

**湖北大学知行学院**

**Zhixing College Of Hubei University**

**学士学位论文**

**题目： 基于node.js和vue.js的二手交易平台的设计和实现**

**论文作者： 元萌萌**

**学 号： 1531050066**

**系 部： 计算机与信息工程系**

**专 业： 计算机科学与技术**

**指导教师： 龙文佳**

论文提交日期： 2019 年 4 月 10 日

**目 录**

**绪论** 1

**1 课题总体介绍** 1

1.1 课题介绍 1

1.2 课题目的及意义 1

1.3 主要研究功能 1

**2 前端小程序开发分析** 3

2.1 微信小程序开发流程 3

2.2 微信小程序数据渲染 3

2.3 小程序端请求服务端数据 4

2.3.1 配置说明 4

2.3.2 请求数据的具体方法 4

2.3.3 小程序图片上传 5

**3 管理页面开发分析** 6

3.1 前序准备 6

3.2 项目目录结构和基础配置 7

3.3 登录和权限管理流程 7

3.3.1 登录操作 8

3.3.2 权限验证操作 8

3.3.3 控制页面路由 8

3.3.4 为什么不选择完全由后端来控制页面路由 8

3.3.5 具体的实现 9

3.4 构建vue单页 9

3.4.1 页面布局 9

3.4.2 页面基础结构和内容 10

3.5 和服务端进行交互 10

3.6 构建和发布 11

**4 服务端开发分析** 11

4.1 选择node.js的原因以及环境搭建 11

4.2 关于koa2的项目目录结构说明 12

4.3 编写REST风格API 12

4.3.1 接口和前端的交互 12

4.3.2 接口规范 13

4.3.3 接口实现 14

4.4 数据库操作 14

4.4.1 为什么选择MySQL 14

4.4.2 如何访问数据库 15

4.5 服务端构建 16

**结论** 17

**参考文献** 18

**附录** 19

**致谢** 20

**基于node.js和vue.js的二手交易平台的设计和实现**

**摘 要**

本项目使用到的技术栈包括：node.js，vue.js以及微信小程序。选择小程序作为客户端，管理后台选用vue搭建。后端则选择了性能优越的node.js。整个项目采用了最新的前后端分离的开发模式，让前端和后端做各自擅长的事，区别于混合开发。这样可以最大程度的降低耦合。

值得一提的是后端采用REST（Representational State Transfer）风格的软件架构模式。REST强调组件交互的可伸缩性，接口的通用性，组件的独立部署，以及用来减少交互延迟，增强安全性，封装遗留系统的中间组件（intermediary components）。REST是一种设计API的模式。最常用的数据格式是JSON。由于JSON能直接被JavaScript读取，所以，以JSON格式编写的REST风格的API具有简单，易读，易用的特点。REST也已经成为Web API的标准。

编写API的好处在于：API本身就是把Web App的功能全部封装，然后通过API来操作数据，可以极大地把前端和后端的代码隔离，使得后端代码易于测试，前端代码编写更简单。此外，如果我们把前端页面看作是一种用于展示的客户端，那么API就是为客户端提供数据，操作数据的接口。这种设计可以获得极高的扩展性。例如，当用户需要在手机上购买商品时，只需要开发针对iOS和Android的两个客户端，通过客户端访问API，就可以完成通过浏览器页面提供的功能，而后端代码基本无需改动。

**【关键词】**前后端分离 REST架构风格 Web API

**Architectural Styles and the Design of Network-based Software Architectures**

**Abstract**

The technology stack used in this project includes: node.js, vue.js, and WeChat Mini Program. Select the Mini Program as the client, and the management background to use vue to build. The backend chose node.js with superior performance. The entire project adopts the latest development mode of front end and back end separated, so that the front end and the back end do their own good things, which is different from hybrid development. This minimizes coupling.

It is worth mentioning that the back end adopts the REST (Representational State Transfer) style software architecture mode.REST emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems. REST is a model for designing APIs. The most common data format is JSON. Since JSON can be read directly by JavaScript, the RESTful API written in JSON format is simple, easy to read, and easy to use. REST has also become the standard for Web APIs.

The advantage of writing an API is that the API itself encapsulates the functions of the Web App and then manipulates the data through the API. It can greatly isolate the front-end and back-end code, making the back-end code easy to test and the front-end code writing easier. In addition, if we think of the front-end page as a client for display, the API is the interface that provides data and operations data for the client. This design can achieve extremely high scalability. For example, when a user needs to purchase an item on a mobile phone, only two clients for iOS and Android need to be developed, and the function provided through the browser page can be completed through the client access API, and the back-end code basically does not need to be changed.

