毕业设计（论文）



|  |  |
| --- | --- |
| 题 目： | 跨平台电子商城系统 |
| 学 生： | 陈杰 |
| 指导老师： | 陈丽晖 |
| 院 别： | 软件学院 |
| 专 业： | 信息管理与信息系统 |
| 班 级： | 1502 |
| 学 号： | 3158111226 |

2019年1月

福建工程学院本科毕业设计（论文）作者承诺保证书

本人郑重承诺：本篇毕业设计（论文）的内容真实、可靠。如果存在弄虚作假、抄袭的情况，本人愿承担全部责任。

学生签名：陈杰

2019 年 1 月 13 日

福建工程学院本科毕业设计（论文）指导教师承诺保证书

本人郑重承诺：我已按有关规定对本篇毕业设计(论文)的选题与内容进行了指导和审核，且提交的毕业设计（论文）终稿与上传至“大学生论文管理系统”检测的电子文档相吻合，未发现弄虚作假、抄袭的现象，本人愿承担指导教师的相关责任。

指导教师签名：

2019 年 1 月 日

跨平台电子商城系统

摘要：近几年随着手机的普遍和兴起，从曾今的只有windows和Linux系统的PC端到目前的各大平台的主流系统(Android,IOS,塞班，各大PC操作系统等),他们之间的应用软件相互独立，代码的可复用性小，开发周期长，平台差异大等，针对每个平台都要开发出一款基于该平台下的产品和解决方案，给各大公司带来了极大的开发成本和极低的开发效率，不同的开发平台都得招聘一队开发组人员，而且各大平台应用市场长时间的审核，给产品需要快速迭代的企业带来了极大的损失，常常因为此版本功能做好了，等待安卓或者IOS等主流市场审核时，往往需要等待很久的审批，如果该更新涉及一些日期活动，他们有相应的时效性，等待审核时时间过长往往会影响用户的体验和该类活动在收益上极大的缩小。如何在人力和成本上尽量的节省开支和实现产品快速的迭代，成为了当今互联网企业很棘手的问题，本系统主要尝试实现跨平台一个技术多端渲染(IOS,Android,H5)，本系统在需要多端开发的情况下，能很大程度上减少开发周期，系统开发的代码量以及后期维护的难度，且保留各大平台的特性（android的高交互和性能，web端开发高效快速迭代和动态更新）

**关键词**：前端组件化；工程化;跨平台；JavaEE;前后端分离；Native App；Vue；React；React-Native

Cross-platform e-commerce system

**Abstract**:in recent years as cell phones and rise generally, from once only Windows and Linux PC into the mainstream of the current major platform system (Android, IOS, symbian, major PC operating system, etc.), the application software are independent of each other, between them the reusability of the code is small, the development cycle is long, large difference of platform, etc., for each platform to develop a products and solutions based on the platform, has brought companies great development cost and low efficiency, different development platforms have to hire a team of development personnel, and each big market platform application review for a long period of time, the product need to be fast iterative enterprises has brought great loss, often because the version function ready, waiting for the mainstream market audit such as Android or IOS, often need to wait for a long time for examination and approval, if the update activity involves some date, they have corresponding timeliness, pending for too long tends to affect the user experience and the class activities greatly narrowed in revenue. How to try to save money on manpower and cost and realize the rapid iteration, become the Internet enterprise is difficult problem, the system is mainly trying to implement a cross-platform technology multiterminal rendering (IOS, Android, H5), the front-end is based on the current mainstream MVVM framework Vue and React - Native FaceBook open source, cross-platform technology, realizing the unity of the technology, in order to solve the current Internet mode complex business requirements and satisfy the current Internet dynamic, open interconnection and parallel development requirements. this system under the condition of the need to develop diverse, can largely reduce the development cycle, the amount of code system development as well as the difficulty of maintenance, and keep the major platform features (android interaction and high performance, web side to develop efficient rapid iteration and dynamic update)Finally, this system under the condition of the need to develop diverse, can largely reduce the development cycle, the amount of code system development as well as the difficulty of maintenance, and keep the major platform features (android interaction and high performance, rapid development of colleges and universities web side iteration and free updates).

**Keywords:** front componentization, former section of engineering, cross-platform, JavaEE, separation of front end, Native App, Vue, React-Native, React

目录

[1.绪论 5](#_Toc25589)

[1.1 课题背景和意义 5](#_Toc19481)

[1.2跨平台技术和WebComponent现状 6](#_Toc13802)

[2. 可行性分析 7](#_Toc30347)

[2.1技术可行性 7](#_Toc1311)

[2.2操作可行性 7](#_Toc18412)

[3.需求分析 7](#_Toc21528)

[3.1功能性需求 7](#_Toc4432)

[3.1.1多个商品展示模块 7](#_Toc22049)

[3.1.2 支付模块 8](#_Toc27953)

[3.1.3 分类模块 8](#_Toc9897)

[3.1.4 购物车管理 8](#_Toc12467)

[3.1.5 地址模块 8](#_Toc6127)

[3.1.6 订单模块 8](#_Toc6177)

[3.1.7 评论模块 8](#_Toc25609)

[3.1.8 商品主页模块 8](#_Toc4167)

[3.1.9 个人中心模块 9](#_Toc15199)

[3.2非功能性需求 9](#_Toc19341)

[3.2.1 UI和平面设计 9](#_Toc17363)

[3.2.2 商品详情展示 9](#_Toc2307)

[3.2.3 可扩展性 9](#_Toc30236)

[3.2.4 安全性 9](#_Toc25794)

[3.3 跨平台需求的实现 9](#_Toc29486)

[3.3.1 React&WebComponent 9](#_Toc1093)

[3.3.2 Virtual DOM 10](#_Toc31132)

[3.3.3 React-Native原理 11](#_Toc18095)

[3.4 本章小结 12](#_Toc9052)

[4.系统设计 13](#_Toc8214)

[4.1 跨平台商城的系统设计 13](#_Toc5715)

[4.2 后端系统设计 13](#_Toc16458)

[4.2.1 开发系统和运行环境 13](#_Toc26256)

[4.2.2 系统总体设计 14](#_Toc26800)

[4.2.3 常用工具类 14](#_Toc7975)

[4.2.4 商品模块 14](#_Toc12847)

[4.2.5 支付模块 17](#_Toc20224)

[4.2.6 分类模块 18](#_Toc19262)

[4.2.7 购物车模块 19](#_Toc8483)

[4.2.8 收货地址模块 20](#_Toc14218)

[4.2.9 订单模块 21](#_Toc30230)

[4.2.10 评论模块 22](#_Toc31666)

[4.2.11 个人中心模块 22](#_Toc21642)

[4.3 前端系统设计 23](#_Toc11626)

[4.3.1 “大前端” 23](#_Toc22338)

[4.3.2 webpack 23](#_Toc24640)

[4.3.3 SPA单页面应用 24](#_Toc1505)

[4.3.4 Web Component 24](#_Toc5792)

[4.3.5 对前端生态的总结 25](#_Toc21805)

[5.系统实现 27](#_Toc9750)

[总结 28](#_Toc12511)

[致谢 29](#_Toc28266)

[参考文献 30](#_Toc22724)

# 1 概述

## 1.1 课题背景和意义

随着当今互联网的快速发展，推进我国经济社会创新发展，传统的购物模式已经不能满足大多数人的需求，在线购物，在线支付等已经成为大众的选择，传统企业也面临着很大的挑战，如何正确成功的 转型变得尤为重要，才能为企业创造一个有良好盈利的基于在线商城的渠道。且满足当前互联网模式特性下的在线购物系统，重交互、界面美观简洁、高性能、高迭代、支持android原生接口调用等特性。既能保证在android下渲染为原生组件，通过js调用android预留的接口，以往的传统的原生Android或者APP内嵌浏览器的相关开发方式，两者各有优缺点，原生开发性能高，UI好，但是开发周期长，开发成本也高，上架各大Android应用市场审核时间也长，并不满足很多大型高迭代项目的需求，内嵌浏览器开发速度快，体积小但UI体验不好，基于浏览器端的用户体验并没有原生Android那样流畅和高性能等，两者各有所长，近几年就出现了一些使用JavaScript开发出原生APP的强大技术。既能保证在移动APP下高于h5的UI交互和多列表性能，同时也能满足当今互联网产品快速迭代等特性。

一个大型的系统或者网站他的每个部分并不是独立的，往往会有许多的通用模块或者UI样式以及逻辑脚本等，且每个部分之间相互关联，一个部分的数据变化会触发很多其他相关部分的样式变动或者事件，业务复杂的项目书写JS代码变得棘手，而以往的前端页面开发方式就仅仅是html,css,js脚本，并不能很好地对通用代码或者逻辑进行抽取，css样式的唯一性也是一个很大的痛点，大家都是分开开发，最后合并成几个个CSS文件html页面link标签引入，难免会遇到类名冲突等常见的问题，很不适合大型项目的并行开发，分离开发，后期的维护成本也相对增高，一个问题往往需要花好长的时间去定位并找出解决他。所以基于这些痛点，近几年出现了WebComponent，对前端的页面和UI以及CSS进行抽离，将一个大型系统进行颗粒化，组件化，抽离出公用代码，使用导入导出API进行搭建项目，组建封装好后就像是搭积木一样构建你的系统。

