

《农产品批发市场大数据分析系统》

系 统 详 细 设 计

**目 录**

第一部分 引言 8

一、编写目的 8

二、项目背景 8

三、术语与缩写8

1、HTTP8

2、JSON9

3、B/S架构9

4、RESTFUL风格9

5、Vue9

6、Jsoup9

7、CRUD9

8、范式9

四、参考资料10

第二部分 项目概述11

一、项目描述 11

二、项目功能描述 11

第三部分 总体设计13

一、技术架构设计13

二、核心控制流程 14

1、核心控制流程图14

2、核心控制流程说明15

三、前后端分离（Vue+SpringBoot+MyBatis Plus）环境下的集成配置15

1、package.json15

2、config/index.js17

3、applicationContext.yml20

4、logback.xml21

第四部分 界面设计23

一、界面设计23

1、界面框架设计23

2、PC界面设计23

2.1、欢迎界面设计23

2.2、登录界面设计24

2.3、抓取量查询界面设计24

2.4、实时监测界面设计25

2.5、价格查询界面设计26

2.6、价格走势界面设计26

2.7、价格对比界面设计27

2.8、价格预测界面设计28

第五部分 接口说明文档28

一、用户登录验证接口28

二、用户注册申请接口30

三、获取农产品种类列表接口31

四、获取实时监控中种类数量接口34

五、获取某农产品各省平均价格列表接口35

六、获取省份列表接口37

七、获取农产品品种列表接口39

八、获取实时监测中品种数量接口41

九、获取抓取量查询中品种数量接口42

十、获取抓取量查询中种类占比接口45

十一、价格走势接口46

十二、获取某省份的市场列表接口49

十三、获取某批发市场所有的产品接口51

十四、获取各省批发市场数量接口54

十五、获取实时监测中批发市场数量接口56

十六、获取实时监测中总数据量接口58

十七、获取最新价格数据接口59

十八、价格查询接口61

十九、获取省内各批发市场价格对比接口63

二十、获取价格预测列表接口65

二十一、加载个人信息接口67

二十二、更改个人信息接口68

二十三、更改密码接口70

第六部分 单元模块设计71

一、数据访问层设计71

1、类图设计71

2、类的详细设计描述72

2.1 MarketMapper接口设计72

2.2 PriceMapper接口设计72

2.3 PriceTrendMapper接口设计72

2.4 ProductMapper接口设计73

2.5 ProductTypeMapper接口设计73

2.6 ProvinceMapper接口设计73

2.7 ProvinceAvpriceMapper接口设计74

2.8 TypeMapper接口设计74

2.9 UserMapper接口设计74

2.10 Mapper接口汇总74

二、业务逻辑层设计75

1、类图设计75

2、类的详细设计描述75

2.1 IMarketService接口设计75

2.2 IPriceService接口设计76

2.3 IPriceTrendService接口设计76

2.4 IProductService接口设计77

2.5 IProductTypeService接口设计77

2.6 IProvinceService接口设计77

2.7 IProvinceAvpriceService接口设计77

2.8 ITypeService接口设计77

2.9 IForecastPriceService接口设计78

2.10 IUserService接口设计78

2.11 业务接口实现类汇总78

第七部分 数据库设计79

一、数据库整体结构图79

二、用户管理79

1、user表结构79

2、admin表结构80

三、市场信息管理80

1、province表结构80

2、city表结构80

3、province\_city表结构81

4、market表结构81

四、农产品信息管理 81

1、product表结构 82

2、type表结构82

3、product\_type表结构82

4、price表结构82

五、查询业务管理83

1、province\_avprice表结构83

2、forecast\_price表结构83

3、price表结构84

4、price\_trend表结构84

第八部分 补充设计和说明85

一、配置环境设计85

二、目录结构设计86

1、前端目录结构86

2、后端目录结构87

第一部分 引言

一、编写目的

编写本设计的目的是为了准确阐述农产品大数据分析系统的具体实现思路和方法，即系统的详细架构和实现逻辑，主要包括程序系统的结构以及各层次中每个程序的设计考虑。预期读者为项目全体成员，包括运行维护和测试人员。

二、项目背景

* 系统名称：农产品大数据分析系统
* 任务提出者：略。
* 开发者：略。
* 用户和运行该程序系统的计算中心：略。

三、术语与缩写

**1、HTTP**

HyperText Transfer Protocol，超文本传输协议。是因特网上应用最为广泛的一种网络传输协议，所有的WWW文件都必须遵守这个标准。HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）的。

**2、JSON**

JSON(JavaScript Object Notation, JS 对象简谱) 是一种轻量级的数据交换格式。它基于 ECMAScript (欧洲计算机协会制定的js规范)的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。并且我们也可以使用JSON数据格式来实现前后端分离。

**3、B/S架构**

B/S架构即浏览器和服务器架构模式，是随着Internet技术的兴起，对C/S架构的一种变化或者改进的架构。在这种架构下，用户工作界面是通过WWW浏览器来实现，极少部分事务逻辑在前端（Browser）实现，但是主要事务逻辑在服务器端(Server)实现，形成所谓三层3-tier结构。B/S架构是WEB兴起后的一种网络架构模式，WEB浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。

**4、RESTFUL风格**

REST，即Representational State Transfer的缩写。直接翻译的意思是"表现层状态转化"。

它是一种互联网应用程序的API设计理念：URL定位资源，用HTTP动词GET、POST、PUT、DELETE等方式对于服务器的资源进行操作。

**5、Vue**

Vue 是一套用于构建用户界面的渐进式JavaScript框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，方便与第三方库或既有项目整合。

**6、Jsoup**

Jsoup是一款Java的HTML解析器，可以直接解析某个URL地址，可以通过DOM、CSS以及类似于jQuery的操作方法来取出数据。

**7、CRUD**

CRUD其实是数据库基本操作中的Create(创建)、Read(读取)、Update（更新）、Delete（删除）

**8、范式**

范式是符合某一种级别的关系模式的集合，表示一个关系内部各属性之间的联系的合理化程度。数据库范式分为1NF，2NF，3NF，BCNF，4NF，5NF。一般在我们设计关系型数据库的时候，最多考虑到BCNF。符合高一级范式的设计，必定符合低一级范式。

**1 第一范式（1NF）**

在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。

所谓第一范式（1NF）是指数据库[表](https://baike.baidu.com/item/%E8%A1%A8/9997188)的每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。如果出现重复的属性，就可能需要定义一个新的实体，新的实体由重复的属性构成，新实体与原实体之间为一对多关系。在第一范式（1NF）中表的每一行只包含一个实例的信息。

简而言之，第一范式就是无重复的列。

**2 第二范式（2NF）**

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或行必须可以被唯一地区分。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。这个唯一属性列被称为主关键字或主键、主码。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性。如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。

简而言之，第二范式就是非主属性完全依赖于主关键字。

**3 第三范式（3NF）**

满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

例如，存在一个部门信息表，其中每个部门有部门编号（dept\_id）、部门名称、部门简介等信息。那么在图3-2的员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。

简而言之，第三范式就是属性不依赖于其它非主属性。

四、参考资料

列出有关的参考资料，如：

1. 本项目的经核准的计划任务书或合同、上级机关的批文；

《技术服务合同》

《项目章程》

《里程碑计划》

《开发计划》

《交付物清单》

2. 属于本项目的其他已发表的文件；

《需求说明》

《概要设计》

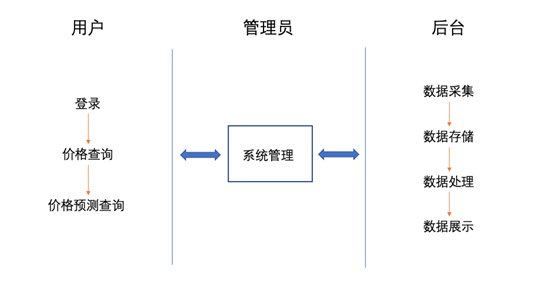
3. 本文件中各处引用到的文件资料，包括所要用到的软件开发标准。

《Java语言编码规范(Java Code Conventions)》

第二部分 项目概述

一、项目描述

近几年随着电子商务的兴起，农产品的网上销售也模式也逐渐发展起来，各种涉农信息的发布也改变了传统模式，直接利用网络资源进行宣传与销售。而原有的大型果蔬农产品批发市场，也逐步改变原有的管理销售模式，采用新的电子商务销售系统，对农产品交易和农资信息发布进行有效的整合，为人们在线进行农产品交易提供虚拟网络环境和保障农产品电子商务顺利运营的管理环境。 在此，为实现一个能为各地果蔬交易商提供及时有效的农产品信息，为了能够以更快更便捷的方式向大家展示农产品的各类信息，在保证信息真实可靠的前提下，决定利用各地各大果蔬农产品批发市场的交易系统所提供的交易信息，开发一款农产品交易市场的大数据分析系统。