**【Key words】** Representational State Transfer (REST)  Front-end and back-end separated Web API

**绪论**

关于小程序简介：小程序是一种全新的连接用户与服务的方式，它可以在微信内被便捷地获取和传播，同时具有出色的使用体验。微信小程序并非凭空冒出来的一个概念。当微信中的 WebView 逐渐成为移动 Web 的一个重要入口时，微信就有相关的 JS API 了。小程序的主要开发语言是 JavaScript ，小程序的开发同普通的网页开发相比有很大的相似性。对于前端开发者而言，从网页开发迁移到小程序的开发成本并不高，但是二者还是有些许区别的。

​vue.js是一套构建用户界面的渐进式框架。与其他重量级框架不同的是，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，并且非常容易学习，非常容易与其它库或已有项目整合。另一方面，Vue 完全有能力驱动采用单文件组件和Vue生态系统支持的库开发的复杂单页应用。

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。 Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。本项目采用了后端分离的开发模式，后端的api接口依据REST风格设计，符合现代Web架构模式。

**1 课题总体介绍**

1.1 课题介绍

本系统分为小程序用户端和平台后台管理两个部分。小程序端的开发设计遵循用户友好原则，方便用户发布二手商品和浏览购买商品以及购买后的操作包括：评论，点赞，买家秀等；后台管理提供给平台使用，主要功能包括：人员管理，二手商品信息管理等。

1.2 课题目的及意义

时代的发展，科技的进步，新旧物品的交替日新月异，每个家庭，每个人都会有更新过的物品在闲置，如何处理这些物品也成为了人们日常关注的问题。有人会拿到当地的二手交易市场去交易，有人会因离二手市场较远或二手物品不易搬运而把它当废品卖掉甚至扔掉。这不仅不利于环保，还浪费了社会资源。

由于二手交易平台的诞生就使得二手交易不再局限于一定要去二手交易市场实行交易。由于网络的方便性，让很大部分人选择了上二手交易平台发布二手买卖信息，足不出户就可以买卖二手，极大的方便了老百姓。可以说二手交易平台极大的推动了用户买卖二手的热情，也促进了社会资源的有效利用。

1.3 主要研究功能

具体的小程序端功能实现如下：

（1）小程序属于微信生态，通过微信授权来简化登录注册流程，具有良好的用户体验。

（2）用户发布二手商品，支持文字图片上传。

（3）首页通过轮播图等形势展示热门商品。

（4）用户发布的商品信息经过后台审核后在小程序端按类型进行展示。

（5）用户可通过关键字搜索指定的商品信息。

（6）用户可以在找到合适满意的商品后获取到卖家的联系方式。平台不参与买家和买家的交易。

（7）使用平台完成交易，买家可以在平台上传买家秀。

对于后台管理页面的具体功能如下：

（1）小程序端用户信息的管理。

（2）用户发布的二手商品信息的管理。

（3）首页轮播图和热门商品信息的管理。

（4）买家上传的买家秀信息管理。

**2前端小程序开发分析**

2.1微信小程序开发流程

开发小程序的第一步，需要拥有一个小程序 AppID，后续的所有开发流程会基于这个 AppID 来完成。小程序的注册非常简单，只需几个操作。使用浏览器打开 https://mp.weixin.qq.com/ 点击立即注册，在打开的页面中选择小程序后，填入相关的信息，就可以完成注册了

有了AppID 还需要下载安装小程序开发者工具，这些准备工作结束之后还需要了解小程序的目录结构以及如何配置：

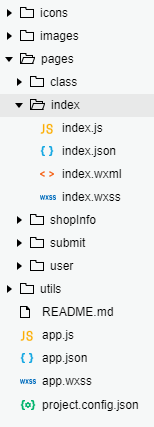


图2.1小程序目录结构

小程序包含一个描述整体程序的 app 和多个描述各自页面的 page。一个小程序主体部分由三个文件组成，必须放在项目的根目录，其中app.js控制小程序逻辑；app.json包含小程序公共配置；app.wxss则是小程序公共样式表。一个小程序页面由四个文件组成，分别是：js 文件包含页面逻辑；wxml文件编写页面结构；json文件则是页面配置；wxss 文件包含页面样式表。

