

本科毕业设计

论文题目：基于vue和node的移动端外卖平台

前端设计与实现

学生姓名：文超

学号：201324040231

专业：信息管理与信息系统

指导教师：王鑫

学 院：管理科学与工程学院

2017 年 5 月 22 日

毕业论文（设计）内容介绍

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 论文（设计）  题 目 | 基于vue与node的移动端外卖平台前端设计与实现 | | | | | |
| 选题时间 | 2016年12月 | | 完成时间 | 2017年5月 | 论文（设计）  字数 | 约11000字 |
| 关 键 词 | WebAPP 前端开发 Vue Node 外卖系统 | | | | | |
| 论文（设计）题目的来源、理论和实践意义：  来源： 老师指导下学生自拟。  理论和实践意义：  在大学生普遍创业的大环境下，如何快速、高效的完成相关互联网产品（介绍网站等）的制作，已经成为困扰广大创业者的关键问题。其核心就是软件开发的难易程度。传统开发移动app需要同时兼顾Android、IOS双系统，难度较大，而采用WebAPP的方式则可以大幅度减少开发难度，同时兼顾兼容性。本文通过对一个移动端外卖平台系统的研究设计，就WebAPP的全新应用方式进行了探讨和分析，并提出了一些注意问题和建议。 | | | | | | |
| 论文（设计）的主要内容及创新点：  本文的系统，WebAPP采用HTML5，CSS3，javascript编程语言进行前端开发，并采用node.js进行后台开发，mongoDB进行数据库开发，相比于传统的PHP+SQL后台方式，node+mongoDB采用javascript语言，这对熟悉前端开发的开发者来说极为友好，并且在性能上也具有一定的优势。而在前端上，采用了MVVM的vue框架，相比于传统的网站开发，极大地简化了网站的开发步骤，并且大大增加了易用性、易扩展性，使得大学生创业者们可以在更加专注业务实现的基础上，更快的搭建现代化的移动APP。 | | | | | | |
| 附：论文（设计） | | 本人签名： 年 月 日 | | | | |

目 录

[中文摘要 1](#_Toc483420652)

[英文摘要 2](#_Toc483420653)

[1、绪论 3](#_Toc483420654)

[1.1 课题背景 3](#_Toc483420655)

[1.2 外卖平台的特点以及网站技术的发展 3](#_Toc483420656)

[2、相关技术研究 5](#_Toc483420657)

[2.1 node.js简介 5](#_Toc483420658)

[2.2 非关系型数据库 5](#_Toc483420659)

[2.3 前端框架vue.js简介 6](#_Toc483420660)

[3、外卖平台系统的需求分析 8](#_Toc483420661)

[3.1 系统的功能需求 8](#_Toc483420662)

[3.2 具体需求页面介绍 9](#_Toc483420663)

[3.3 前端上的需求 12](#_Toc483420664)

[4、外卖平台系统的设计 13](#_Toc483420665)

[4.1 系统的设计原则 13](#_Toc483420666)

[4.2 系统的架构设计 13](#_Toc483420667)

[4.3 系统用例介绍 15](#_Toc483420668)

[5、外卖平台系统的实现 16](#_Toc483420669)

[5.1 MonogDB数据集合 16](#_Toc483420670)

[5.2 node控制层部分核心模块的实现 19](#_Toc483420671)

[5.3 vue框架 前端实现 28](#_Toc483420672)

[5.4 外卖平台系统的测试 34](#_Toc483420673)

[6、总结与展望 36](#_Toc483420674)

[6.1 论文总结 36](#_Toc483420675)

[6.2 论文实现的意义 36](#_Toc483420676)

[6.3 存在的问题以及未来的展望 37](#_Toc483420677)

[参考文献 38](#_Toc483420678)

[附录 39](#_Toc483420679)

基于vue与node的移动端外卖平台前端设计与实现

摘 要

随着互联网的快速发展，移动互联网的快速成长，以移动应用为主的新一代应用程序成为越来越多的人的首要选择。其中包括以Android以及IOS为主的两大阵营，这两个移动操作系统的应用程序需要通过不同的编程语言来编写，其难度较大。而本文的主要工作就是以vue前端框架、node.js作为后台语言、mongoDB作为数据库为例，实现一个基于HTML5的移动端外卖平台系统。这将为广大大学生、创业者提供了一个新的解决问题的思路。基于vue、node.js、mongoDB的移动端外卖平台，提供了外卖平台常见的如登录注册、购买评论等功能，同时该系统具有轻量化、易维护、易扩展的功能。

**关键词：**WebAPP 前端开发 Vue Node 外卖系统

Design and Implementation of Mobile Terminal Takeout Platform Based on Vue and Node.js

ABSTRACT

With the rapid development of the Internet, the rapid growth of mobile Internet, mobile applications based on a new generation of applications become more and more people's first choice. Including the Android and IOS, the two mobile operating system applications need to write through a different programming language, it’s more difficult. The main work of this paper is to Vue Front-End framework, Node.js as the background language, mongoDB as a database, for example, to achieve a HTML5 mobile take-off platform system. This will provide the majority of college students, entrepreneurs, a new way to solve the problem. Based on Vue, Node.js, mongoDB mobile terminal take-off platform, provides a take-away platform common such as login registration, purchase reviews and other functions, while the system has a lightweight, easy to maintain, easy to expand the function.

**KEY WORDS：**web app, front-end development, vue, node, take away platform

# 1、绪论

## 1.1 课题背景

在五六年前移动互联网以及移动设备不发达的时候，人们就餐无非是自己做、外出就餐两种。对于大学生来说，只能选择食堂就餐。但是近几年随着智能手机的性能越来越强大、移动互联网的急速发展，人们生活的方式也随之发生变化。为了节约时间并且吃到多种多样的食物，外卖系统开始在移动互联网上大规模爆发。

外卖平台在移动端实现的途径通常分为移动APP以及WEBAPP两种，作为校园外卖平台，移动APP需要开发至少Android以及IOS两种，并且后续维护升级需要人力较多，因此采用WEBAPP进行H5开发，从成本、维护、升级上进行考虑是十分合适的。

## 1.2 外卖平台的特点以及网站技术的发展

### 1.2.1 外卖平台的特点

作为一个可以快速应用的外卖平台，可以从前端（页面效果）以及后端（业务逻辑的实现）来介绍其特点。首先表现层应具有以下特点：

（1）店铺页面，包括商品列表、商品详情、分类、评价、店铺信息等页面效果实现

（2）商家、用户各自的管理页面

（3）提升用户体验，在购物时提供动画效果

业务逻辑层应具有以下特点：

（1）用户商家登录注册的实现

（2）商家、用户数据库的形成，可以进行买家购买、评论，卖家接单发货，双方修改信息等交互

### 1.2.2移动端开发的现状

智能手机已快速发展，许多应用都会在移动端有相应的实现，本项目也是其中之一。目前市面上主流的两大操作系统是IOS和Android。然而面对这两个操作系统，移动端的开发一直处于较为复杂的状态，不同操作系统具有较高的学习成本。

IOS应用开发必须使用Object-c语言（或者新推出的swift语言），而且要对于OC有深入的理解才能写出优秀的移动应用。这其中涉及了OC的语言核心以及苹果为其准备的庞大API，这对于大学在校生是一个很大的他挑战。

Android应用开发需要熟练掌握java语言，同时要学习Android API应用开发手册与优秀的Android程序。

这两个平台开发者仍需要掌握不同的开发环境，IOS的开发环境只能在macOS下的Xcode[1]，Android开发则需要借助Android studio，其中涉及复杂的开发流程以及测试、查错功能，都是需要较高的学习成本。

然而只是掌握语法核心显然是无法胜任多平台的开发工作。如果要进行多平台开发，需要开发者具有强大的专业知识，熟知两个平台的众多API。但是对于大学生创业或者校园项目的开展这是很不符合实际的，因此移动web app就应用而生，浏览器跨平台解决方案是最简单而高效的。

每个移动平台都具有游览器功能，利用html（5），css（3），javascript写出来的程序可以运行在浏览器内核中（移动端多为统一的webkit内核），从而达到跨平台的目的[2]。

### 1.2.3 传统网站开发的现状以及发展

网站开发的技术在发展过程中经历了几个比较重要的时间点。

（1）以往网站在页面更新的时候，需要通过设置url来跳转页面，获取、更新数据需要重新刷新、渲染页面。但是2004年谷歌推出的gmail采用了ajax技术（异步javascript与XML）[3]，使得网页获取数据、渲染页面不需要完全刷新页面，只需要通过获取的新的数据对页面进行局部渲染，使用户体验大大提升。例如以往注册账户，需要点击注册之后，才会进行数据库查询刷新页面通知账户是否可用，使用ajax技术之后在输入完用户名的同时便会异步发送请求查询并在页面中局部刷新页面。

（2）以往进行网站开发需要使用源生的html、javascript进行编程，需要通过js脚本手动控制DOM（html文档对象模型）来实现页面效果[4]，复杂的操作将大大分散开发者精力。框架的出现改变了这一情况，例如vue、react等框架可以虚拟化dom，将视图层与控制层分离，使开发者可以专注于业务的实现。

（3）node.js的诞生，以往的后台需要使用C#，java，php等语言，但是node.js的诞生标志着用于前台的脚本语言javascript同样可以进行后台编程，这为小型公司创业者提供了更低人力成本的方式。

# 2、相关技术研究

## 2.1 node.js简介

Node.js（下文简称为node）是一个可以快速构建网络服务与应用的平台，其是基于Chrome V8引擎的JavaScript运行环境[5]，而V8引擎具有非常快的性能以及非常好的性能。

