# 《Shop系统Node》说明文档

### 1、学号：

### 2、姓名：

* 技术框架 Vue+Express+Mysql

前端部分使用了当下流行的Vue框架，而UI框架部分使用了移动端的VantUI，能实现自适应布局，非常符合当下人们的需求。

* 界面功能

登录与注册界面

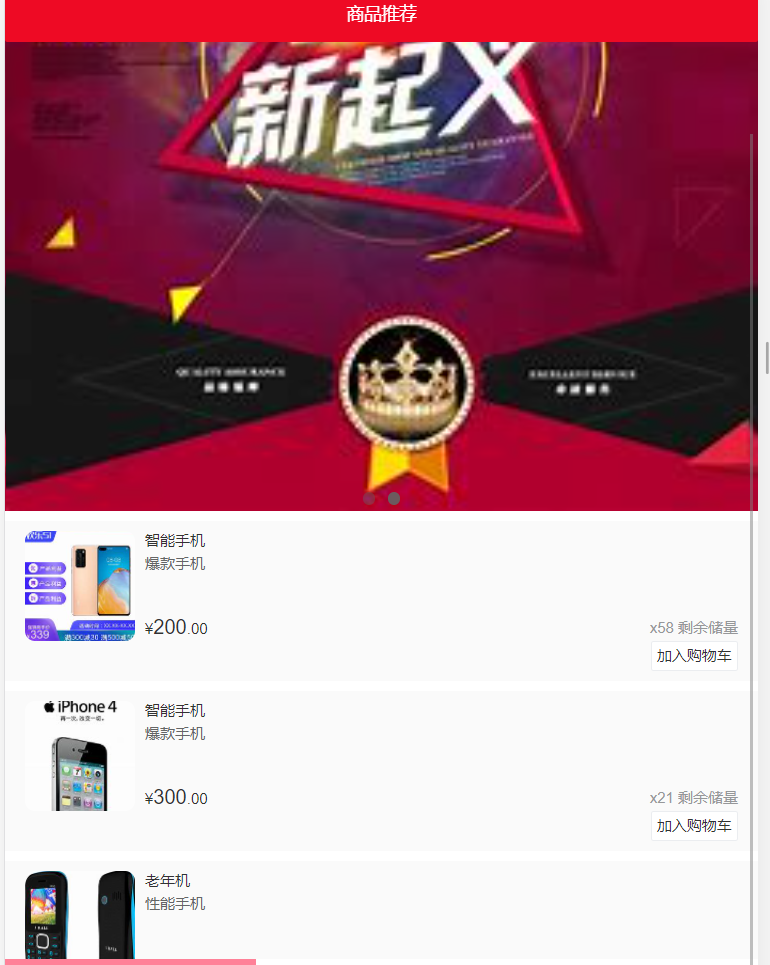




关键代码

<template>  
<div class="login">  
 <van-tabs type="card">  
 <van-tab title="DENGLU❤">  
 <van-form @submit="onSave">  
 <van-field  
 v-model="userAccount"  
 name="user\_account"  
 label="用户名"  
 placeholder="用户名"  
 :rules="[{ required: true, message: '请填写用户名' }]"  
 />  
 <van-field  
 v-model="password"  
 type="password"  
 name="password"  
 label="密码"  
 placeholder="密码"  
 :rules="[{ required: true, message: '请填写密码' }]"  
 />  
 <div style="margin: 16px;">  
 <van-button round block type="info" native-type="submit">提交</van-button>  
 </div>  
 </van-form>  
 </van-tab>  
 <van-tab title="ZHUCE😀">  
 <van-form @submit="onSubmit">  
 <van-field  
 v-model="username"  
 name="username"  
 label="用户名"  
 placeholder="用户名"  
 :rules="[{ required: true, message: '请填写用户名' }]"  
 />  
 <van-field  
 v-model="userAccount"  
 name="userAccount"  
 label="账号名"  
 placeholder="账号名"  
 :rules="[{ required: true, message: '请填写账号' }]"  
 />  
 <van-field  
 v-model="telephone"  
 type="text"  
 name="phone"  
 label="手机号"  
 placeholder="手机号"  
 :rules="[{ required: true, message: '请填写手机号' }]"  
 />  
 <van-field  
 v-model="password"  
 type="password"  
 name="password"  
 label="密码"  
 placeholder="密码"  
 :rules="[{ required: true, message: '请填写密码' }]"  
 />  
 <van-field  
 v-model="confirmPass"  
 type="password"  
 name="confirmPassword"  
 label="确认密码"  
 placeholder="再输入一次你的密码哦"  
 :rules="[{ required: true, message: '请确认',validator:Validator }]"  
 />  
 <van-field  
 v-model="address"  
 type="text"  
 name="address"  
 label="输入地址"  
 placeholder="请输入你的地址"  
 :rules="[{ required: true, message: '请输入你的地址',validator:Validator }]"  
 />  
  