了解目录结构是所有开发的第一步，然后需要了解开发流程中最关键的一步：页面语法和API的使用。其中js文件不需要多说，就是JavaScrapt语法；wxml可以类比html；wxss则可以类比css。关于小程序的API都是微信提供的各种基础接口，主要涵盖路由，界面，网络，数据缓存，媒体，位置，转发，画布，文件，设备，Worker等。

2.2微信小程序数据渲染

小程序的页面渲染和vue很相似，采用数据绑定来进行条件和列表渲染，wxml 中的动态数据均来自对应 Page 的 data，其中还可以使用this.setData({})来更新数据。在wxml文件内，数据绑定使用 Mustache 语法（双大括号）将变量包起来。以首页的banners为例，在index.js文件中：

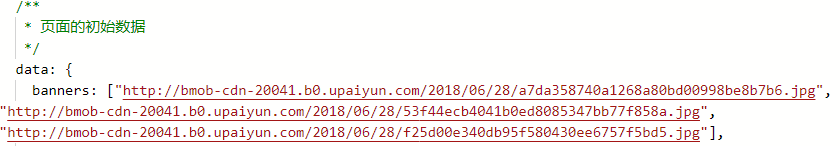


图2.2小程序index.js代码

在index.wxml文件内对应的渲染代码为：

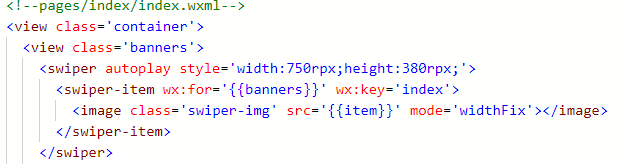


图2.2小程序index.wxml代码

2.3 小程序端请求服务端数据

小程序端不可能都是静态数据，更多的数据是需要从服务端请求的，出于安全考虑，小程序只可以跟指定的域名与进行网络通信。包括普通 HTTPS 请求（wx.request）、上传文件（wx.uploadFile）、下载文件（wx.downloadFile) 和 WebSocket 通信（wx.connectSocket）。

2.3.1配置说明

配置流程：服务器域名请在 「小程序后台-设置-开发设置-服务器域名」 中进行配置，配置时需要注意：域名只支持 https (wx.request、wx.uploadFile、wx.downloadFile) 和 wss (wx.connectSocket) 协议；域名不能使用 IP 地址或 localhost；可以配置端口，如 https://myserver.com:8080，但是配置后只能向 https://myserver.com:8080 发起请求。如果向 https://myserver.com、https://myserver.com:9091 等 URL 请求则会失败。

如果不配置端口。如 https://myserver.com，那么请求的 URL 中也不能包含端口，甚至是默认的 443 端口也不可以。如果向 https://myserver.com:443 请求则会失败。

域名必须经过 ICP 备案；

出于安全考虑，api.weixin.qq.com 不能被配置为服务器域名，相关API也不能在小程序内调用。 开发者应将 AppSecret 保存到后台服务器中，通过服务器使用 getAccessToken 接口获取 access\_token，并调用相关 API；对于每个接口，分别可以配置最多 20 个域名。

2.3.2请求数据的具体方法

小程序中使用wx.request()接口来发起 HTTPS 网络请求。具体的请求参数如下：

表2.3.2 wx.request()请求参数

|  |  |
| --- | --- |
| 属性 | 说明 |
| url | 接口地址 |
| data | 请求的具体参数 |
| header | 设置请求header |
| method | 请求的方法，包括GET,POST,PUT, DELETE等 |
| success | 请求成功的回调函数 |

关于data 参数说明：最终发送给服务器的数据是 String 类型，如果传入的 data 不是 String 类型，会被转换成 String 。转换规则如下：对于 GET 方法的数据，会将数据转换成 query string；对于 POST 方法且 header['content-type'] 为 application/json 的数据，会对数据进行JSON 序列化；对于 POST 方法header['content-type'] 为 application/x-www-form-urlencoded 的数据，会将数据转换成 query string。首页请求二手商品的代码如下：



图2.3.2小程序wx.request()代码

2.3.3小程序图片上传

小程序中上传图片需要用到wx.chooseImage()接口和wx.uploadFile()接口。chooseImage接口作用是从本地相册选择图片或使用相机拍照。uploadFile作用是把本地选中的图片上传到制定服务器。

表2.3.3 wx. chooseImage ()请求参数

|  |  |
| --- | --- |
| 属性 | 说明 |
| count | 最多可以选择的图片张数 |
| sizeType | 所选的图片的尺寸 |
| sourceType | 选择图片的来源 |
| success | 请求成功的回调函数 |