最后，基于前后端分离 的 开发方式快速并行开发，提供在线加载组件界面（从url获取资源后脚本转换成原生android页面）渲染，通过路由跳转切换页面实现快速并行开发，满足当今复杂业务项目分工合作和不同页面通信的适配方案。

### 1.2 跨平台技术和WebComponent现状

目前的web技术体系在中国已经成熟，随着中国互联网的发展和庞大的用户体系，新技术如雨后春笋般萌发，也经受住了一些大企在实际业务中的考验，包括支付，跨平台，前端组件化等优秀的解决方案和众多成功案例,例如:国内已经有众多企业选择使用weex进行跨平台开发（weex重在解决磨平移动web和原生android两个平台的差异性--write once run anywhere，按照官方的开发方式即可写出同时运行多端的代码，相比react native的开发方式--learn once，write anywhere，最终还是要用相同的技术写出两套代码来运行的移动web和android），基于Weex渲染的国内各大APP如（手淘，手机天猫，优酷，闲鱼，飞猪，UC浏览器，菜鸟裹裹，饿了么等），他们均是当前移动互联网下对三大特性要求较高的系统。

WebComponent基于前端组件化思想的优秀框架也被国内外大量互联网公司所使用，他们之间不同的学习成本与学习周期长度。React是13年FaceBook公司开源的基于JavaScript框架，用简介高效的语法构建用户界面。react学习成本较高周期长，对前端javascript要求比较高，独有的jsx语法更是开发利器，虚拟DOM高效渲染页面，配合react-router,redux,mobx等构建丰富的SPA单页面应用,2017年FaceBook开源了基于React框架的跨平台原生native应用开发框架React-native，吸引了国内外众多优秀的开发者，为其开源增添自己的力量，各种场景下的解决方案，正式由于其开源比国内的一些跨平台技术提前，相比与Weex而言，他拥有更多的开源开发者贡献代码，更加完备的官方文档，虚拟Dom(将html元素以js对象的形式映射出来，彻底将JS脱离于浏览器端，运行在其他第三方平台)，DIFF算法（脱离浏览器渲染引擎的高效绘制和计算以更新样式和属性）等特性，能完美的跨平台到Android，IOS原生组件、桌面程序等，特点：react中一切皆JS，css，html均使用js渲染，且文档齐全，基于React-native优秀的开源组件也最多，论坛也最活跃。包括近几年的新平台-微信小程序，百度小程序，支付宝小程序等，均能使用Vue、React进行开发，极大的缩短了开发周期）。vue是国内开发者尤大开发的优秀开源MVVM库，在国外也有很大的使用人群，由于其设计的优美简洁的API以及html内的模板语法，一些熟悉jsp语法的后端开发者也可以快速上手，数据层与视图层绑定，每个数据设置Object.defineProperty实现数据劫持，利用发布订阅模式，每次数据改动时自动触发页面重新更新数据，开发者再也不用频繁的DOM操作，只要关系业务逻辑改变数据即可。

# 可行性分析

2.1技术可行性

当今互联网已经有众多的公司使用跨平台方式开发移动APP，类似目前主流的微博，饿了吗，美团，手淘等，他们的APP内都内置了跨平台技术，可像浏览器加载html页面一样从服务器加载JS脚本渲染为原生页面。商城包括两个平台，一个是基于浏览器平台的后台管理和用户商城系统，一个是基于原生APP的用户商城APP，前台浏览器平台使用WebComponent技术渲染UI，NativeAPP使用React-Native渲染为原生native页面，各大社区的活跃性也都很高。自己在前端方面也研究了很多时间，因此，该系统在技术上是可行的。

后端JAVA更是无可厚非了，java EE一整套的开发体系，长达二十多年的技术沉淀，既简化了开发难度缩短了开发周期，经典的 MVC分层模式，使代码的可维护，可读性都有极大的提高。

2.2操作可行性

通过参考淘宝电商的交互和界面，尽可能符合人们使用习惯以及交互并简化操作流程，增加系统使用人群，并添加测试用户，咨询他们的意见和反馈并做出一定修改。因此，在操作上是可行的 。

# 3.需求分析：

3.1功能性需求

3.1.1多个商品展示模块

商品信息选购是整个系统最重要的模块，也是最基本的功能，模块包括商品分类展示，按照日期、存货、价格对数据进行排序，更方便直观的让用户找到自己想找的东西。商品搜索功能，根据商品名称进行模糊查询。多数量的结果进行分页操作，并使用懒加载，当用户浏览到底部时自动加载下一页，在不影响用户体验下减少网络流量和服务器负担。管理员对商品的管理-修改商品的信息，存货，价格，名称和商品富文本介绍等信息，商品的上下架功能，下架的商品不能被查询到，主要方便管理员对商品的控制性以及突发状况下的紧急处理办法。

3.1.2 支付模块

支付模块是卖家收益的前提，随着支付宝的普及，越来越多的人都开始使用支付宝支付商品，体验无现金生活带来的快捷和方便，本系统对接阿里巴巴蚂蚁金服的后台数据，并通过蚂蚁金服第三方支付中心进行付款，用户付款成功后收到支付宝回调结算成功，插入到支付宝订单信息表中。

3.1.3 分类模块

用户可以根据具体的分类查询该制定分类或者子分类或者下级分类下的商品，考虑到商品的复杂性，分类往往有多个层级，例如：手机可能按品牌分，各大平牌下还可以再按照价格或者适用人群细分。对于管理员来说，主要提供分类的增删改查，为该分类添加子分类，实现分类下转的功能。

3.1.4 购物车管理

用户可以对要买的商品以及对应的商品数量选择加入购物车中，并在购物车页面对商品数量进行操作，动态展示出操作后的对应的商品金额和总价，对商品进行全选和反选功能，最后对选择的商品进行删除或者结算。

3.1.5 地址模块

地址模块用于存储用户的地址信息，即用户的收货地址，购物车结算前用户需要选择或者新增收获地址并确定后才能进行下一步付款操作。

3.1.6 订单模块

订单是每笔交易的结果，订单状态有待付款，已付款，待收货，已签收和取消订单，通过每笔订单记录可以查询到该笔订单的支付宝流水账号，支付金额，所买商品，购买时间，付款时间，支付方式等，所选的地址等其他信息，用户已付款后管理员可以选择对该订单发货，用户就可以对订单的商品进行签收，签收后评论。

3.1.7 评论模块

用户签收该笔订单后可以对该订单下的商品进行评论，打分和评价，评论成狗后展示在商品的主页面。

3.1.8 商品主页模块

展示当前商品的基本信息，每个商品分为一个主图和多个附图，

其他一些基本信息和富文本介绍，以及购买过该商品的其他用户的评价等，当用户在商品列表看到自己喜欢的东西后就可以进入商品主页进行查看这些信息。

3.1.9 个人中心模块

个人中心主要是维护一些自己的信息，收货地址数据，个人信息数据，密码修改，忘记密码密码找回，展示用户的订单的所有状态和数量，已付款多少订单，未付款多少，有多少待收货以及待评价等。

3.1.10 商户数据分析模块

商户数据分析模块要提供给商户管理系统的功能，以及各个模块下的数据的分析，使用各种统计图更形象只管的展示出各个时间段和月份的模块数据，发现更多数据的意义和价值。

3.2非功能性需求

3.2.1 UI和平面设计

开发出所有当前主流系统的一整套体系产品，包括Android，IOS，PC浏览器端一套产品，并实现UI下的高交互和流畅的用户体验，以及页面的美观性和各不同分辨率的适配性。

3.2.2 商品详情展示

商品详情展示下的富文本信息可编辑多种格式切可以插入图片且支持常见的文字格式和字体样式，为用户展示更多更自定义化的信息，提高商品的介绍的可读性。

3.2.3 可扩展性

系统后期维护和升级时应当给给后台的接口预留更多的配置，以适应系统后期复杂多变的业务环境，以及需求的扩展和业务的变化，对前后端代码都要尽可能的模块化，添加新功能或者修改以往业务时不会影响到其他模块的正常运行。

3.2.4 安全性

本系统有多个用户，用户信息的安全性，交易双方的安全性，数据的安全性，系统都应该尽可能的防范于未然，在安全上下保障客户和商家，避免利益丢失。

3.2.5 开发快速迭代性

动态性：页面之间关系复杂多变，实时更新活动或者新增新页面时，以往的原生android开发显得棘手。开放互联：如今移动应用的内容和信息都是相互孤立的，应用之间的交流变得非常复杂和困难，也缺乏一定的标准和规范化的容器来承载。并行研发：移动应用的开发需要并行研发的能力，当一个移动应用发展到一定程度规模的时候，能否支撑大规模的并行研发就成为了一件非常关键而又重要的事情，性能：当今移动应用应该有良好的视觉效果和丰富的用户体验。

3.3 跨平台需求的实现

跨平台是本系统的系统特色，跨平台的本质是抹平各个平台的差异性，即用一种技术统一各平台下的行为样式和表现能力。

3.3.1 React&WebComponent

React是一个用户界面的UI构建框架(库)，它专注于MVC中的View层的渲染，所以React经常还伴随React-Redux或者Mobx来搭配使用，他们用来管理数据状态，控制React什么时候该重新渲染视图了。一个React组件的编写基本代码：