 <div style="margin: 16px;">  
 <van-button round block type="info" native-type="submit">提交</van-button>  
 </div>  
  
  
 </van-form>  
 </van-tab>  
 </van-tabs>  
 </div>  
  
  
</template>  
  
<script>  
 import {areaList} from "@vant/area-data";  
  
 export default {  
 name: 'login',  
 data() {  
 return {  
 areaList: areaList,  
 searchResult: [],  
 username: '',  
 password: '',  
 confirmPass: '',  
 telephone: '',  
 address: "",  
 userAccount: ''  
 };  
 },  
 methods: {  
 async onSave(e) {  
 let params = {  
 user\_account: e.user\_account,  
 password: e.password}  
 console.log(params)  
  
 const res = await this.$http.postLogin(params);  
 console.log(res);  
 if (res.ret === 0) {  
 localStorage.setItem('token', res.token);  
 }  
 await this.$router.push({path:"/main"});  
  
 console.log(e);  
 },  
 async onSubmit(e) {  
 if (!e.address) {  
  
 }  
 let user = {  
 username: e.username,  
 user\_account: e.userAccount,  
 password: e.password,  
 phone: e.phone,  
 receiveAddress: e.address  
 }  
 await this.$http.postRegister(user);  
 // await this.$http.getAllProducts();  
 // console.log('submit', values);  
 },  
 areaclick() {  
 console.log(1)  
 },  
 areaInfo(values) {  
 console.log(values);  
 },  
 Validator() {  
 return this.password === this.confirmPass;  
 },  
 onChangeDetail(val) {  
 if (val) {  
 this.searchResult = [  
 {  
 },  
 ];  
 } else {  
 this.searchResult = [];  
 }  
 },  
 },  
 };  
</script>  
  
<style scoped>  
  
</style>

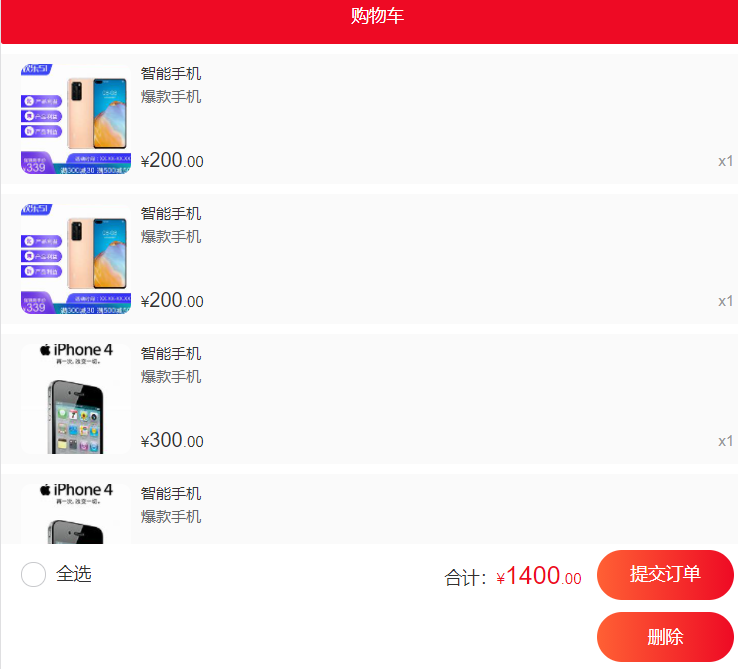
商品页面



关键代码

<template>  
<div>  
 <van-sticky>  
 <van-button type="danger" style="width: 100%">商品推荐</van-button>  
 </van-sticky>  
 <swiper>  
 <swiper-item v-for="(item, index) in banners" :key="index">  
 <img :src="item.img" alt=""/>  
 </swiper-item>  
 </swiper>  
  