二、项目功能描述

项目功能主要分为管理员功能和普通用户功能，管理员功能包括管理员登陆、数据维护、数据检索，而普通用户功能包括用户注册、用户登录、数据查询、请求监控数据、查询预测价格。

**1、管理员登陆：**用户以管理员身份登入系统，拥有管理员功能。

**2、数据维护：**数据维护是管理员向服务器发出请求，对数据库中的一些新的、过期或者有损数据进行新增、更新、删除。

**3、数据检索：**数据检索是管理员向服务器提交查询请求，输入要查询的商品名以及查询的详细条件可以得到一个系统返回结果。

**4、用户注册：**用户注册是游客用户对网站申请账号的功能。

**5、用户登录：**用户以用户名以及密码登入系统，拥有普通用户的功能。

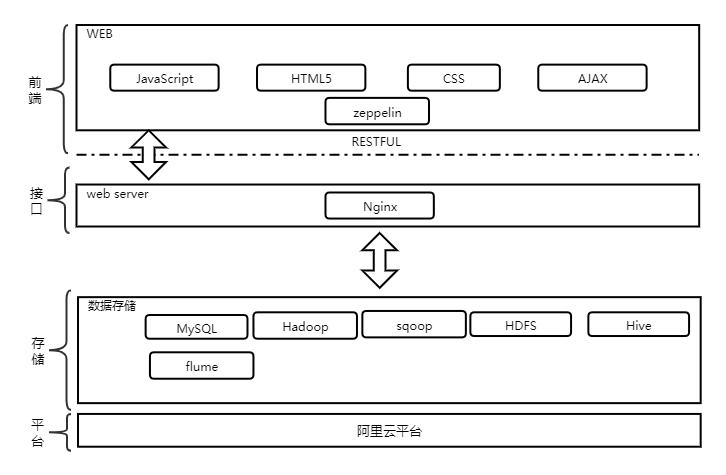
**6、数据查询：**数据查询是用户向服务器提交查询请求，输入要查询的商品名以及查询的详细条件可以得到一个系统返回结果，其中包括商品价格、商品价格走势、商品同期价格对比。

**7、请求监控数据：**请求监控数据功能是用户查看今日的商品数据爬取情况，包括市场总数、商品总数等等。

**8、查询预测价格：**查询预测价格是用户提交商品名以及详细条件后系统对未来七日的价格做出预测并展示，对用户估价起到参考作用

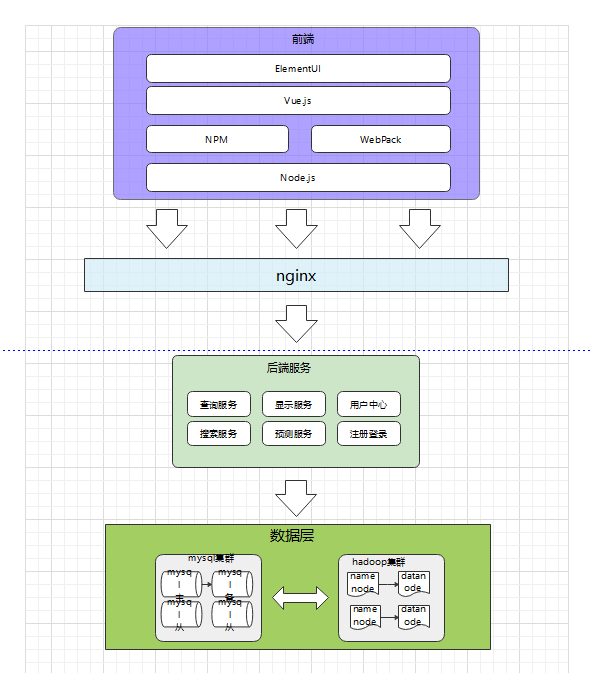
第三部分 总体设计

一、技术架构设计



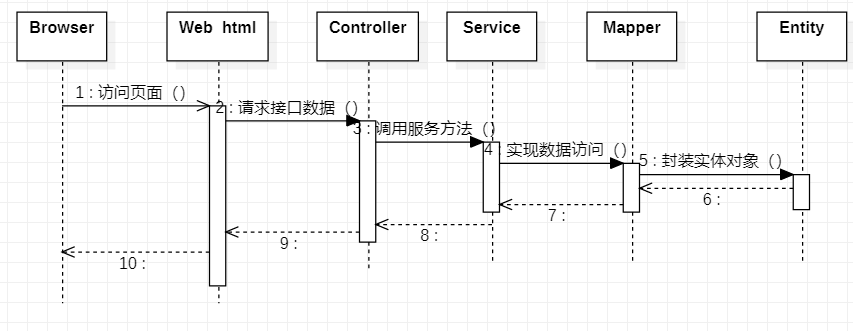
系统采用前端web界面后端数据存储的架构，整个项目使用阿里云的云服务器构建，采用hadoop分布式数据存储管理MySQL中的农产品数据，采用hadoop技术的优点在于可以方便地扩展到数以千计的节点中，具有高扩展性。Hadoop能够在节点之间进行动态地移动数据，并保证各个节点的动态平衡，处理速度非常快，具有高效性。

其开发的结构图如下：



二、核心控制流程

1、核心控制流程图



2、核心控制流程说明

结合上图，针对每一个消息请求，按照消息请求/回复顺序，作如下说明。

|  |  |  |
| --- | --- | --- |
| **编号** | **消息名称** | **消息说明** |
| 1 | 访问页面 | 访问前端界面 |
| 2 | 请求接口数据 | 将前端的请求发向后端控制器 |
| 3 | 调用服务方法 | 控制器层根据请求的接口调用相应的服务 |
| 4 | 实现数据访问 | 在服务层里调用数据访问层的方法 |
| 5 | 封装实体对象 | 根据Mapper添加对象到结果 |
| 6 | 5的返回消息 | 封装的对象 |
| 7 | 4的返回消息 | 封装的对象集合 |
| 8 | 3的返回消息 | 服务调用获得的结果 |
| 9 | 2的返回消息 | JSON格式的请求返回结果 |
| 10 |  | 数据刷新并显示在浏览器上 |

三、前后端分离（Vue+SpringBoot+MyBatis Plus）环境下的集成配置

1、package.json

|  |
| --- |
| {  "name": "market\_vue",  "version": "1.0.0",  "description": "A Vue.js project",  "author": "group09",  "private": true,  "scripts": {  "dev": "webpack-dev-server --inline --progress --config build/webpack.dev.conf.js",  "start": "npm run dev",  "build": "node build/build.js"  },  "dependencies": {  "axios": "^0.19.2",  "echarts": "^4.8.0",  "element-ui": "^2.13.2",  "vue": "^2.5.2",  "vue-router": "^3.0.1"  },  "devDependencies": {  "autoprefixer": "^7.1.2",  "babel-core": "^6.22.1",  "babel-helper-vue-jsx-merge-props": "^2.0.3",  "babel-loader": "^7.1.1",  "babel-plugin-syntax-jsx": "^6.18.0",  "babel-plugin-transform-runtime": "^6.22.0",  "babel-plugin-transform-vue-jsx": "^3.5.0",  "babel-preset-env": "^1.3.2",  "babel-preset-stage-2": "^6.22.0",  "chalk": "^2.0.1",  "copy-webpack-plugin": "^4.0.1",  "css-loader": "^0.28.11",  "extract-text-webpack-plugin": "^3.0.0",  "file-loader": "^1.1.11",  "friendly-errors-webpack-plugin": "^1.6.1",  "html-webpack-plugin": "^2.30.1",  "node-notifier": "^5.1.2",  "optimize-css-assets-webpack-plugin": "^3.2.0",  "ora": "^1.2.0",  "portfinder": "^1.0.13",  "postcss-import": "^11.0.0",  "postcss-loader": "^2.0.8",  "postcss-url": "^7.2.1",  "rimraf": "^2.6.0",  "semver": "^5.3.0",  "shelljs": "^0.7.6",  "style-loader": "^1.2.1",  "uglifyjs-webpack-plugin": "^1.1.1",  "url-loader": "^0.5.8",  "vue-loader": "^13.3.0",  "vue-style-loader": "^3.0.1",  "vue-template-compiler": "^2.5.2",  "webpack": "^3.6.0",  "webpack-bundle-analyzer": "^2.9.0",  "webpack-dev-server": "^2.9.1",  "webpack-merge": "^4.1.0"  },  "engines": {  "node": ">= 6.0.0",  "npm": ">= 3.0.0"  },  "browserslist": [  "> 1%",  "last 2 versions",  "not ie <= 8"  ]  } |

