**分类号：TP311.5 U D C：D10621-408-（2022）5503-0**

**密 级：公 开 编 号：2018121032**

**成都信息工程大学**

**学位论文**

**基于Vue的汽车资讯网站的设计与实现**

|  |  |
| --- | --- |
| **论文作者姓名：** | **向峰** |
| **申请学位专业：** | **网络工程** |
| **申请学位类别：** | **工学学士** |
| **指导教师姓名（职称）：** | **王燚（教授）** |
| **论文提交日期：** | **2022年6月1日** |

**汽车资讯网站的设计与实现**

**摘要：**本文介绍了汽车资讯网站的功能和相关开发技术。汽车资讯网站基于Vue框架以及Express框架进行开发，通过基于WebSocket的SocketIO进行消息双向通信，ElementUI作为UI库提供样式，前后端都采用js作为开发语言，采用MySQL做为数据库提供数据源支持，阿里云轻量级服务器部署的Tomcat提供资源提供服务。本系统目前采用客户端、商家端、管理端三合一模式，一套系统，三方使用，轻便高效。本系统实现了账户的注册、登录、客户端的资讯展示、新车的多条件查询搜索、车辆详情、新车的购买入库、询问卖家、二手车的挂售信息设置、出售、购买，车友会的品牌选择、新帖的发布、新帖详情查看、评论的发布，个人信息页面的头像上传、信息设置、注销、申请成为商家。商家可以进行新车信息上传发布、修改车辆信息、删除车辆信息。系统管理员可以进行新帖的发布。经过多次测试使用，已经实现汽车资讯网站所需的相关功能。

**关键词：**汽车资讯；Vue；Express；SocketIO；

**Design and Implementation of Automobile Information Website**

**Abstract:**This paper introduces the function and related development technology of an automobile information website.The automotive information website was developed based on Vue and Express , built as a two-way communication platform by using SocketIO on this website.It was developed based on JavaScript and useing MySQL as the database. In addition, it provide functionality of online pictureand videos viewing by config an  an online tomcat server instance.This Web application provides functions for merchant, administrator and clients all together, includingslogin, buying cars, car exhibition and other functions.People can also sell their cars on this platform. In addition, users can communicate with each other in the car club. Merchants can add car information and administrators can modify vehicle news.

**Key words:**Carinformation;Vue;Express;SocketIO；

**目 录**

论文总页数：32页

[1 引 言 1](#_Toc21194)

[1.1课题背景 1](#_Toc21804)

[1.2课题内容简介 1](#_Toc17508)

[2 相关技术概述 2](#_Toc18384)

[3 系统设计 4](#_Toc28103)

[3.1需求概述 4](#_Toc16612)

[3.2需求分析 4](#_Toc14194)

[3.3功能模块设计 7](#_Toc1860)

[3.4数据库设计 8](#_Toc13261)

[3.5 用户权限设计 14](#_Toc30437)

[4 汽车资讯网站的开发与实现 15](#_Toc15736)

[4.1 页面请求响应 15](#_Toc15453)

[4.2 代理/打包设置 16](#_Toc319)

[4.3 动态路由生成 17](#_Toc405)

[4.4 后端结构 18](#_Toc6286)

[4.5 登录注册模块 19](#_Toc15587)

[4.6新车与二手车选购模块 21](#_Toc16532)

[4.7 车库模块 23](#_Toc7716)

[4.8 车友会模块 24](#_Toc23418)

[4.9 资讯管理 25](#_Toc12695)

[4.10 汽车信息管理 25](#_Toc5208)

[4.11 个人中心模块 26](#_Toc20091)

[结 论 29](#_Toc976)

[参考文献 30](#_Toc22470)

[致 谢 31](#_Toc25203)

[声 明 32](#_Toc18287)

1 引 言

1.1课题背景

人们生活物质水平的提高，让车房都成了一大消费热点，而消费的主要对象主要是30-40岁工薪阶层，因为这一年龄层的人群大多已经有了相对稳定的工作收入，对汽车的消费热情较高同时需求也比较大。这个年龄段的人一方面消费理想，在购车前需要进行大量的调研；同时他们又工作繁忙，没有很多的时间进行比较和筛选。各大城市往往会开办一些汽车展，以供人们近距离的了解各品牌汽车。同时各类汽车实体店也有配套的汽车咨询服务以及过户，保险，维修等服务。线下汽车市场最大的好处就在于为用户提供了一站式服务，能够让用户更加真切得观看、试驾到自己有意向的汽车。同时线下交易也能一手交钱一手交货，买的更加放心，避免了一些不必要的损失[5]。

但线下汽车资讯存在一些固有的缺点。无论是展览会还是4S店，都需要巨大的人力成本，店铺、场地租用成本。最终这些成本会提高交易价格。此外，线下看车买车还存在巨大的局限性：客户一天最多到几家4S店咨询，不仅得到的信息有限，还费心费时费力。由于线下看车的这些不足，催生了众多的汽车网站，让用户线上看车买车。然而这些网站上存在大量的“二房东”。由于用户根本无法与卖家直接沟通，既不安全也不实惠。

综合来看，线下看车存在效率低、成本高等不利因素，而线上的汽车网站又不能给用户提供完全的咨询和交易的可靠性，因此，又必须开发一个即能够克服两者缺点，同时又能提供两者优势的汽车资讯网站。

1.2课题内容简介

基于上述分析，有必要设计一款能够让用户在方便看车的同时能够直接与商家或者二手车卖家对话的网站。该网站提供简单便捷的注册方式，高效的选车工具，用户能够快捷的选择到自己喜欢的车辆，网站风格简约，没有花里胡哨的广告，提升用户体验。买家可以与卖家一对一通信，拿到第一手汽车资料，买的放心。不仅如此，用户还可以在车友会于其他用户交流，了解到更多关于汽车的信息，相互参谋，避免非理性消费。此外，网站也为商家提供管理界面，让商家能够及时快捷的对汽车进行管理。

本网站需要实现用户选购买卖汽车，车友会，汽车管理，资讯发布展示等功能。同时将客户端、商家端、管理员端三端合一，依据用户权限获取各自能够访问的页面，便捷的同时提升网页各模块的复用性。

2 相关技术概述

（1） Vue2

Vue是用于构建用户界面的渐进式JavaScript框架。通过组件化模式提高了代码的可复用性，同时让代码更加便于维护。声明式的编码方式与Vue提供的OptionsAPI让开发人员在不需要重复操作真实DOM节点的情况下能够多次复用DOM节点，提高开发效率，也让开发人员将开发关注重心注重于数据层面而非页面的显示上[1,2]。