Import React from “react”

Export default Class MyComponent Extends React.Component{

Constructor(props){

//构造器，接收props，即父组件传递过来的参数

Super(props);

This.state={

Data:”数据”

}

//state中存放render()方法返回的jsx中需要被渲染或者有状态的的数据

}

ComponentWillDidMount(){

Cosnole.log(“我被初始化了”)

//生命周期函数，对应还有组件销毁，组件实例化等

}

Render(){//渲染函数，返回的jsx内容会被编译成VirtualDom,最后浏览器渲染成真实DOM对象

Return(

<div>this is jsx{{this.state.Data}}</div>

)

}

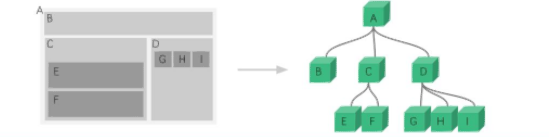
}

Export default 为Javascript模块化语法，在其他js脚本中可以使用import XXX from “组件相对路径” 来导入该组件

Class 为es6的声明类的语法，目前浏览器普遍支持的是ES5的语法，所以还要使用babel来对代码进行转换为ES5

MyComponent 组件名（既类名）

Extends React.Component继承自React.Component类



上图用代码表示为:

<A>

<B/>

<C><E/><F/></C>

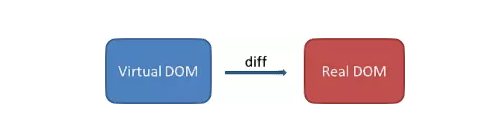
<D><G/><H/><I/></D>

</A>

一个完整的应用就是组件与组件之间props传值，每个组件处理自己的逻辑，从而达到组件复用，多个组件拼接成一个应用或者页面。

3.3.2 Virtual DOM

浏览器端的DOM对象的操作是非常昂贵且耗费性能的，大量的DOM操作应用会变得非常卡顿，每次DOM改变浏览器都会重新绘制DOM，从而引出了Virtual DOM机制，其实大部分的DOM变化都是可以预见的，Virtual DOM与浏览器DOM形成一个映射，Virtual DOM改变时最终结果映射到真实DOM，用户触发数据变化时，Virtual DOM进行diff算法查找前后的Virtual DOM结构的变化并生成补丁，根据补丁渲染到浏览器的DOM中，例如：当用户改变组件内的state的值的时候，将张三改成了李四，Virtual DOM就会对比改变前和改变后的，并生成一个diff对象，告诉真实DOM又哪里需要变化，你要改变节点了。



浏览器的节点

<div name=“chenjie”class=”wrap”><p>我是真实节点</p></div>

使用Virtual DOM来表示：

\_dom={

uid:'asdasdasa4a5as74d5a4',

Type:div,

Attributes:[{attributeKey:’name’,attributeValue:’chenjie’},{attributeKey:’class’，attributeValue:’wrap’}],

Textnode:null,

Children:{

uid:'asdasdasa4a5as54564',

Type:p,

Attributes:[],

Textnode:’我是真实节点’，

Children:null

}

}

当用户改变节点内的内容时，例如把我是真实节点改了我是虚拟节点，此时生成一个diff对象，用来记录两次的差异性

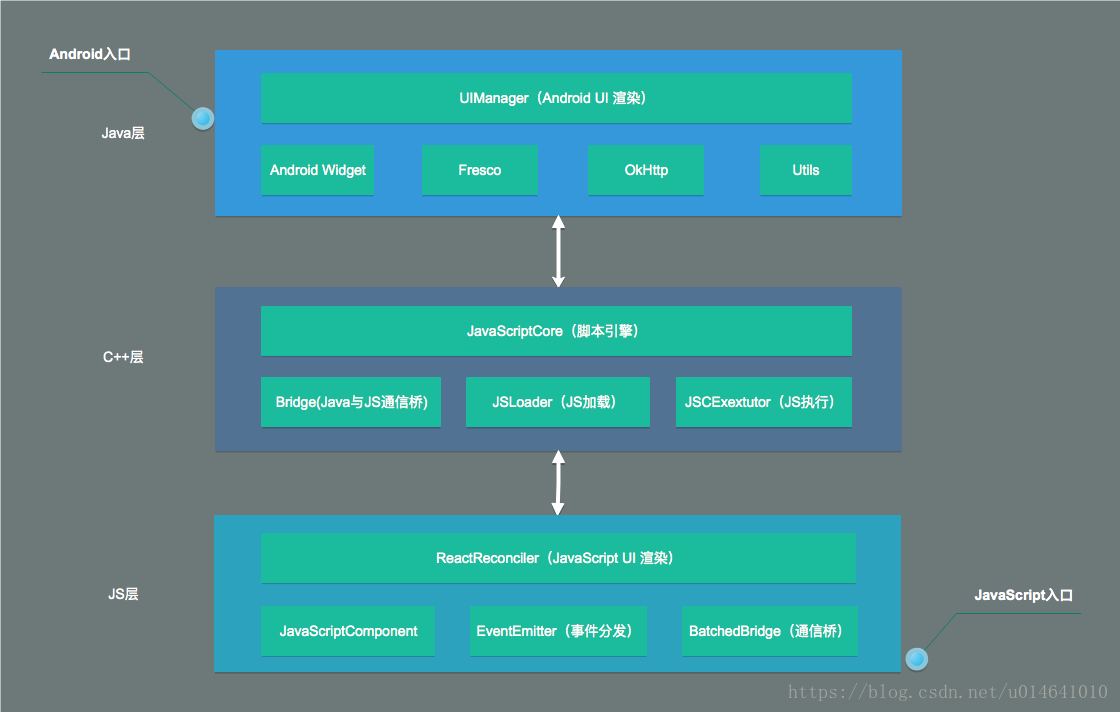
Diff={  
 changUId:asdasdasa4a5as54564’,

Value:”我是虚拟节点”

}

之后把改变的部分映射到浏览器DOM中。

3.3.3 React-Native原理



（1）Java层：主要提供一个Android的渲染引擎(UIManager)将JavaScript映射为一些组件的功能，例如OKHttp:服务器获取数据，Fresco:Android图片加载等。

（2）C++层:主要封装了一个JavaScriptCore，用来解析js代码，开发者就可以在脱离浏览器引擎的情况下使用ES6，ES7的新语法特性，如：箭头函数，const，let等API，Bridge桥接了Java与js，从而实现js调用Java的函数，java调用js。基于C++的JSLoader主要用于加载一些静态文件和JavaScriptCore，JSCExextutor主要用与执行js文件。

（3）JS层：加载React框架，并执行对应代码，将Virtual DOM映射到java层，

Virtual DOM改变后生成diff通知java的UIManage绘制对应的视图。JavascriptComonent用于构建jsx构建的React组件。

（4）跨平台最主要的特性就是依赖Virtual DOM ，它将DOM脱离了浏览器，并搬到了内存中进行操作，因此可以在任何操作系统平台下根据Virtual DOM构建一套属于该平台规则的UI体系。

3.4 本章小结

本章通过说明React框架的一些基础语法和跨平台VirtualDOM的原理和依赖，从而更好的理解跨平台的原理和实现。

# 4. 系统设计

4.1 跨平台商城的系统设计

整个系统采用前后端分离的方式，即前端使用Ajax进行request请求处理，使用当今流行的Axios框架处理异步，后台返回json数据格式，系统初期，应统一后端返回数据格式，每个返回的json中都应该含有status属性，成功为0异常为一，并把数据放在data中，例如，前台操作成功的结果应该为

Json={

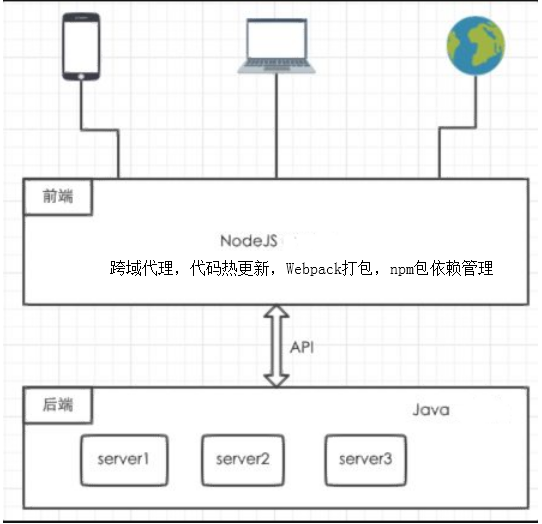
Status:0,

Data:[....]