2、config/index.js

|  |
| --- |
| 'use strict'  // Template version: 1.3.1  // see <http://vuejs-templates.github.io/webpack> for documentation.  const path = require('path')  ***module***.***exports*** = {  dev: {  // Paths  assetsSubDirectory: 'static',  assetsPublicPath: '/',  proxyTable: {  '/api': {  target: 'http://localhost:8081', //源地址  changeOrigin: true, //改变源  pathRewrite: {  '^/api': '' //路径重写  }  }  },  // Various Dev Server settings  host: 'localhost', // can be overwritten by process.env.HOST  port: 8080, // can be overwritten by process.env.PORT, if port is in use, a free one will be determined  autoOpenBrowser: false,  errorOverlay: true,  notifyOnErrors: true,  poll: false, // <https://webpack.js.org/configuration/dev-server/#devserver-watchoptions->  */\*\**  *\* Source Maps*  *\*/*  // <https://webpack.js.org/configuration/devtool/#development>  devtool: 'cheap-module-eval-source-map',  // If you have problems debugging vue-files in devtools,  // set this to false - it \*may\* help  // <https://vue-loader.vuejs.org/en/options.html#cachebusting>  cacheBusting: true,  cssSourceMap: true  },  build: {  // Template for index.html  index: path.resolve(\_\_dirname, '../dist/index.html'),  // Paths  assetsRoot: path.resolve(\_\_dirname, '../dist'),  assetsSubDirectory: 'static',  assetsPublicPath: '/',  */\*\**  *\* Source Maps*  *\*/*  productionSourceMap: true,  // <https://webpack.js.org/configuration/devtool/#production>  devtool: '#source-map',  // Gzip off by default as many popular static hosts such as  // Surge or Netlify already gzip all static assets for you.  // Before setting to `true`, make sure to:  // npm install --save-dev compression-webpack-plugin  productionGzip: false,  productionGzipExtensions: ['js', 'css'],  // Run the build command with an extra argument to  // View the bundle analyzer report after build finishes:  // `npm run build --report`  // Set to `true` or `false` to always turn it on or off  bundleAnalyzerReport: process.env.npm\_config\_report  }  } |

3、applicationContext.yml

|  |
| --- |
| Server:  port: 8081  spring:  application:  name: market  aop:  proxy-target-class: true  datasource:  type: com.alibaba.druid.pool.DruidDataSource  driver-class-name: com.mysql.cj.jdbc.Driver  url: jdbc:mysql://47.92.195.80:3306/agricultural\_products\_database?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Shanghai&useSSL=false  username: root  password: WHU\_group09  druid:  validation-query: SELECT 1 FROM DUAL  initial-size: 10  min-idle: 10  max-active: 200  min-evictable-idle-time-millis: 300000  test-on-borrow: false  test-while-idle: true  time-between-eviction-runs-millis: 30000  pool-prepared-statements: true  max-open-prepared-statements: 100  mybatis-plus:  type-aliases-package: com.ada.marketbackend.\*.entity  global-config:  db-config:  id-type: *auto*  logic-delete-field: 1  logic-not-delete-value: 0  configuration:  map-underscore-to-camel-case: true  cache-enabled: false  jdbc-type-for-null: '*null*'  log-impl: org.apache.ibatis.logging.stdout.StdOutImpl  logging:  config:  classpath: logback.xml  logging.level.org.springframework.baoot.autoconfigure: ERROR |

4、logback.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <configuration>  <property name="LOG\_PATTERN"  value="%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -%msg%n" />  <property name="LOG\_LEVEL" value="INFO"/>  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  <encoder>  <pattern>${LOG\_PATTERN}</pattern>  </encoder>  <filter class="ch.qos.logback.classic.filter.ThresholdFilter">  <level>${LOG\_LEVEL}</level>  </filter>  </appender>  <root level="${LOG\_LEVEL}">  <appender-ref ref="STDOUT" />  </root>  </configuration> |