 <van-card  
 :num="item.stock + ' 剩余储量'"  
 :price="item.price"  
 :desc="item.product\_desc"  
 :title="item.product\_name"  
 :thumb="item.image"  
 v-for="(item, index) in goods"  
 :key="index"  
 >  
 <template #tags>  
 <van-tag  
 plain  
 type="danger"  
 v-for="(item, index) in item.keyWord"  
 :key="index"  
 >{{ item }}</van-tag  
 >  
</template>  
  
<template #footer>  
<van-button size="mini" @click="intoCart(item.product\_id, item.price,index)"  
 >加入购物车</van-button  
 >  
</template>  
</van-card>  
</div>  
</template>  
<script>  
 import { Swiper, SwiperItem } from "./Swiper/useSwiper";  
 import { get } from "../utils/http"  
 import {Notify} from "vant";  
  
 export default {  
 components: {  
 Swiper,  
 SwiperItem,  
 },  
 data() {  
 return {  
 active: 0,  
 banners: [{img: 'https://tse2-mm.cn.bing.net/th/id/OIP.wzzaurqikEdc67s1MQ0wegHaGo?w=188&h=180&c=7&o=5&dpr=1.25&pid=1.7'},{img: 'https://tse2-mm.cn.bing.net/th/id/OIP.wzzaurqikEdc67s1MQ0wegHaGo?w=188&h=180&c=7&o=5&dpr=1.25&pid=1.7'}],  
 goods: [],  
 };  
 },  
 async created() {  
 let allProducts = await this.$http.getAllProducts();  
  
 this.goods = allProducts.data;  
 console.log(this.goods)  
 },  
 props: {  
 list: {  
 default() {  
 return [  
 ];  
 },  
 },  
 },  
 methods: {  
 async intoCart(id, price,index) {  
 let res = await this.$http.addToMyCart({amount: 1, productId: id, totalPrice: 1 \* price, price: price});  
 if (res.ret === 0) {  
 Notify({type: 'success', message:'加入购物车成功'});  
 }  
 this.goods[index].stock -= 1;  
  
 },  
 },  
 };  
</script>  
<style scoped>  
 .todo-container li {  
 border: solid 1px white;  
 background-color: beige;  
 margin-top: 1em;  
 padding: 0.2em;  
 }  
 .main {  
 font-size: 0.7em;  
 }  
</style>

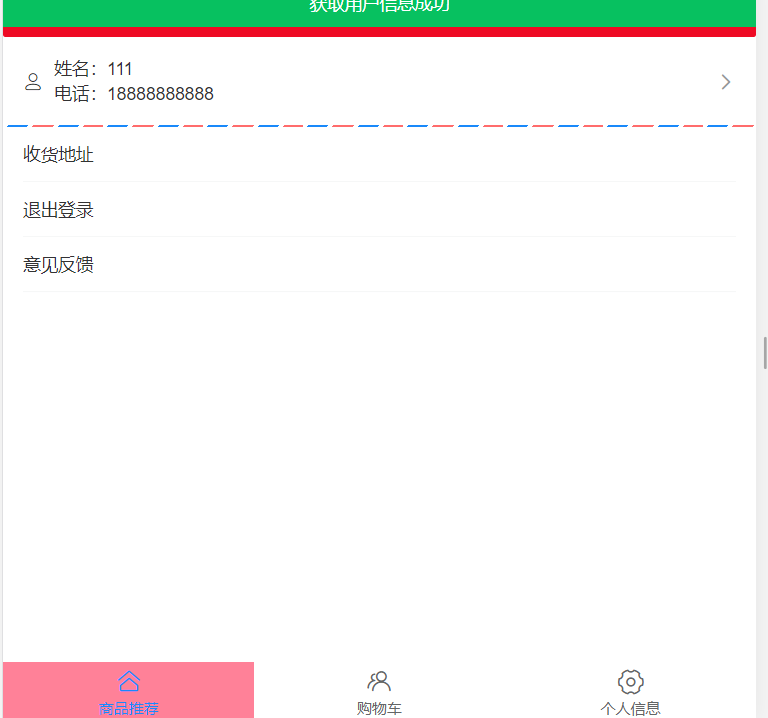
购物车界面



关键代码

<template>  
<div class="cart">  
 <van-sticky>  
 <van-button type="danger" style="width: 100%"> 购物车 </van-button>  
 </van-sticky>  
  
