**信息工程系**

**vue程序设计项目**

**考核设计报告**

**题 目：vue后台管理系统**

**班 级**

**学 号**

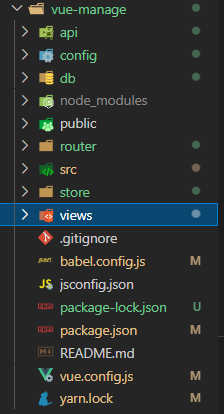
**姓 名**

**二O二一 年 十二 月**

**一、项目基础**

1. vue2.x
2. vuex
3. vue-router
4. axios
5. element-ui
6. vue-cli4.0
7. lang
8. echarts
9. mockjs

## 二、项目目录与介绍



|  |  |  |
| --- | --- | --- |
| 目录 | | 说明 |
| api | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| Api->home.js | | 使用mockjs |
| Api->permission.js | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| Api->user.js | | 删除用户改成post请求 |
| Api->axios.js | | axios二次封装 |
| Api->mock.js | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| Config->index.js | | axios二次封装 |
| Public->favicon.ico | | init |
| Public->index.html | | init |
| Router->index.js | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| Src->assets->images | | 页面相关图片 |
| Src->assets->common.less | | header添加面包屑 |
| Src->assets->home.less | | 添加less样式文件 |
| Src->assets->index.less | | 添加less样式文件 |
| Src->assets->reset.less | | 添加less样式文件 |
| Src->components | | 退出登录功能 |
| Src->components->CommonAside.vue | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| Src->components->CommonForm.vue | | 封装Form为Vue组件，用户管理页面添加新增用户按钮 |
| Src->components->CommonHeader.vue | | 退出登录功能 |
| Src->components->CommonTable.vue | | table分页功能 |
| Src->components->CommonTag.vue | | 封装Form为Vue组件，用户管理页面添加新增用户按钮 |
| Src->components->ECharts.vue | | ECharts图表封装成Vue组件 |
| Src->App.vue | | 侧边栏路由跳转功能 |
| Src->main.js | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| Store->index.js/Store->tab.js/Store->user.js | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| views | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| Views->home | | ECharts图表封装成Vue组件 |
| Views->login | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| Views->mall | | 完善路由页面 |
| Views->other | | 完善路由页面 |
| Views->user | | 删除用户改成post请求 |
| Views->Main.vue | | Tag切换与删除 |
| .gitignore | | init |
| README.md | | docs: add feature branch |
| babel.config.js | | ElementUI改为按需引入 |
| jsconfig.json | | init |
| package-lock.json | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| package.json | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| vue.config.js | | 添加vue-router，关闭语法检查 |
| yarn.lock | | 升级vue-router，添加js-cookie，完成登录逻辑 |
| 目录 | 说明 | |
| api | 升级vue-router，添加js-cookie，完成登录逻辑 | |
| Api-> |  | |
|  |  | |
|  |  | |
| config | axios二次封装 | |
| public | init | |
| router | 升级vue-router，添加js-cookie，完成登录逻辑 | |
| src | docs: update README | |
| store | 升级vue-router，添加js-cookie，完成登录逻辑 | |
| views | 升级vue-router，添加js-cookie，完成登录逻辑 | |
| .gitignore | init | |
| README.md | docs: add feature branch | |
| babel.config.js | ElementUI改为按需引入 | |
| jsconfig.json | init | |
| package-lock.json | 升级vue-router，添加js-cookie，完成登录逻辑 | |
| package.json | 升级vue-router，添加js-cookie，完成登录逻辑 | |
| vue.config.js | 添加vue-router，关闭语法检查 | |
| yarn.lock | 升级vue-router，添加js-cookie，完成登录逻辑 | |

## 三、axios的封装和使用

在项目里面对axios封装能够节省不少的代码和时间，首先在 api 目录下新建axios.js文件

import axios from 'axios'

import config from '../config'

const baseUrl = process.env.NODE\_ENV === 'development' ? config.baseUrl.dev : config.baseUrl.pro

class HttpRequest {

    constructor(baseUrl) {

        this.baseUrl = baseUrl

    }

    getInsideConfig() {

        const config = {

            baseUrl: this.baseUrl,

            header: {}

        }

        return config

    }

    interceptors(instance) {

        // 添加请求拦截器

        instance.interceptors.request.use(function (config) {

            // 在发送请求之前做些什么

            return config;