第四部分 界面设计

一、界面设计

1、界面框架设计

ElementUI

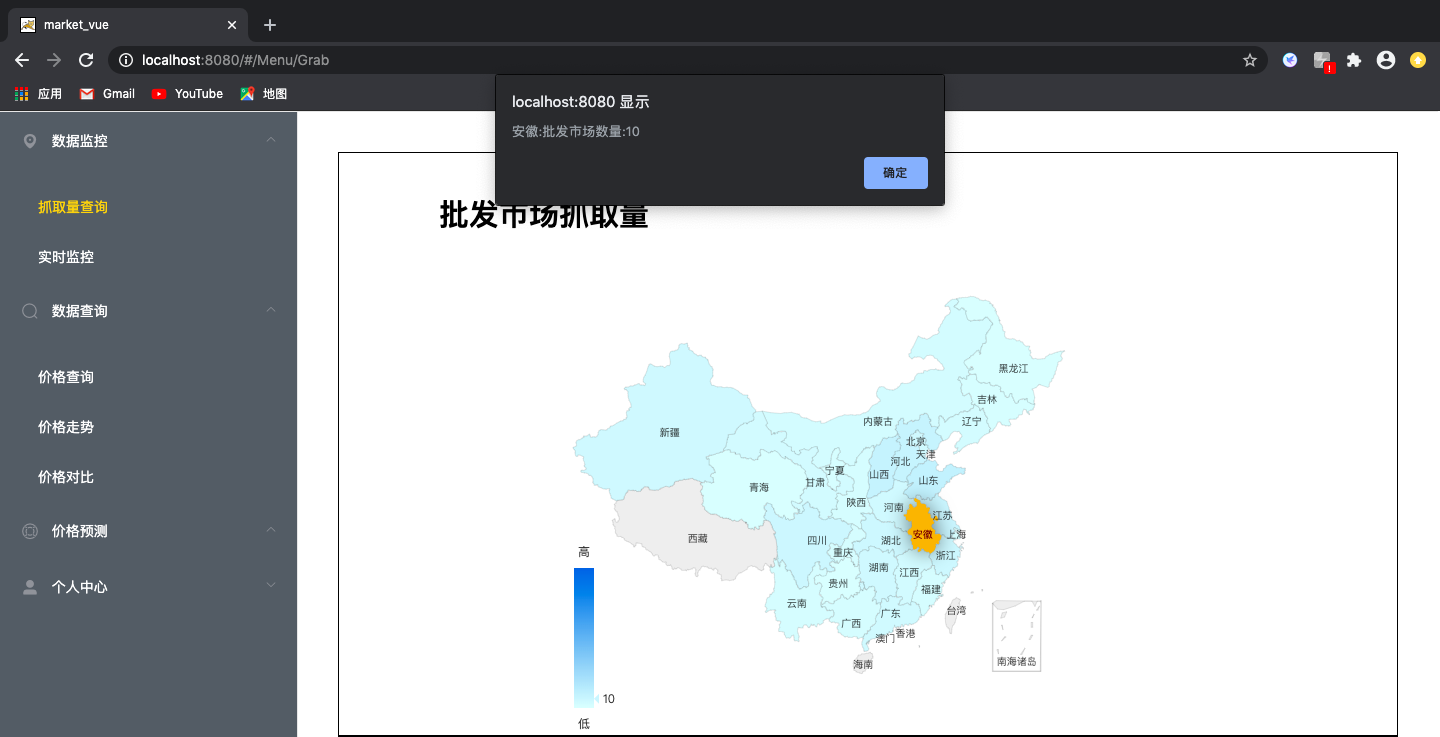
2、PC界面设计

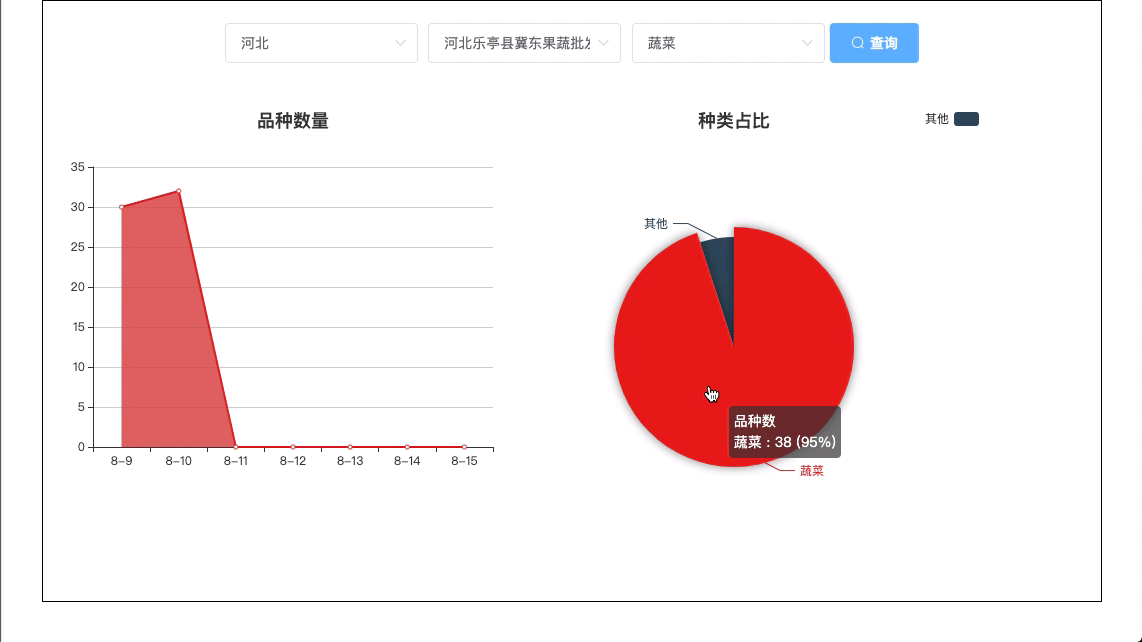
**2.1、欢迎界面设计**

2.2、登录界面设计



2.3、抓取量查询界面设计

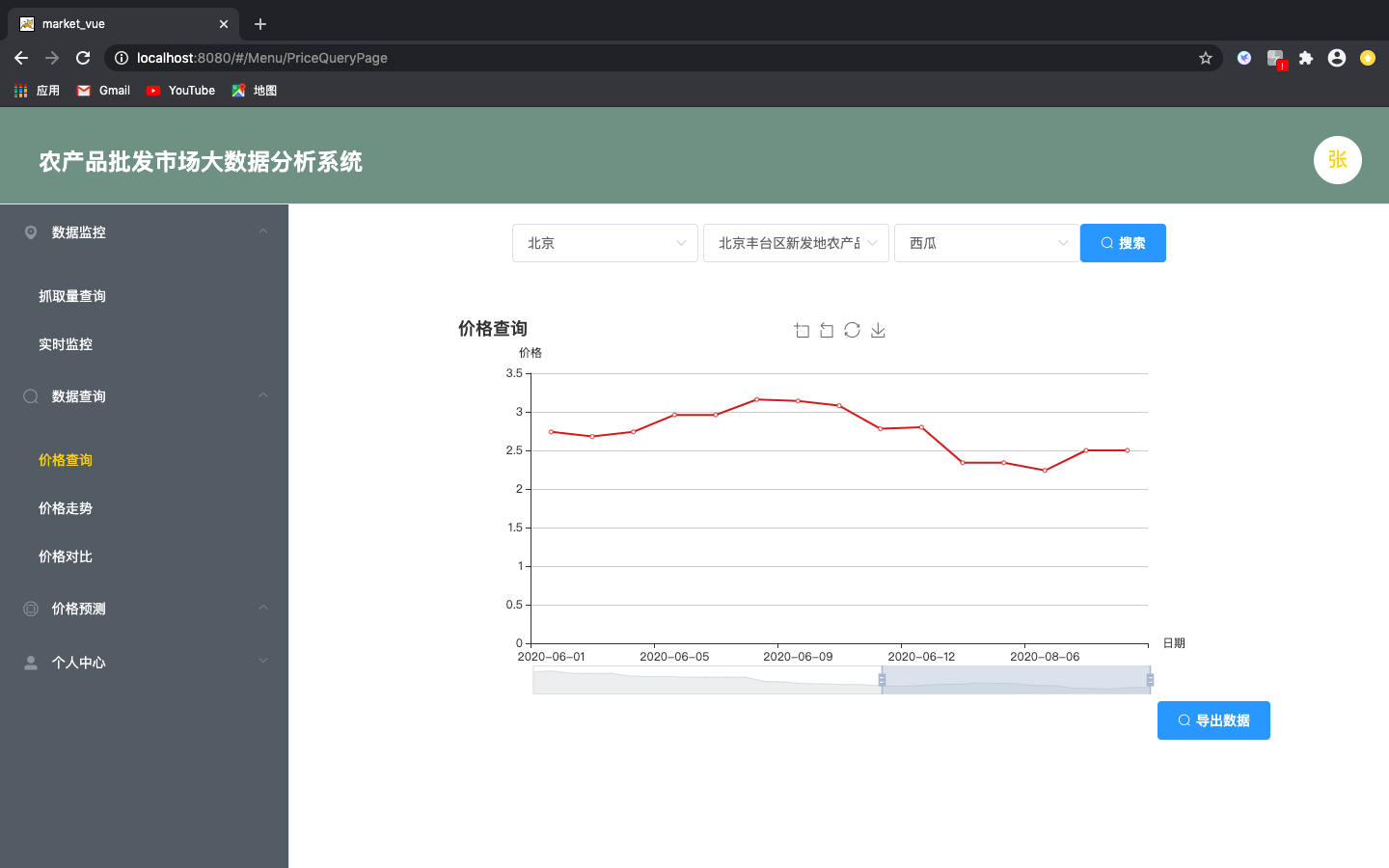




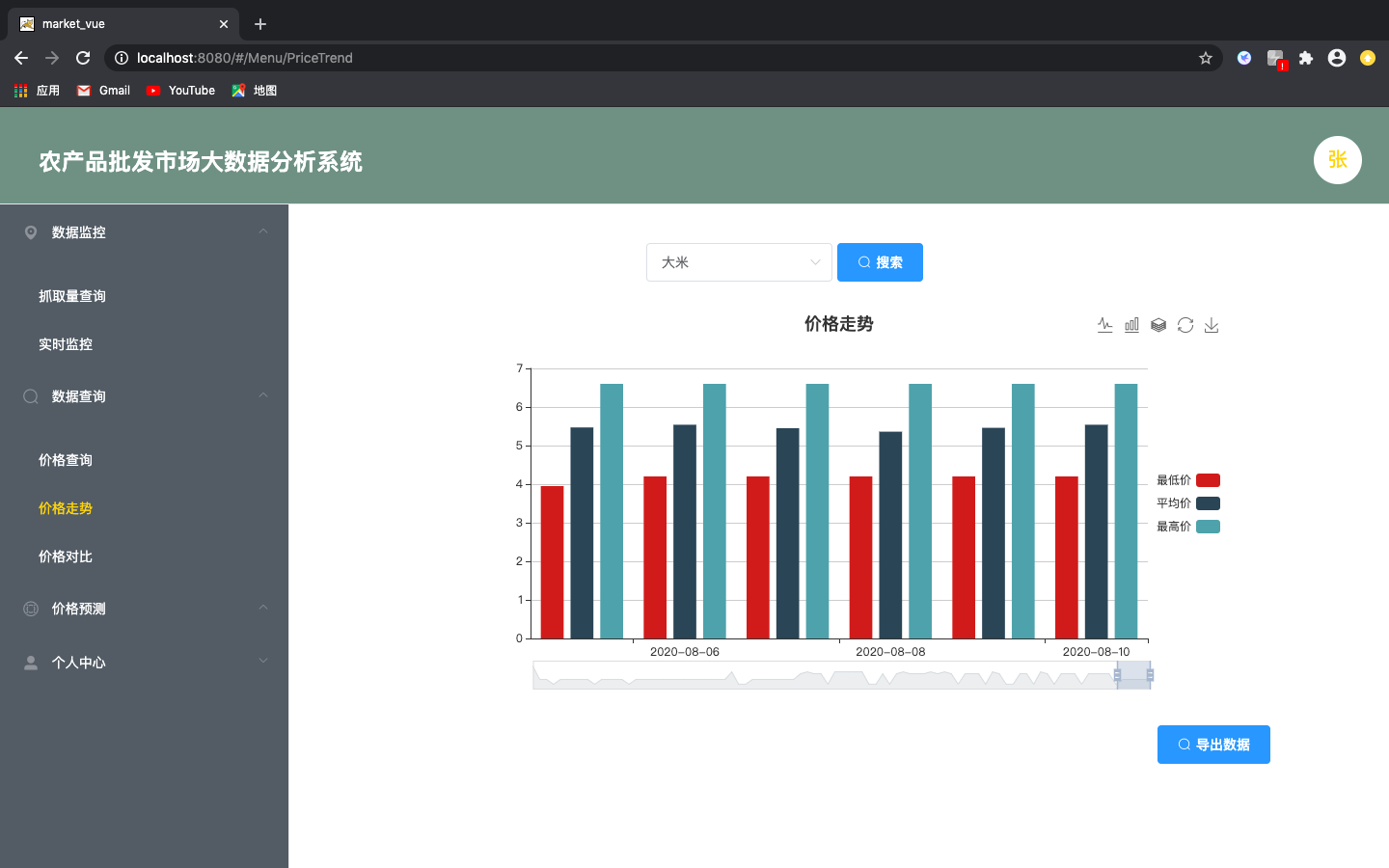
2.4、实时监测界面设计



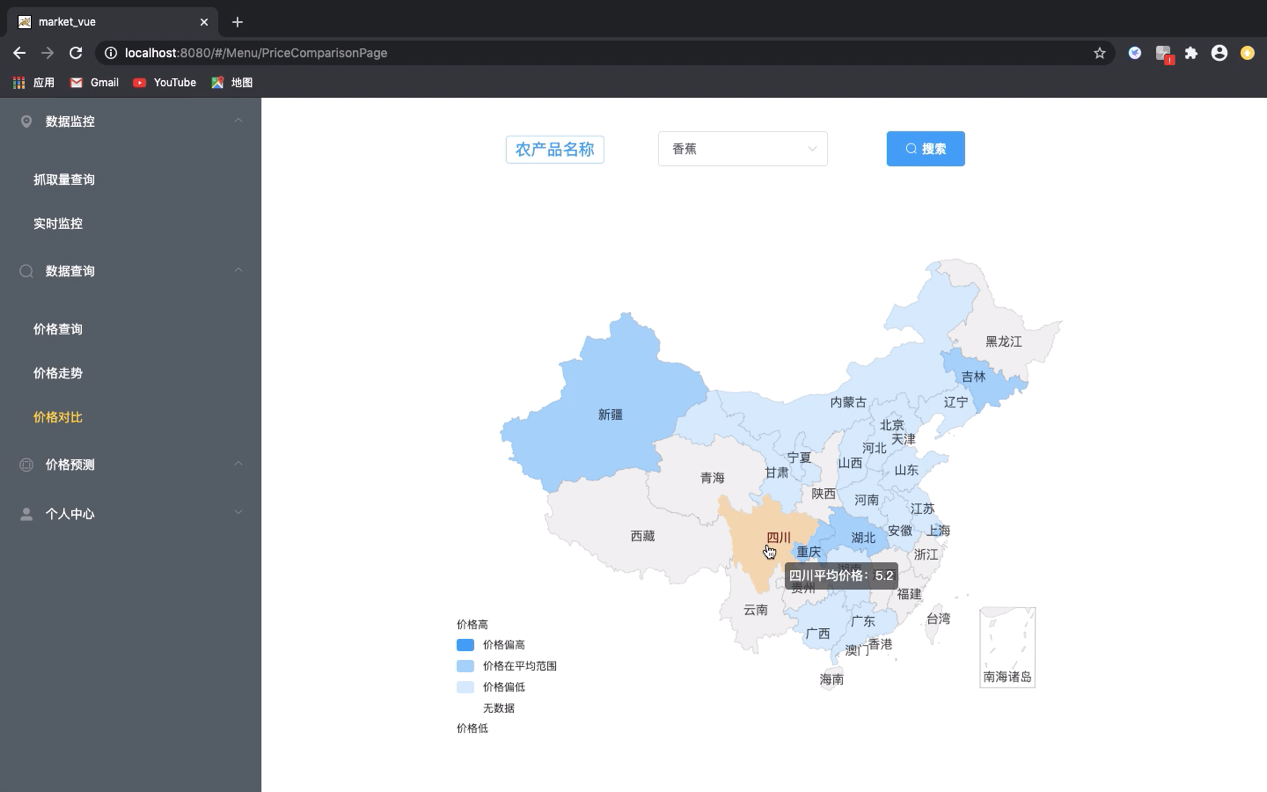
2.5、价格查询界面设计

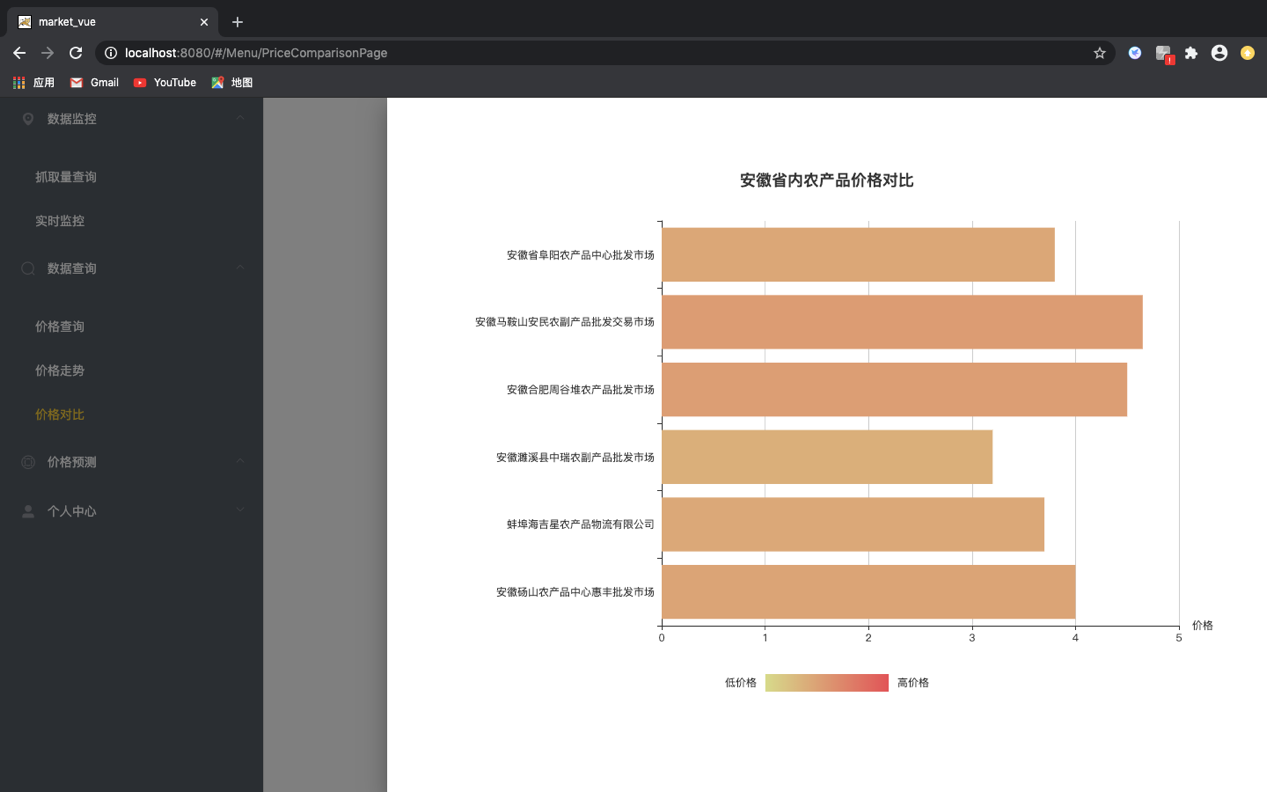


2.6、价格走势界面设计



2.7、价格对比界面设计





2.8、价格预测界面设计



第五部分 接口说明文档

一、用户登录验证接口

**1、接口功能**

对登录页面中用户输入的用户名和密码进行验证

**2、接入URL**

<http://localhost:8081/user/loginCheck>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 用户名 | userName | true | String | 用户登录时输入的用户名 |
| 用户密码 | userPass | true | String | 用户登录时输入的密码 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| status | String | 返回结果状态。200:正常；500:错误 |
| msg | String | 返回结果信息 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/user/loginCheck?userName=aaaa&userPass=12345>

如果用户名aaaa以及密码12345均正确，则返回结果为

{

"status": "200",

"msg": "Success"

}

如果用户名aaaa或者密码12345错误，则返回结果为

{

"status": "500",

"errorMsg": "Fail"

}

二、用户注册申请接口

**1、接口功能**

注册新用户

**2、接入URL**

<http://localhost:8081/user/regAdd>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 用户名 | userName | true | String | 用户注册时输申请的用户名 |