通过使用插值语法和指令语法，Vue以极低的性能成本让数据在页面上展示。Vue页面的生命周期大致分为四个阶段，因此Vue提供了8个生命周期函数分别让页面在不同时段执行用户所需要的功能，例如Mounted阶段可以用于发送Ajax请求，BeforeDestroy阶段可以用于清除登录信息，清除定时器等。

Vue中有组件化与数据驱动两大概念，在不同的Vue版本中具有不同的实现方式。Vue2中提供OptionsAPI来对页面提供数据支持，以下就是OptionsAPI中的重要属性：

Data:用于保存当前Vue页面初始化时的变量，这些在data中定义的变量可以在Html代码中直接展示，也可以在Js中调用，让变量能够在Html中动态变化。

Computed:计算属性，根据data数据发生变化就会重新解析模板，也可以直接读取。

（2） Express

Express 基于 Node.js 的Web应用程序开发框架，可以为网页和APP提供后台服务。Express框架建立在Node.js内置的HTTP模块上。Express框架有一个重要的特点就是中间件，即处理HTTP请求的函数，通过Next()将请求传递给下一个中间件。它每次接受Req，Res，Next三个参数，分别代表请求，响应和下一个中间件。Express 4.0版本后开始，Express的路由器功能成了一个单独的组件Express.Router。Express.Router和小型Express应用程序一样，有自己的Use、Get、Param和Route方法。Express的路由器可以通过解析路由路径以处理不同的HTTP请求。

（3） MySQL

MySQL是关系型数据库的典型代表，是当前最流行的数据库之一，由MySQLAB公司开发。MySQL将数据存在于不同的表中，通过字段、主键、外键等。定义各种表内与表间的关系。MySQL使用的是最标准的SQL语言，体积小，速度快，开放源码，也因此成为了大多数中小型网站热衷的数据库。MySQL还有针对不同规模应用的三种架构，分别是单点，复制，集群，可以针对性的提供自己的功能。

（4） Element UI

Element UI是当前最火的UI库之一，它不依赖于Vue，确实当前和Vue配合的最紧密最方便的UI库。它由饿了么前端团队开发，UI风格简约实用，提供大量提前写好的属性以及模板以供开发者使用，极大地提高了开发效率，他的组件化也是非常契合Vue的组件化思维，让Vue项目开发者无需大量重复写一些小组件，而将更加关注数据的处理。在引入方面，ElementUI提供全局引入和按需引入两种方式，根据用户不同需求进行使用。

（5） Vue-cli

Vue-cli是基于Vue进行开发的项目脚手架，可以快速进行Vue项目的快速搭建。Vue cli可以将Vue生态中的各个工具进行标准化管理，因为它默认内置Webpack工具，且有自带合理的包管理配置，可以扩展其他插件以保证插件与项目的平稳衔接。在安装Vue-cli之后，仅仅通过简单的命令行即可完成Vue项目的创建[6]。在创建过程中可通过可视化的方式对Vue项目进行预设引入模块配置。

（6） Webpack

Webpack是一种模块打包工具，使用Webpack可以管理项目中模块之间的依赖关系，可以将各模块之间的文件进行打包生成web页面所需要的Html、Css、Js以及其他静态资源文件，让开发过程更加高效。Webpack会自动将各模块之间的调用关系进行分析，优化并生成最后的静态资源。Webpack将整个项目当成整体，从给定的主文件开始寻找项目中所有的依赖文件，通过Loaders来进行处理，将这些文件编译转化打包成为浏览器可以识别的Js文件。除了Loaders，Webpack还可以使用Plugin来对打包进行优化压缩，用以生成体积更小的Bundle文件，使页面加载时更快[6,7]。

3 系统设计

3.1需求概述

本网站开发所要达成的目的是在方便客户对于汽车的选购的同时满足客户能够多方面获取汽车讯息的需求。因此系统需要汽车新闻、交流工具、车友会以及汽车选购市场等功能。并要求界面简洁，流程简单且安全。此外，为避免线下二手车市场的复杂情况，还应该提供二手车交易功能，通过使用该功能，用户可以自主提交二手车发售消息，从而为客户提供直接交流的平台。

网站管理方面，超级管理员和商户应该可以在自己权限范围内对汽车资讯、新车信息进行管理。在提交数据过程中，前台应当负责对于数据进行验证，符合规范的情况下才允许数据上传。

服务器方面，消息发布功能有上传文件的需求，因此需要搭建在线服务器。本课题采用阿里云轻量化服务器，在其中部署Tomcat服务，提供在线资源。

3.2需求分析

（1） 登录注册模块

登录：登录模块需要对用户信息进行验证，通过验证结果告知用户登录结果。登录成功后，需要让用户保持登录状态。系统登录流程如图3.1所示。

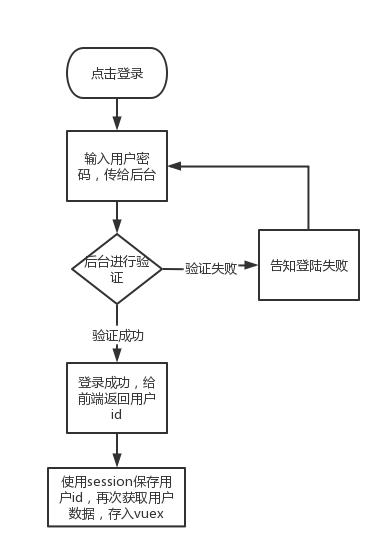


图 3.1 用户登录流程图

注册：注册页面需要用户填写用户名、密码等信息。后台需要对用户提交的数据进行查重。查重通过则完成用户名密码填写，但需要对未让用户填写的字段进行默认值设置。用户注册流程如图3.2所示。

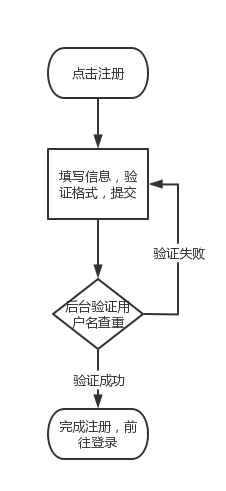


图 3.2 用户注册流程图

（2） 新车/二手车选购模块

市场存在的汽车品牌数量庞大，因此需要设计一个条件查询工具供用户使用。

用户选订汽车提交支付信息后，后台需要对支付密码、余额等以此进行检验。完成订购后，需要在订单表和车库表插入新订单与新车信息。如果是二手车购买，则需要修改车辆所属人。汽车交易流程如图3.3所示。