 <van-card  
 :num="item.amount"  
 :price="item.price"  
 :desc="item.product\_desc"  
 :title="item.product\_name"  
 :thumb="item.image"  
 v-for="(item, index) in cartGoods"  
 :key="index"  
 >  
 <template #tags>  
 <van-tag  
 plain  
 type="danger"  
 v-for="(item, index) in item.keyWord"  
 :key="index"  
 >{{ item }}</van-tag  
 >  
</template>  
  
</van-card>  
<van-submit-bar  
 :price="totalPrice"  
 button-text="提交订单"  
 @submit="onSubmit"  
 class="van-submit-bar"  
 >  
  
 <van-checkbox v-model="checked">全选</van-checkbox>  
</van-submit-bar>  
  
<van-submit-bar  
 button-text="删除"  
 @submit="onDelete"  
 class="van-submit-bar1"  
 >  
  
</van-submit-bar>  
  
  
</div>  
</template>  
<script>  
 import { get } from "../utils/http";  
 export default {  
 props: {},  
 data() {  
 return {  
 active: 0,  
 show: false,  
 value: null,  
 checked: false,  
 cartGoods: [],  
 totalPrice: 0  
 };  
 },  
 methods: {  
 onSubmit() {},  
 async onDelete() {  
 if (this.checked) {  
 let size = this.cartGoods.length;  
 let cartCopy = this.cartGoods;  
 this.cartGoods = [];  
 this.totalPrice = 0;  
 for (let i = 0; i < size; i++) {  
 await this.$http.deleteMyCart({cartId:cartCopy[i].cart\_id});  
 }  
 }  
 }  
 },  
 async created() {  
 // let cartArr = JSON.parse(sessionStorage.getItem("cartGoods")) || [];  
 let cartArr = await this.$http.getMyCart();  
 this.cartGoods = cartArr.data;  
 console.log(this.cartGoods.length)  
 for (let i = 0; i < this.cartGoods.length; i++) {  
 this.totalPrice += ~~this.cartGoods[i].price \* 100;  
 }  
 },  
 };  
  
</script>  
<style scoped>  
 .van-submit-bar {  
 bottom: 100px;  
  
 }  
 .van-submit-bar1 {  
 bottom: 50px;  
 }  
 .cart {  
 padding-bottom: 100px;  
 }  
</style>

我的界面



关键代码

<template>  
<div>  
 <van-sticky>  
 <van-button type="danger" style="width: 100%">我的信息</van-button>  
 </van-sticky>  
  
 <van-contact-card  
 type="edit"  
 :name="user.username"  
 :tel="user.phone"  
 @click="showMore"  
 />  
 <van-contact-list  
 v-model="chosenContactId"  
 :list="list"  
 default-tag-text="默认"  
 v-if="isEdit"  
 @Add="onAdd"  
 class="van-contact-list"  
 />  
  
 <van-list :finished="finished">  
 <van-cell v-for="(item,index) in listName" :key="item" @click="test(index)" :title="item"/>  
 </van-list>  
  
  
 </div>  
</template>  
<script>  
 import {Notify} from "vant";  
  
 export default {  
 props: {},  
 data() {  
 return {  
 isEdit: false,  
 active: 0,  
 loading: false,  
 finished: false,  
 listName: ["收货地址", "退出登录", "意见反馈"],  
 chosenContactId: "2",  
 user: {},  
 list: [  
  
 ],  
 };  
 },  
 computed: {  
 showWho() {  
 return {  
 name: "张三",  
 tel: "13000000000",  
 };  
 },  
 },  
 methods: {  
 showMore() {  
 this.isEdit = !this.isEdit;  
 },  
 async logout() {  
 console.log('我在这');  
 if (localStorage.getItem('token')) {  
 localStorage.removeItem('token');  
 const res = await this.$http.deleteLogout();  
 console.log(res);  
 await this.$router.push('/');  
 }  
 },  
 test(t) {  
 switch (t) {  
 case 0:  
 Notify({type: 'success', message: '点击上方'})  
 break;  
 case 1:  
 this.logout();  
 break;  
 case 2:  
 Notify({type: 'warning', message: '暂未开通该功能'})  
 break;  
 default:  
 break;  
 }  
 console.log(t)  
 },  
 onAdd() {  
  