| 用户密码 | userPass | true | String | 用户注册时申请的密码 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| status | String | 返回结果状态。200:正常；500:错误 |
| msg | String | 返回结果信息 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/user/regAdd?userName=abcd&userPass=123456789&userPhone=null>

如果注册成功，则返回结果为

{

"status": "200",

"msg": "Success"

}

如果用户名abcd已经存在，则返回结果为

{

"status": "500",

"errorMsg": "用户名已存在"

}

如果是其他原因注册失败，则返回结果为

{

"status": "500",

"errorMsg": "注册失败"

}

三、获取农产品种类列表接口

**1、接口功能**

获取农产品所有的种类

**2、接入URL**

<http://localhost:8081/type/getTypeList>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| typeID | String | 返回农产品种类对应的ID编号 |
| typeName | String | 返回农产品种类的名称 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/type/getTypeList>

返回结果为

{

    "data": [

        {

            "typeID": "13073",

            "typeName": "粮油"

        },

        {

            "typeID": "13075",

            "typeName": "蔬菜"

        },

        {

            "typeID": "13076",

            "typeName": "水果"

        },

        {

            "typeID": "13077",

            "typeName": "花"

        },

        {

            "typeID": "13079",

            "typeName": "畜产品"

        },

        {

            "typeID": "13080",

            "typeName": "水产品"

        }

    ],

    "status": "200"