}

操作失败应返回｛status:1,msg:”失败原因”｝规定好后方便前台处理,

并使用node代理后台端口处理浏览器跨域问题，该商城系统主要架构图如下:



4.2 后端系统设计

4.2.1 开发系统和运行环境

系统开发环境:

语言：Java

后端服务器:TomCat9.0

数据库：Mysql 5.6

后台框架:MyBatis+SPring+SpringMVC

按照传统的MVC的设计思想开发

4.2.2 系统总体设计

该系统分为管理员模块和用户模块，管理员分别是：分类管理模块，商品管理模块，订单管理。用户分别是：购物车模块，浏览商品，收货地址模块，支付模块，评论模块。

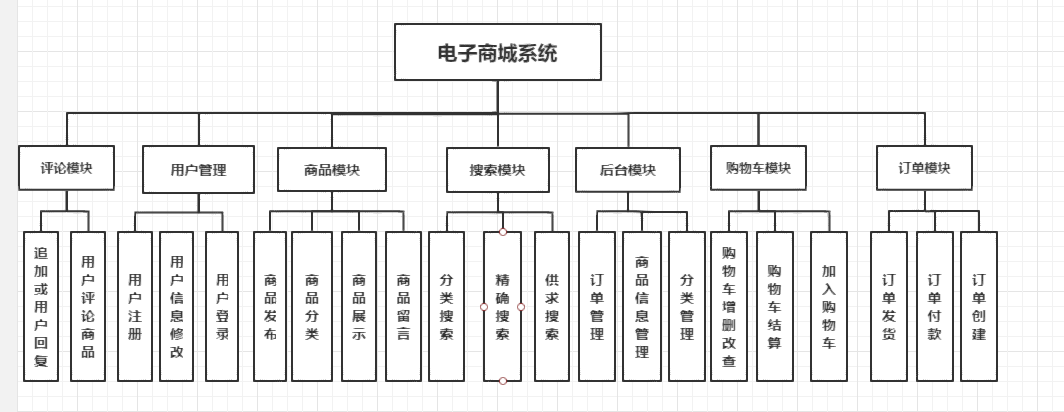


图 4.2.2.1 功能模块图

4.2.3 常用工具类

（1）Const:存储静态变量信息，防止人工输入错误导致系统出错，例如：角色是否管理员（IsAdmin），支付方式(PaymentType)，订单状态(OrderStatus)等。

（2）CacheToken:缓存Token，用户根据密保问题找回答案验证成功后，返回给用户一个有效期为两个小时的token，修改密码时根据token的有效性验证是否修改密码。

（3）FTPUtils：封装ftp上传图片的功能，本系统的图片存储使用第三方ftp图片服务器存储，后端服务不直接存储任何图片，只提供API接口服务。

（4）BigDecimalUtil:由于系统涉及金额，浮点型数据在java内会丢失精度，例如，0.00052在java中最末尾几位数会发生随机变化，变成0.00052000000000000684，避免发生不必要的意外，使用BigDecimal将数据转换为字符串进行加减，不再丢失数据精度。

（5）MD5Util：为了保护用户信息的安全性，避免数据库被黑客攻击后拿到密码，对用户密码进行md5加盐加密，提高系统的安全性。

（6）ServerResponse:返回序列化的Json数据类，更节省网络流量，使用ServerResponse.createSuccess()和ServerResponse.createByErrorCodeMessage()方法定义返回成功态和失败态，返回对应的状态码和提示语。

4.2.4 商品模块

1. 搜索功能

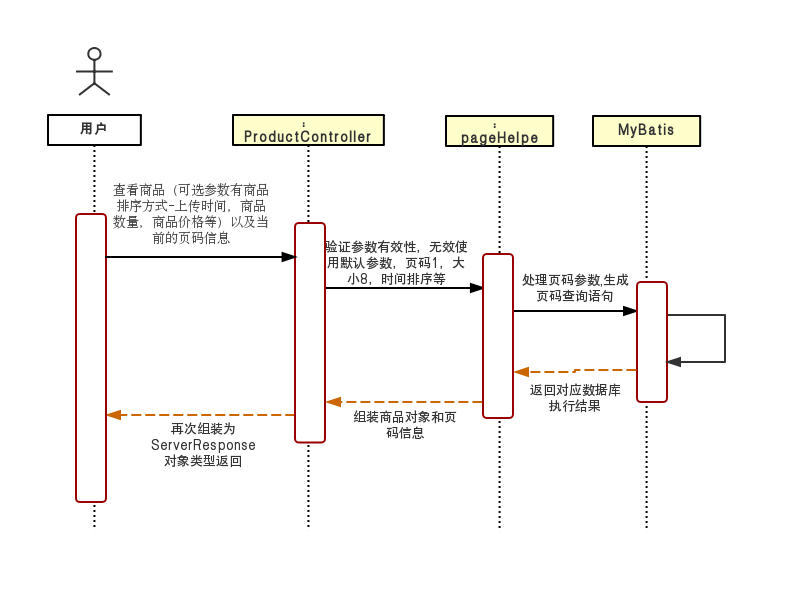


图 4.2.4.1 搜索功能时序图

参数：关键字，分类ID，当前页，一页的大小，排序功能。关键字和分类ID可不传，默认全局查询。

界面设计:Android端用户可根据喜好动态修改界面布局，动态根据价格，存货和上传日期排序，并添加对应的关键字搜索，滑动到底部时自动加载下一页并显示加载图标，给用户一个良好的用户体验。



图 4.2.4.2 搜索功能界面图

1. 管理员增加商品

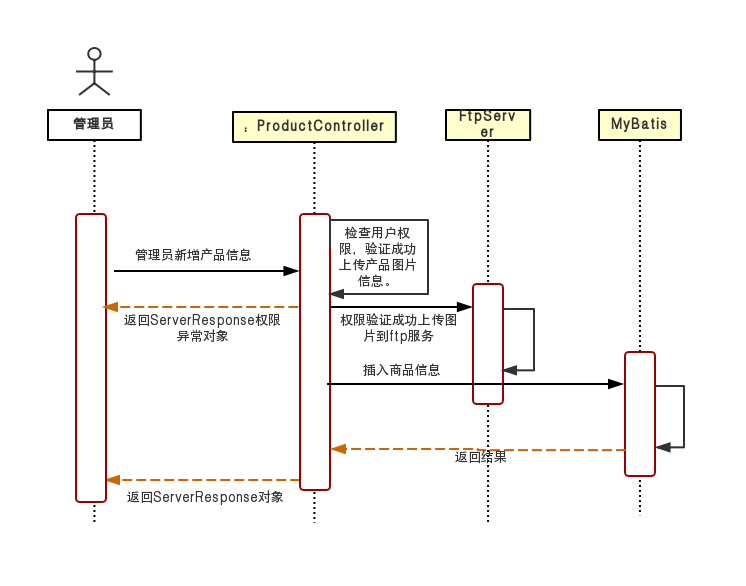


图 4.2.4.3 上传功能时序图

验证当前是否有用户登录；

验证当前登录对象是否是管理员身份；

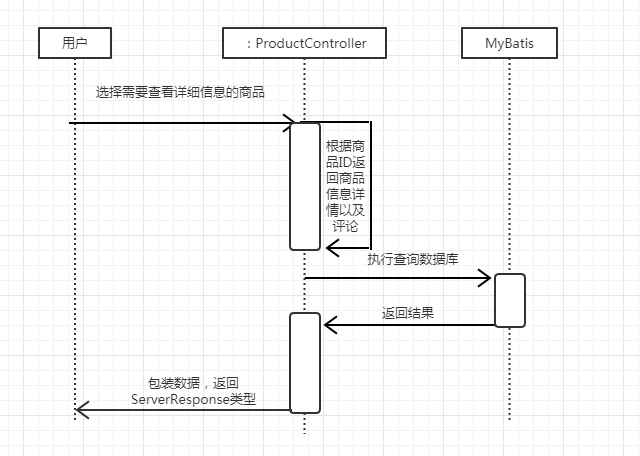
验证通过后插入Product（产品）信息（富文本，产品名称，主图，副图，价格，数量，描述，分类，是否上架等信息）。

1. 管理员上下架商品

图 4.2.4.4 上下架商品用例图

管理员在列表中对商品上下架进行选择，操作成功后实时刷新数据。

1. 商品主页



用户在商品列表根据商品主图和商品名称定位到自己想要购买的商品，点击商品进入商品主页面查看商品详情（商品副图，价格，存货，富文本描述信息等）。

界面设计:主页使用轮播图展示副图信息，并展示该商品购买后的用户的评价，以及该商品的富文本描述，使用类似word的格式更直观的展示商品的信息。



4.2.5 支付模块

使用支付宝蚂蚁金服官方提供的沙箱服务模拟线上交易转账，生成应用公钥，每次交易后向支付宝提交交易订单号以及交易信息（金额，数量，时间日期等），支付宝扣款成功后产生回调信息，用支付宝官方提供的工具验证支付宝回调是否由支付宝发出，成功修改订单状态为已付款。