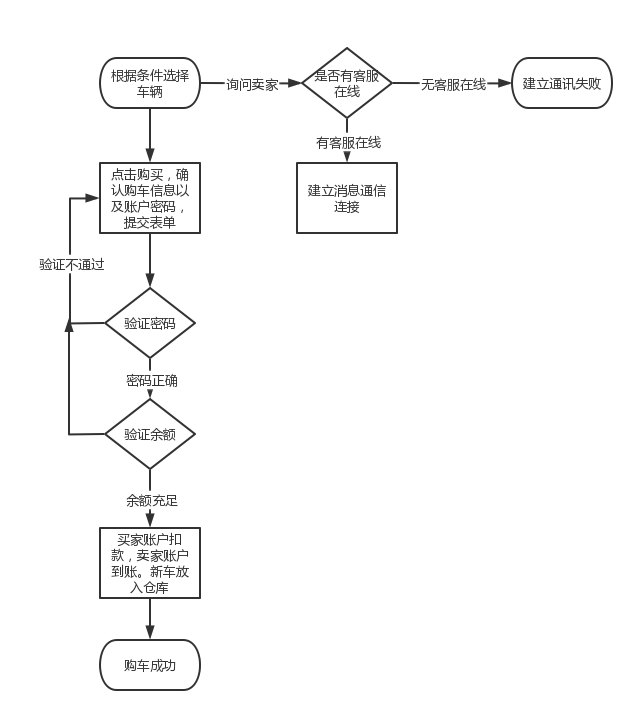


图 3.3 汽车交易流程图

（3） 车库模块

个人车库中主要涉及已购车辆的展示与个人车辆挂售的功能，因此需要提供用户修改车辆挂售信息的工具。每一次修改挂售信息，后台都需要校验用户提供的支付密码，以防他人操作。车库挂售流程如图3.4所示。

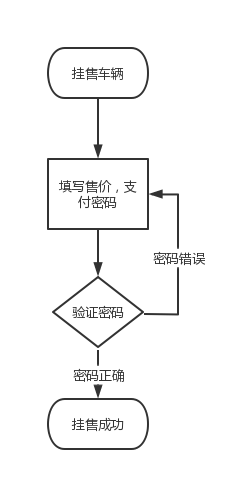


图 3.4 汽车挂售流程图

（4） 车友会模块

车友会首页会提供所有品牌车友会的入口。选择一个车友会进入之后，会展示车友会近期发表的帖子的简述信息，点击查看全文可以浏览帖子的具体内容。每个帖子提供评论功能，也可以查看近期评论并能够发布个人评论。用户发布帖子需要同时能够发布图片与文字。

（5） 车辆管理模块

车辆管理模块需要商家权限，同时需要提供新车上传，车辆信息编辑以及删除功能。

（7） 资讯管理模块

资讯管理模块需要超级管理员权限。需要提供资讯上传，编辑，删除等功能。

（8） 个人中心模块

个人中心模块需要完成的功能主要包含信息的更改，头像上传。用户只能够选择的本地文件上传头像。用户可以选择个人身份，可以申请成为某品牌客服。

3.3功能模块设计

（1） 登录注册模块

登录：对于用户信息验证，均在后台完成。后台根据用户名查找对应密码完成验证并返回用户id。前端将用户id保存如session即可保持登录状态。紧接着再次通过用户id查询出除密码外的所有用户信息，前端将用户信息保存在Vuex中以便随时调用。

注册：用户注册信息提交后，后台调用查重功能对用户注册信息查重。通过查重后，对于余额、头像等字段分别设置了默认值，以避免数据库查询时报错。

（2） 新车/二手车选购模块

车辆选购模块包含三个小模块，分别是选车、与商家交谈、购买付款。

选车环节为满足客户需求，设置了多条件选择器，可以通过品牌、车型、价格区间、名称等多个条件进行筛选，以此提供符合用户条件的车辆列表以供用户选购。

为了让用户得以了解到信息能够清楚明了，为用户提供与商家发消息的平台。只要相应商家有客服在线的情况下，就可以随时进行咨询。一旦双方建立了通讯，即使客服或用户已经下线，未读的消息也会在下次通讯时展示出来。

点击购买车辆后，会出现表单，其中会有车辆相关信息以供用户确认，用户需要输入支付密码并提交表单。表单会在后台以此对支付密码，账户余额进行验证，验证通过之后就会进行金额转移的流程。新车完成购买之后，会新增订单记录，并将新车入库。二手车完成购买之后，除了新增订单记录，车库中二手车的所属人信息会进行相应更新。

（3） 车库模块

个人车库中主要涉及已购车辆的展示，个人车辆的挂售、取消挂售的功能。

当车主需要将自己的车辆卖出时，会出现表单让用户填写售价以及个人支付密码等信息。填写完毕之后，后台对车主密码进行验证，验证通过即将车辆添加到二手车市场。个人车库中的该车也将处于挂售状态，车主可随时取消挂售。

在二手车市场中的车被购买之后，车辆所属人会被修改成买家。

（4） 车友会模块

车友会首页通过查询车友会表获取全部车友会信息。在每个车友会论坛中，默认展示最近的十条帖子。每条帖子默认只展示简要信息，但可以通过点击查看详情按钮查看帖子具体内容。

发布帖子功能采用的是富文本文本框，可以往里面直接添加图片，并可以设置字体。

（5） 车辆信息管理模块

车辆信息管理模块在route.js中设置了身份限制，只有商家身份可以进入。

新车上传以及编辑功能必须给提供车辆名字，展示图，售价，相关参数等信息，否则无法上传或保存信息成功。车辆展示图将通过调用云服务器上的文件上传接口，先完成图片上传保存，再获取保存路径与其他车辆信息一起存入数据库中。

（7） 资讯管理模块

资讯管理模块在route.js文件中设置了身份限制，只有超级管理员身份能够进入。新增/编辑功能按照要求填写相应字段即可进行资讯的上传与保存。上传接口部署在服务器里共开发环境调用，文件资源的上传与调用都由服务器提供，资讯信息的图片与信息保存方式与车辆管理模块的设计方式类似。

1. 个人中心模块

个人中心的头像通过Element组件库的头像上传组件以及服务器端的文件上传接口完成上传，上传时直接存放访问路径到数据库中用户表的头像字段。个人其他信息的修改，通过表单信息提交到后端更新用户信息接口即可完成修改。