Node的运行环境一般是独立于浏览器的，为了弥补JavaScript对二进制数据的支持不足，node新增了buffer类，从而进行高效的二进制数据处理。并且根据了V8引擎进行了性能优化，在底层核心（底层模块）node使用c++编写，具有将为强大的性能优势[6]。

性能方面，node的核心思想是单线程、事件驱动、非阻塞I/O。单线程需要配合事件循环机制来进行并行操作，同时意味着可以大幅度减少服务器开销[7]。Node采用的非阻塞I/O，其基础的理论依据是I/O的代价是高昂的，尤其是磁盘I/O与网络I/O，node创始人Ryan Dahl在JSConf大会表示磁盘与网络的I/O需要的CPU时间是访问内存的几万倍甚至几十万倍（如图2-1），然而当前网络应用需要频繁的访问磁盘与网络，可见提高I/O过程中的CPU利用率对网络应用是十分重要的[8]。

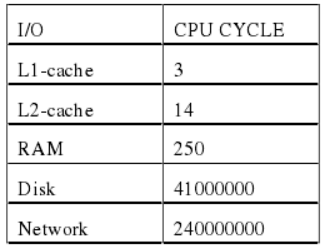


图2-1 开销

为了简化开发，node采用模块化开发的方式，遵循CMD（common module definition）规范，Module模块类似于C语言中的类库，其封装了如http，fs（文件系统），url等基础功能模块，同时通过npm（包管理器）可以轻易引入各种扩展包[9]。

Node的编程语言为javascript，javascript的匿名函数、闭包等特性特别适合事件驱动、异步编程风格。同时前后端的编程风格较为统一，利于个人轻量级开发[10]。

## 2.2 非关系型数据库

### 2.2.1 非关系型数据库NoSQL简介

非关系型数据库指的是no only sql，是随着web2.0应用而兴起的新型数据库。在web2.0的应用中，传统sql数据所擅长的复杂查询并不会应用特别多，同时对于数据库事物的一致性要求并不高[11]。

大数据量的web2.0系统都十分忌讳复杂的sql查询，在系统设计之初就应避免这样的查询出现（例如本项目只利用到了3个数据表），只是用单表的主键查询等功能便足以应对。

简而言之，关系型数据库的“重功能”并不适用于新型web应用，而高效、简洁的非关系型数据库则很容易胜任。

并且非关系型数据库可以处理超大量的数据，运行环境不需要太高要求，易扩展、大数据量、高性能、十分灵活的数据结构等特性都十分适用于web应用。

### 2.2.2 MongoDB——面向文档的分布式数据库

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引[12]。

## 2.3 前端框架vue.js简介

Vue是一套构建用户界面的渐进式框架，与其他重量级框架（angular，体积庞大、功能繁多）不同的是，vue采用自底向上增量开发的设计。Vue的核心库只关注视图层，它有着较为简单的上手难度，还便于与第三方库或者既有项目进行整合[13]，下面将介绍其重要特点。

### 2.3.1 轻量化

相比于Angular的160kb，70多kb的vue的优势不仅体现在体量上，专注于核心库可以加快其核心方法的速度。

### 2.3.2 数据双向绑定

作为提高开发效率最重要的一个特性，Vue、Angular、React等主流框架都实现了这个特性。传统网站开发时进行表单输入，需要手动书写js来进行视图层与数据层的数据同步，但是vue通过设置类似于ES5的访问器属性来监听表单文本的变化，同时监听数据层的变化，来达到实时更新的效果。DOM Listeners和Data Bindings是实现双向绑定的关键。DOM Listeners监听页面所有View层DOM元素的变化，当发生变化，Model层的数据随之变化；Data Bindings监听Model层的数据，当数据发生变化，View层的DOM元素随之变化。因此只需要在html元素中增加v-model属性便可以轻松实现数据双向绑定。

### 2.3.3 指令

指令是Vue最强大的功能之一，它作用在HTML元素上，将指令绑定在元素上时，会给绑定的元素添加一些特殊行为，可将指令视作特殊的HTML属性，例如：

（1）v-for指令：可以循环遍历数据，根据内容批量生成html结构，并且具有各自的作用域。传统开发方式是在js中，在遍历数据的循环内书写html字符串代码，根据数据进行替换，然后增加到DOM上。V-for指令大大简化了这一操作。

（2）v-show指令：可以根据指定数据的值来决定是否对某一html元素进行显示或隐藏。传统开发需要经过较为复杂的DOM操作。

Vue还支持自定义指令，大大扩展了html元素的功能，简化了开发，使开发者可以专注业务的实现而不是DOM的操作。

### 2.3.4 组件化开发

Vue建议使用vue-cli脚手架进行开发，每一个路由（单页面）或者每一个页面中的小部分都是一个组件，每一个组件都是一个.vue文件，其完整的包含了视图层（html，css）和数据层（数据，js交互）。

组件化意味着更加清晰地项目结构，组件间可以相互引用，分类公用组件将大幅度提高开发效率，同时组件间的数据是可以继承的，实现了组件之间的通信。

# 3、外卖平台系统的需求分析

## 3.1 系统的功能需求

外卖平台系统本身是一个具有较多功能的系统，本外卖平台系统主要是实现外卖平台一些常见的需求。

首先是功能上的需求：

### 3.1.1 登录、注册

买家、卖家都应具有的注册、登录功能。两者采用不同的数据库模型，在注册时应通过Ajax实时检测用户名是否占用。登录成功会局部刷新页面而不是重新刷新整个页面。

### 3.1.2 购买

用户可以对不同店铺的商品进行选购，在选购金额达到最低价后将会提交订单，同时计算上运费。

### 3.1.3 商家对于订单的操作

商家可以根据订单情况设置已接单、正在派送的状态（默认是未接单）。

### 3.1.4 买家收货以及评论

对于已经设置为正在派送的订单，买家可以设置状态为 已签收，并且可以评论，包括送达时间、评分、评价。

### 3.1.5 个人信息/店铺信息的修改

买家可以对自己的联系方式、收货地址进行修改。买家可以修改店铺信息。

### 3.1.6 卖家增加、修改商品信息

卖家可以更改商品的售价、简介、图片等，可以设置新的分类，并将原有商品重新分类。也可以上传新的商品。

具体页面展示如下部分：

## 3.2 具体需求页面介绍

图3-2-1是登录界面，可以选择不同的账户类型进行登录，注册的话是统一的模块，选择不同的用户类型便可以进行注册。



图3-2-1，登录注册界面

下图3-2-2是买家详情页，登陆成功后点击“查看个人中心”便可进入，所有订单展示为此用户所下订单，每个订单都可以展开，里面展示订单详情，如果订单状态为“正在派送”，则可以选择“收货”，并进行订单评价。评价将会插入店家的整体评价以及每个商品中的评价。用户注册时没有设置联系方式，需要在个人中心手动设置。收货地址也是如此，可以存储多个地址，选择一个默认收货地址。如果下单时没有设置联系方式或者没有收货地址，则不能进行下单操作。



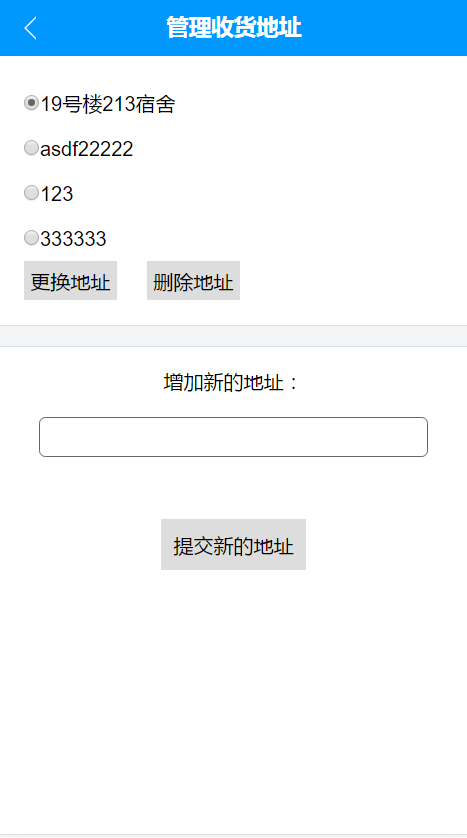
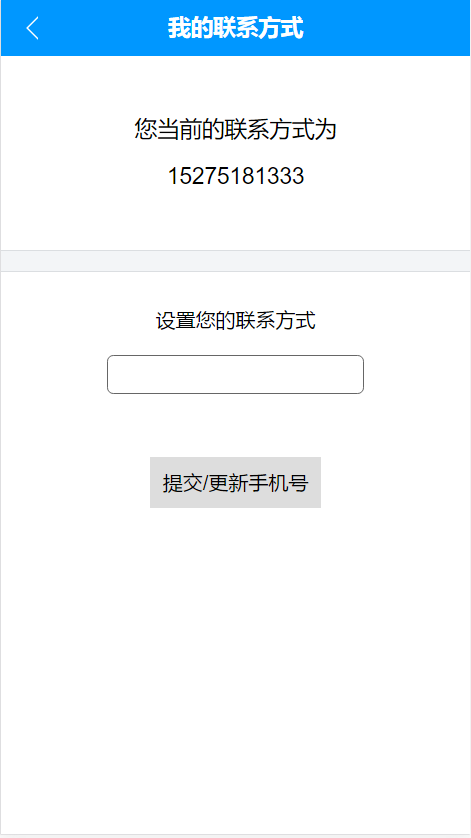