表2.3.3 wx. uploadFile ()请求参数

|  |  |
| --- | --- |
| 属性 | 说明 |
| url | 开发者服务器地址 |
| filePath | 要上传文件资源的路径 |
| name | 文件对应的 key，开发者在服务端可以通过这个 key 获取文件的二进制内容 |
| header | HTTP请求 Header，Header 中不能设置 Referer |
| formData | HTTP 请求中其他额外的 form data |
| success | 请求成功的回调函数 |

二手物品图片上传时的代码如下：

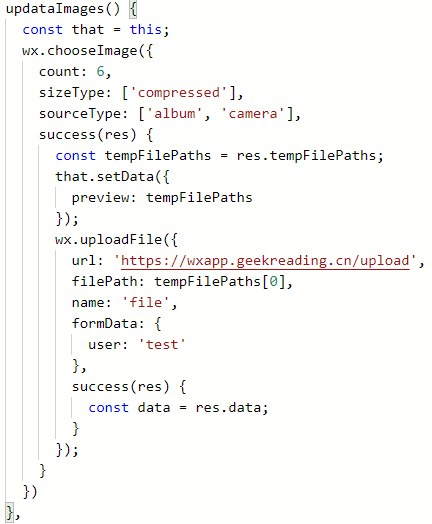


图2.3.3小程序图片上传代码

**3管理页面开发分析**

3.1前序准备

管理页面使用vue前端框架开发，需要在本地安装node和git。本项目技术栈基于 ES2015+，vue，vuex，vue-router ，vue-cli，axios 和 element-ui，所有的请求数据都使用Mock.js进行模拟。

3.2项目目录结构和基础配置

build文件与构建相关；config文件包含配置相关；dist文件下是最终编译生成的页面内容；node\_modules文件是npm模块；src中是所有页面的源代码；static包括第三方不打包资源，例如：富文本；.babelrc文件是babel-loader 的配置文件；eslintrc.js是eslint 配置项；.gitignore是git忽略项；favicon.ico是页面的icon图标； index.html是首页的html模板；package.json是webpack打包的配置文件。src下的文件目录也在下面列出：api包含了所有请求；assets包含主题，字体等静态资源；components包含公用组件； directive包含全局指令；icons包含所以的icons；router是路由页面；store是全局store管理；syles包括全局样式；utils包含全部的公共方法；views所有页面的view都在这个文件下；App.vue入口页面；main.js入口加载组件初始化等。

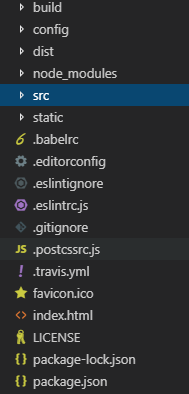
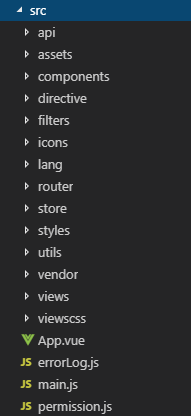
 

图3.2管理页面目录结构

首先需要配置的是config下的dev和prod件，dev是指开发环境配置，而prod则是生产环境配置。使用module.exports来把配置参数以对象的形式暴露出来，其中包括了调用的api地址，项目名称，npm运行指令和短信接口是否启用。

3.3登录和权限管理流程

后台项目有别于其它的项目，权限验证与安全性是非常重要的，可以说是一个后台项目一开始就必须考虑和搭建的基础核心功能。本项目所要做到的是：不同的权限对应着不同的路由，同时侧边栏也需根据不同的权限，异步生成。这里先简单说一下，实现登录和权限验证的思路。

3.3.1登录操作

其中首先面对的是登录：当用户填写完账号和密码后向服务端验证是否正确，验证通过之后，服务端会返回一个token，拿到返回的token之后会将这个token存到浏览器的cookie中，保证刷新页面后能记住用户登录状态，然后前端会根据token再去拉取一个user\_info的接口来获取用户的详细信息例如：用户权限，用户名等。

3.3.2权限验证操作

再来说关于权限验证：通过token获取用户对应的role，动态根据用户的role算出其对应有权限的路由，通过router.addRoutes动态挂载这些路由。

上述所有的数据和操作都是通vuex全局管理控制的。需要补充说明的是：刷新页面后 vuex的内容也会丢失，所以需要重复上述的那些操作。

在接下来的每一次请求都是要带token来验证权限的，我的做法是在axios里进行封装，这样一来就不用每个请求手动来配置token，还可以在axios里做一些统一的异常处理，这是一劳永逸的做法。