3.4数据库设计

关于汽车资讯网站数据库关联见图3.5。

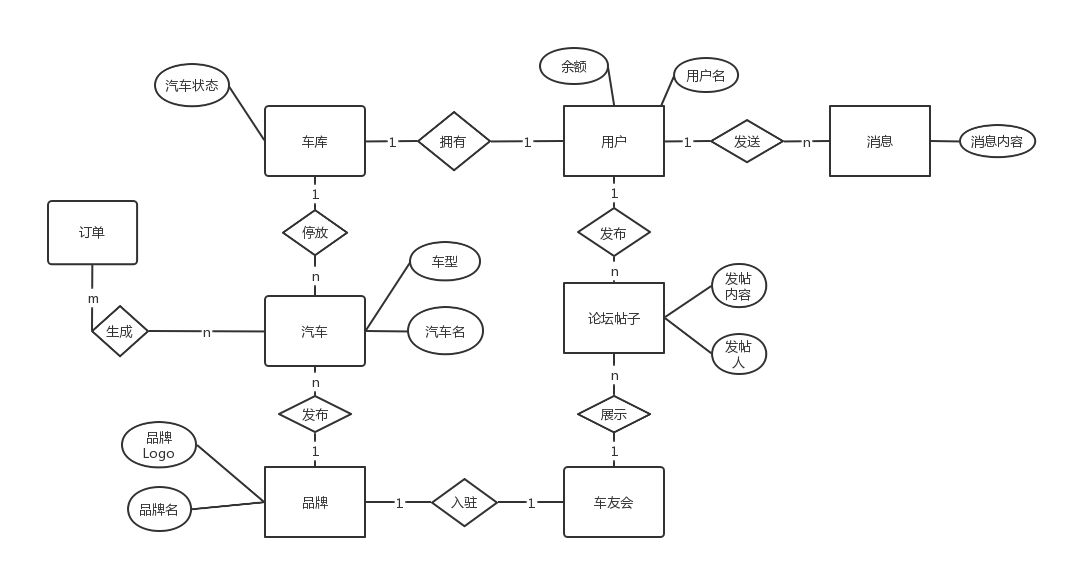


图3.5 数据库关系ER图

图3.1的ER图表示汽车资讯网站数据库各表之间的关联。车库表用于存放已经被购买的车辆，其中包含车辆id、购买人id、当前汽车状态等信息。车库表通过汽车id与用户id分别于汽车库和用户表关联，可以获知买车的具体用户及具体车辆。品牌表记录品牌信息，负责发布汽车。汽车表有品牌id以查询具体品牌。品牌表与车友会表一一对应，通过品牌id相关联。论坛帖子表用于记录已发布的论坛贴，通过车友会id和用户id字段可以分别获知发布帖子的车友会平台以及发帖人信息。用户发布消息会记录到消息表，每条记录会包含发送者/接受者的id、消息内容、发送时间等信息。

（1） 车库表

车库表信息：车库表id，汽车表id，用户表id，车牌号，里程数，订单id，售出状态（在库/在售/已售），定价。

车库表结构如表3.1所示。

表3.1 车库表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| barn | id |  | int(11) | PRI |
| barn | carid | 汽车id | int(11) |  |
| barn | userid | 用户id | int(11) |  |
| barn | car\_number | 车牌号 | varchar(11) |  |
| barn | km | 里程数 | int(11) |  |
| barn | orderid | 订单id | int(11) |  |
| barn | sellstate | 汽车状态 | varchar(255) |  |
| barn | pricing | 价格 | int(11) |  |

（2） 品牌表

品牌表信息：品牌表id，品牌名，品牌Logo图片链接，品牌首字母。

品牌表结构如表3.2所示。

表3.2 品牌表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| brand | id |  | int(11) | PRI |
| brand | brand\_name | 品牌名称 | varchar(255) |  |
| brand | brand\_logo | 品牌logo | varchar(255) |  |
| brand | brand\_intial | 品牌首字母 | varchar(255) |  |

（3） 汽车表

汽车表信息：汽车表id，汽车名字，车型（轿车/SUV/MPV/跑车），品牌表id，品牌名，车辆简介，车辆缩略图链接，车辆展示图列表地址，价格下限，价格上限，产地，生产日期，是否为新车（0为false,1为true），公里数，发动机型号，变速箱型号，汽车评分，参与汽车评分人数。

汽车表结构如表3.3所示。

表3.3 汽车表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| car | id |  | int(11) | PRI |
| car | car\_name | 汽车名字 | varchar(255) |  |
| car | car\_type | 车型 | varchar(255) |  |
| car | car\_brandid | 品牌号 | varchar(255) |  |
| car | car\_brand | 品牌名 | varchar(255) |  |
| car | car\_introduce | 车辆简介 | text |  |
| car | car\_showimg | 车首展图地址 | varchar(255) |  |
| car | car\_exhibition\_list | 车展览图列表地址 | text |  |
| car | car\_price\_low | 起始价格 | int(11) |  |
| car | car\_price\_high | 顶配价格 | int(11) |  |
| car | car\_manufacture\_addr | 产地 | varchar(255) |  |
| car | car\_manufacture\_date | 生产日期 | varchar(255) |  |
| car | isnew | 是否是新车 | tinyint(255) |  |
| car | kmruned | 已跑公里数 | int(255) |  |
| car | car\_engine | 发动机型号 | varchar(255) |  |
| car | car\_at | 变速箱型号 | varchar(255) |  |
| car | car\_score | 汽车评分 | double(255,0) |  |
| car | car\_score\_nums | 参与汽车评分的人数 | int(255) |  |

（4） 汽车资讯表

汽车资讯信息：汽车资讯id，资讯名称，资讯缩略图链接，资讯类型（文字资讯/视频资讯），视频资讯链接，文字资讯内容，发布日期，播放次数。

汽车资讯表结构如表3.4所示。

表3.4 汽车资讯表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| car\_news | id |  | int(11) | PRI |
| car\_news | news\_title | 资讯名称 | varchar(255) |  |
| car\_news | news\_headimg\_url | 资讯缩略图url | varchar(255) |  |
| car\_news | news\_content\_type | 资讯类型 | varchar(255) |  |
| car\_news | news\_video\_url | 视频资讯url | varchar(255) |  |
| car\_news | news\_text | 文字资讯内容 | text |  |
| car\_news | news\_time | 发布日期 | varchar(255) |  |
| car\_news | playtimes | 播放次数 | int(11) |  |

（5） 汽车订单表

汽车订单表信息：汽车订单表id，汽车表id，用户表id，购买时间，订单类型（0为新车，1为二手车），卖家id（新车为品牌id，二手车为上任车主id），交易额。

汽车订单表结构如表3.5所示。

表3.5 汽车订单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| car\_order | id |  | int(11) | PRI |
| car\_order | carid | 汽车id | int(11) |  |
| car\_order | userid | 用户id | int(11) |  |
| car\_order | order\_time | 购买时间 | varchar(255) |  |
| car\_order | order\_type | 订单类型 | int(255) |  |
| car\_order | sell\_id | 卖家id | int(11) |  |
| car\_order | sellprice | 交易额 | int(10) |  |

（6） 聊天记录表

聊天记录表信息：聊天记录表id，发送方用户id，接收方用户Id,接收方是否已读消息（0为false，1位true），消息内容。

聊天记录表结构如表3.6所示。

表3.6 聊天记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| chatlist | id | 聊天表id | int(11) | PRI |
| chatlist | from\_userid | 发送信息的用户id | int(11) |  |
| chatlist | to\_userid | 接收信息的用户id | int(11) |  |
| chatlist | readed | to方是否已读信息 | int(11) |  |
| chatlist | content | 评论内容 | text |  |