        }, function (error) {

            // 对请求错误做些什么

            return Promise.reject(error);

        });

        // 添加响应拦截器

        instance.interceptors.response.use(function (response) {

            // 2xx 范围内的状态码都会触发该函数。

            // 对响应数据做点什么

            return response;

        }, function (error) {

            // 超出 2xx 范围的状态码都会触发该函数。

            // 对响应错误做点什么

            return Promise.reject(error);

        });

    }

    request(options) {

        const instance = axios.create()

        options = {...this.getInsideConfig(), ...options}

        this.interceptors(instance)

        return instance(options)

    }

}

export default new HttpRequest(baseUrl)

## 四、全局组件的注册和使用

import Vue from 'vue'

import App from './App'

import 'element-ui/lib/theme-chalk/index.css';//'element-ui/lib/theme-chalk/index.css'

import {

    Button,

    Container,

    Header,

    Main,

    Aside,

    Menu,

    Submenu,

    MenuItem,

    MenuItemGroup,

    Dropdown,

    DropdownMenu,

    DropdownItem,

    Row,

    Card,

    Col,

    Table,

    TableColumn,

    Breadcrumb,

    BreadcrumbItem,

    Tag,

    Form,

    FormItem,

    Input,

    Select,

    Option,

    Switch,

    DatePicker,

    Dialog,

    Pagination,

    MessageBox,

    Message,

} from 'element-ui'

import router from '../router'

import './assets/less/index.less'//./assets/less/index.less

import store from '../store'

import '../api/mock'

import http from 'axios'

Vue.config.productionTip = false

Vue.use(Button)

Vue.use(Container)

Vue.use(Header)

Vue.use(Main)

Vue.use(Aside)

Vue.use(Menu)

Vue.use(Submenu)

Vue.use(MenuItem)

Vue.use(MenuItemGroup)

Vue.use(Dropdown)

Vue.use(DropdownMenu)

Vue.use(DropdownItem)

Vue.use(Row)

Vue.use(Card)

Vue.use(Col)

Vue.use(Table)

Vue.use(TableColumn)

Vue.use(Breadcrumb)

Vue.use(BreadcrumbItem)

Vue.use(Tag)

Vue.use(Form)

Vue.use(FormItem)

Vue.use(Input)

Vue.use(Select)

Vue.use(Option)

Vue.use(Switch)

Vue.use(DatePicker)

Vue.use(Dialog)

Vue.use(Pagination)

Vue.prototype.$http = http

Vue.prototype.$confirm = MessageBox.confirm

Vue.prototype.$message = Message

// 路由守卫逻辑

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

    // store.commit('clearToken')

    store.commit('getToken')

    const token = store.state.user.token

    console.log('@token', token)

    if (!token && to.name !== 'login') {

        next({name: 'login'})

    } else if (token && to.name === 'login') {

        next({name: 'home'})

    } else {

        next()

    }

})

new Vue({

    store,

    router,

    render: h => h(App),

    created() {

        store.commit('addMenu', router)

    },

}).$mount('#app')

## 五、路由的分区和nprogress的使用

import Vue from 'vue'

import VueRouter from 'vue-router'

Vue.use(VueRouter)

const routes = [

    {

        path: '/',

        name: 'Main',

        component: () => import('../views/Main'),

        children: [

            // {

            //     path: '/home',

            //     name: 'home',

            //     component: () => import('../views/home')

            // },

            // {

            //     path: '/user',

            //     name: 'user',

            //     component: () => import('../views/user')

            // },

            // {

            //     path: '/mall',

            //     name: 'mall',

            //     component: () => import('../views/mall')

            // },

            // {

            //     path: '/page1',

            //     name: 'page1',

            //     component: () => import('../views/other/pageOne')

            // },

            // {

            //     path: '/page2',

            //     name: 'page2',

            //     component: () => import('../views/other/pageTwo')

            // },

        ],

    },

    {

        path: '/login',

        name: 'login',

        component: () => import('../views/login/login.vue')

    },

]

const router = new VueRouter({

    mode: 'history',

    routes

})