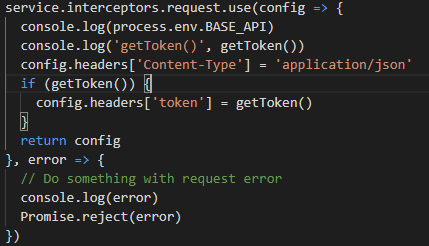


图3.3.2在axios里封装token配置的代码

3.3.3控制页面路由

权限控制的主要思路在于页面路由。一般来说前端会有一份路由表，它表示了每一个路由可访问的权限。但是，要清楚这些控制都只是页面级别的，说到底前端来做权限控制不是绝对安全的，最终都需要后端来处理这些权限。

在现在的项目中，前端只是控制页面级权限，不同的权限下显示的侧边栏是完全不同的，后端则会在前端请求接口时验证这些操作。具体的验证方式为：每一个发给后台的请求，无论是GET还是POST都需要前端在请求的header里面携带用户token发给后端。之后，后端再根据该用户的token 来验证此用户是否有权限执行该操作。若没有权限则抛出一个对应的状态码，前端检测到该状态码，做出相对应的操作；如果有权限则得到数据进行其后的操作。

3.3.4为什么不选择完全由后端来控制页面路由

在很多项目中的路由表是于后端根据用户的权限动态生成的，那为什么本项目没有采取这种方式呢？原因如下：

首先，本项目计划会不断的迭代，如果前端开发一个新的页面还要让后端重新配置路由和权限，那就回到了曾经的前后端不分离时代，不符合本项目前后端分离的开发模式。

其次，在本项目中，虽然后端确实也有权限验证，但在后端的验证更多的时候是针对业务来划分的，但对于前端来说并不能按业务来划分权限。

还有一点是就在vue的2.2.0版本之前异步挂载路由是很麻烦的事。好在官方最终也给出了新的接口来优化异步挂载。

3.3.5具体的实现

首先在创建vue实例时将vue-router进行挂载，但这个时候vue-router挂载一些登录或者不用权限的公用的页面。

之后，当用户登录后，获取用role，将role和路由表每个页面的需要的权限作比较，生成最终用户可访问的路由表。

最后，调用router.addRoutes(store.getters.addRouters)添加用户可访问的路由。再使用vuex管理路由表，根据vuex中可访问的路由渲染侧边栏组件。

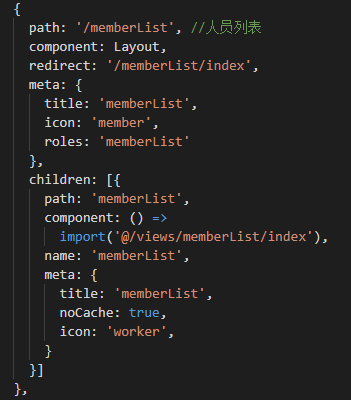


图3.3.5 页面路由的实现代码

3.4构建vue单页

本项目的UI选择了element-ui库，主要原因在于：有大厂背书（饿了么前端组开源项目） 。 尽管核心开发只有两到三个人，但至少不用担心哪天就不维护。

得益于vue的生态圈优异，社区活跃。这里还有一些其它的UI框架包括：ivew，个人开源框架，美观度和交互性都不错，有种介于Element和Ant之间的感觉。 vue-admin，也是一个不错的选择，vue官方的一套admin系统UI。

3.4.1页面布局

页面整体布局是一个产品最外层的框架结构，往往会包含导航、侧边栏、面包屑以及内容等。想要开始一个前端项目，先要确定基础布局。本项目大部分页面都是基于这个layout 的，除了个别页面如：login , 404, 401等页面没有使用该layout。如果你想在一个项目中有多种不同的layout也是很方便的，只要在一级路由那里选择不同的layout组件就可以了。

3.4.2页面基础结构和内容

vue单页的结构包括：style，template以及script标签。其中style控制页面样式；template控制页面结构；script控制页面逻辑和数据处理。