图3-2-2，买家个人中心界面

下图3-2-3为卖家详情页，其组件与买家详情页一样，通过判断用户类型（通过props继承）来决定显示部分。订单管理同买家，可以根据订单状态决定接单、派送，若此订单买家已评价则可以在页面中显示买家评价。管理商品页面，可以设置商品分类（增加、删除），点击商品后可以修改商品信息以及分类。点击新增商品按钮后可以新增商品。



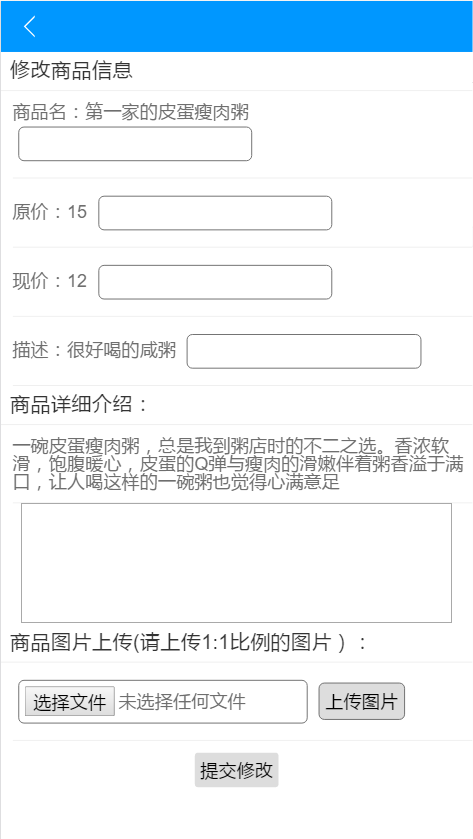
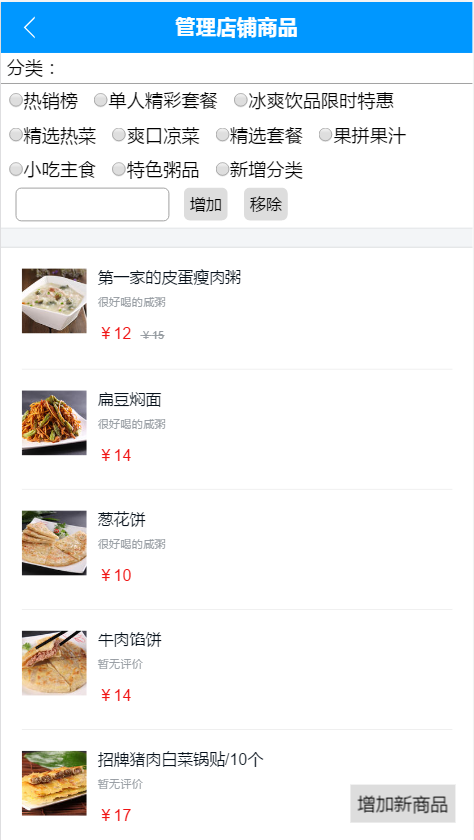


图3-2-3，卖家个人中心界面

下图3-2-4为商品页购买展示，点击购物车后购物车栏展开，可以统一调整商品数量，然后提交订单。



图3-2-4，购买商品

## 3.3 前端上的需求

### 3.3.1 无刷新的SPA（单页面应用）

随着前端技术的发展，单页面应用也逐渐流行起来，其具有类似于桌面应用的及时性，可以提供类似于app的体验。

本系统没有链接的直接跳转（强制刷新页面），链接的跳转导致的页面的刷新将会减弱用户的体验，因此本系统在运行时采用hash路由，根据不同的hash决定不同的显示组件，使得用户在使用上可以达到无缝体验。

### 3.3.2 购买页面的H5动画

HTML5（以及CSS3）原生支持动画，在用户挑选店家商品时，应提供类似于APP端相似的滑动、购物车掉落小球等效果，这些均通过CSS3实现。

# 4、外卖平台系统的设计

## 4.1 系统的设计原则

### 4.1.1 系统数据的互通性

数据的互通性是外面平台最基本的功能之一，因为采用单页面应用，不存在界面的刷新，因此用户、卖家数据更新之后，本页面数据并没有重新获取。因此需要在更新数据之后统一获取新数据，这也体现了数据与页面分离的思想。

### 4.1.2 开放性

考虑到未来系统存在安全性、性能、扩展功能的升级（多指后台），前端与后台应严格分开，前端统一接口标准，如需升级只需要修改后台代码，前端统一调用。

### 4.1.3 前端的易扩展、易维护性

作为一个互联网应用，未来会进行较多功能扩展、UI升级，因此本系统应具有良好的可扩展性以及令人清晰的分层，提高可维护性。

Vue提供的组件化开发可以使每个模块都成为一个组件，并可以将利用率较高的组件进行封装以便多次调用。而后期的升级维护也只需要找到那个部分的组件进行修改。扩展时也只需要增加额外的vue组件，在原有组件上引用即可[14]。

## 4.2 系统的架构设计

本系统部署于服务器上，数据存储在数据库中，用户通过浏览器访问系统，因此本系统的架构为浏览器/服务器（Browser/Server，B/S结构），如图4-1所示：

User

DataBase Server

ase Server

ser

User

Internet

Web Server

DataBase Server

图4-1 B/S结构

在本B/S结构上，其核心是WEB服务器，数据请求、网页的生成、数据访问、应用程序的执行都在WEB服务器上执行[15]。当用户访问此系统（浏览网页）时，从在客户端浏览器输入URL开始，向WEB服务器发送HTTP请求，经过4次握手后WEB服务器将向客户端浏览器返回一个响应，浏览器根据响应的内容开始渲染网页。

本B/S结构系统包括：视图层、逻辑层、数据层。视图层（表现层）是用户与系统之间的接口，逻辑层负责所有的业务逻辑（后台业务逻辑），数据层与逻辑层直接接触，进行数据的处理已配合业务逻辑的实现。

本系统整体架构如下图（4-2）所示：

浏览器（视图层）

HTML CSS JavaScript

Web服务器（逻辑层）

Node.js

数据库服务器（数据层）

MongoDB

图4-2 整体架构

本外卖平台系统的前端实现采用HTML+CSS+JavaScript来实现，使用Vue框架进行整合开发。HTML+CSS负责页面结构以及特效的制作，JavaScript负责前端逻辑的实现以及与后台的接口交互。

逻辑层（后台）采用Node.js，其与数据库MongoDB配合为前端返回接口响应。

## 4.3 系统用例介绍

### 4.3.1 买家用例

对于买家，系统为其提供的主要功能包括：登录、注册、购买、修改联系方式、设置收货地址、确定收货、评价。其用例图如下图4-3所示：

登录

注册

修改联系方式

设置收货地址

确定收货

评价

购买

图4-3 买家用例

### 4.3.2 卖家用例

对于买家，系统为其提供的主要功能包括：登录、注册、设置订单状态、设置店铺信息、修改商品、增加商品。其用例图如下图4-4所示：

登录

注册

设置店铺信息

修改商品

增加商品

设置订单状态

图4-4 卖家用例

# 5、外卖平台系统的实现

## 5.1 MonogDB数据集合

本系统分为三个集合，分别是用户集合（User），卖家集合（Seller），订单集合（Order）

（1）User：（每个集合MongoDB都会自动生成唯一标识\_id，下面将不再列出）

var userSchema = new Schema({  
 "username": String,  
 "password": String,  
 "order": [String],  
 "phone": String,  
 "address": [String]  
})

username和password为用户的账号密码，密码采用sha256方式加密（控制层加密后已sha256字符串存储）。Phone为收货手机号，如果为空则不能执行购买操作。Address为收货地址，因为NoSQL是松散数据结构，因此可以直接定义数据类型为数组，这是传统数据库所不能提供的便利。

（2）Seller：

var sellerSchema = new Schema({  
 "seller": {  
 "username": String, //商家登录名  
 "password": String, //商家密码  
 "checkout": Number, //当为1时，表示暂不合格不上首页  
 "name": String, //店铺名  
 "description": String, //描述（快递描述，首页）  
 "deliveryTime": Number, //送达时间  
 "score": Number, //综合评分  
 "serviceScore": Number, //服务态度  
 "foodScore": Number, //商品评分  
 "rankRate": Number, //高于周边商家（69.2）  
 "minPrice": Number, //起送价  
 "deliveryPrice": Number, //运费  
 "ratingCount": Number, //评论总数  
 "sellCount": Number, //销量  
 "bulletin": String, //商家公告（介绍）  
 "supports": [],  
 "avatar": String, //店家头像  
 "pics": [String], //店家照片  
 "infos": [String], //商家信息  
 },  
  
 "goods": [  
 {  
 "name": String,  
 "foods": [  
 {  
 "image": String,  
 "icon": String,  
 "ratings": [  
 {  
 "avatar": String,  
 "text": String,  
 "rateType": String,  
 "rateTime": Number,  
 "username": String  
 }  
 ],  
 "name": String, //餐品名  
 "price": String, //价格  
 "oldPrice": String, //原价  
 "description": String, //介绍  
 "sellCount": Number, //销量  
 "rating": Number, //好评率（100）  
 "info": String, //商品信息  
 }  
 ]  
 }  
 ],  
 "ratings": [  
 {  
 "username": String,  
 "rateTime": Number,  
 "deliveryTime": Number, //用户填写的送达时间  
 "score": Number, //用户打分  
 "rateType": Number, //0推荐1吐槽  
 "text": String,  
 "avatar": String,  
 "recommend":[String] //购买的商品  
 }  
 ]  
})

在卖家集合中，整合了卖家信息、商品、评论区三个子集合。其中商品子集是根据商品分类来组织，同一商品可以存在于多个分类。每个商品内存在评论的对象数组，用户每次评论都会将评论增加到评论区子集合以及每一个相关商品的评论中。