}

四、获取实时监控中种类数量接口

**1、接口功能**

获取实时监控中的种类数量，即统计最新抓取数据中所有农产品的种类数量

**2、接入URL**

<http://localhost:8081/type/getTypeNumber>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| result | int | 返回最新爬取数据中所有农产品的种类数量 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/type/getTypeNumber>

返回结果为

{

    "data": [

        {

            "result": 5

        }

    ],

    "status": "200"

}

五、获取某农产品各省平均价格列表接口

**1、接口功能**

获取某农产品各省最新的平均价格列表

**2、接入URL**

<http://localhost:8081/province_avprice/getProvinceAvpriceByProduct>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 农产品名称 | productName | true | String | 要获取各省平均价格列表的农产品名称 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| name | String | 省份的名称 |
| value | double | 该农产品在对应省份的平均价格 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/province_avprice/getProvinceAvpriceByProduct?productName=西瓜>

返回结果为

{

    "data": [

        {

            "name": "北京",

            "value": 10

        },

        {

            "name": "湖北",

            "value": 5

        }

    ],

    "status": 200

}

六、获取省份列表接口

**1、接口功能**

获取省份列表

**2、接入URL**

<http://localhost:8081/province/getProvinceList>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| provinceID | String | 返回省份对应的ID编号 |
| provinceName | String | 返回省份名称 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/province/getProvinceList>

返回结果为

{"data":[{"provinceID":"11","provinceName":"北京"},{"provinceID":"12","provinceName":"天津"},{"provinceID":"13","provinceName":"河北"},{"provinceID":"14","provinceName":"山西"},{"provinceID":"15","provinceName":"内蒙古"},{"provinceID":"21","provinceName":"辽宁"},{"provinceID":"22","provinceName":"吉林"},{"provinceID":"23","provinceName":"黑龙江"},{"provinceID":"31","provinceName":"上海"},{"provinceID":"32","provinceName":"江苏"},{"provinceID":"33","provinceName":"浙江"},{"provinceID":"34","provinceName":"安徽"},{"provinceID":"35","provinceName":"福建"},{"provinceID":"36","provinceName":"江西"},{"provinceID":"37","provinceName":"山东"},{"provinceID":"41","provinceName":"河南"},{"provinceID":"42","provinceName":"湖北"},{"provinceID":"43","provinceName":"湖南"},{"provinceID":"44","provinceName":"广东"},{"provinceID":"45","provinceName":"广西"},{"provinceID":"46","provinceName":"海南"},{"provinceID":"50","provinceName":"重庆"},{"provinceID":"51","provinceName":"四川"},{"provinceID":"52","provinceName":"贵州"},{"provinceID":"53","provinceName":"云南"},{"provinceID":"54","provinceName":"西藏"},{"provinceID":"61","provinceName":"陕西"},{"provinceID":"62","provinceName":"甘肃"},{"provinceID":"63","provinceName":"青海"},{"provinceID":"64","provinceName":"宁夏"},{"provinceID":"65","provinceName":"新疆"},{"provinceID":"99","provinceName":"新疆建设兵团"}],"status":200}

七、获取农产品品种列表接口

**1、接口功能**

获取农产品品种列表

**2、接入URL**

<http://localhost:8081/product/getProductList>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| productID | String | 返回农产品品种对应的ID编号 |
| productName | String | 返回农产品品种的名称 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/product/getProductList>

返回结果为

{

    "data": [

        {

            "productID": "13083",

            "productName": "籼米"

        },

        {

            "productID": "9060834",

            "productName": "葡萄干(二级)"

        },

        {

            "productID": "9060839",

            "productName": "晚籼稻"

        },

        {

            "productID": "9060841",

            "productName": "早籼稻"

        },

        {

            "productID": "9060853",

            "productName": "枸杞一级"

        }

    ],

    "status": 200

}

八、获取实时监测中品种数量接口

**1、接口功能**

获取实时监测中的品种数量，即统计最新抓取数据中所有农产品的品种数量

**2、接入URL**

<http://localhost:8081/product/getLatestProductNumber>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| result | int | 返回最新爬取数据中所有农产品的品种数量 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/product/getLatestProductNumber>

返回结果为

{

    "data": [

        {

            "result": 262

        }

    ],

    "status": "200"

}

九、获取抓取量查询中品种数量接口

**1、接口功能**

在抓取量查询中，获取最近7天每一天的某批发市场的某种类的所有农产品品种的数量

**2、接入URL**

<http://localhost:8081/product/getProductNoByType>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 市场名称 | marketName | true | String | 要获取相应数据的批发市场名称 |
| 农产品种类 | type | true | String | 要获取相应数据的农产品种类名称 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| date | Date | 日期 |
| value | int | 对应日期的农产品品种数量 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/product/getProductNoByType?marketName=A市场&type=水果>

返回结果为

{

    "data": [

        {

            "date": "2020-08-13",

            "value": 20

        },

        {

            "date": "2020-08-12",

            "value": 21

        },

        {

            "date": "2020-08-11",

            "value": 25

        },

        {

            "date": "2020-08-10",

            "value": 27

        },

        {

            "date": "2020-08-09",

            "value": 30

        },

        {

            "date": "2020-08-08",

            "value": 30

        },

        {

            "date": "2020-08-07",

            "value": 30

        }

    ],

    "status": "200"

}

十、获取抓取量查询中种类占比接口

**1、接口功能**

在抓取量查询中，获取所有抓取数据中某批发市场的某农产品种类的占比

**2、接入URL**

<http://localhost:8081/product/getTypePercents>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 市场名称 | marketName | true | String | 要获取种类占比数据的市场名称 |
| 农产品种类 | type | true | String | 要获取相应种类占比的农产品种类名 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| name | String | 有两个值，请求参数中的type值和”其他“ |
| value | int | name对应的数据数量 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/product/getTypePercents?marketName=北京丰台区新发地农产品批发市场&type=水果>

返回结果为

{

    "data": [

        {

            "name": "其他",

            "value": 38

        },

        {

            "name": "水果",

            "value": 4

        }

    ],

    "status": "200"

}

十一、价格走势接口

**1、接口功能**

获取某农产品品种的价格走势，包括最大值、最小值以及平均值

**2、接入URL**

[http://localhost:8081/price\_trend/getPriceTrend](http://localhost:8081/price_trend/getPriceTrendByProductID)

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 农产品品种编号 | productID | true | String | 要获取相应数据的农产品品种编号 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| average | double | 相应农产品品种的平均价格 |
| date | Date | 该条数据对应的日期 |
| highest | double | 相应农产品品种的最高价格 |
| lowest | double | 相应农产品品种的最低价格 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/price_trend/getPriceTrend?productID=13084>

返回结果为

{

    "data": [

        {

            "average": 5.47,

            "date": "2020-08-05",

            "highest": 6.6,

            "lowest": 3.95

        },

        {

            "average": 5.54,

            "date": "2020-08-06",

            "highest": 6.6,

            "lowest": 4.2

        },

        {

            "average": 5.45,

            "date": "2020-08-07",

            "highest": 6.6,

            "lowest": 4.2

        },

        {

            "average": 5.36,

            "date": "2020-08-08",

            "highest": 6.6,

            "lowest": 4.2

        },

        {

            "average": 5.46,

            "date": "2020-08-09",

            "highest": 6.6,

            "lowest": 4.2

        },

        {

            "average": 5.54,

            "date": "2020-08-10",

            "highest": 6.6,

            "lowest": 4.2

        }

    ],

    "status": 200

}

十二、获取某省份的市场列表接口

**1、接口功能**

获取某省份的市场列表

**2、接入URL**

<http://localhost:8081/market/getMarketByProvinceName>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 省份名称 | provinceName | true | String | 要获取市场列表的省份名称 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| marketID | String | 返回批发市场对应的编号 |
| marketName | String | 返回批发市场对应的市场名 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/market/getMarketByProvinceName?provinceName=湖北>

返回结果为

{

    "data": [

        {

            "marketID": "20687",

            "marketName": "湖北洪湖市农贸市场"

        },

        {

            "marketID": "20690",

            "marketName": "湖北浠水城北农产品批发大市场"

        },

        {

            "marketID": "2550029",

            "marketName": "湖北孝感市南大批发市场"

        },

        {

            "marketID": "8756123",

            "marketName": "湖北两湖平原农产品交易物流中心"

        }

    ],

    "status": 200

}

**十三、获取某批发市场所有的产品接口**

**1、接口功能**

获取一个批发市场的农产品列表

**2、接入URL**

<http://localhost:8081/product/getProductListByMarketID>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 市场ID | marketID | 是 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
|  | String | 返回省份的名称 |
| value | int | 返回对应省份的批发市场的数量 |

**7、接口示例**

请求提交方法为：[http://localhost:8081/product/getProductListByMarketID?marketID=21035](http://localhost:8081/product/getProductListByMarketID?marketID=21035" \t "_blank)

返回结果为

{

"data": [

{

"productID": "13084",

"productName": "大米"

},

{

"productID": "13087",

"productName": "大豆"

},

{

"productID": "13095",

"productName": "面粉"

},

{

"productID": "13096",

"productName": "绿豆"

},

{

"productID": "13097",

"productName": "富士苹果"

},

{

"productID": "13099",

"productName": "红小豆"

},

{

"productID": "13100",

"productName": "甜橙"

},

{

"productID": "13103",

"productName": "香蕉"

},

.......