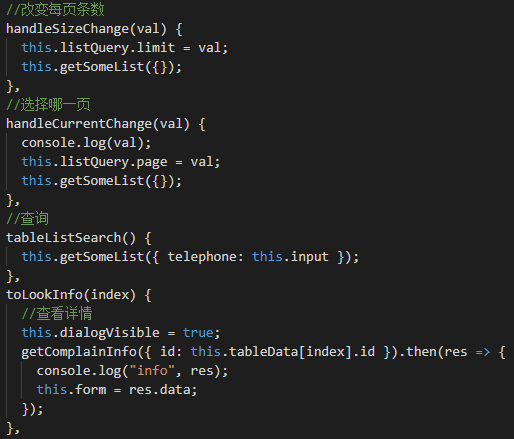


图3.4.2单页中的查询代码

3.5和服务端进行交互

单页需要和服务端进行交互才可以动态获取数据。这也是前后端分离的核心。一个完整的前端从UI到服务端的交互处理流程大概是这样的：首先从UI 组件交互操作开始；触发调用统一管理的api service请求函数；然后再使用封装的request.js发送请求到服务器；之后就可以获取到服务端返回数据，本项目返回的数据统一为json格式；最终还需要前端更新 请求到的data，也就是数据；这就是一个完整的流程。

上面提及的这些请求都放在@/src/api文件内，使用来自request.js的统一封装好的请求方式。这样就可以实现全局的request拦截器，response拦截器以及统一的错误处理，统一做了超时处理，全局baseURL设置等。



图3.5api中获取商品详情接口

3.6构建和发布

当项目开发完毕，只需要运行一行命令就可以打包应用：

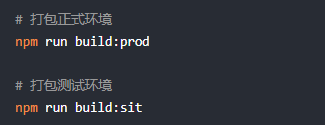


图3.6打包命令

构建打包成功之后，会在根目录生成dist文件夹，里面就是构建打包好的文件，通常会包含有index.js ，index.css以及index.html等静态文件。到这一步就完成了整个二手交易平台项目的所有前端开发。

**4服务端开发分析**

4.1选择node.js的原因以及环境搭建

本项目服务端使用node.js语言进行开发，选择了比较流行的koa2框架。选择node.js的原因主要有两个：第一，本身我从事前端开发选用node.js没有重新学习语言的成本；其次在于node.js自身的性能很高，它是天生异步的，可以从性能方面节约更多的服务端资源。在众多的node.js框架中（包括：Express，Sails.js，koa，Meteor，DerbyJS，Total.js，restify等）选择koa2的原因在于相比koa第一个版本，新版支持最新的ES6和ES7的语法，异步的回调函数可以摆脱那种臃肿的地狱回调了，取而代之的是使用Promise对象来执行异步，或者使用更先进的async和await，可以轻松地把一个function变为异步模式。

关于环境搭建也非常简单。node.js依旧是遵循javascript语法，只是运行在node环境下，区别于在浏览器里运行的js。所以，首先需要安装node环境，同时会附带安装npm也就是node环境下的包管理工具。

4.2关于koa2的项目目录结构说明

后端项目的目录结果如下：

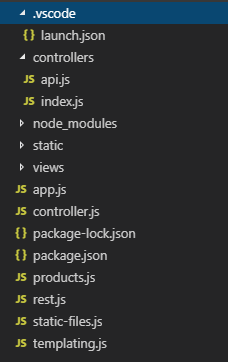


图4.2服务端目录结构

其中包括：.vscode文件包含VScode IDE的配置文件；controllers/api.js接口文件也就是采用REST风格封装的api；controllers/index.js可以理解为MVC的控制器文件；node\_modules文件包含所有的npm包；static下是静态文件，例如页面引用的字体，css等；views下是页面文件；app.js是koa的项目入口；controller.js是扫描注册的控制器文件；package.json为项目的说明性配置文件。

4.3编写REST风格API

在本项目中最值得一提的就是前后端分离模式，其中的核心就是后端不再操作页面的DOM，而选择利用接口来返回json格式数据给前端，无论是小程序客户端还是管理页面都可以接收和发送json给服务端，因为本身json就是JavaScript的一种对象。

4.3.1接口和前端的交互

整个的前后端交互流程其实非常简单也非常直接，后台的管理系统也就是本项目的服务端可以把数据通过REST API以接口的形势暴露给客户端和管理页面，在本项目中客户端就是小程序，管理页面就是vue编写的pc页面。如下图所示：