（3）Order：

var orderSchema = new Schema({  
 "date": Number,  
 "ordernumber": Number,  
 "process": Number, //0 刚提交订单 1 正在制作 2 正在发货 3 已经签收 5 已经评价  
 "user": String,  
 "seller\_id": String,  
 "seller\_name": String,  
 "price": Number,  
 "foods": [],  
 "phone": String,  
 "address": String,  
 "rating": {}  
})

## 5.2 node控制层部分核心模块的实现

控制层接口说明如下表：

|  |  |
| --- | --- |
| 接口名 | 借口说明 |
| allSeller | 获取全部商家数据，便于首页渲染 |
| checkLogin | 页面进入新状态时检查是否登录 |
| login | 登录 |
| logout | 登出 |
| checkexist | Ajax接口，实时监测邮箱是否被占用 |
| createuser | 根据请求不同创建买家/卖家 |
| pay | 支付 |
| addorder | 增加订单 |
| getorderlist | 根据卖家\_id获取商家订单列表 |
| getsellerorder | 根据买家\_id获取商家订单列表 |
| changephone | 改变买家手机号 |
| addaddress | 增加买家收货地址 |
| changeadr | 修改买家默认收货地址 |
| deleteadr | 删除买家收货地址 |
| jiedan | 卖家设置接单操作 |
| paisong | 卖家设置派送操作 |
| shouhuo | 买家选择收货 |
| subrating | 买家发表评论 |
| addinfo | 增加商家信息 |
| deinfo | 删除商家信息 |
| changebaseinfo | 修改商家基础信息 |
| changebulletin | 修改商家公告 |
| addType | 增加商品分类 |
| removeType | 删除商品分类 |
| changegood | 更改商品 |
| addpic | 表单上传图片 |
| addgood | 增加商品 |

### 5.2.1 实时监测注册用户名是否被占用

相比于传统网站中，点击注册才会查询用户名（邮箱）是否已占用，采用ajax方案则可以监听input元素的change事件来实时查询，代码如下：

前端：

<input type="text" @change="checkusername" v-model="subinfo.username" placeholder="6~18位不允许中文特殊字符">

@change是vue封装的改变事件，v-model将input的内容双向绑定到数据中，当文本改变且失去焦点后触发事件

checkusername() {  
 if(! /^\w{5,17}$/.test(this.subinfo.username)){  
 this.usernamebol = 1  
 }else {  
 this.$http.get('/checkexist?'+'username='+this.subinfo.username + '&subtype=' + this.subinfo.subtype).then((res) => {  
 //false 表示未被占用  
 if(res.body == 'false'){  
 this.usernamebol = 0;  
 }else{  
 this.usernamebol = 1;  
 }  
  
 })  
 }  
}

向WEB服务器发送get请求，附带username参数以及注册类别（买家/卖家），在后台得到相应：

exports.checkExist = function(req,res){  
 var username = req.query.username;  
 var type = req.query.subtype; //true用户 false商家  
 if(type == 'true') {  
 User.checkExist(username,function(bollean){  
 res.send(bollean.toString());  
 })  
 }else if(type == 'false'){  
 Seller.checkExist(username,function(bollean){  
 res.send(bollean.toString());  
 })  
 }  
}

由于采用的MongoDB是文档型数据库，在router.js开始引入mongo文件（js文件）便可以在后续调用。每个数据集合可以存在自己的内部方法：

//检查用户是否被占用  
userSchema.statics.checkExist = function(username,callback){  
 //this指向类名，非Schema  
 this.find({'username': username},function(err,results){  
 if(results.length == 0){  
 callback(false);  
 }else{  
 callback(true);  
 }  
 });  
};

其中采用node的函数式编程风格，调用时将函数作为参数使用，这也体现了javascript作为一个开放式语言用作服务器编程的优势（足够的灵活性）。

### 5.2.2 登录

登录除了匹配数据库内容外，设置session也是核心之一。设置session可以使用户在这个浏览器运行期间（当前浏览器未关闭），服务器记住用户的登录状态而不用重新登陆。Node服务器设置session使用了express-session中间件：

var session = require('express-session');

var app = express()  
app.use(session({  
 secret: 'houtai',  
 // 过期时间，10天  
 expires : new Date(Date.now() + 1000 \* 60 \* 60 \* 24 \* 10),  
 resave: false,  
 saveUninitialized: true  
}));

在控制器层便可以使用session：

//登陆  
//-1服务器错误  
//-2没有这个用户  
//-3密码不正确  
//1登陆成功  
exports.login = function (req,res) {  
 var form = new formidable.IncomingForm();  
 form.parse(req, function(err, fields, files){  
 if(err){  
 res.send("-1");  
 return;  
 }  
 var hash = crypto.createHash("sha256");  
 var username = fields.username;  
 var password = fields.password;  
 var passwordSha256 = hash.update(password).digest("hex");  
 var type = fields.logintype;  
 if(type) {  
 User.find({'username': username},function (err ,results) {  
 if(err) { res.send("-1"); return;}  
 if(results.length == 0){  
 res.send("-2");  
 return;  
 }  
 if(results[0].password != passwordSha256){  
 res.send("-3");  
 return;  
 }  
 req.session.login = true;  
 req.session.username = username;  
 req.session.usertype = true; //用户状态为用户，false为商家  
 res.send( {  
 username: results[0].username,  
 type: type,  
 order: results[0].order,  
 phone: results[0].phone,  
 address: results[0].address  
 });  
 })  
 }else{  
 Seller.find({'seller.username': username},function (err,results) {  
 if(err) { res.send("-1"); return;}  
 if(results.length == 0){  
 res.send("-2");  
 return;  
 }  
 if(results[0].seller.password != passwordSha256){  
 res.send("-3");  
 return;  
 }  
 req.session.login = true;  
 req.session.username = username;  
 req.session.\_id = results[0].\_id;  
 req.session.usertype = false; //用户状态为用户，false为商家  
 res.send( {  
 username: results[0].seller.username,  
 type: type,  
 seller: results[0].seller,  
 ratings: results[0].ratings,  
 goods: results[0].goods  
 });  
 })  
 }  
 })  
}

在登录过程中需要将用户的密码使用sha256加密，使用了crypto中间件作为加密方式。

登录的post请求中包含买家/卖家的标记信息type，根据type不同查询不能的数据集合，根据查询的情况返回不同的状态吗供前端调用。在匹配成功后，将会设置session让服务器记住此用户，设置了：

（1）login：表示是否登录，布尔值

（2）username：登录的用户名

（3）usertype：用户的登录状态

（4）\_id：数据集合的\_id，mongo通过\_id进行查询有着十分明显的性能优势。

然后根据用户类型不同返回对应的对象，作为状态存储在前端数据内。

### 5.2.3 用户发表评论

评价需要增加到所有相关商品的rating中，并且在Seller子集ratings也要增加：

exports.subrating = function (req,res) {  
 var form = new formidable.IncomingForm();  
 form.parse(req,function(err,fields,files){  
 if(err){  
 res.send("-1");  
 return;  
 }  
 var rating = fields;  
 var foods = fields.recommend;  
 var orderid = fields.orderId;  
 Seller.find({"\_id": rating.seller\_id},function (err,results) {  
 if(err){res.send("-1");return;}  
 results[0].ratings.push(rating);  
 foods.forEach(function (v,i,arr) {  
 results[0].goods.forEach(function (\_v,\_i,\_arr) {  
 \_v.foods.forEach(function (\_\_v,\_\_i,\_\_arr) {  
 if(\_\_v.name === v){  
 results[0].goods[\_i].foods[\_\_i].ratings.push(rating);  
 }  
 })  
 })  
 })  
 results[0].save();  
 Order.find({"\_id": orderid},function (err,doc) {  
 if(err){res.send("-2");return;}  
 doc[0].process = 5;  
 doc[0].rating = rating;  
 doc[0].save();  
 res.send("1")  
 })  
 })  
 })  
}

遍历数组采用ES5新增的forEach方法，对传来的food数组进行遍历，对Seller集合中匹配的商品进行添加。因为商品嵌套较深因此进行了3层循环。

### 5.2.4 增加商品

增加商品是外卖平台系统较为重要的一项功能。增加商品分类两步：

（1）上传图片：

采用的方式是H5新增的表单属性：

<form id="uploadForm" enctype="multipart/form-data" method="post">  
 <div class="main"><div><input class="iconsub" type="file" accept="image/png,image/,image/jpg,image/jpeg" name="tupian" /><input @click="uppic" class="sctp" type="button" value="上传图片"></div></div>  
</form>

Enctype=”multipart/form-data”表示该表单上传的是二进制数据，表单里的input的值将以2进制的方式发送到后台。后台接受图片接口：

exports.addpic = function (req,res) {  
 var form = new formidable.IncomingForm();  
 //设置上传目录  
 form.uploadDir = "./uploads";  
 form.parse(req, function(err, fields, files) {  
 //确认有文件上来  
 if(files.length == 0){  
 res.send("非法操作，请上传图片！");  
 return;  
 }  
 //改名逻辑  
 //改名，给上传的文件加上.jpg后缀  
 fs.rename("./" + files.tupian.path , "./" + files.tupian.path + ".jpg",function(err){  
 if(err){  
 res.send("改名失败，请重新上传图片！");  
 return;  
 }  
 res.send(req.session.newpicname)  
 })  
 });  
}

