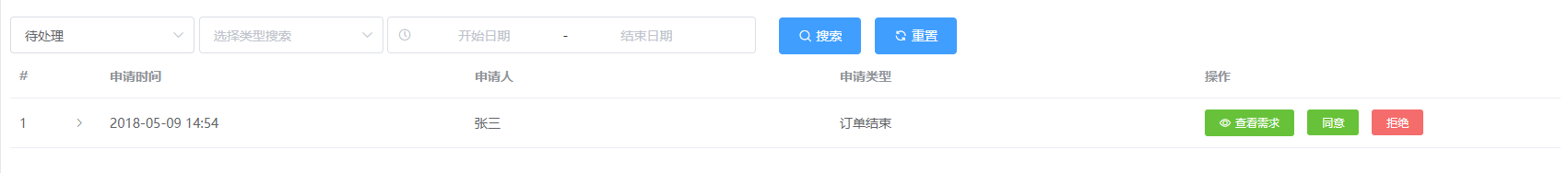


图5.12 查询刚生成的订单

到此，消息推送测试完成。

## 5.4 测试结论

经过以上测试，本系统完整的实现了预期的业务功能，能很好地解决汽修厂和技师的业务需求。

# 6 结论

经过不懈努力，最终通过利用springboot，vue，微信小程序开发完成了本系统的开发，共完成了以下功能：

1. 汽修厂的信息统计模块

PC端首页可动态的展示整个汽修厂的数据统计，待办事务的数量，最近七天的数据统计。

1. 汽修厂的请求处理模块

汽修厂管理员可对技师的订单请求或者租赁请求进行处理。

1. 汽修厂的需求管理模块

可对汽修厂的需求进行添加，添加过程中制定需求类型，并能够对需求进行状态追踪，随时进行管理。

1. 汽修厂的技师管理模块

实现了查看最近合作技师，管理签约技师，向最近合作的技师发送签约请求，以及查看自己的签约签约请求状态。

1. 汽修厂的订单管理模块

实现了对订单的管理，以及对生产该订单的需求查看，包括对已经结束的订单进行评分，该评分也将影响处理该订单的技师的评分。

1. 汽修厂的工具管理，租赁管理模块

实现了汽修厂对工具管理，如增加工具，删除工具等，并可直接查看被租工具的租赁信息以及租赁者的信息。

1. 汽修厂的工厂信息以及管理员信息管理模块

利用高德地图API实现了修改公司定位的功能，以及修改公司其他信息，管理员个人信息等。

1. 技师的抢单模块

实现了技师在移动端对汽修厂发布的需求进行的抢单操作，能有有效的实现用户抢单正常，以及效率性。

1. 技师的注册登陆模块

实现了技师针对本系统的登陆，注册功能。

1. 技师的租赁工具模块

实现了技师对汽修厂的工具进行请求租赁，管理自己的租赁信息等功能

1. 技师的信息管理

实现了技师对自己热个人信息进行修改的功能。

1. 技师的订单管理

实现了技师对自己处理的订单进行管理，包括订单结束请求，订单中断请求的发送。

1. PC端与移动端的消息推送模块

利用goeasy实现了PC端与移动的即时消息推送，保证了汽修厂管理员能够及时处理事务。

由于本系统涉及到的数据过多，可能在平时大量查询过程中偶尔会有卡顿，还需要优化查询功能，优化数据库设计，或者使用非关系数据库存储一些数据。此外，本系统的WEB端目前各个浏览器展示页面会有些许不同，这是由于还未做对各种浏览器的兼容，还应当进行优化。对于汽修厂的签约模块。仅仅实现了汽修厂发送签约请求，应当增加合同书的上传和下载，后期可能增加第三方的WEB端的word处理工具进行处理。目前PC端的用户仅仅面向汽修厂的管理员，角色过于单一，后期可能会增加多种角色，处理不同的事务。后期可能还需要再增加一套后台管理系统面向开发者使用，主要对日志，汽修厂，用户，系统设置进行管理，还可能会与更多的汽修厂合作，使技师能有更多的抢单机会。