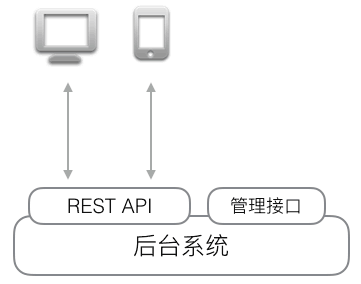


图4.3.1前后端交互模式

4.3.2接口规范

由于选择使用REST风格进行接口封装，所以遵循REST的编写规范是前提，需要说明的是这些规范不是强制的，但是遵循规范可以让代码更易于理解方便扩展。

例如，商品Product就是一种资源。获取所有Product的URL如下：

GET /api/products

而获取某个指定的Product，例如，id为123的Product，其URL如下：

GET /api/products/123

新建一个Product使用POST请求，JSON数据包含在body中，URL如下：

POST /api/products

更新一个Product使用PUT请求，例如，更新id为123的Product，其URL如下：

PUT /api/products/123

删除一个Product使用DELETE请求，例如，删除id为123的Product，其URL如下：

**DELETE** /api/products/123

资源还可以按层次组织。例如，获取某个Product的所有评论，使用：

GET /api/products/123/reviews

当我们只需要获取部分数据时，可通过参数限制返回的结果集，例如，返回第2页评论，每页10项，按时间排序：

GET /api/products/123/reviews?page=2&size=10&sort=time

4.3.3接口实现

在koa中实现REST接口要做的第一步就是npm安装好需要的标准库，之后在app.js文件内引用这些库。做好准备工作就可以在api.js文件内编写api接口，时刻需要记得遵循上面的约定。

其实在上面的步骤中，我们可以看到一个重点就是app.use(bodyParser())这个语句，它给koa安装了一个解析HTTP请求body的处理函数。如果HTTP请求是JSON数据，我们就可以通过ctx.request.body直接访问解析后的JavaScript对象。

然后到api.js中我们就可以添加一个GET接口来请求数据库里的商品，代码如下：



图4.3.3api.js中的GET接口代码

实际上暴露接口就是在处理函数的async，然后具体做法就是设置Content-Type和对应的设置Response Body这两项。

然后我们通过客户端来请求这个商品接口就可以得到如下的json数据：



图4.3.3接口请求到的数据

4.4数据库操作

服务端的作用就是处理数据，那就绕不开数据库的设计和操作。node.js生态提供了关于操作MySQL数据库的驱动，只需要使用npm安装就可以使用。

4.4.1为什么选择MySQL

MySQL属于关系型数据库，现在有很多流行的NoSQL数据库，它们一般都宣称自己速度快，存储规模可以是MySQL的几十倍甚至上百倍，但是这些仅仅是在特定的场景下，而且拥有SQL语句的基础，上手MySQL可以节省更多的时间。

再来看在同类的关系型数据库里：Oracle，SQL Server，DB2这些都是需要付费的，再加上各自的生态有很大的局限性，相比之下的MySQL既有免费版本可选，又拥抱开源，生态非常棒。所以，选择MySQL的理由很充分了。

4.4.2如何访问数据库

在node.js里访问MySQL数据库主流方法就是利用ORM技术（Object-Relational Mapping），简单讲就是把数据库里的表结构映射在了js的对象上，这样一来就可以避免去写复杂而冗余的底层SQL语句。在本项目中就选用了Sequelize框架，正如上面提到的Sequelize就是一个基于ORM的node框架。

编码之前需要做的是编写一个MySQL的配置文件，具体内容如下：



图4.4.2MySQL配置文件内容

之后创建一个Sequelize实例，具体代码如下：

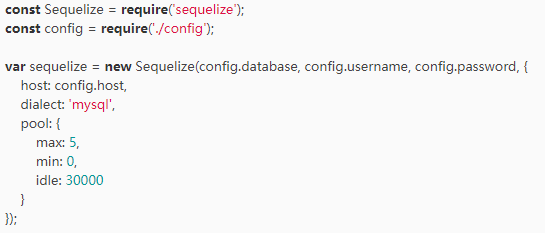


图4.4.2Sequelize实例

最后还需要给Sequelize对应一个数据表映射，也可以称为定义模型：

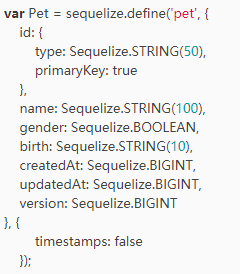


图4.4.2建立映射关系

到这里为止后端就实现了数据库操作，并且这样的效率是远远超过写SQL语句的。

4.5服务端构建

服务端的作用就在于处理数据并暴露接口给小程序和后台管理页面，现在这些步骤都已经完成。最后只需要上传到云服务器在命令行里执行npm run app.js指令前端就可以请求数据。