在实际上线项目开发时，图片应上传至专有CDN（内容分发网络），返回CDN路径，本系统将模拟上传至本地//uploads，改名后返回响应内容为图片路径。

（2）发送创建商品请求：

包含返回的图片地址（如果前端未返回图片地址则不能进行请求），接口为：

exports.addgood = function (req,res) {  
 var form = new formidable.IncomingForm();  
 form.parse(req,function(err,fields,files){  
 if(err){  
 res.send("-1");  
 return;  
 }  
 if(!fields.newPicRef){  
 res.send("-2");  
 return;  
 }  
 var gooditem = {  
 "image": fields.newPicRef,  
 "icon": fields.newPicRef,  
 "info": fields.info,  
 "rating": 0,  
 "sellCount": 0,  
 "description": fields.description,  
 "name": fields.name,  
 "price": fields.price,  
 "oldPrice": fields.oldPrice,  
 "ratings": []  
 }  
 Seller.find({"\_id": req.session.\_id},function (err,results) {  
 if(err){res.send("-1");}  
 results[0].goods.forEach(function (v,i) {  
 fields.typeNum.forEach(function (\_v,\_i) {  
 if(\_v == i){  
 results[0].goods[i].foods.push(gooditem);  
 }  
 })  
 })  
 results[0].save();  
 res.send("1");  
 })  
 })  
}

因为商品的数据结构为，商品按照类别分类，每个分类下可以存在相同商品，因此需要遍历goods子集合匹配name，然后进行添加。

## 5.3 vue框架 前端实现

传统网站（非单页面应用）的开发，常见多个html文件，每个html文件是一个网页，体验不好而且混杂。一般单页面应用开发时，只有一个html文件，层次不清不利于开发。Vue使用组件化开发模式，每一个组件为一个.vue文件，然后由webpack（工程化工具）解析打包，层次更加清晰。如图5-2：

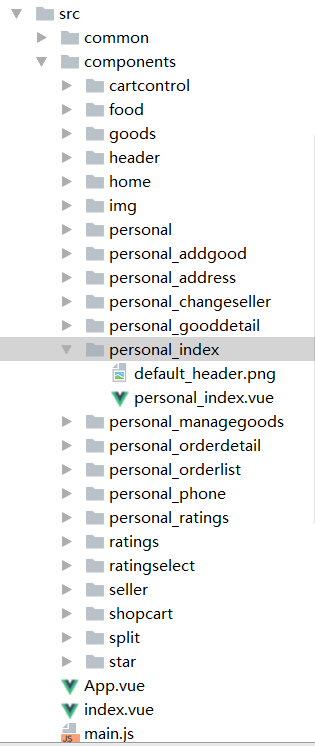


图5-2

每个文件夹包含一个组件文件（XXX.vue）以及此组件所依赖的静态图片。main.js作为Vue项目的入口文件，作用为引入所有的vue组件文件，书写路由并根据路由调用不同的组件。

Index.html是vue项目的入口html文件， 作为总路由载体

<router-view :allseller="allseller" :seller="showseller" :userdetail="userdetail" ></router-view>

其中所传数据为app创建时获取的数据，并继承给后续组件。

路由结构为：

router.map({  
 '/home':{  
 component: home //首页  
 },  
 '/personal': {  
 component: personal, //个人中心  
 subRoutes: {  
 '/': {  
 component: personal\_index //个人中心首页  
 },  
 '/orderlist': {  
 component: personal\_orderlist //个人中心-订单列表  
 },  
 '/address': {  
 component: personal\_address //个人中心-收货地址  
 },  
 '/phone': {  
 component: personal\_phone //个人中心-手机号  
 },  
 '/changeinfo': {  
 component: personal\_changeseller //个人中心-改变卖家信息  
 },  
 '/managegoods': {  
 component: personal\_managegoods //个人中心-管理商品  
 }  
 }  
 },  
 '/detail':{  
 component: App, //商家店铺页  
 subRoutes: {  
 '/':{  
 component: goods //默认商品页  
 },  
 '/goods':{  
 component: goods //商品页  
 },  
 '/ratings':{  
 component: ratings //评论页  
 },  
 '/seller': {  
 component: seller //商家详情页  
 }  
 }  
 },  
})

### 5.3.1 核心模块vue实现

本节将以核心vue模块——APP.vue简单介绍其前端部分。

一个vue模块分为三个部分：

（1）html模板部分

（2）javascript数据部分（主要是对vue实例的扩展）

（3）样式部分（本系统使用stylus书写css样式）

<template></template>  
<script type="text/ecmascript-6"></script>  
<style lang="stylus" rel="stylesheet/stylus"></style>

APP.vue大模块视图如下图5-3：



图5-3

其模板分为三部分：

（1）公用头部模块：header

（2）APP.vue的模板：商品—评论—商家的tab栏

（3）下方具体的路由

<template>  
 <div>  
 <v-header :seller="seller.seller"></v-header>  
 <div class="tab border-1px">  
 <div class="tab-item">  
 <a v-link="{path:'/detail/goods'}">商品</a>  
 </div>  
 <div class="tab-item">  
 <a v-link="{path:'/detail/ratings'}">评论</a>  
 </div>  
 <div class="tab-item">  
 <a v-link="{path:'/detail/seller'}">商家</a>  
 </div>  
 </div>  
 <router-view :seller="seller" :userdetail="userdetail" keep-alive></router-view>  
 </div>  
</template>

选择不同的tab将会切换不同的路由，第（3）部分的路由模块将会切换到不同的模块：goods、ratings、seller。例如切换至ratings路由，ratings模块将会以呈现：

图5-3-2

每个模块都存在相应的js部分，这部分主要作为向外暴露一个vue实例化对象，其中一般包括：

<script type="text/ecmascript-6">  
 import split from 'components/split/split';  
  
 export default {  
 props: {  
  
 },  
 data() {  
   
 },  
 created() {  
  
 },  
 methods: {  
  
 },  
 components: {  
 split  
 }  
 }  
</script>

Props：从父组件继承的属性。

Data：本组件存储的数据。

Created：组件创建时执行的方法，通常是ajax获取初始信息，或者根据props的数据初始化data数据。

Methods：本组件的方法，点击等等方法都在此处定义，例如登录注册的ajax请求。

Components：本组件所依赖的其他组件，在开始通过import关键字引入其他模块，并在comonents中声明，就可以在模板出通过<Component></Component>方式调用模块。

## 5.4 外卖平台系统的测试

### 5.4.1 测试环境：

本机测试，客户端服务器均为本机。配置包括：处理器 Inter i7 2.6GHz，内存为8G，硬盘容量256GB。操作系统 windows 10，配置环境：node.js v6.2.0，mongoDB：v3.4.1。

### 5.4.2 启动

因为本系统存在虚拟数据，因此在项目首次启动时可以注入虚拟数据来进行模拟，启动步骤：

（1）在cmd中执行启动数据库命令：mongod –dbpath PATH (PATH为本地数据库路径，实际环境中可以托管至web上不需要启动)。

（2）将//models/Seller.js 116~124行代码取消注释（json注入数据库）

（3）在项目根目录执行命令 npm run dev 此时监听端口8060

（4）将（2）中代码重新注释

（5）浏览器访问<http://localhost:8060>

注：第一个卖家存在账号密码，请至Seller.js代码中查看注释。

### 5.4.3 平台功能测试

经过上面的步骤系统已可以运行（测试环境），由于node是单线程，出现错误代码会中断运行并返回错误代码，因此测试中主要以黑盒测试为主，将主要功能进行测试。

依照第三章需求部分以及第五章接口部分进行逐个测试，经测试所有功能均已实现。

### 5.4.4 系统功能测试总结

经测试，本系统的功能达到预期期望，各项功能均能正常使用。在使用体验上，由于采用单页面应用的方式，页面间的切换快速流畅。

在界面UI上，商品部分采用了网上的better-scroll库实现了类似移动端源生APP的滚动特效，通过CSS3新增的动画实现了购物车小球掉落、点击商品详情页面横向滚过的特效，用户体验良好。

# 6、总结与展望

本文对于基于vue、node.js的移动端外卖平台系统的设计背景、技术方案、实现过程进行了较为详细的介绍，本章将对以上内容进行总结分析，对本设计存在的意义进行总结，并对项目存在的问题以及下一步的发展进行展望。

## 6.1 论文总结

本文从外卖平台的定义、传统网站开发的方式开始，分别针对技术方案、功能需求、项目的架构设计、项目的实现以及测试等方面，详细的介绍了基于vue、node.js的移动端外卖平台前端设计与实现。

本文主要完成的工作如下：

（1）相关背景介绍。针对传统web开发、移动端源生APP的开发现状，介绍了基于web的移动端app的开发优势，针对个人、小团体来说，web开发具有语言的统一性，更好适配。

（2）技术相关介绍。介绍了后台所用全新技术node.js，数据库mongoDB，并且介绍了目前前端领域炽手可热的前端框架vue.js，针对原有前端开发的缺陷进行了优势补充，其新增的特性对于现代web开发具有相当大的意义。

（3）。针对外卖平台系统的常见需求进行介绍。

（4）实现以及测试介绍系统需求介绍。在确定技术方案、需求的前提下，运用vue以及node完成了开发工作，在这一部分中着重对关键需求进行了介绍，并对前端关键组件进行了视图展示。在测试中所有功能运行良好。

## 6.2 论文实现的意义

随着互联网的快速发展，移动应用显得尤为重要。而在大学中，大学创业也变得尤为普遍，特别是在互联网创业方面，而如何快速、低成本的搭建一个app平台则显得尤为重要。通过web app的方式，可以免去IOS、Android两个平台复杂的源生开发学习，并且使用web开发，具有较强的灵活性、开放性、易扩展性，而HTML5、CSS3带来的新特性则使网页在新技术的加持下做到更好的用户体验，对于广大创业同学来说，使用web app代替各个平台的源生app不失为一个高性价比的选择。