 }  
  
 },  
 async created() {  
 let userInfo = await this.$http.getUserinfo();  
 this.user = userInfo.data[0];  
 const res = await this.$http.getMyAddress();  
 const user = await this.$http.getUserinfo();  
 console.log(user)  
 let i = res.data.length;  
 for (let j = 1; j<= i; j++) {  
 console.log(res.data)  
 if (j == 1) {  
 this.list.push({id: j - 1, name: user.data[0].username, tel: res.data[j - 1].address, isDefault: true})  
 } else {  
 this.list.push({id: j - 1, name: user.data[0].username, tel: res.data[j - 1].address})  
 }  
 }  
  
 }  
 };  
</script>  
<style scoped>  
 .todo-container li {  
 border: solid 1px;  
 background-color: beige;  
 margin-top: 1em;  
 border-radius: 2em;  
 padding: 1em;  
 }  
  
 .van-contact-list >>> .van-contact-list\_\_bottom {  
 display: none;  
 }  
  
 .van-contact-list {  
 padding-bottom: 20px;  
 }  
</style>

登录之后状态的保留

本项目没有用传统的cookie-session机制，而是用了jwt，即jsonwebtoken技术。

Json web token (JWT), 是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准（[(RFC 7519](https://link.jianshu.com?t=https://tools.ietf.org/html/rfc7519)).该token被设计为紧凑且安全的，特别适用于分布式站点的单点登录（SSO）场景。JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该token也可直接被用于认证，也可被加密。

因此用户登陆之后，服务器需要将用户的信息用一定的签名算法签名之后将用户的信息返还给客户端，而客户端每次请求都需要将生成的token带上，用于验证是否已经登陆。

关键代码

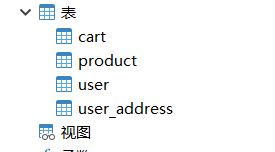
const login = (req, res, next) => {  
 let user = req.body;  
 userService.getUserByNameAndPassword(user, (error, data, fields) => {  
 if (error) {  
 console.log(error);  
 next();  
 } else {  
 if (data.length !== 1) {  
 next(new Error('用户名密码不正确'));  
 } else {  
  
 hash.update(user.password);  
  
 const {user\_account} = data[0];  
  
 console.log(user\_account)  
 let token = "Bearer " + jwt.sign({user\_account}, secretKey);  
  
 res.send({  
 msg: '登陆成功',  
 token,  
 ret: 0  
 })  
 }  
 }  
 })  
}

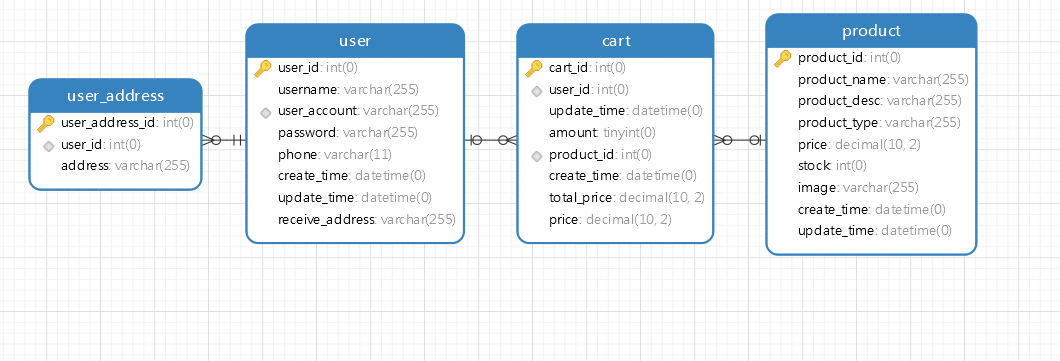
instance.interceptors.request.use(config => {  
  
if (localStorage.getItem("token") ) {  
console.log(this)  
config.headers['Authorization'] = localStorage.getItem("token");  
}  
  
if (config.method === 'get') {  
config.params = {  
...config.params,  
}  
}  
  
return config  
}, error => {  
return Promise.reject(error)  
})

后端部分

后端采用Express+Mysql的架构进行运行。其中Express 是一种保持最低程度规模的灵活 Node.js Web 应用程序框架，为 Web 和移动应用程序提供一组强大的功能。而Mysql 是一款安全、跨平台、高效的，并与 PHP、Java 等主流编程语言紧密结合的数据库系统。