**结论**

一：微信小程序的页面布局最好选择flex布局，从设备兼容性上来看flex布局无论在低版本的iOS设备还是在屏幕不规则的安卓设备上都不会出现显示问题。

二：从产品交互的角度来说，客户端的一些按钮，icon的可点击区域可以适当做大，因为在移动设备上的交互不同于pc端，可能出现无法点击的情况。

三：无论是在服务端还是在vue项目中尽量不要使用国内的npm镜像来安装模块或者包，可能会出现意想不到的麻烦。

四：后端管理页面在添加新的路由时，最后可以和新的views页面文件一一对应，有助于二次开发或者项目重构。

五：在编写REST风格的接口时要区分它与HTTP请求是不同的，主要在于REST请求的Content-Type为application/json。当然返回给客户端的也是json格式的数据。

**参考文献**

[1] 高性能MySQL（第3版）. 编者: (美)Baron Schwartz，(美)Peter Zaitsev，(美)Vadim Tkachenko . 译者:宁海元，周振兴，彭立勋. 电子工业出版社,2013

[2] Vue.js实战.编者：梁灏. 清华大学出版社,2017

[3] JavaScript设计模式.编者：(美)Addy Osmani. 人民邮电出版社,2013

[4] Node.js实战 第2版.编者：[英] 亚历克斯•杨（Alex Young），布拉德利•马克（Bradley Meck），[美] 麦克•坎特伦（Mike Cantelon） 著，吴海星 译. 人民邮电出版社,2018

[5] 深入理解Nginx：模块开发与架构解析（第2版）.编者：陶辉著. 机械工业出版社,2016

[6] Hands-On Microservices with Node.Js.编者：Diogo Resende & 著. 出版社Packt Publishing,2018

[7] Vue.js in Action.编者：Eric Hanchett / Benjamin Listwon.出版社Manning Publications,2018

[8] Web前端开发技术实验与实践——HTML5、CSS3、JavaScript.编者：储 久良 著 出版社: 清华大学出版社,2018

[9] 重构JavaScript.编者: [美] 埃文 伯查德 著. 中国电力出版社,2018.

[10] 算法导论（原书第3版）.编者：[美] Thomas H.Cormen，[美] Charles E.Leiserson，[美] Ronald L.Rivest，[美] Clifford Stein 著，殷建平，徐云，王刚 等 译. 机械工业出版社,2012

[11] 数据产品设计.编者：艾达 著. 电子工业出版社,2017

[12] 数据库系统：设计、实现与管理（基础篇）（原书第6版）.编者：[英] 托马斯•康诺利（Thomas Connolly），卡洛琳•贝格（Carolyn Begg） 著，宁洪 译. 机械工业出版社,2016

[13] MySQL 8 Cookbook.编者：[美] Karthik，Appigatla 著.电子工业出版社,2018

**附录:**

整个项目系统的将最新的web开发流程展现了出来，无论是前后端分离的设计模式，还是后端采用REST接口设计风格，都是现在的主流做法。关于系统稳定性在最初的后端技术选型就考虑到了，node.js在这方面可以说是当之无愧的语言。

关于客户端的选择，也是经过多次考虑的。选择使用微信小程序最重要的就在于可以兼顾安卓和iOS用户，不需要去开发两个系统下的客户端APP。后端管理系统的选择则是因为考虑到登陆流程和vue拥有很棒的生态。

**致 谢**

本次的毕业设计是在龙老师的悉心的指导和帮助下完成的。在这近三个月的开发过程中，我们占用了老师的很多时间，给她添了很多麻烦。但是龙老师依旧没有厌烦，每次都很耐心地给我们讲解、纠错。可以说，老师在我们这次的毕业设计中花费了很大的心血。在她的耐心指导下，我们系统地学习了软件开发的过程，从需求分析、总体设计到详细设计，我们由一无所知到比较熟练的掌握点点滴滴都离不开老师的指教！在此，我代表我们向老师表示衷心的感谢，谢谢他这两个月的帮助与指导！

四年时间很快过去了，我即将毕业，心中有万般的不舍。四年来，在班主任及系、院领导的关心帮助下，我不仅学到了很多专业知识，也学到了很多为人处世的道理。在此，我深深地感谢四年来帮助过、关心过我的所有老师！

同时，我的各个方面的进步，也离不开全体同学在学习、生活各方面的支持与帮助。所以，我也要向他们表示由衷的感谢。