## 6.3 存在的问题以及未来的展望

### 6.3.1 现阶段的问题

由于时间以及技术的限制，本系统还存在一定的问题有待改善，node.js作为单线程应用，如果其中出现问题就会终止服务并抛出错误，因为使用环境较为复杂，特别是在多用户情况下，因此系统的稳定性还需在实践中长期测试并逐渐完善。

本系统只在商品页面实现了较为美观的UI设计，个人管理中心使用了比较原始的css技巧，后期可以逐步的、组件分次的进行设计优化。

本系统模拟本地实现静态资源服务，在实际项目中，通常使用在线的静态资源服务（例如七牛云），而储存时仅需储存cdn链接。

### 6.3.2 未来的展望

Web技术的发展是十分快速的，不仅仅是前端，后端发展也是。Node.js拥有庞大的社区支持，其稳定性、可靠性、安全性将会大大提升，而可以应用的中间件、npm包也会越来越多，方便开发。

前端框架也进入了MVVM时代，本项目使用的是vue1.0版本，近期推出的vue2.0版本使用了虚拟DOM技术，在性能上有着较大的提升。同时近期火热的React Native甚至提供了跨平台（IOS Android）的解决方案。在未来有业务需求时，可根据需求的具体情况选择不同的框架，但是有一点是决定的，采用框架开发的效率、性能、可扩展性将远远大于传统的网站开发。

**参考文献：**

[1]王伟军，孙晶. Web2. 0的研究与应用综述[J]. 情报科学，2008, 25(12): 1907-1913 .

[2]李军，高并发Web系统的设计与优化[D]. 北京:北京交通大学，2009, 1-15.

[3]冯振兴. Ajax技术在Web系统中的应用研究[D].北京林业大学.2008

[4]Jeremy Keith.《JavaScript DOM编程艺术》[M[.人民邮电出版社.2011.

[5]蒋文旭，辛阳.大型高并发 Web 应用系统架构分析与设计[EB/OL].

http://www.paper.edu.cn/releasepaper/content/201203-557.html. 2012

[6]陈瑶. 基于Node.js高并发web系统的研究与应用[D].电子科技大学.2014.

[7]王金龙,宋斌,丁锐. Node.js:一种新的Web应用构建技术[J]. 2015.

[8]彭娜. 基于Node.JS博客系统的设计与实现[D].大连理工大学.2012.

[9]npm社区[EB/OL]. https://www.npmjs.com/.2015.

[10]朴灵.《深入浅出Node.js》[M[.人民邮电出版社.2013.

[11]Brad Dayley.《Node.js+MongoDB+AngularJS Web开发》[M[.电子工业出版社.2015

[12]Kristina Chodorow.《MongoDB权威指南》[M].人民邮电出版社.2013.

[13]vue社区[EB/OL]. https://forum.vuejs.org.2015.

[14]张耀春、黄轶等.《Vue.js权威指南》[M[.电子工业出版社.2016.

[15]王越. 基于node.js的微博系统的设计与实现[D].电子科技大学.2014.

**附 录：**

其附件为后台核心路由router.js全部代码：

router.js:

var formidable = require("formidable");

var fs = require("fs");

var crypto = require("crypto"); //加密

var url = require("url");

var \_ = require("underscore");

var Seller = require('../models/Seller');

var User = require('../models/User');

var Order = require('../models/Order');

//检查是否登录

exports.checkLogin = function(req,res){

if(req.session.login == true){

if(req.session.usertype) {

User.find({'username': req.session.username},function (err ,results) {

if(err) { res.send("-1"); return;}

res.send( {

username: results[0].username,

type: req.session.usertype,

order: results[0].order,

phone: results[0].phone,

address: results[0].address

});

})

}else {

Seller.find({'seller.username': req.session.username},function (err,results) {

if(err) { res.send("-1"); return;}

res.send( {

username: results[0].seller.username,

type: req.session.usertype,

seller: results[0].seller,

ratings: results[0].ratings,

goods: results[0].goods

});

})

}

}else{

res.send({

data: {

username: '',

type: true

},

errno: 0

})

}

};

//登陆

//-1服务器错误

//-2没有这个用户

//-3密码不正确

//1登陆成功

exports.login = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req, function(err, fields, files){

if(err){

res.send("-1");

return;

}

var hash = crypto.createHash("sha256");

var username = fields.username;

var password = fields.password;

var passwordSha256 = hash.update(password).digest("hex");

var type = fields.logintype;

if(type) {

User.find({'username': username},function (err ,results) {

if(err) { res.send("-1"); return;}

if(results.length == 0){

res.send("-2");

return;

}

if(results[0].password != passwordSha256){

res.send("-3");

return;

}

req.session.login = true;

req.session.username = username;

req.session.usertype = true; //用户状态为用户，false为商家

res.send( {

username: results[0].username,

type: type,

order: results[0].order,

phone: results[0].phone,

address: results[0].address

});

})

}else{

Seller.find({'seller.username': username},function (err,results) {

if(err) { res.send("-1"); return;}

if(results.length == 0){

res.send("-2");

return;

}

if(results[0].seller.password != passwordSha256){

res.send("-3");

return;

}

req.session.login = true;

req.session.username = username;

req.session.\_id = results[0].\_id;

req.session.usertype = false; //用户状态为用户，false为商家

res.send( {

username: results[0].seller.username,

type: type,

seller: results[0].seller,

ratings: results[0].ratings,

goods: results[0].goods

});

})

}

})

}

//登出

exports.logout = function(req,res){

req.session.login = false;

req.session.username = null;

res.send('1');

}

//检查注册邮箱是否被占用

exports.checkExist = function(req,res){

var username = req.query.username;

var type = req.query.subtype; //true用户 false商家

if(type == 'true') {

User.checkExist(username,function(bollean){

res.send(bollean.toString());

})

}else if(type == 'false'){

Seller.checkExist(username,function(bollean){

res.send(bollean.toString());

})

}

}

//创建用户

//-1服务器错误

//1成功

exports.createuser = function(req,res){

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

var hash = crypto.createHash("sha256");

var passwordSha256 = hash.update(fields.password).digest("hex");

var type = fields.subtype;

console.log(type)

if(type == 'true') {

var u = new User({

"username": fields.username,

"password": passwordSha256

});

u.save(function(err){

if(err){

res.send("-1");

return;

}

res.send("1");

})

}else if(type == 'false'){

var s = new Seller({

"seller": {

"username": fields.username,

"password": passwordSha256,

"checkout": 1

},

"goods": [],

"ratings": []

});

s.save(function(err){

if(err){

res.send("-1");

return;

}

res.send("1");

})

}

// Seller.find({'seller.name': '第二家'},function (err, results) {

// if(err) { res.send("-1"); return;}

// results[0].seller.username = fields.username;

// results[0].seller.password = passwordSha256;

// results[0].save()

// })

})

};

var setDateString = function () {

return new Date().getFullYear().toString() + (new Date().getMonth() <9 ? ('0' + (new Date().getMonth()+1).toString()) : (new Date().getMonth()+1).toString()).toString() + (new Date().getDate() <9 ? ('0' + (new Date().getDate()).toString()) : (new Date().getDate()).toString()).toString() + (parseInt(Math.random()\*10000)).toString()

}

//支付创建订单

//-1服务器错误

//返回订单mongo \_id成功

exports.pay = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

var order = fields;

order.date = +new Date();

order.ordernumber = setDateString();

order.process = 0;

var v = new Order(order);

v.save(function(err){

if(err){

res.send("-1");

return;

}

res.send(v.\_id);

})

})

}

//为用户增加订单

exports.addorder = function (req,res) {

var id = req.query.id;

User.find({'username': req.session.username},function (err,results) {

if(err){res.send("-1");return;}

results[0].order.push(id);

results[0].save(function () {

res.send("1")

})

})

}

//获取订单列表

exports.getorderlist = function (req,res) {

// if(!req.session.type) {return;}

User.find({'username': req.session.username},function (err,results) {

if(err){res.send("-1");return;}

var orderlistid = results[0].order;

Order.find()

.where('\_id').in(orderlistid)

.exec(function (err,results) {

res.send(results)

})

})

}

//获取商家订单列表

exports.getsellerorder = function (req,res) {

if(!req.session.\_id) {return;}

Order.find({'seller\_id': req.session.\_id},function (err,results) {

if(err){res.send("-1");return;}

res.send(results)

})

}

//修改用户手机号

exports.changephone = function (req,res) {

var phone = req.query.phone;

User.find({"username": req.session.username},function (err,results) {

if(err){res.send("-1");return;}

console.log(results)

results[0].phone = phone;

results[0].save(function () {

res.send("1")

});

})

}

//增加收货地址

exports.addaddress = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

var address = fields.address;

User.find({"username": req.session.username},function (err,results) {

if(err){res.send("-1");return;}

results[0].address.push(address);

results[0].save();

res.send("1")

})

})

}

//修改收货地址

exports.changeadr = function (req,res) {

var num = req.query.num;

User.find({"username": req.session.username},function (err,results) {

if(err){res.send("-1");return;}

var i = results[0].address.splice(num,1);

results[0].address.unshift(i);

results[0].save();

res.send("1")

})

}

//删除收货地址

exports.deleteadr = function (req,res) {

var num = req.query.num;

User.find({"username": req.session.username},function (err,results) {

if(err){res.send("-1");return;}

results[0].address.splice(num,1);

results[0].save();

res.send("1")

})

}

//接单

exports.jiedan = function (req,res) {

var num = req.query.num;

Order.find({"\_id":num},function (err,results) {

if(err){res.send("-1");return;}

results[0].process = 1;

results[0].save();

res.send("1")

})