（7） 车友会分类表

车友会分类表：车友会表id，所属品牌id。

车友会分类表结构如表3.7所示。

表3.7 车友会分类表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| club | id | 车友会id | int(11) | PRI |
| club | brandid | 所属品牌id | int(11) |  |

（8） 评论表

评论表信息：评论表Id,评论类型（车友会帖子评论/资讯评论），评论人id，评论内容，评论时间。

评论表结构如表3.8所示。

表3.8 评论表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| comment | id |  | int(11) | PRI |
| comment | comment\_type | 评论类型 | varchar(255) |  |
| comment | from\_id | 评论来源 | int(11) |  |
| comment | userid | 评论人id | int(11) |  |
| comment | comment\_content | 评论内容 | text |  |
| comment | comment\_time | 评论时间 | varchar(255) |  |

（9） 帖子表

帖子表信息：帖子id，所属车友会id，发帖人，帖子简介，帖子详细内容，发帖时间。

帖子表结构如表3.9所示。

表3.9 帖子表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| invitation | id | 帖子id | int(11) | PRI |
| invitation | clubid | 所属车友会id | int(11) |  |
| invitation | userid | 发帖人 | int(11) |  |
| invitation | brief\_context | 帖子简介 | varchar(255) |  |
| invitation | detail\_context | 帖子详细内容 | text |  |
| invitation | time | 发帖时间 | varchar(255) |  |

（10） 用户信息表

用户表信息：用户表id，账号，密码，账户名，头像，余额，Socket连接号，用户身份，账户所属公司id，是否在线。

用户信息表结构如表3.10所示。

表3.10 用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段 | 名称 | 数据类型 | 主键 |
| user | id |  | int(11) | PRI |
| user | account | 账号 | varchar(255) |  |
| user | password | 密码 | varchar(255) |  |
| user | name | 账户名 | varchar(255) |  |
| user | headimg | 头像 | varchar(255) |  |
| user | balance | 余额，单位w | int(255) |  |
| user | Socketid | Socketid | varchar(255) |  |
| user | usertype | 用户身份:normal/store/admin | varchar(255) |  |
| user | companyid | 账户所属公司的id | int(11) |  |
| user | isonline | 是否在线,1为true,0为false | tinyint(1) |  |

3.5 用户权限设计

* 用户刚进网页时处于未登录状态，只能访问选车，资讯，车友会页面，但无法进行任何提交，购买操作。一旦进行上述操作会被拦截并要求完成登录。
* 用户注册后初始身份即为普通用户，普通用户可以访问选车，资讯，车友会，车库等界面，能够进行发帖，评论，购买汽车，管理车库等操作。普通用户在个人信息中心可以申请成为商家用户。
* 商家用户无法进入选车，车库等界面，但允许访问车辆管理系统，能够对车辆进行新增上传，以及现有车辆的编辑。
* 超级管理员可以进入任意界面，较其他权限可以访问资讯管理界面，能够对资讯进行上传发布编辑。

4 汽车资讯网站的开发与实现

4.1 页面请求响应

（1） 登录验证

因为发布评论，交易等大量功能需要用户处于在线状态，因此对于用户的请求设置了请求拦截以验证用户登录状态，主要是通过验证session中存储的myid键。只有当用户信息已经存在时，请求才会真正向后端发出。

登录功能代码如下：

request.interceptors.request.use(config => {

    if (!sessionStorage.getItem('myid')) {

        return Promise.reject({

            code: 20001,

            message: '请先完成登录'

        })

    } else {

        return config

    }

}, err => {

    return Promise.reject(err)

})

（2） 错误拦截

无论是成功还是失败的请求，都在响应对象里定义了三个属性表示相关信息：code表示自定义响应码，message表示响应信息，data表示返回的数据。

当后台校验数据需要报告验证失败或者请求失败时，需要返回自定义错误信息。定义code:20000为正确信息，code:20001为错误信息。当响应拦截器获取到code:20001的信息时，就会返回promise.reject对象，告诉前台失败信息。

错误拦截实现代码如下：

request.interceptors.response.use(response => {

    if (response.data.code != 20000) {

        return Promise.reject(response.data)

    } else {

        return response //拦截处理响应结果，直接返回需要的数据

    }

}, err => {

    console.log(err)

    return Promise.reject(err)

})

4.2 代理/打包设置

设置反向代理是为了避免开发时的跨域问题，以及相关WebSocket通讯的权限问题。可以根据请求baseurl替换成自己希望的目标请求地址。

打包代码如下：

'/loc': {

        pathRewrite: {

          '^/loc': ''

        },

        target: "http://localhost:3201",

        ws: true, // 是否开启 websokets

        secure: false, // 是否安全，https 为 true，http 为 false

        changeOrigin: true,

      },

打包设置是为了将前端项目生成浏览器可是识别的静态资源文件。并且可以使用Plugin的插件对打包进行优化，将生成的静态资源文件进一步压缩，减小打包文件的大小从而提高页面的加载速度。

打包核心代码如下：

configureWebpack: {

    Plugins: [

      new CompressionPlugin({

        test: /\.(js|Css)?$/i, // 哪些文件要压缩

        filename: '[path].gz[query]',　// 压缩后的文件名

        algorithm: 'gzip',　// 使用gzip压缩

        minRatio: 1,　// 压缩率小于1才会压缩

        deleteOriginalAssets: false // 删除未压缩的文件，谨慎设置，如果希望提供非gzip的资源，可不设置或者设置为false

      })

    ]

  }

4.3 动态路由生成

本系统将客户端，管理端，商家端进行整合，涉及权限问题，因此都需要根据不同身份权限来对路由进行处理，让用户无法跨权限操作访问。每个用户在数据库中都有一个身份字段，路由文件中使用路由元数据中的limits数组存储能够访问该网页的身份。

每次当网页载入时，会根据登录用户身份与limits中的身份限制过滤出用户可以访问的路由列表并保存到Vuex数据仓库中。

路由配置核心代码：

Routerjs:

       meta: {

          title: '首页',

          sidebar: true,

          require\_login: false,

          icon: 'el-icon-menu',

          limits: ['admin', 'store', 'normal']

        }

过滤路由函数会在页面初始化和登录时调用，生成可访问的路由列表。

过滤路由列表的函数实现如下:

export function filtrouters() {

  let myroutes = routes.options.routes[0].children

  if (store.getters.getme) {

    let usertype = store.getters.getme.usertype

    myroutes = myroutes.filter(function (item) {

      return item.meta.sidebar && item.meta.limits.includes(usertype)

    })

  } else {

    myroutes = myroutes.filter(function (item) {

      return item.meta.sidebar && !item.meta.require\_login

    })