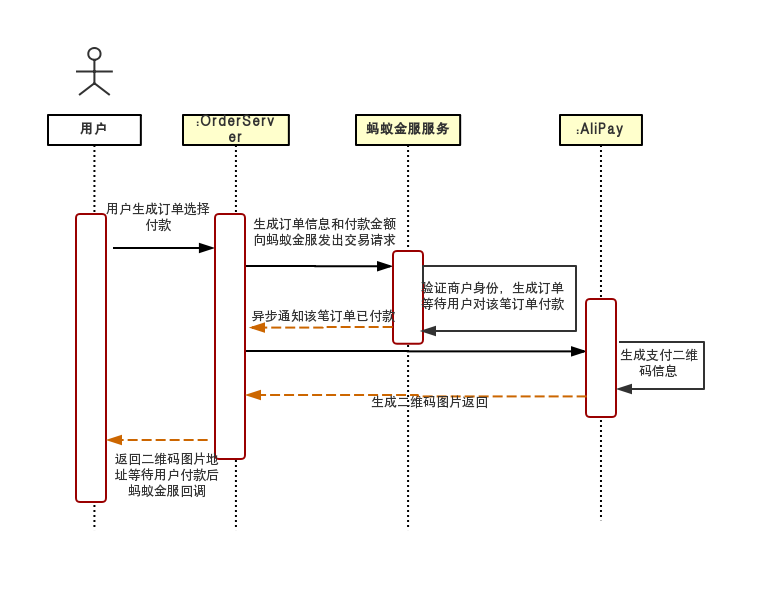


图 4.2.4.5 支付宝付款时序图

4.2.6 分类模块

（1） 管理员增加分类

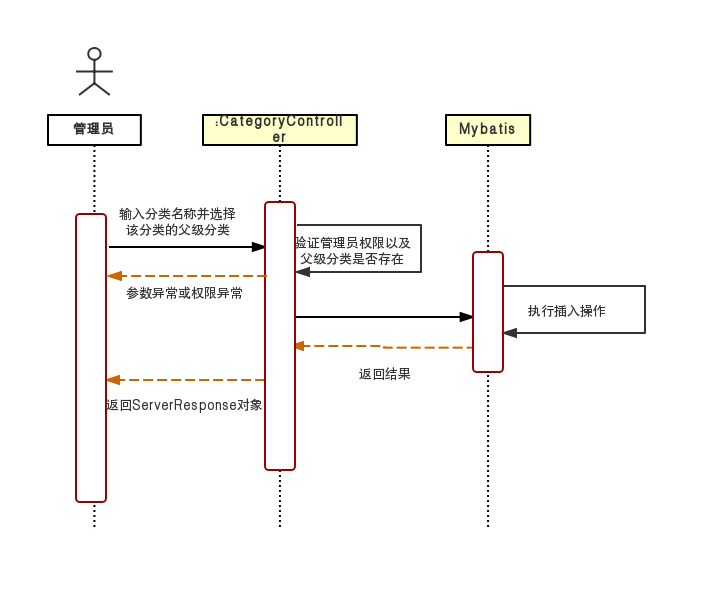


图 4.2.4.6 管理员新增分类时序图

管理员新增分类，由于商品的复杂性，分类的层级可能是多层级，在数据库设计时要从字段中体现出该分类的父级和该分类的子级，进而根据分类进行操作。

1. 分类查询模块

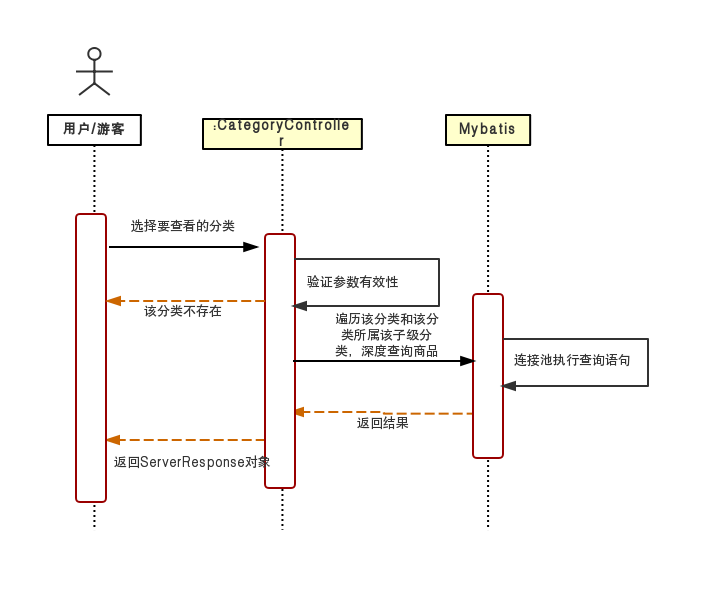
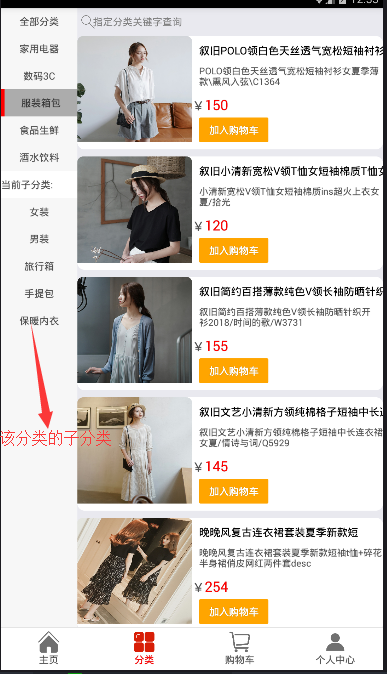


图 4.2.4.7 分类查询时序图

分类查询用于用户查询商品时动态展示该分类下的商品信息，以及动态展示分类下的子分类，子分类的下级分类，并支持关键字下的分类查询。

界面设计：

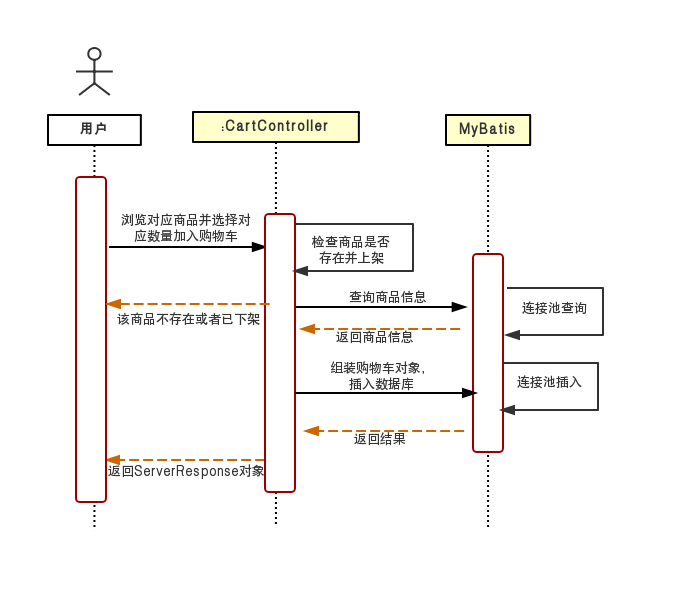


4.2.7 购物车模块

（1）加入购物车

用户浏览商品信息，可对商品选择对应的数量加入购物车，在前台购物车中增加对应的商品和数量并显示对应的购物车信息。

图 4.2.4.7 加入购物车时序图



（2）购物车操作

用户登录后可对自己的购物车内的商品进行操作，批量删除和批量结算，全选和全不选商品，根据当前购物车状态自动计算选中的商品的总价(由于购物车功能涉及金额，所以关于购物车的一切操作都放在后端处理，根据商品是否选中和对应的数量动态在后台计算当前需要支付的价格，前端仅修改状态和数量以及删除和支付)

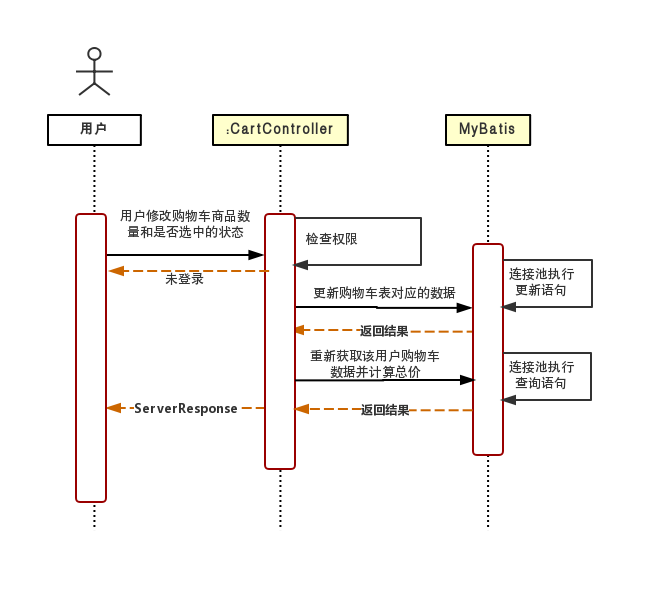


图 4.2.4.8 购物车操作时序图

界面设计：

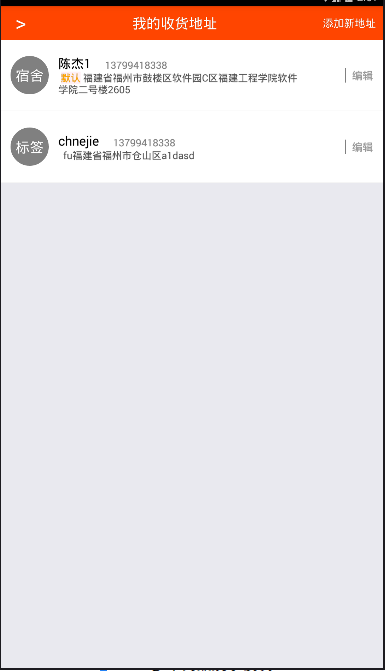


4.2.8 收货地址模块

新增收货地址

用户登录系统后在我的地址中可以管理收货地址，修改，新增以及删除，设为默认等，每次购物车结算生成订单时使用默认地址或从地址列表中选择一个设为商品的收货地址。

界面设计:



4.2.9 订单模块

用户在购物车内选择需要结算的商品和数量后跳转到结算页面进行结算时，选择对应的收货地址后，结算生成订单数据，包括该商品的总价和订单内的商品以及对应的数量，此时该笔订单状态为未付款，生成后跳转到支付宝结算页面，等待用户付款和支付宝异步回调通知，确认付款成功状态修改为已付款。生成支付宝订单信息表，记录支付宝流水账号和交易时间交易类型交易金额和商品价格等信息。

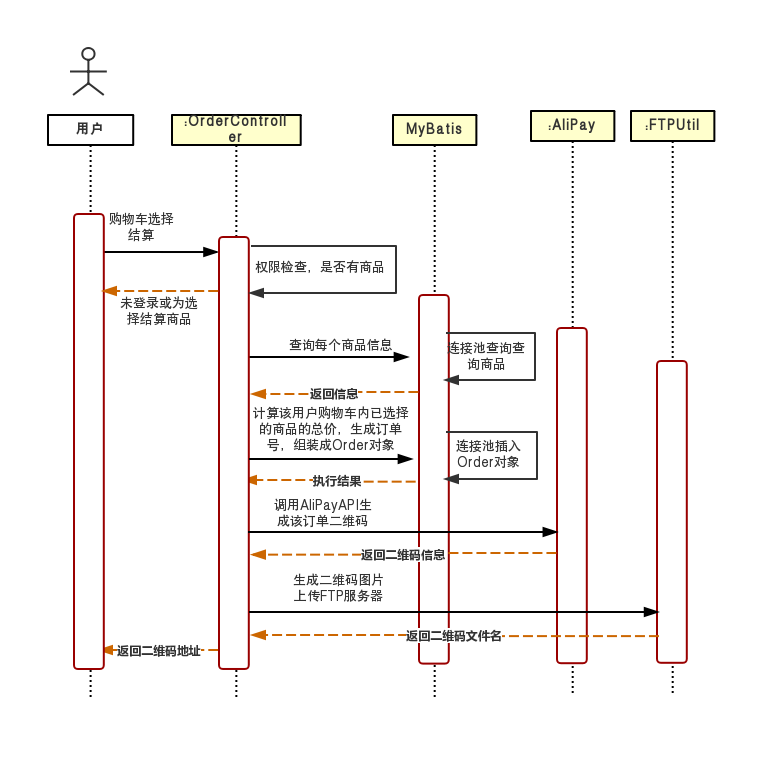


图 4.2.4.8 订单生成时序图

界面设计：

用户在个人中心界面可进入订单管理，对所属订单进行付款，收货，评论等操作。



4.2.10 评论模块

当订单生成并付款后，管理员对该订单内的商品进行发货，发货后用户可选择签收功能，对该笔订单进行已签收，签收后可对该订单内的商品进行评价操作。

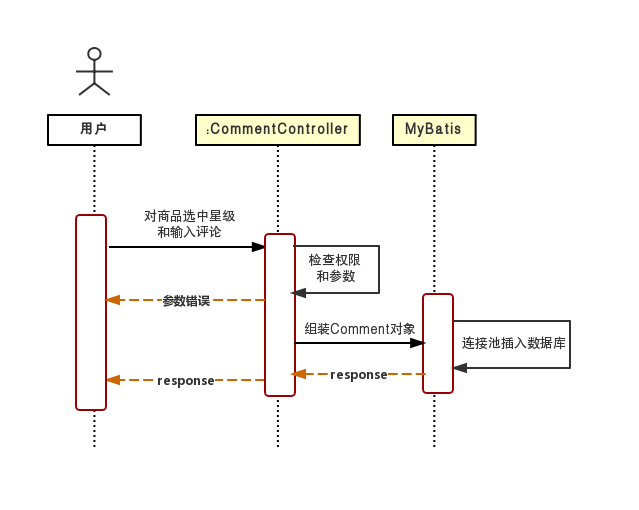


图 4.2.4.9 用户评论时序图

界面设计：

评论模块可选择对应已收货的商品进行评论打星等操作，成功后展示在对应商品的主页面。

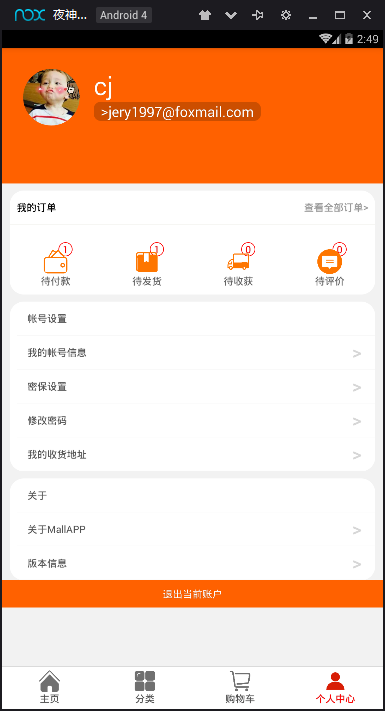


4.2.11 个人中心模块

个人中心模块主要作为一些有关用户信息操作的入口页面和用户相关的信息展示，例如地址管理，我的订单管理，我的信息等,通过该入口可以对未付款的订单进行付款，对订单进行收货，

对已收货的订单的商品进行评论打分功能。可以进入我的地址进行对地址的增删改查功能。

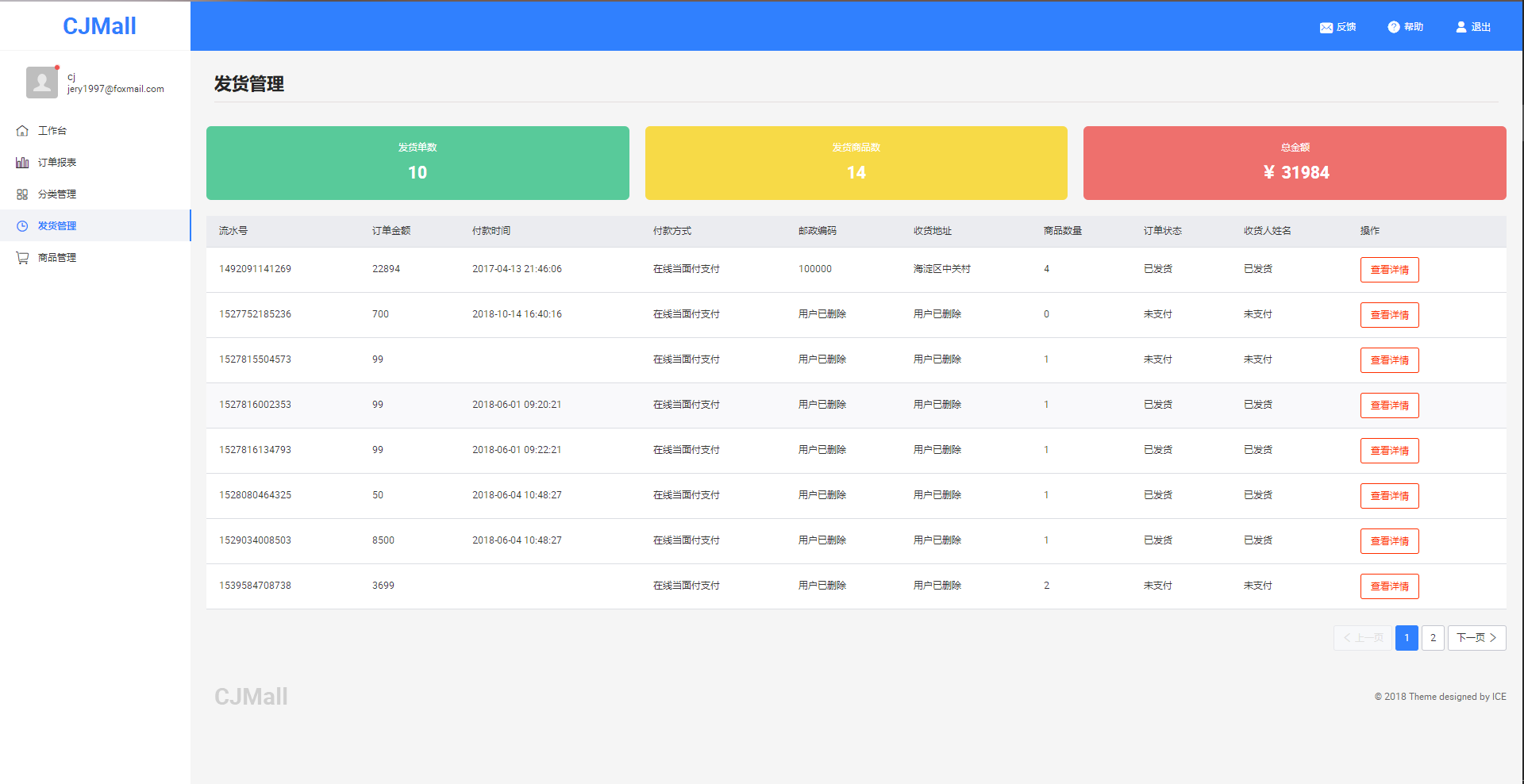
界面设计:



4.2.12 管理员管理模块

管理员模块可在后台查看系统每个时期的数据，以及收入，商品比例等，通过各种图表的形式更直观的展示出各时期的差异，以及订单，分类商品的数据分析等。

界面设计:



4.3 前端系统设计

4.3.1 “大前端”

随着近几年nodeJS的兴起，其基于chome的V8引擎的javascript运行时，V8引擎的性能远比其他脚本语言，配合npm的包管理工具，使前端也能像后端那样拥有一个属于自己的生态环境，使用模块化API将js文件导入导出，使代码更方便管理和后期维护，将依赖信息放置于package.json中，类似maven从远程仓库拉取项目依赖，真正将js这款语言脱离了浏览器，运行在操作系统上并充当服务器。

在此之前，js仅仅只是一款运行在浏览器内的script(脚本)，并没有很多类似于一款语言的特性和能力，例如：class，extend，装饰器等一款面对对象语言该有的特性和API，随着浏览器的普及和移动浏览器端的需求，用户变得越来越多，项目复杂度也越来越高，ECMAScript（定义javascript标准语法的组织）组织开始对js进行完善，浏览器也开始更新版本兼容ES5，ES6，ES7等语法特性，使得到现在js真正成为一款语言，ES6的class，extend，箭头函数等特性，目前众多浏览器已经可以识别并运行ES6语法，ES7的包装模式@，强大的异步操作API Async await语法被定义为js异步终极解决方式，抛开了以前的回调，promise等繁琐复杂的回调和嵌套。

在短短几年内，node取得了巨大的成功，在开源社区github上，node高居第二，npm下载量和模块数量也多的惊人，express，koa，egg这种的基于node下的框架，在企业界的应用也开始变得广泛，LinkedIn的移动平台已经全部从RuBy迁移到node，原有的机器数量减少了十分之一，雅虎，微软这样的大公司，也有好多应用迁移到node了，国内的阿里巴巴，腾讯，网易，新浪等公司很多线上产品也纷纷改用node开发。

4.3.2 webpack

Webpack是一个web应用程序静态打包工具，他会将项目的所有依赖和模块化代码解析打包为一个或者多个bundle文件，基于nodeJS，所有的组件化模块化代码项目都需要经过webpack等其他打包工具打包为浏览器可运行的静态文件。Webpack还提供了许多的常用功能和工具，利用websocket通信实现代码热更新，解析sass，less代码，配合babel打包ES6和ES7等高级ECMASript定义的语法并将他们转换为ES5，即浏览器端可执行的代码，还可以处理css的方言，各种浏览器内核的一些方言-webkit- -moz-的写法并自动加上对应的前缀。

4.3.3 SPA单页面应用

以往的web端页面与页面之间的状态不能共享，当页面刷新或跳转到其他页面时，页面的数据状态丢失，无法获取上一个页面的一些数据和状态，用html5提出的local-storage和session-storage虽然可以解决浏览器的存储问题，但是过多的存储也会变得难以维护，数据量多的时候命名等也是一些难题，SPA页面就相当于一个应用app一样，他有全局状态，页面切换不刷新页面，上一个页面的数据和状态不会丢失，所以开始使用spa页面开发应用，使用vue-router和react-router处理单页面应用，具体实现原理有两种：

（1）路由hash化，在浏览器地址后加#符号实现路由hash化，路由更改后页面无法刷新，通过监听window.addEventListener(“hashchange”,function(){})浏览器hash路由的变化，发生变化时动态根据配置好的路由规则，具体的路由需要展示的组件达到页面切换的效果，封装好一套url解析规则得到浏览器传递的值，且不会有用户等待浏览器加载页面的时间，更好的用户体验。

（2）路由hash化存在一定的弊端，会使锚点功能变得不可用，html中用来用户快速定位并跳转到某个部分时，使用#加dom的Id进行跳转，此时使用了hash路由，会使锚点功能不可用，所以引出了另外一种实现spa页面的方法——使用浏览器history API进行页面切换，当我们切换页面时，点击回退按钮的时候浏览器不会刷新页面，使用浏览器提供的API window .history.pushState(,, “nofreash”) 向浏览器创建新的历史记录，最后在浏览器对象创建事件监听window.addEventListener("popstate", function() {});也可以根据url规则显示对应的DOM，而且此模式对路由完全没有任何副作用，完全不影响路由的其他操作。

4.3.4 Web Component

前面跨平台需求实现已经讲解了基于React的Web Component基本实现，配合router路由管理，渲染出一套基于前端路由的实现，根据路由的匹配和路由的传值规则?name=cj，选择该路由下对应的Web Component进行显示，从而达到页面切换的效果。浏览器原生支持的原生Web Component方式，正是此类没有被所有浏览器全部支持的技术，开始对前端组件化产生积极的影响，google力推WebComponent的开发方式，然而到目前为止，也仅仅只有自家的谷歌浏览器以及欧朋和一些新版本浏览器支持部分API，结果始终不理想，既然浏览器厂商对组件化不重视，为什么大家不通过js来推动组件化开发，于是出现了很多像是vue，react，angular此类的组件化框架，浏览器端不允许跨文件加载组件，那就用基于node的webpack来分文件导入导出来组织我们的代码，最终将项目文件打包成一个或多个js文件浏览器运行就好，很好的解决了代码重用性，加快了开发效率，每个WebComponent中都有自己的逻辑和代码块，就是我们常见的浏览器内自带的音频，视频播放器等此类的标签，又叫Shadow Dom（影子Dom），他将内部的逻辑和样式代码封装了起来，很大程度上减少了代码的重复性和样式代码的冲突。

Web Component 由四个部分组成，用于封装一个常用或者需定制的元素。

Custom elements(用户自定义元素)：允许开发者在html文档中定义除去原有标签外的其他标签的dom元素类型，即自定义元素标签。

Shadow Dom（影子DOM）:一种明确的dom树结构，用来描述一个WebComponent的dom构成和层级，封装Dom树到附加元素，并将其作为一个独立性的DOM加载到文档，组件与外部隔离，Javascript作用域和样式互不影响，完全与外部文档隔离，这样就可以使代码块被局部私有化以及样式局部化，避免和全局或用户自定义的样式类名和js代码的命名等起不必要的冲突。

HTML templates（HTML模板）：开发者可以申明自己标记的dom片段，他们会被解析为html，加载时并不会被看到也不会被加载进dom树种构建，并在运行时被实例化和展示出来,在模版内也支持加载js和css文件，从而不和外界代码和样式起冲突。

<template id=”hello-word” >

<script src=”./../XX/XX.js”></script>

<link rel="stylesheet" href=”./../xx/xx.css”/>

<p>Hello Word!!!</p>

</template>

HTML import （HTML引用）：可以允许开发者在其他文件中定义和创建新的模版，在代码中可以引入该模版，帮助开发者更好的组织代码结构和逻辑。

使用标签导入模版<link rel="import" href="./../xxxx.html">后就能使用模版了。

<hello-word></hello-word>

4.3.5 对前端生态的总结

总体而言，前端的生态主要体现在三个层次，就像是软件需要很好的运行也需要有良好的硬件支持一样。

1. 浏览器厂商的支持

ECMAScript组织的定义标准化js语法，需要有良好的前瞻性，浏览器厂商对该组织语法和功能API的实现，并在下个版本加入应有的功能，浏览器厂商一般都能很快的支持最新的语法和提出的草案。

1. 用户客户端的版本

目前很多开发上的障碍都在用户的版本这边受到很大限制，例如，国内还是拥有许多政府企业还在用低版本IE浏览器，自以为自己能因为强大的用户群和windows系统自带的原因就可以不顾ECMAScript的标准，弄自己独有的js语法和css写法，直到目前，下版本的window浏览器也将开始使用webkit内核，微软开始宣布放弃IE和Edge浏览器，调查结果也显示很多用户最不会更新的软件就是浏览器，好像觉得更不更新没什么两样，从而导致浏览器端无法使用许多新特性，进而开发者为了浏览器端为了能在不影响自己开发的前提下产生了许多插件，浏览器禁止跨域文件加载，无法对文件进行IO读写，有各种方言，产生了基于node语言的webpack来处理一些前端生产上自动化的问题；浏览器对Web Component思想的实现不重视，于是产生了三大组件化开发框架；用户不更新客户端还在用只支持ES5语法的浏览器，于是产生了babel来编译开发者书写的ES6，ES7等高版本语法特性，将他转换为ES5可执行的代码；逻辑代码为什么全要后端去写，即增加后端的开发时间，又浪费服务器性能，node的高性能异步非阻塞IO这么适合做这些事情 ，于是有了基于node的（ssr）服务端渲染。

1. 开发者

开发者的思想总是跑在最前的，ECMAScript组织的语法许多也都是开发者通过提议后一步一步草案后最终加入并被浏览器所支持，三者的更新迭代的步伐必须统一，不然就会促使一些第三方工具的产生，最终解决开发上的问题。