}

//派送

exports.paisong = function (req,res) {

var num = req.query.num;

Order.find({"\_id":num},function (err,results) {

if(err){res.send("-1");return;}

results[0].process = 2;

results[0].save();

res.send("1")

})

}

//收货

exports.shouhuo = function (req,res) {

var num = req.query.num;

Order.find({"\_id":num},function (err,results) {

if(err){res.send("-1");return;}

results[0].process = 3;

results[0].save();

res.send("1")

})

}

//发表评论

exports.subrating = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

var rating = fields;

var foods = fields.recommend;

var orderid = fields.orderId;

Seller.find({"\_id": rating.seller\_id},function (err,results) {

if(err){res.send("-1");return;}

results[0].ratings.push(rating);

foods.forEach(function (v,i,arr) {

results[0].goods.forEach(function (\_v,\_i,\_arr) {

\_v.foods.forEach(function (\_\_v,\_\_i,\_\_arr) {

if(\_\_v.name === v){

results[0].goods[\_i].foods[\_\_i].ratings.push(rating);

}

})

})

})

results[0].save();

Order.find({"\_id": orderid},function (err,doc) {

if(err){res.send("-2");return;}

doc[0].process = 5;

doc[0].rating = rating;

doc[0].save();

res.send("1")

})

})

})

}

//增加商家信息

exports.addinfo = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

var info = fields.info;

Seller.find({"\_id": req.session.\_id},function (err,results) {

if(err){res.send("-1");}

// console.log(info,results[0].seller)

results[0].seller.infos.push(info);

results[0].save();

res.send("1");

})

})

}

//删除商家信息

exports.deinfo = function (req,res) {

var num = req.query.num;

Seller.find({"\_id": req.session.\_id},function (err,results) {

if(err){res.send("-1");return;}

results[0].seller.infos.splice(num,1);

results[0].save();

res.send("1")

})

}

//修改商家基础信息

exports.changebaseinfo = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

Seller.find({"\_id": req.session.\_id},function (err,results) {

if(err){res.send("-1");}

// console.log(info,results[0].seller)

results[0].seller.deliveryPrice = fields.deliveryPrice

results[0].seller.name = fields.name;

results[0].seller.description = fields.description;

results[0].seller.minPrice = fields.minPrice;

results[0].save();

res.send("1");

})

})

}

//修改商家公告

exports.changebulletin = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

Seller.find({"\_id": req.session.\_id},function (err,results) {

if(err){res.send("-1");}

// console.log(info,results[0].seller)

results[0].seller.bulletin = fields.bulletin

results[0].save();

res.send("1");

})

})

}

//删除商品分类

exports.removeType = function(req,res){

var num = req.query.num;

Seller.find({"\_id": req.session.\_id},function (err,results) {

if(err){res.send("-1");return;}

results[0].goods.splice(num,1);

results[0].save();

res.send("1")

})

}

//增加商品分类

exports.addType = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

Seller.find({"\_id": req.session.\_id},function (err,results) {

if(err){res.send("-1");}

results[0].goods.push({

"foods":[],

name: fields.newType

})

results[0].save();

res.send("1");

})

})

}

//更改商品

exports.changegood = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

var detail = fields;

var name = detail.oldname;

Seller.find({"\_id": req.session.\_id},function (err,results) {

if(err){res.send("-1");}

results[0].goods.forEach(function (v,i) {

v.foods.forEach(function (\_v,\_i) {

// \_v.forEach(function (\_\_v,\_\_i) {

if(\_v.name === name){

for(k in detail){

if(detail[k])

\_v[k] = detail[k];

}

}

// })

})

})

results[0].save();

res.send("1");

})

})

}

//上传图片

exports.addpic = function (req,res) {

var form = new formidable.IncomingForm();

//设置上传目录

form.uploadDir = "./uploads";

form.parse(req, function(err, fields, files) {

//确认有文件上来

if(files.length == 0){

res.send("非法操作，请上传图片！");

return;

}

//改名逻辑

//改名，给上传的文件加上.jpg后缀

fs.rename("./" + files.tupian.path , "./" + files.tupian.path + ".jpg",function(err){

if(err){

res.send("改名失败，请重新上传图片！");

return;

}

req.session.newpicname = files.tupian.path.substr(8)+'.jpg';

res.send(req.session.newpicname)

})

});

}

//增加商品

exports.addgood = function (req,res) {

var form = new formidable.IncomingForm();

form.parse(req,function(err,fields,files){

if(err){

res.send("-1");

return;

}

if(!fields.newPicRef){

res.send("-2");

return;

}

var gooditem = {

"image": fields.newPicRef,

"icon": fields.newPicRef,

"info": fields.info,

"rating": 0,

"sellCount": 0,

"description": fields.description,

"name": fields.name,

"price": fields.price,

"oldPrice": fields.oldPrice,

"ratings": []

}

Seller.find({"\_id": req.session.\_id},function (err,results) {

if(err){res.send("-1");}

results[0].goods.forEach(function (v,i) {

fields.typeNum.forEach(function (\_v,\_i) {

if(\_v == i){

results[0].goods[i].foods.push(gooditem);

}

})

})

results[0].save();

res.send("1");

})

})

}

exports.allSeller = function (req,res) {

Seller.find({},function (err,results) {

if(err){res.send("-1");return;}

res.json({

errno: 0,

data: results

})

})

}

**指导教师意见**

|  |
| --- |
| （包括选题的意义，资料收集或实验方法、数据处理等方面的能力，论证或实验是否合理，主要观点或结果是否正确，有何独到的见解或新的方法，基础理论、专业知识的掌握程度及写作水平等，并就该论文是否达到本科毕业论文水平做出评价）  文超同学的学士学位论文（设计）的题目是《基于vue与node的移动端外卖平台前端设计与实现》，该论文选题合理，紧扣当前热点问题。随着我国互联网+战略的大力推广，大学生创业日益增多，这对广大大学生创业的移动应用产出有着积极地作用，使得创业者们可以更快的进行创业。  该论文从前端开发背景知识介绍入手，进一步阐述系统开发的需求、架构，并系统的阐述了其相对于传统网站开发的优势，并提出了不错的建议，具有一定的时代性，也具有很强的现实性和很好的应用价值。  总的来说，该论文（设计）观点明确，论据丰富，条理清晰，结构严谨，语言通顺，专业知识运用得当，体现出较高的专业素养、较强的分析问题能力及较好的写作水平，达到了本科生毕业论文（设计）的水平，同意论文答辩。  成绩：  指导教师（签名）：  年 月 日 |

注：成绩按优、良、中、合格、不合格五级分制计。

**评阅人意见**

|  |
| --- |
| （包括选题的意义，资料收集或实验方法、数据处理等方面的能力，论证或实验是否合理，主要观点或结果是否正确，有何独到的见解或新的方法，基础理论、专业知识的掌握程度及写作水平等，并就该论文是否达到本科毕业论文水平做出评价）  文超同学的选题为《基于vue与node的移动端外卖平台前端设计与实现》，论文选题合理，工作量适中，符合专业培养目标。目前前端开发在移动端应用十分热门，在创业行为中作用广泛。  本文通过对移动端外卖平台系统的研究与设计，提出了自己的问题，阐述了自己的意见和看法，具有很强的现实意义。  论文表明，该同学查阅了较多的文献资料，具备了一定的文献综述和资料整理能力。内容较完整，层次结构清晰，主要观点正确，分析比较透彻，结构合理，对策建议有可操作性。论文文字较流畅，语言功底良好，显示了良好的写作水平，达到了本科生的培养目标。  综上所述，该论文达到了学士学位论文水平，同意论文答辩。  成绩：  评阅人（签名）：  年 月 日 |

注：成绩按优、良、中、合格、不合格五级分制计。

**答辩委员会意见**

|  |
| --- |
| （应根据论文内容和答辩情况，并参考指导教师意见、评阅人意见对论文的综合水平做出具体评价）  在答辩过程中，文超同学介绍了论文的主要观点和基本框架并回答了答辩委员的提问。答辩表明:该同学对前端开发应用在移动应用中有着较为深入的分析研究，观点具有一定的新意。思路清晰，能熟练运用各种理论解决实际问题，分析深刻。答辩人叙述流利、完整，回答问题正确合理。但该论文内容中仍有些微不足之处，若能有充分的数据例证将会使文章更加有说服力。  经答辩委员会讨论，一致同意文超同学通过学士论文答辩。  成绩：    答辩委员会主任（签名）：  年 月 日 |

**学院学位分委员会意见**

|  |
| --- |
| 综合指导教师、评阅教师的关于文超同学的“基于vue与node的移动端外卖平台前端设计与实现”一文的写作意见以及答辩委员会的答辩意见，经学院学位分委员会讨论，一致同意文超同学通过论文答辩，建议授予学士学位。  成绩：  学位分委员会主任（签名）： （公章）  年 月 日 |

注：成绩按优、良、中、合格、不合格五级分制计。

**山东师范大学本科毕业论文（设计）题目审批表**

学院（部）：管理科学与工程学院(章) 系/教研室：信息管理系 时间：2016年12月28日

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课  题  情  况 | 题目名称 | 基于vue与node的移动端外卖平台前端设计与实现 | | | | |
| 课题性质 | A基础研究 B基础应用研究 C应用研究 √ | | | | |
| 教师姓名 | 王鑫 | 职称 | 讲师 | 学位 | 硕士 |
| 课题来源 | A.科研 B.生产 C.教学 D. 学生自拟  E. 其它√ | | | | |
| 成果类别 | A.论文 B.设计 √ | | | | |
| 主要  研究  内容  与  研究  目标 | 课题主要研究内容：  传统网站项目脚本代码集中后期难维护扩展，使用后台路由频刷新页面不利于移动端用户体验。本项目使用Node作为后台，MongoDB作为数据库，vue.js作为前端框架，使用HTML5开发移动端SPA，MVVM模式来简化开发成本，组件化开发将利于后期维护。  预期目标：  实现一个移动端单页面应用，具有轻量化、高效、后期易扩展维护的特点，可以实现买卖双方常规的交互操作。  选题学生（签名）： 年 月 日  指导教师（签名）： 年 月 日 | | | | | |
| 系/教研室审题意见 | 同意  负责人（签名）: 年 月 日 | | | | | |
| 学院审批意见 | 同意  学院学位分委员会主任（签名或盖章）: 年 月 日 | | | | | |