  }

  store.commit('setmyroutes', myroutes)

  console.log(store.getters.getmyroutes);

}

4.4 后端结构

（1） 项目入口

项目后端采用Express框架，入口文件为app.js，在app.js中主要是对全局资源的引入，包括全局用到的中间件，请求入口。App.js中需要对请求路由进行注册分配，将不同的请求根据路径分发到不同的路由层。另外还需要引入post请求的处理中间件，因为Express默认没有对post请求参数的接收方法，因此需要引入额外引入。

（2） 路由层

路由层负责接收分配的请求，并将请求传递给控制层。

（3） 控制层

控制层是真正处理请求的地方，路由层分配的请求都会对应控制层的一个方法。参数req，res分别是请求体，响应体，可以获得参数，进行操作，然后返回前端所需要的数据。

（4） Socket服务端

引入全局Socket.IO-server工具，在前端初次连接服务端时，服务端的Socket对象会为前端分配Socketid来标识用户。前后端的Socket连接主要是通过Socket.on和Socket.emit两个方法来进行侦听和触发，从而完成双向通信。

（5）其他工具

主要包括数据库连接和sql语句工具等，在控制层可以快速调用这些已经封装好的js文件中的方法。

4.5 登录注册模块

（1） 注册

注册功能需要用户提交用户名密码等信息，前端进行验证后提交至后端。后端会在查重之后进行数据库的信息提交。也并非所有信息都需要用户提交，比如头像，在初次注册时会设置默认头像。

查重实现：

async cnki(table, columnName, columnval) {

        var str = `select \* from ${table} where ${columnName} = '${columnval}'`

        var res = await sqlQuery(str)

        return res.length > 0 ? true : false

    }

  注册功能实现效果如图4.1。



图 4.1 注册页面图

（2） 登录

输入账户名密码，后端进行验证。验证完成之后即可返回用户信息。前端会将用户id保存至session中，做持久化存储。将其余的用户信息保存如Vuex的store中，方便随时调用。同时触发Socket事件，更新数据库中该用户被分配的Socketid，并设置登录状态，每次关闭页面或者主动注销时就会清除Vuex中的信息和session中的userid。在session中保存userid的作用就在于，当页面刷新，就会根据sessionid重新给后端发出请求，重新获得用户数据。

用户信息保存实现：

sessionStorage.setItem( "myid", JSON.stringify(res.data.data.userid) );

                Socket.emit("setSocket", sessionStorage.getItem("myid"));

                getUser(sessionStorage.getItem("myid")).then((res) => {

                  this.$store.commit("setme", res.data.data);

                }).then(() => {

                  filtrouters()

                });

更新数据库信息:

Socket.on('setSocket', async function (userid) {

           let sqlStr = "update user set Socketid=?,isonline=? where id=?"

            let result = await sqlQuery(sqlStr, [Socket.id, 1, userid])

        })

登录页面实现效果如图4.2所示。



图 4.2 登录页面图

4.6新车与二手车选购模块

（1）汽车筛选

在选购页面，默认按照价格降序展示车辆。但根据多条件选择器可以选择出用户需要的车辆。多条件选择器首先可以通过首字母选择品牌，之后通过车型、价格、名字来进行选择。我通过一个数组来保存用户设置的条件。这些条件都有默认值，每一次更改条件都会将这个条件数组完整的传回后端，后端根据条件进行拼接sql语句完成查询。

后端筛选实现：

var str = `select \* from car where 1=1`

            var term = ''

            if (req.query.myscreen[0] != -1) {

                term += ` aznd car\_brandid = '${req.query.myscreen[0]}'`

            }

            if (req.query.myscreen[1] != -1) {

                term += ` and car\_price\_low >= ${req.query.myscreen[1]} and car\_price\_high <= ${req.query.myscreen[2]}`

            }

            if (req.query.myscreen[3] != -1) {

                term += ` and car\_type = '${req.query.myscreen[3]}'`

            }

            if (req.query.searchstr != '') {

                term += ` and car\_name like '%${req.query.searchstr}%'`

            }

            term += ` order by id desc limit ${req.query.pagenum},${req.query.reqnum}`

            var sqlres = await sqlQuery(str + term)

汽车筛选实现效果如图4.3所示。

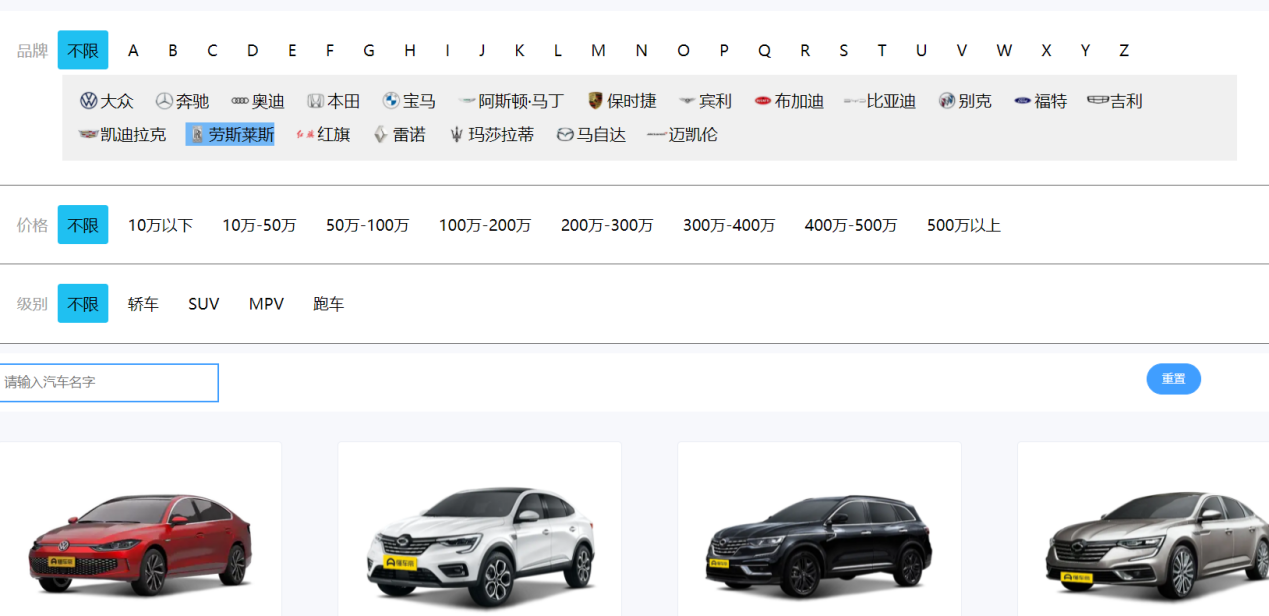


图 4.3 汽车选购界面图

（2）交易

交易模块需要用户提交购车表单，其中包含用户支付密码以及选购的车辆信息。信息传回后端会进行支付密码以及余额的验证。全都验证通过后完成整个交易流程。

交易流程后台实现：

let carsql = await sqlQuery(`select \* from car where id=${req.body.carid}`)

            let car = carsql[0]

            let pwdcheck = await modusers.pwdcheck(req.body.userid, req.body.paypwd)

            if (!pwdcheck.check) {

                return res.json({

                    code: 20001,

                    message: "密码验证失败",

                })