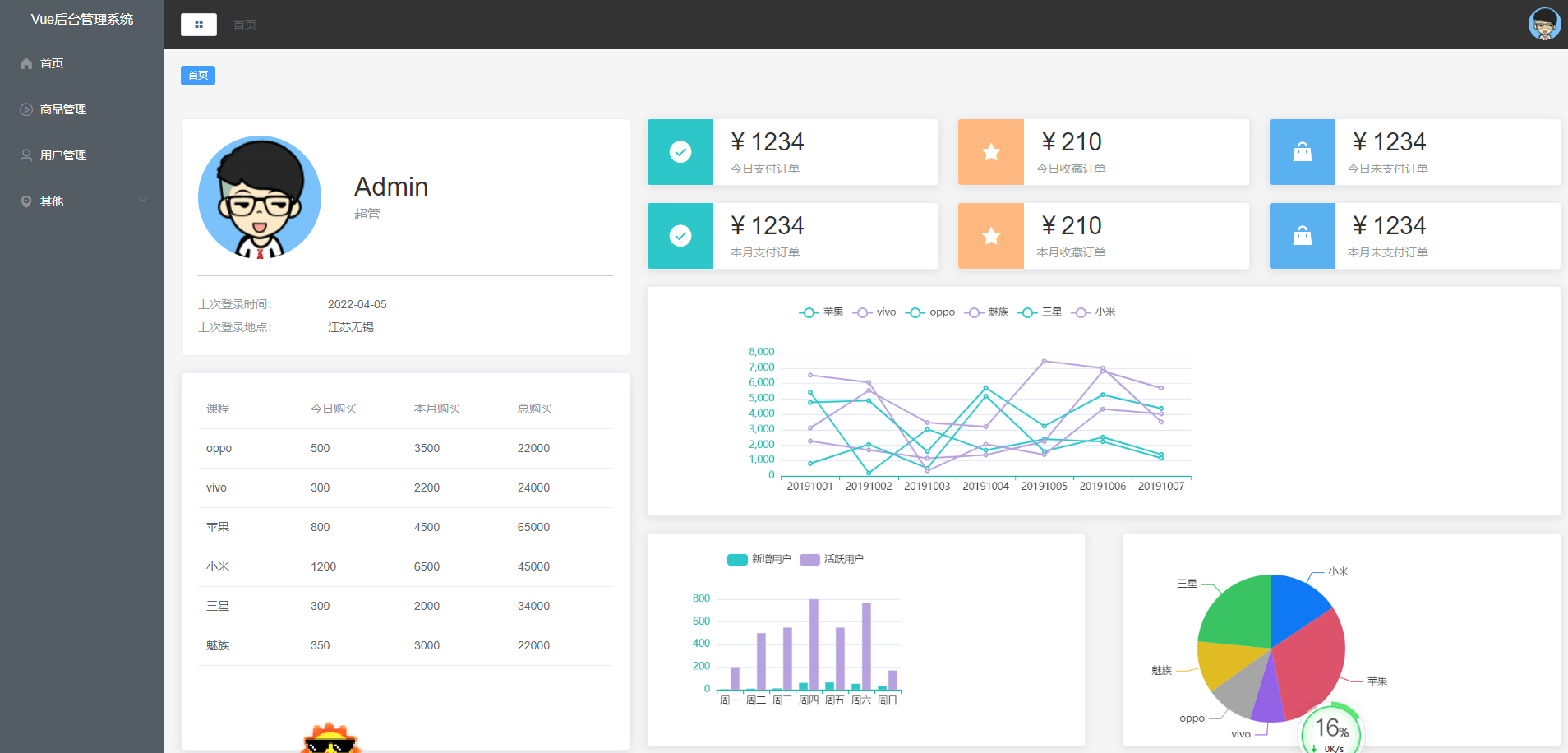
export default router

# 六 系统展示

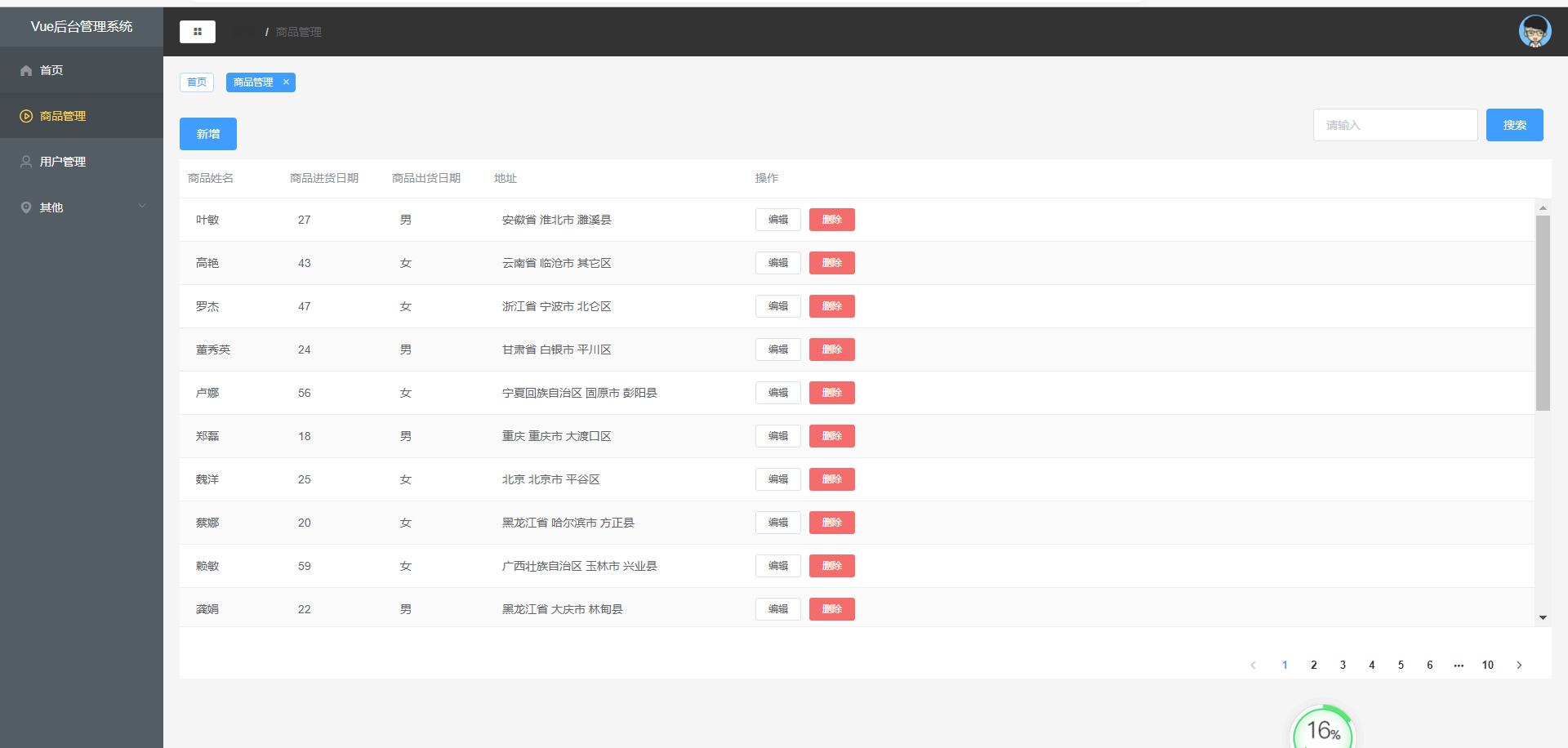
登录



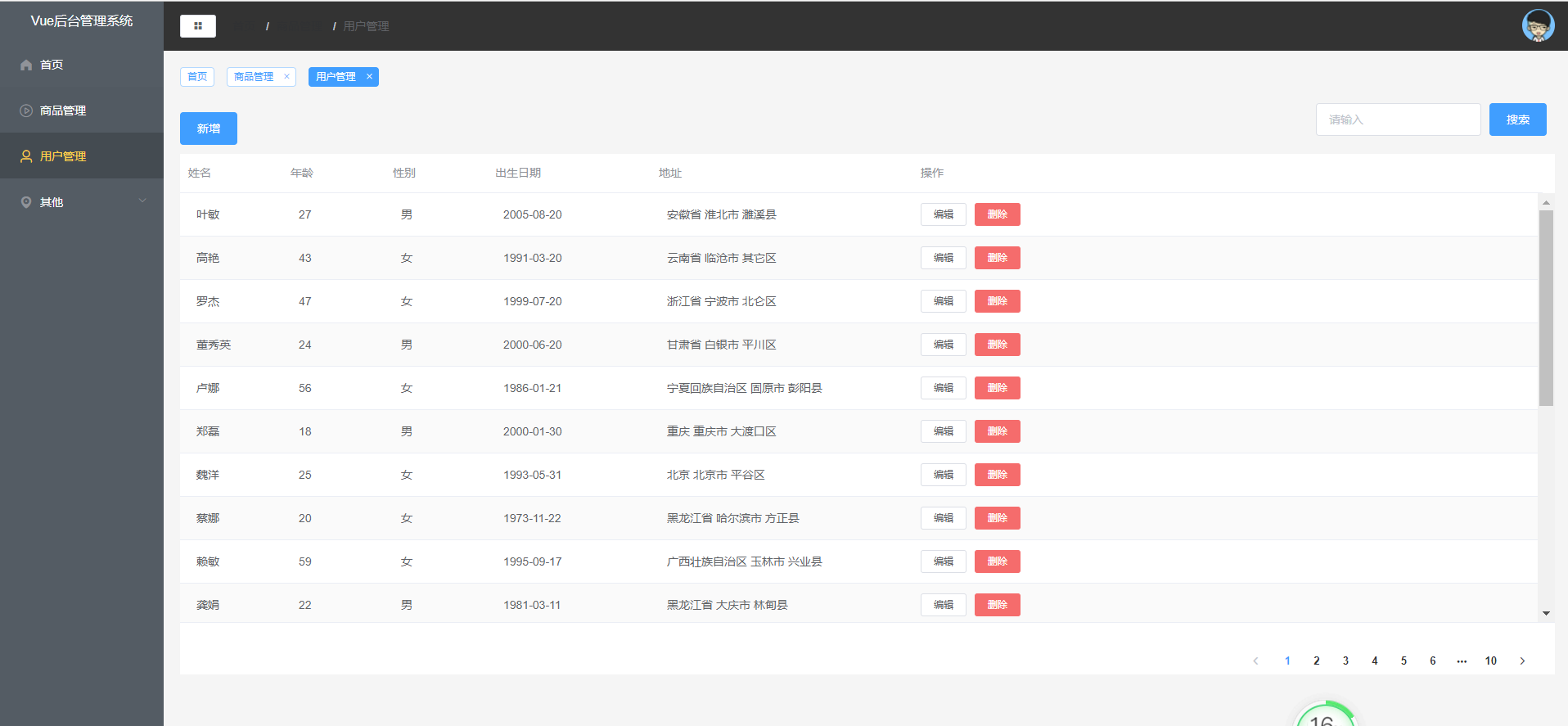
首页



商品管理



用户管理



其他

访问日志



# 七．项目总结

在刚刚结束得项目中，感觉收获良多，今日便为刚刚结束得项目做出总结

1.代码得熟练