数据库设计





数据库连接池部分

//加载mysql数据库连接库  
let mysql=require("mysql2");  
  
//调用mysql.createPool方法创建商品连接池  
let pool = mysql.createPool({  
 connectionLimit : 10,  
 multipleStatements:true,  
 host : '127.0.0.1',  
 user : 'root',  
 password : '123456',  
 database : 'shop'  
});  
  
  
pool.on('acquire', function (connection) {  
 console.log('Connection %d acquired', connection.threadId);  
});  
pool.on('connection', function (connection) {  
 connection.query('SET SESSION auto\_increment\_increment=1')  
});  
pool.on('enqueue', function () {  
 console.log('Waiting for available connection slot');  
});  
pool.on('release', function (connection) {  
 console.log('Connection %d released', connection.threadId);  
});  
  
module.exports=pool;

分层：DAO层、业务层、控制层、，DAO层用于操作数据库，业务层专注于业务的处理，控制层用于控制来自客户请求的分发。

DAO层关键代码

//使用连接池  
let pool=require('../dao/pool');  
//密码采用sha256加密  
const crypto = require('crypto');  
  
  
//根据用户名密码获取用户信息，供登录使用  
function getUserByNameAndPassword(user,callback){  
 const hash = crypto.createHash('sha256');  
 hash.update(user.password)  
 pool.query("select \* from user where user\_account = ? and password = ?",  
 [user.user\_account,hash.digest("hex")],  
 callback  
 );  
}  
  
//添加用户信息，供注册使用  
function addUser(user,callback){  
 const hash = crypto.createHash('sha256');  
  
 hash.update(user.password);  
 pool.query("insert into user(username,user\_account,password,phone,receive\_address) values(?,?,?,?,?)",  
 [user.username,user.user\_account,hash.digest('hex'), user.phone,user.receiveAddress],  
 callback  
 );  
}  
  
//获取用户信息,用于从token中获取用户信息  
function getUser(userAccount, callback) {  
 pool.query("select \* from user where user\_account = ?",  
 [userAccount],  
 callback  
 );  
}  
  
//注销用户  
function deleteUser(userAccount, callback) {  
 pool.query("delete from user where user\_account = ? ", [userAccount],function (err, data, fields) {  
 callback(err, data, fields);  
 })  
}  
  
//添加收货地址  
function addAddress(user, callback) {  
 pool.query(  
 'insert into user\_address(user\_id, address) values (?,?)',  
 [user.userId, user.address],  
 callback  
 )  
}  
//删除收货地址  
function deleteAddress(user, callback) {  
 pool.query(  
 'delete from user\_address where user\_id = ? and address = ? ',  
 [user.userId, user.address],  
 callback  
 )  
}  
  
function getAddress(user, callback) {  
 pool.query(  
 'select \* from user\_address where user\_id = ? and address = ?',  
 [user.userId, user.address],  
 callback  
 )  
}  
  
function getMyAddress(userId, callback) {  
 pool.query(  
 `select \* from user\_address where user\_id = ?`,  
 [userId],  
 callback  
 )  
}  
  
  
//指明可供其它模块使用的模块  
module.exports={  
 getUserByNameAndPassword,  
 addUser,  
 deleteUser,  
 getUser,  
 addAddress,  
 deleteAddress,  
 getAddress,  
 getMyAddress  
};

业务层关键代码

// let express = require('express');  
let userService = require('../services/userservice');  
const jwt = require('jsonwebtoken');  
const secretKey = '123456';  
const crypto = require('crypto');  
let hash = crypto.createHash('sha256');  
  
  
//用户注册，需要判断账号是否被注册  
const register = (req, res, next) => {  
  
 let user = req.body;  
 // test.getUser()?.addAddress()  
 // console.log(user);  
 userService.getUser(user.user\_account, (error, data, fields) => {  
 console.log(data)  
 if (data.length === 1) {  
 res.send({ret: 1, message: '该账户已经注册'});  
 } else {  
 //添加账户  
 userService.addUser(user, function (error, data, fields) {  
 if (error) {  
 next(error);  
 } else {  
 //获取账户ID  
 userService.getUserByNameAndPassword(user, (error, data, fields) => {  
 if (error) {  
 next(error);  
 } else {  
 console.log(data[0].user\_id)  
 //添加收货地址  
 userService.addAddress({  
 userId: data[0].user\_id,  
 address: user.receiveAddress  
 }, (error, data, fields) => {  
 if (error) {  
 next(error);  
 } else {  
 res.send({ret: 0, message: '注册成功'});  
 }  
 })  
 }  
 })  
  