            }

let balancecheck = await modusers.balancecheck(car.car\_price\_low, req.body.userid)

            if (!balancecheck.check) {

                return res.json({

                    code: 20001,

                    message: "账户余额不足",

                })

            }

            modusers.deduct(car.car\_price\_low, req.body.userid)

      var ordersql = await sqlQuery(`insert into car\_order (carid,userid,order\_time,order\_type,sell\_id)

            values (${car.id},${req.body.userid},'${new Date()}',0,${car.car\_brandid})`)

            var car\_number = "川E 456E06"

            var barnsql = await sqlQuery(`insert into barn (carid,userid,car\_number,km,orderid)values (${car.id},${req.body.userid},'${car\_number}',0,${ordersql.insertId})`)

4.7 车库模块

提供在库的车辆提供出售服务，且汽车有在库、挂售两种状态。用户若要卖出车辆，需要设置定价以及输入自己的支付密码，完成验证后，即可将车辆挂售。挂售的车辆会出现在二手车市场。后台对挂售的主要操作是，更改车辆状态。在其他用户购买此车完成支付后，当完成交易，会进行金额的转移和车辆所属人的更新。

购买操作实现：

let carbarns = await sqlQuery(`select \* from barn where id=${req.body.barnid}`)

let carofbarn = carbarns[0]

let pwdcheck = await modusers.pwdcheck(carofbarn.userid, req.body.paypassword)

if (!pwdcheck.check) {

                return res.json({

                    code: 20001,

                    message: "密码验证失败",

                })

}

let updatesql = await sqlQuery(`update barn set sellstate='在售',pricing=${req.body.pricing}

             where id=${req.body.barnid}`)

            return res.json({

                code: 20000,

                message: "汽车挂售成功",

            })

车库模块实现效果如图4.4所示。

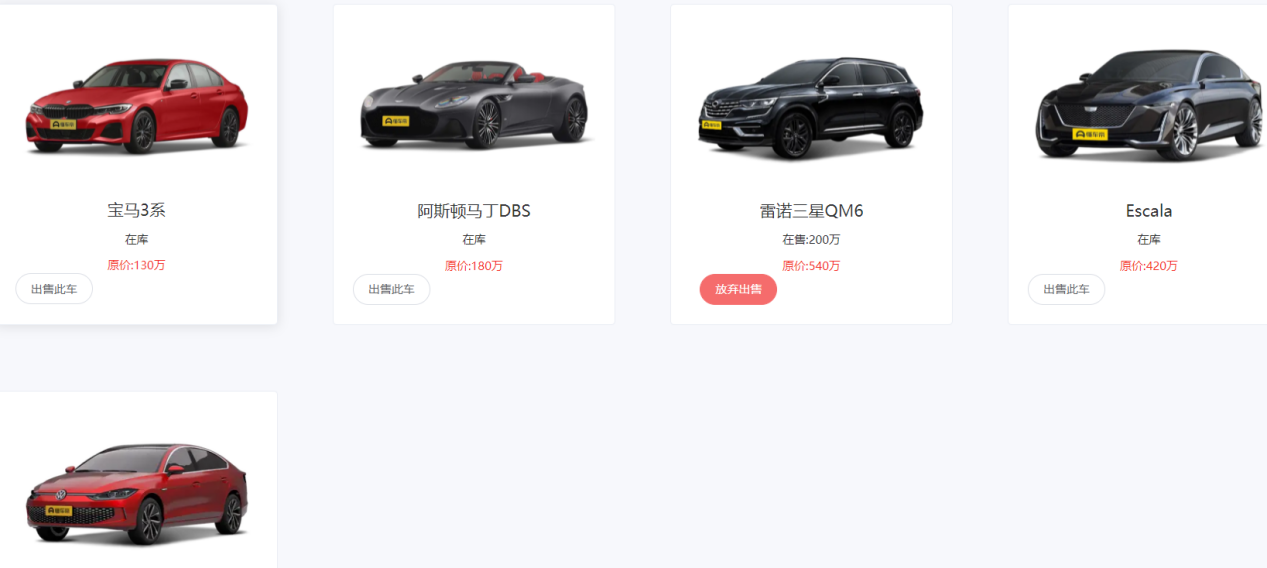


图 4.4 车库界面图

4.8 车友会模块

车友会发帖功能与评论功能的实现方式类似，帖子功能使用了富文本，意味着用户可以在输入框中贴图。帖子与评论的提交会顺带提交用户信息，后台就会进行插入数据的操作，记录发布者，发布时间以及内容。

发帖功能代码实现：

var sqlres = await sqlQuery(`insert into invitation (clubid,userid,brief\_context,detail\_context,time) values(${req.body.club\_id},${req.body.userid},'${req.body.brief\_context}','${req.body.detail\_context}','${new Date()}')`)

            return res.json({

                code: 20000,

                message: "发布成功",

                data: req.body

            })

4.9 资讯管理

资讯管理主要是由超级管理员负责。有新增、编辑、删除三个功能。引用了Element的表单组件进行快捷展示。新增或者编辑时对每个字段都有相应的规则进行文本校验，只有当校验列表都符合要求，才能进行表单提交。

资讯新增代码实现：

var str = `insert into car\_news (news\_title,news\_headimg\_url,news\_content\_type,news\_video\_url,news\_text,news\_time) value (?,?,?,?,?,?)`

            let sqldata = [

                req.body.news\_title,

                req.body.news\_headimg\_url,

                req.body.news\_content\_type,

                req.body.news\_video\_url,

                req.body.news\_text,

                modusers.getCurrentTime()

            ]

              var sqlres = await sqlQuery(str, sqldata)

4.10 汽车信息管理

汽车的信息管理主要是由商户进行负责，发布人在新增/编辑车辆时同样需要提供车名、展示图等相关信息，发布时不但会将表单内容提交至后台，同时会将发布人所属的公司id一起提交，按照公司id来决定车辆所属品牌，因此汽车管理模块只有商户能够进入。