在这次项目中，发现自己重大问题，即许多知识点仅浮于皮毛，并未深沉理解其真正含义，对于一项功能得实现理解仅限于教材与视频教程所讲并为深挖期原理，只知生搬硬套，导致错误频出，

后续应多加练习并熟读相关知识点文档，理解并深刻记忆其应用。

2.关于接口文档与需求文档

接口文档与需求文档是我们前端工程师最需要得两样工具，前者告诉我们需要怎样拿到数据或者是拿怎样得数据或者需要传回怎样得数据，后者则告诉我们需要完成怎样得功能做出什么样得效果，

这次项目中就因没有熟读文档，导致数据传回失败和功能缺失需要重新更改代码，白白浪费很多时间。

3.团队合作

项目得完成并不是一个人得战斗，而是需要和你的队友共同完成，一个人的力量是有限得，当发现不会的地方应该积极与队友请教交流，大家的共同目的都是一样的，都是希望项目以最快的速度和最好的状态上线，应该时刻保持不懂就问，不会就查的工作状态。团队中大家都会使用GitHub或者码云这样的工具来合并各自的代码，因此要熟记这些工具的使用方法，不要因自己的失误而导致整个项目的滞后，同样代码的书写规范也很重要，多写注释遵守Eslint书写规范，这样代码才会美观和一目了然，方便队友查看与修改。