**山东师范大学**

**本科毕业论文（设计）开题报告**

论文题目： 基于vue与node的移动端外卖

平台前端设计与实现

学院名称： 管理科学与工程学院

专 业： 信息管理与信息系统

学生姓名： 文超

学 号： 201324040231

指导教师： 王鑫

2017年 3 月 8 日

|  |
| --- |
| **一、选题的性质** 应用研究 |
| 1. **选题的目的和意义** 2. 选题的目的：   加深对于前端框架vue和后台node.js的学习了解，熟悉完成一个较完善的APP前端实现的具体步骤。结合在校所学HTML、CSS、JavaScript，使用较为前沿的技术完成一个可扩展、易维护的外卖平台APP。   1. 选题的意义：   提高自己的编程实战能力，学习并了解模块化开发、MVVM的编程思想，能够掌握独立开发小型应用的能力。并且可以为未来学习前后端构架、新型数据库思想做一定的铺垫。 |
| **三、与本课题相关的国内外研究现状，预计可能有所创新的方面**  随着互联网以及移动互联网的快速发展，网站作为其主要载体，其技术也获得了极大发展，网站的制作已远不是传统的ASP.NET等传统技术所能够满足的了。除了部分国有企业、传统企业依旧因为历史原因以及换代成本依旧保留原始技术，国内外各大公司已全面进入了前后端分离、注重页面结构语义化、js实现更复杂的交互的新时代。同时在脚本语言JavaScript上，随着前端的极大发展，js已成为当今炽手可热的编程语言，其所能做到的事情远非数年前写个验证脚本那么简单，node.js的出现，也标志的js可以作为后台语言与传统的PHP、JAVA在一定程度上同台竞技。并且Facebook，twtter，Google等公司也纷纷推出各类前端框架已适应全新的环境，而vue正是国内开发者所开发的获得世界认同的框架。  本课题着重为移动端前端实现，利用vue框架与node.js后台，可以已较低成本实现简单的外卖系统，并且依托vue框架强大的模块化开发、视图数据分离等功能，相较于传统的网站，在用户体验、加载速度，以及后期的代码维护、功能扩展上都有很大的提升。 |
| 1. **课题研究的可行性分析**   本项目使用vue作为前端框架，其语言为html、css、javascript，通过这些构建出前端页面，依靠html5/css3制作交互优良的移动网站，并依托vue.router作为前端路由，实现完全的单页面应用。在开发过程中，使用vue的组件化特性，数据绑定特性，指令、服务、过滤器等简化开发流程，并更加利于后期代码维护。后台使用node.js，是一个可以使用javascript编写的后台语言。数据库使用mongoDB，是一个使用JSON（javascript对象格式）风格储存数据的非关系型数据库，在储存JSON上具有很大的性能优势。  通过以上技术可以实现一个app的全栈开发。 |

|  |
| --- |
| 1. **课题研究的策略、方法和步骤**   策略：  （1）选定课题后，进行与本课题相关的现状调查，构思其基本框架。  （2）充分利用图书馆借阅技术书籍，以及互联网上的技术文章，对该课题所需技术进行深入学习。  （3）在老师的指导下实现课题项目并写出论文，完善课题业务需求等。  （4）通过对本课题的整体把握，以及现有环境的研究，提出自己设计（论文）的创新点。  方法：  进行本地开发，本地模拟服务器后台以及数据库数据。可以进行本地的预览、调试。如有需要可以发布至线上。  步骤：  （1）第一阶段：学习所需技术，总体规划项目架构，完成初步后台数据建模等，模拟假数据供前台开发。  （2）第二阶段：进行前端开发，必要时进行后台的接口开发。  （3）第三阶段：进行前后端联调，着重测试登录、注册、购买、评价等基础逻辑的实现。  （4）第四阶段：进行功能性扩展，如卖家开店、上架商品等较负责功能的实现。  （5）第五阶段：进行功能性自测以及细节优化，并完成论文。 |
| 1. **预期成果形式描述**   论文 |
| **七、指导教师意见**  选题合理，具有可行性，同意开题。  指导教师（签名）：  年 月 日 |
| **八、学院学位分委员会意见**  同意开题    学院学位分委员会主任（签名）：  年 月 日 |

表5（学生教师合用）

**山东师范大学本科毕业论文（设计）教师指导记录表**

学院（部）：管理科学与工程学院 专业：信息管理与信息系统

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 论文（设计）题目：基于vue与node的移动端外卖平台前端设计与实现 | | | | | | | |
| 学生姓名 | 文超 | 学号 | 201324040231 | 指导教师 | 王鑫 | 职称 | 讲师 |
| 计划完成时间：2017年5月 | | | | | | | |
| 指导情况纪录（含指导时间、指导内容）  1、2016年11月22日，导师召集开会研讨论文选题，最终确定论文题目。  2、2017年1月20日，根据所选题目，进行写作前的准备工作。  3、2017年3月4日，导师指导写作思路及方法，开始着手写作。  4、2017年3月20日，上交初稿，导师进行修改及指导。  5、2017年4月8日，修改后上交二稿，导师进一步详细修改及指导。  6、2017年4月19日，导师针对论文的格式等方面进行指导。  7、2017年5月15日，导师对论文全局进行指导，最终定稿，准备论文答辩。  学生（签名）：  指导教师（签名）： | | | | | | | |
| 学院学位分委员会主任（签名）： 年 月 日 | | | | | | | |

注：本科论文（设计）的指导应不少于5次，如表格空间不足可另附页。

表6（学院用）

**山东师范大学本科毕业论文（设计）答辩记录表**

学院（部）：管理科学与工程学院（章）专业：信息管理与信息系统

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 论文（设计）题目：基于vue与node的移动端外卖平台前端设计与实现 | | | | | | |
| 学生姓名 | 文超 | | | 学 号 | 201324040231 | |
| 指导教师 | 王鑫 | | | 职 称 | 讲师 | |
| 答辩时间 | 2017-05-18 | | | 答辩地点 | 长清校区C421 | |
| 答辩委员会名单 | 姓 名 | 性别 | 职 称 | | 职 务 | 其它 |
| 曲建华 | 女 | 副教授 | | 主席 |  |
| 王鑫 | 女 | 讲师 | | 委员 |  |
| 张鸿雁 | 女 | 讲师 | | 委员 |  |
| 王文平 | 女 | 讲师 | | 委员 |  |
| 答辩记录：  1.问：这个系统是用什么语言编写的？  答：前端页面是用html，css，javascript，采用vue框架，后端服务器采用node.js，是用javascript语言。  2.问：能否在多个系统上运行？  答：本系统暂时在本地进行运行调试，网上购买node服务器较为昂贵，数据库可以部署在网上（免费），在本地运行时，本机作为服务器，同一局域网下可以进行访问测试。  3.问：前端网站开发有许多种方式，为什么要采用这一种方法？有什么优势？  答：前端开发目前趋向于框架式开发，采用MVC、MVM、MVVM等设计模式可以简化开发过程，提高开发效率，并有利于后期维护。者相比于传统而开发方式，可以使开发者更加专注于业务的实现，将繁琐的、重复的网站开发编码过程进行简化，更具有现实开发意义。  记录人（签名）： 答辩委员会主任（签名）：  年 月 日 年 月 日 | | | | | | |

山东师范大学本科毕业论文（设计）摘要

学院：管理科学与工程学院 专业：信息管理与信息系统 班级：13级1班

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | | 文超 | | | 学号 | 201324040231 | | | 指导教师 | | 王鑫 | |
| 论文（设计）  题 目 | | | 基于vue与node的移动端外卖平台前端设计与实现 | | | | | | | | | |
| 关键词 | | | WebAPP 前端开发 Vue Node 外卖系统 | | | | | 论文（设计）字数 | | | | 约11000字 |
| 内容摘要：  随着互联网的快速发展，移动互联网的快速成长，以移动应用为主的新一代应用程序成为越来越多的人的首要选择。其中包括以Android以及IOS为主的两大阵营，这两个移动操作系统的应用程序需要通过不同的编程语言来编写，其难度较大。而本文的主要工作就是以vue前端框架、node.js作为后台语言、mongoDB作为数据库为例，实现一个基于HTML5的移动端外卖平台系统。这将为广大大学生、创业者提供了一个新的解决问题的思路。基于vue、node.js、mongoDB的移动端外卖平台，提供了外卖平台常见的如登录注册、购买评论等功能，同时该系统具有轻量化、易维护、易扩展的功能。  同时前端开发的时效性也很强，前端开发作为目前最热门的开发领域之一，每年都有大量、新型、热门的技术涌入到实际中来，为广大技术爱好者、小型创业者提供了更加丰富多样的应用解决方案，具有很强的现实意义。 | | | | | | | | | | | | |
| 成绩 |  | | | 学院负责人（签名） | | |  | | | 年 月 日 | | |

注：文科论文摘要不少于500字，理科不少于300字。

本页一式两份，一份装入学生档案，一份由学院保存