汽车编辑代码实现：

let carstr = `update car set car\_name=?,car\_at=?,car\_engine=?,car\_introduce=?,car\_manufacture\_addr=?,car\_price\_high=?,car\_price\_low=?,car\_showimg=?,car\_type=? where id=?`

let data = [ req.body.car\_name, req.body.car\_at,req.body.car\_engine, req.body.car\_introduce,req.body.car\_manufacture\_addr,req.body.car\_price\_high,req.body.car\_price\_low,req.body.car\_showimg,req.body.car\_type,req.body.id]

 let carres = await sqlQuery(carstr, data)

车辆信息修改管理实现效果图4.5所示。



图 4.5 车辆修改信息界面图

4.11 个人中心模块

（1）头像上传

头像上传主要涉及的知识点是图片的上传。前端的上传主要是使用的Element的图片上传组件。在获取到文件之后，将需要保存的地址一起发送给后端。后端有文件上传接口，在获取到文件之后根据请求参数来保存图片。在后台完成文件的保存之后会告诉前端文件的真实保存路径，前端再次发出请求完成用户信息修改，也就是将用户的头像地址进行更新。

文件上传接口实现：

async uploadfile(req, res, next) {

        try {

            var des\_file = `/www/server/tomcat/webapps/ROOT/guloti/GulotiResource/upload/${req.body.path}`

            // res.json(req.body)

            // var des\_file = "./upload/" + req.files[0].originalname;

            fs.readFile(req.files[0].path, function (err, data) {

                fs.writeFile(des\_file, data, function (err) {

                    if (err) {

                        fs.unlinkSync(req.files[0].path);

                    } else {

                        var response = {

                            message: 'File uploaded successfully',

                            url: `http://47.108.230.246:8080/guloti/GulotiResource/upload/${req.body.path}`

                        };

                        fs.unlinkSync(req.files[0].path);

                        res.json(response)

                    }

                });

            });

        }

    }

（2）注销

注销功能的流程与关闭页面的轮流一样，根据session中的用户id清除用户数据库中的Socketid，清除Vuex里的用户信息，最后完成退出页面完成注销。

注销代码实现：

logout() {

      //清除会话

      sessionStorage.removeItem("myid");

      this.$store.commit("setme", null);

      Socket.emit("logout");

      filtrouters()

      this.$message({

        message: "注销成功",

        type: "success",

      });

      this.$router.push("/");

    },

结 论

本次毕业设计我完成的是一个综合性比较强的汽车网站开发，以Vue+Express为基础技术完成开发，除此之外，还需搭建云服务器完成包括图片、视频、文件与数据库在内的各种资源的提供工作。

在开发过程中，我遇到了各种问题，但在反复调试和收集相关资料的情况下都能逐渐解决，提高了我开发过程中独立解决的能力。与此同时还学会了各种新工具的使用，例如接口调试工具Postman，服务器连接工具Finalshell等，收获不可谓不多。

本次毕业设计只是我开发道路上的一个起点，道阻且长。唯有在今后的开发工作中多多学习多加锻炼，才能让自己的技术专业水平更上一层楼。而对于他人的帮助与教诲我也会悉心聆听牢记于心。

参考文献

[1] [Vue.js在前端开发应用中的性能影响研究](https://kns.cnki.net/kcms/detail/detail.aspx?filename=DZZZ202010020&dbcode=CJFQ&dbname=CJFDTEMP&v=xq4phlLVx-kQPw7zszIvhiecKUbgl_CTRfY-RUFtHIDL17NScx6z9zgvY2KonTAC" \t "https://kns.cnki.net/kcms/detail/frame/kcmstarget)[J].唐斌斌,叶奕.电子制作.2020(10)

[2] [基于MVVM模式的Extjs框架在前端界面设计中的应用研究](https://kns.cnki.net/kcms/detail/detail.aspx?filename=DNZS201605037&dbcode=CJFQ&dbname=CJFD2016&v=osVPV8X7cZ4IYEKH18hRPoJJ1Xy-8Lx3T-1x1Az_NZaHF7hLNTqw1684cF7iQnil" \t "https://kns.cnki.net/kcms/detail/frame/kcmstarget)[J].肖小岚,刘振宇.电脑知识与技术.2016(05)

[3] [2015年我国汽车配件行业发展趋势分析](https://kns.cnki.net/kcms/detail/detail.aspx?filename=QCLJ201510021&dbcode=CJFD&dbname=CJFD2015&v=QMmrIho0yrRZvg-5Sjaqx41TaTGdmwxgxn9Zq60p0Ca0qBZcalLvMOOLlt4nX0fC" \t "https://kns.cnki.net/kcms/detail/kcmstarget)[J].汽车零部件.2015(10)

[4] [基于Web前端开发技术特点及优化分析](https://kns.cnki.net/kcms/detail/detail.aspx?filename=TXWL202118017&dbcode=CJFQ&dbname=CJFDTEMP&v=TVNvMD85LNfseoU_7_hnchxe-hSKK3VUOr4cT_fgOvvNvc3DGnEa6_zUUE_P-TPe" \t "https://kns.cnki.net/kcms/detail/frame/kcmstarget)[J].孙鹏飞,崔兰超.中国新通信.2021(18)

[5] 马超群, 赵海龙. 汽车市场需求预测建模及其应用研究[J]. 湖南大学学报：社会科学版, 2009, 23(4):7.

[6] 陈鲱. Web前端开发技术以及优化方向分析[J]. 新媒体研究, 2015, 1(3):2.

[7] 吴睿. Web前端开发技术以及优化研究[J]. 中国新通信, 2016(17):1.

[8] 聂常红. Web前端开发技术:Html、Css、JavaScript[M]. 人民邮电出版社, 2013.

致 谢

在此非常感谢在王燚老师在毕业设计开发以及毕业论文修改阶段给予我的帮助。每逢遇到无法解决的难题，王燚老师都能够及时指出我的问题所在并悉心指导助我解决，在此表达衷心感谢！

同时感谢参与论文评审的各位老师与专家们！

作者简介：

姓 名：向峰 性别：男

出生年月：1999年12月 民族：汉族

E-mail:863983969@qq.com

声 明

本论文的工作是2022年4月至2022年5月在成都信息工程大学网络空间安全学院完成的。文中除了特别加以标注地方外，不包含他人已经发表或撰写过的研究成果，也不包含为获得成都信息工程大学或其他教学机构的学位或证书而使用过的材料。

关于学位论文使用权和研究成果知识产权的说明：

本人完全了解成都信息工程大学有关保管使用学位论文的规定，其中包括：

（1）学校有权保管并向有关部门递交学位论文的原件与复印件。

（2）学校可以采用影印、缩印或其他复制方式保存学位论文。

（3）学校可以学术交流为目的复制、赠送和交换学位论文。

（4）学校可允许学位论文被查阅或借阅。

（5）学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

除非另有科研合同和其他法律文书的制约，本论文的科研成果属于成都信息工程大学。

特此声明！

作者签名：

年 月 日