}

十四、获取各省批发市场数量接口

**1、接口功能**

获取各个省份的批发市场的数量

**2、接入URL**

<http://localhost:8081/market/getMarketNoByProvince>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| name | String | 返回省份的名称 |
| value | int | 返回对应省份的批发市场的数量 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/market/getMarketNoByProvince>

返回结果为

{

    "data": [

        {

            "name": "湖南",

            "value": 7

        },

        {

            "name": "广东",

            "value": 6

        },

        {

            "name": "广西",

            "value": 3

        },

        {

            "name": "重庆",

            "value": 3

        },

        {

            "name": "四川",

            "value": 9

        },

        {

            "name": "新疆",

            "value": 7

        },

        {

            "name": "新疆建设兵团",

            "value": 1

        }

    ],

    "status": "200"

}

十五、获取实时监测中批发市场数量接口

**1、接口功能**

获取最新抓取数据中批发市场的数量

**2、接入URL**

<http://localhost:8081/market/getLatestMarketNumber>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| result | int | 返回最新抓取数据中批发市场数量 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/market/getLatestMarketNumber>

返回结果为

{

    "data": [

        {

            "result": 2

        }

    ],

    "status": "200"

}

十六、获取实时监控中总数据量接口

**1、接口功能**

获取最新抓取数据中总的数据量

**2、接入URL**

<http://localhost:8081/price/getTotalDataVolume>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| result | int | 返回最新抓取数据中总的数据量 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/price/getTotalDataVolume>

返回结果为

{

    "data": [

        {

            "result": 10

        }

    ],

    "status": "200"

}

十七、获取最新价格数据接口

**1、接口功能**

获取最新抓取的价格数据

**2、接入URL**

<http://localhost:8081/price/getLatestPrice>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 无 | 无 | 无 | 无 | 无 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| market | String | 数据对应的市场名称 |
| product | String | 数据对应的品种名称 |
| date | Date | 数据对应的日期 |
| province | String | 数据对应的省份 |
| price | double | 数据对应的价格 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/price/getLatestPrice>

返回结果为

{

    "data": [

        {

            "market": "山东临邑县临南蔬菜大市场",

            "date": "2020-08-14",

            "product": "国光苹果",

            "province": "山东",

            "price": 4

        },

        {

            "market": "北京丰台区新发地农产品批发市场",

            "date": "2020-08-14",

            "product": "香蕉",

            "province": "北京",

            "price": 5

        }

    ],

    "status": "200"

}

十八、价格查询接口

**1、接口功能**

获取指定批发市场的指定农产品的所有时间的价格列表

**2、接入URL**

<http://localhost:8081/price/getPriceByMarketAndProduct>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 批发市场名 | marketName | true | String | 用户在价格查询页面中选中的批发市场 |
| 农产品品种名 | productName | true | String | 用户在价格查询页面中选中的农产品品种 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| date | Date | 返回数据的日期 |
| price | double | 返回查询的价格 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/price/getPriceByMarketAndProduct?marketName=北京顺义区顺鑫石门农产品批发市场&productName=大米>

返回结果为

{

    "data": [

        {

            "date": "2020-08-03",

            "price": 4.9

        },

        {

            "date": "2020-08-04",

            "price": 4.9

        },

        {

            "date": "2020-08-05",

            "price": 4.9

        },

        {

            "date": "2020-08-06",

            "price": 4.9

        },

        {

            "date": "2020-08-07",

            "price": 4.9

        },

        {

            "date": "2020-08-08",

            "price": 4.9

        },

        {

            "date": "2020-08-09",

            "price": 4.9

        }

    ],

    "status": 200

}

十九、获取省内各批发市场价格对比接口

**1、接口功能**

获取某特定省份内各批发市场最近一天的价格列表

**2、接入URL**

<http://localhost:8081/price/getPriceByProvinceAndProduct>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 省份名 | provinceName | true | String | 价格对比页面中用户选中的省份 |
| 农产品品种名 | productName | true | String | 价格对比页面中用户选择的农产品品种 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| marketName | String | 返回的市场名 |
| price | double | 对应市场的价格 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/price/getPriceByProvinceAndProduct?provinceName=河北&productName=面粉>

返回结果为

{

    "data": [

        {

            "marketName": "荷花坑批发市场",

            "price": 5.8

        }

    ],

    "status": 200

}

**二十、获取价格预测列表接口**

**1、接口功能**

获取某省份的某市场中某产品的价格列表，包括日期、预测价格、实际价格信息

**2、接入URL**

<http://localhost:8081/forecast_price/getForecastPriceByMarketAndProduct>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数 | 必选 | 类型 | 说明 |
| 市场ID | marketID | true | String | 价格预测页面中用户选中的市场 |
| 产品ID | productID | true | String | 价格预测页面中用户选择的农产品品种 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 返回字段 | 字段类型 | 说明 |
| date | String | 日期 |
| forprice | String | 某市场当日该产品的预测价格 |
| price | String | 某市场当日该产品的价格 |

**7、接口示例**

请求提交方法为：[http://localhost:8081/forecast\_price/getForecastPriceByMarketAndProduct?marketID=20531&productID=13083](http://localhost:8081/forecast_price/getForecastPriceByMarketAndProduct?marketID=20531&productID=13083" \t "_blank)

返回结果为

{

"data": [

{

"date": "2020-08-09",

"forprice": 3.1,

"price": 3.2

},

{

"date": "2020-08-10",

"forprice": 3.2,

"price": 3.3

}

],

"status": "200"

}

二十一、加载个人信息接口

**1、接口功能**

获取该登录的个人信息

**2、接入URL**

<http://localhost:8081/user/getUser>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数 | 类型 | 说明 |
| 用户名 | userName | String | 登录用户名称 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 参数名 | 字段类型 | 说明 |
| userID | String | 登录用户的ID |
| userName | String | 登录用户名称 |
| userPass | String | 登录用户的密码 |
| userPhone | String | 登录用户的电话号码 |
| userPermission | String | 登录用户权限 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/user/getUser?userName=王五>

返回结果为

{

"data":{

"userID":"11",

"userName":"王五",

"userPass":"aaaa",

"userPermission":"1",

"userPhone":"null"

},

"status":"200"

}

二十二、更改个人信息接口

**1、接口功能**

更改该登录用户的个人信息，包括用户名与电话号码

**2、接入URL**

<http://localhost:8081/user/resetUserInfo>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数 | 类型 | 说明 |
| 用户名 | userName | String | 登录用户名称 |
| 新的用户名 | newName | String | 用户更改后的名称 |
| 用户电话号码 | userPhone | String | 用户的电话号码 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 参数名 | 字段类型 | 说明 |
| status | String | 返回结果状态。200:正常；500:错误 |
| msg | String | 返回结果信息 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/user/resetUserInfo?&userName=www&userPhone=13857394571&newName=David>

返回结果为

如果更改成功，则返回结果为

{

"status": "200",

"msg": "Success"

}

如果更改失败，则返回结果为

{

"status": "500",

"msg": "系统错误"

}

二十三、更改密码接口

**1、接口功能**

更改该登录用户的登录密码

**2、接入URL**

<http://localhost:8081/user/resetPass>

**3、支持格式**

JSON

**4、HTTP请求方式**

POST

**5、请求参数**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数 | 类型Pass | 说明 |
| 用户名 | userName | String | 登录用户名称，用于查找用户 |
| 用户密码 | userPass | String | 登录用户密码 |
| 新的用户密码 | newUserPass | String | 用户更改后的密码 |

**6、返回字段**

|  |  |  |
| --- | --- | --- |
| 参数名 | 字段类型 | 说明 |
| status | String | 返回结果状态。200:正常；500:错误 |
| msg | String | 返回结果信息 |

**7、接口示例**

请求提交方法为：<http://localhost:8081/user/resetPass?&userName=李四&userPass=aaaa&newUserPass=123456>

返回结果为

如果更改成功，则返回结果为

{

"status": "200",

"msg": "Success"