 }  
 });  
 }  
 })  
  
  
}  
  
//用户登录  
const login = (req, res, next) => {  
 let user = req.body;  
 userService.getUserByNameAndPassword(user, (error, data, fields) => {  
 if (error) {  
 console.log(error);  
 next();  
 } else {  
 if (data.length !== 1) {  
 next(new Error('用户名密码不正确'));  
 } else {  
  
 hash.update(user.password);  
  
 const {user\_account} = data[0];  
  
 console.log(user\_account)  
 let token = "Bearer " + jwt.sign({user\_account}, secretKey);  
  
 res.send({  
 msg: '登陆成功',  
 token,  
 ret: 0  
 })  
 }  
 }  
 })  
}  
  
const logout = (req, res) => {  
 res.json({ret: 0, message: '退出成功'});  
}  
  
//获取用户信息  
const userinfo = (req, res, next) => {  
  
 let auth = jwt.decode(req.headers['authorization'].split(' ')[1]);  
  
 userService.getUser(auth['user\_account'], (error, data, details) => {  
 if (error) {  
 next(new Error(error));  
 } else {  
 res.json({ret: 0, message: '获取用户信息成功', data: data});  
 }  
 })  
}  
  
const addAddress = (req, res, next) => {  
 let user = req.body;  
 userService.addAddress(user, (error, data, details) => {  
 if (error) {  
 next(new Error(error));  
 } else {  
 res.send({ret: 0, message: '新增收货成功'})  
 }  
 })  
}  
  
const getMyAddress = (req, res, next) => {  
 let t = req.body;  
 let auth = req.headers['authorization'];  
 let account = jwt.decode(auth.split(' ')[1])['user\_account'];  
 userService.getUser(account, (error, data, fields) => {  
 if (data.length === 0) {  
 res.send({ret: 1, message: '用户不存在'})  
 } else {  
 userService.getMyAddress(data[0].user\_id, (error, data, fields) => {  
 if (error) {  
 next(error)  
 } else {  
 res.send({ret: 0, data});  
 }  
 })  
 }  
 })  
  
}  
  
  
  
  
const deleteAddress = (req, res, next) => {  
 let t = req.body;  
 let address = t.address;  
 let auth = req.headers['authorization']  
 let account = jwt.decode(auth.split(' ')[1])['user\_account']  
 userService.getUser(account, (error, data1, fields) => {  
 if (data1.length === 0) {  
 res.send({ret: 1, message: '用户不存在'})  
 } else {  
 userService.getAddress({userId: data1[0]['user\_id'], address: address}, (error, data, fields) => {  
 if (data.length === 0) {  
 res.send({ret: 1, message: '当前收货地址不存在'})  
 } else {  
 if (data1[0]['address'] === address) {  
 res.send({ret: 1, message: '默认收货地址无法删除'});  
 } else {  
 userService.deleteAddress({  
 userId: data[0]['user\_id'],  
 address: address  
 }, (error, data, fields) => {  
 if (error) {  
 next(error);  
 } else {  
 res.send({ret: 0, message: '删除收货地址成功'});  
 }  
 })  
 }  
 }  
 })  
 }  
 })  
  
}  
  
module.exports = {  
 register,  
 login,  
 logout,  
 userinfo,  
 addAddress,  
 deleteAddress,  
 getMyAddress  
}

控制层关键代码

const {  
 register,  
 login,  
 logout,  
 userinfo,  
 addAddress,  
 deleteAddress,  
 getMyAddress} = require('../controller/users')  
let express = require('express');  
let router = express.Router();  
  
//默认根接口  
router.get('/user', (req, res, next) => {  
 // res.render('index');  
});  
  
//注册  
router.post('/register', register);  
//登录  
router.post('/login', login);  
//注销  
router.delete('/logout', logout);  
//获取用户信息  
router.get('/userinfo', userinfo);  
  
router.post('/addAddress', addAddress);  
  
router.delete('/deleteAddress', deleteAddress);  
  
router.get('/getMyAddress', getMyAddress)  
  
module.exports = router;