# 7 总结与体会

通过这次设计，使我受益匪浅。在整个开发的过程中，让我自学了springboot，vue的技术，让我了解了很多在课堂上无法了解到的新技术，同时也让我明白，开发一款功能略微齐全的系统是有多么不容易，也让我学会了与小伙伴协同开发时需要注意的事项，并从中提升自己的团队协作能力。

整套系统的服务器开发采用springboot开发，当然也是基于spring，java的框架，让我对java基础更加熟悉，对开源框架有了更新的认识，比如在校内我们常常使用的是spring + springMvc + mybatis框架的组合开发项目，在开发过程中能明显感觉到我们的开发需要使用很多的配置文件，显得十分繁琐，而springboot让我们大大减少了这些配置，这也让我对springboot的注解更加了解。由于需要减少对数据库的压力，使用到了redis非关系型数据库作为数据缓存工具，加快对数据的访问速度，从中使我深入了解掌握了springboot框架与第三方框架的集成，maven的仓库配置与使用。

对于系统的WEB端开发，使我自学了vue框架，让我了解了MVVM模型的意义，了解了数据双向绑定的含义，使我的开发效率得以提升，在学习vue的过程中，还使我学会了如何调用第三方接口，比如系统用到的高德地图API，Echart的API，包括goeasy的API，不仅如此，更让我学会了前端开发需要的包管理工具npm。

由于本系统针对汽修厂与技师开发，在开发之前需要对系统的需求有个全面的了解，因此需要调研以获取需求，在开发初期查阅相关资料，了解汽修方面的知识，相信这些知识在今后的学习中也能起到至关重要的作用。 总之，整套系统让我巩固了现有知识，学习了新的知识，更让我学会了与同学合作完成一件事，这是很难能可贵的。

# 致 谢

回想当初选择命题到现在完成该系统已经过去四个多月的时间，非常感谢帮助过我的同学，老师。

在此我要特别感谢我的指导老师谭老师，是他对待工作严谨，认真的态度教会了我在学习，完成毕设的过程中应该让自己的思维缜密，也让我有动力坚持了几个月的学习并写出自己的毕设。尤其需要感谢在确定命题的前几周，谭老师与我分析的系统的需求，同时也确定了我的开发方向，使我有了对一个系统架构的初步认识，让我明白一个系统的开发数据库设计十分重要，数据库的设计好坏决定后期的编码能走多远，能有多便捷。

感谢谭老师，感谢知乎，百度贴吧，Github，CSDN论坛帮助过我的朋友们，另外需要感谢与我一起完成的本系统的刘同学，是我们一起努力才能完成这样一个较好的系统。

# 参考文献

* 1. 张峰.应用SpringBoot改变web应用开发模式[J].科技创新与应用,2017(23):193-194.
  2. 麦冬,陈涛,梁宗湾.轻量级响应式框架Vue.js应用分析[J].信息与电脑(理论版),2017(07):58-59.
  3. Clarence Ho. Using MyBatis in Spring[M].Apress:2012-06-15.
  4. 谭一鸣. 基于微服务架构的平台化服务框架的设计与实现[D].北京交通大学,2017.
  5. 符培炯,杜忠军.Spring在实现MVC构架中的应用[J].计算机技术与发展,2006(06):236-238+241.
  6. Balaji Varanasi,Sudha Belida. Maven Dependency Management[M].Apress:2014-06-15.
  7. 刘向阳,朱根,王庆飞.《JavaWeb应用开发技术》课程教学改革探索[J].福建电脑,2013,03:77-78.
  8. Metsker S J.Java设计模式[M].第2版.电子工业出版社,2012年09月.
  9. 郭琳.JavaWeb中错误异常的处理方法分析[J].电子技术与软件工程,2015,13:85-86.
  10. 张雪云,牟艳,张九博.基于微信小程序的电源监测管理系统设计[J].计算机与现代化,2017(12):98-102+107.