4.4 数据库设计

4.4.1 系统表结构设计

结合系统的业务逻辑以及该系统的需求设计出一下表结构：

表 4‑1用户表（mall\_user）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | username | VARCHAR | 50 |  |  | 否 | 用户名 |
| 3 | password | VARCHAR | 50 |  |  | 否 | Md5加密后的密码 |
| 4 | m\_email | VARCHAR | 50 |  |  | 否 | 邮箱 |
| 5 | m\_phone | VARCHAR | 20 |  |  | 否 | 电话 |
| 6 | question | VARCHAR | 100 |  |  | 否 | 找回密码问题 |
| 7 | answer | VARCHAR | 100 |  |  | 否 | 找回密码答案 |
| 8 | role | int | 1 |  |  | 否 | 角色权限 |
| 9 | Create-time | DATETIME |  |  |  | 否 | 创建时间 |
| 10 | Update-time | DATETIME |  |  |  | 否 | 修改时间 |

表 4‑2商品表（mall\_product）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | category\_id | Int | 11 |  |  | 否 | 分类表外键 |
| 3 | name | VARCHAR | 100 |  |  | 否 | 商品名称 |
| 4 | subtitle | VARCHAR | 200 |  |  | 否 | 商品副标题 |
| 5 | main\_image | VARCHAR | 100 |  |  | 否 | 产品主图 |
| 6 | sub\_images | VARCHAR | 500 |  |  | 否 | 图片副图地址 |
| 7 | detail | LONGTEXT |  |  |  | 否 | 富文本详情 |
| 8 | price | decimal | (20,2) |  |  | 否 | 价格,单位-元 |
| 9 | stock | int | 11 |  |  | 否 | 库存数量 |
| 10 | status | int | 1 |  |  | 否 | 商品状态 |
| 11 | create\_time | DATETIME |  |  |  | 否 | 创建时间 |
| 12 | update\_time | DATETIME |  |  |  | 否 | 更新时间 |

表 4‑3分类表（mall\_category）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | parent\_id | int | 11 |  |  | 否 | 父类id |
| 3 | name | VARCHAR | 20 |  |  | 否 | 类别名称 |
| 4 | status | INT | 1 |  |  | 否 | 类别状态1 |
| 5 | create\_time | DATETIME |  |  |  | 否 | 创建时间 |
| 6 | update\_time | DATETIME |  |  |  | 否 | 更新时间 |

表 4‑4评论表（mall\_comment）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | productId | int | 11 |  |  | 否 | 商品外键 |
| 3 | userId | int | 11 |  |  | 否 | 用户外键 |
| 4 | username | VARCHAR | 100 |  |  | 否 | 评论的用户名 |
| 5 | cStarts | int | 1 |  |  | 否 | 星级 |
| 6 | cContent | VARCHAR | 500 |  |  | 否 | 内容 |
| 7 | cTime | DATETIME |  |  |  | 否 | 评论时间 |

表 4‑5购物车表（mall\_cart）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | user\_id | int | 11 |  |  | 否 | 用户ID外键 |
| 3 | product\_id | int | 11 |  |  | 否 | 商品id外键 |
| 4 | quantity | int | 11 |  |  | 否 | 数量 |
| 5 | checked | int | 1 |  |  | 否 | 是否选择 |
| 6 | create\_time | DATETIME |  |  |  | 否 | 创建时间 |
| 7 | update\_time | DATETIME |  |  |  | 否 | 更新时间 |

表 4‑6订单表（mall\_order）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | order\_no | BIGINT | 20 |  |  | 否 | 订单号 |
| 3 | user\_id | int | 11 |  |  | 否 | 用户id |
| 4 | shipping\_id | int | 11 |  |  | 否 | 地址ID |
| 5 | payment | decimal | (20,2) |  |  | 否 | 实际付款金额 |
| 6 | payment\_type | int | 1 |  |  | 否 | 支付类型 |
| 7 | postage | int | 10 |  |  | 否 | 运费, |
| 8 | status | int | 1 |  |  | 否 | 订单状态 |
| 9 | payment\_time | DATETIME |  |  |  | 是 | 支付时间 |
| 10 | send\_time | DATETIME |  |  |  | 是 | 发货时间 |
| 11 | end\_time | DATETIME |  |  |  | 是 | 交易完成时间 |
| 12 | close\_time | DATETIME |  |  |  | 是 | 交易关闭时间 |
| 13 | create\_time | DATETIME |  |  |  | 否 | 创建时间 |
| 14 | update\_time | DATETIME |  |  |  | 否 | 更新时间 |

表 4‑7订单快照表（mall\_order\_item）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | user\_id | int | 11 |  |  | 否 | 用户ID外键 |
| 3 | order\_no | BIGINT | 20 |  |  | 否 | 订单号 |
| 4 | product\_id | int | 11 |  |  | 否 | 商品ID |
| 5 | product\_name | VARCHAR | 100 |  |  | 否 | 商品名称 |
| 6 | product\_image | VARCHAR | 500 |  |  | 否 | 商品图片地址 |
| 7 | current\_unit\_price | decimal | (20,2) |  |  | 否 | 购买时价格 |
| 8 | quantity | int | 10 |  |  | 否 | 商品数量 |
| 9 | total\_price | decimal | (20,2) |  |  | 否 | 商品总价 |
| 10 | create\_time | DATETIME |  |  |  | 否 | 创建时间 |
| 11 | update\_time | DATETIME |  |  |  |  | 更新时间 |

表 4‑8支付信息表（mall\_pay\_info）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | user\_id | Int | 11 |  |  | 否 | 用户id |
| 3 | order\_no | BIGINT | 20 |  |  | 否 | 订单号 |
| 4 | pay\_platform | Int | 10 |  |  | 否 | 支付平台 |
| 5 | platform\_number | VARCHAR | 200 |  |  | 否 | 支付宝流水号 |
| 6 | platform\_status | VARCHAR | 20 |  |  | 否 | 支付宝支付状态 |
| 7 | create\_time | DATETIME |  |  |  | 否 | 创建时间 |
| 8 | update\_time | DATETIME |  |  |  |  | 更新时间 |

表 4‑9收货地址表（mall\_shipping）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **标识** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 是 | 否 | 主键自增长 |
| 2 | user\_id | int | 11 |  |  | 否 | 用户id |
| 3 | receiver\_name | VARCHAR | 50 |  |  | 否 | 收货姓名 |
| 4 | receiver\_phone | VARCHAR | 20 |  |  | 否 | 收货固定电话 |
| 5 | receiver\_city | VARCHAR | 100 |  |  | 否 | 省份 |
| 6 | receiver\_address | VARCHAR | 200 |  |  | 否 | 详细地址 |
| 7 | receiver\_zip | VARCHAR | 6 |  |  | 否 | 邮编 |
| 9 | create\_time | DATETIME |  |  |  | 否 | 创建时间 |
| 10 | update\_time | DATETIME |  |  |  |  | 更新时间 |

# 5.系统实现

本系统后台逻辑业务主要使用JAVAEE开发，使用MAVEN管理版本依赖；

后台数据分析使用Nodejs下阿里巴巴开源后台框架egg开发，由于数据分析语句涉及大量表连接和频繁的数据库查询，基于node语言特性的非阻塞异步IO来处理多条数据库查询时的等待和延时，很大程度上提高了查询的性能和并发量；

5.1 业务逻辑核心工具

5.1.1 token缓存

用户忘记密码时使用密保问题找回密码时，密保验证成功后，如何防止其他用户利用此API进行恶意修改密码，保障系统的安全性，如何判断该用户是否已经验证密保问题，比较好的解决方案就是生成token缓存来确定用户身份，用户验证成功后生成给用户一个有限时间的键值对

按设计具体说明每个功能的实现，关键功能应提供主要或核心代码。

测试方法、用例及测试结果（可截图）。代码（宋体，五号字）严禁截图。

# 总结

主要是对整个工作做回顾总结，如简要总结开发过程中遇到什么问题，如何解决，用什么技术、什么方法，实现什么功能，效果怎样等等。

最后还可围绕技术实现总结几点得失，如自己的技术、业务能力怎样？通过毕业设计是否有提高？分析问题能力、解决问题能力是否得到提升等等，多方面分析。

# 致谢

谢辞应以简短的文字对在课题研究和论文撰写过程中曾直接给予帮助的人员（例如指导教师、答疑教师及其他人员）表示自己的谢意，这不仅是一种礼貌，也是对他人劳动的尊重，是治学者应有的思想作风。有关自己的体会收获尽量少讲或不讲，如有必要可移到总结里介绍。

# 参考文献

[序号]主要责任者.文献题名[文献类型标识].出版地：出版者，出版年.起止页码（任选）.

[1]刘国钧，陈绍业，王凤翥.图书馆目录[M].北京：高等教育出版社，1957.15-18.

[2]辛希孟，信息技术与信息服务国际研讨会论文集：A集[C].北京：中国社会科学出版社，1994.

[序号]主要责任者.文献题名[J].刊名，年，卷（期）：起止页码.

[3]何龄修.读顾城《南明史》[J].中国史研究，1998，（3）：167-173.

[4]金显贺，王昌长，王忠东，等.一种用于在线检测局部放电的数字滤波技术[J].清华大学学报（自然科学版），1993，33（4）：62-67.

（注意：宋体，五号字，行距23磅，不少于12篇，英文文献不少于两篇，教材等书籍不要多于3篇）