}

如果更改失败，则返回结果为

{

"status": "500",

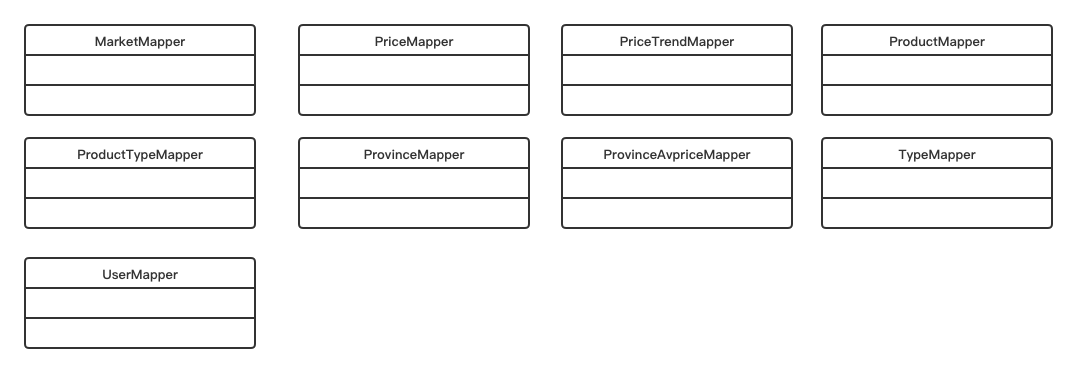
"msg": "原密码错误！请重新输入原密码"

}

第六部分 单元模块设计

一、数据访问层设计

1、类图设计



2、类的详细设计描述

2.1 MarketMapper接口设计

详细描述：

|  |
| --- |
| MarketMapper市场数数据库操作 |
| List<Market> getMarketByProvinceName(@Param("provinceName") String provinceName); |
| List<Map<String,Object>> getMarketNoByProvince(); |
| List<Map<String,Object>> getMarketByProvince(); |
| List<Map<String,Object>> getLatestMarketNumber(); |

2.2 PriceMapper接口设计

详细描述：

|  |
| --- |
| PriceMapper产品价格数据库操作 |
| List<Price> getPriceByMarketAndProduct(@Param("marketName") String marketName, @Param("productName") String productName); |
| List<Price> getPriceByProvinceAndProduct(@Param("provinceName") String provinceName, @Param("productName") String productName); |
| List<Map<String,Object>> getLatestPrice(); |
| List<Map<String,Object>> getTotalDataVolume(); |

2.3 PriceTrendMapper接口设计

|  |
| --- |
| PriceTrendtMapper价格趋势数据库操作 |
|  |
|  |
|  |
|  |

2.4 ProductMapper接口设计

|  |
| --- |
| ProductMapper农产品数量数据库操作 |
| List<Map<String,Object>> getProductNoByType(@Param("marketName") String marketName, @Param("type") String type); |
| List<Map<String,Object>> getProductsByType(@Param("marketName") String marketName, @Param("type") String type); |
| List<Map<String,Object>> getTypePercents(@Param("marketName") String marketName, @Param("type") String type); |
| List<Map<String,Object>> getLatestProductNumber(); |

2.5 ProductTypeMapper接口设计

|  |
| --- |
| ProductTypeMapper农产品类型数据库操作 |
|  |
|  |
|  |
|  |

2.6 ProvinceMapper接口设计

|  |
| --- |
| ProvinceMapper省份数据库操作 |
|  |
|  |
|  |
|  |

2.7 ProvinceAvpriceMapper接口设计

|  |
| --- |
| ProvinceAvprice平均价格数据库操作 |
| List<ProvinceAvprice> getProvinceAvpriceByProduct(@Param("productName") String productName); |

2.8 TypeMapper接口设计

|  |
| --- |
| TypeMapper农产品类型列表数据库操作 |
| List<Map<String,Object>> getTypeNumber(); |

2.9 UserMapper接口设计

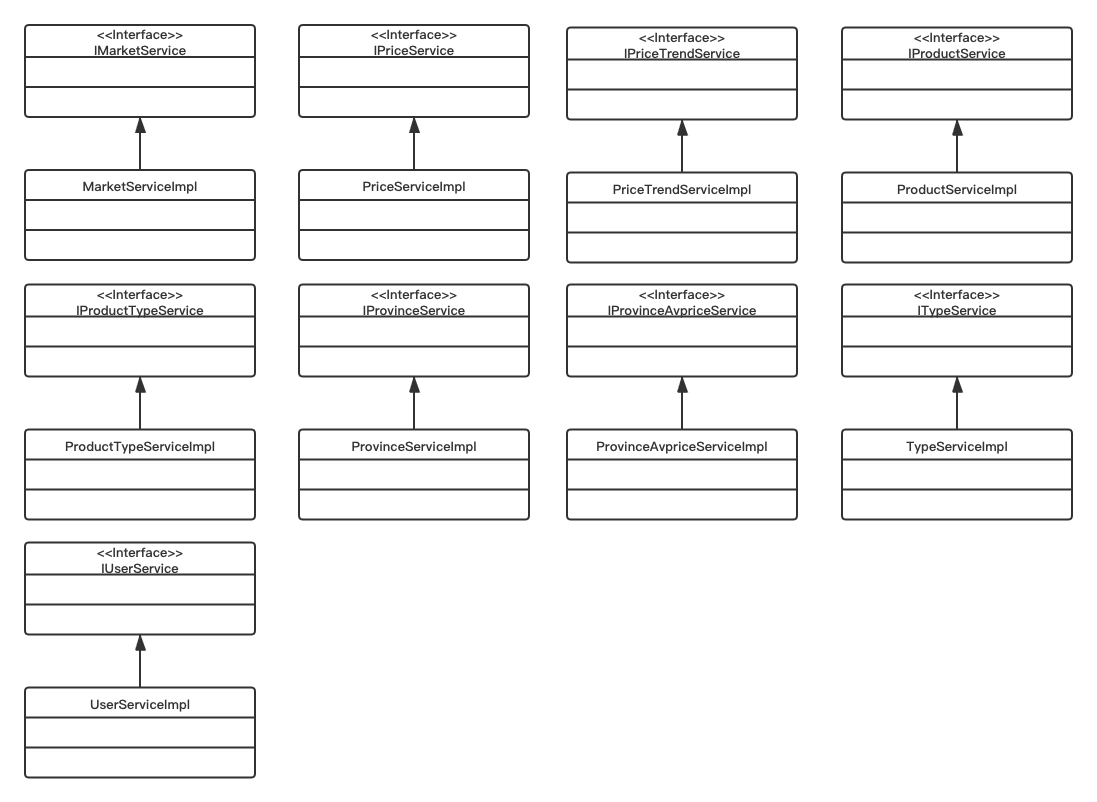
|  |
| --- |
| UserMapper用户数据库操作 |
|  |
|  |
|  |
|  |

2.10 Mapper接口汇总

|  |
| --- |
| 接口 |
| MarketMapper |
| PriceMapper |
| PriceTrendMapper |
| ProductMapper |
| ProductTypeMapper |
| ProvinceMapper |
| ProvinceAcpriceMapper |
| TypeMapper |
| UserMapper |

二、业务逻辑层设计

1、类图设计



2、类的详细设计描述

2.1 IMarketService接口设计

详细描述：

|  |
| --- |
| IMarketService市场查询业务接口 |
| List<Market> getMarketByProvinceName(@Param("provinceName") String provinceName); |
| List<Map<String,Object>> getMarketByProvince(); |
| List<Map<String,Object>> getMarketNoByProvince(); |
| List<Map<String,Object>> getLatestMarketNumber(); |

2.2 IPriceService接口设计

详细描述：

|  |
| --- |
| IPriceService价格查询业务接口 |
| List<Price> getPriceByMarketAndProduct(@Param("marketName") String marketName, @Param("productName") String productName); |
| List<Price> getPriceByProvinceAndProduct(@Param("provinceName") String provinceName, @Param("productName") String productName); |
| List<Map<String,Object>> getLatestPrice(); |
| List<Map<String,Object>> getTotalDataVolume(); |

2.3 IPriceTrendService接口设计

详细描述：

|  |
| --- |
| IPriceTrendService价格趋势查询业务接口 |
| List<Product> getProductList(); |
| List<Map<String,Object>> getLatestProductNumber(); |
| List<Map<String,Object>> getProductNoByType(String marketName, String type); |
| List<Map<String,Object>> getProductsByType(String marketName, String type); |
| List<Map<String,Object>> getTypePercents(@Param("marketName") String marketName, @Param("type") String type); |

2.4 IProductService接口设计

详细描述：

|  |
| --- |
| IProductService产品查询业务接口 |
| List<Product> getProductList(); |

2.5 IProductTypeService接口设计

详细描述：

|  |
| --- |
| IProductTypeService产品种类查询业务接口 |

2.6 IProvinceService接口设计

详细描述

|  |
| --- |
| IProvinceService省份查询业务接口 |
| List<Province> getProvinceList(); |

2.7 IProvinceAvpriceService接口设计

详细描述：

|  |
| --- |
| IProvinceAvpriceService市场查询业务接口 |
| List<ProvinceAvprice> getProvinceAvpriceByProduct(@Param("productName") String productName); |

2.8 ITypeService接口设计

详细描述：

|  |
| --- |
| ITypeService市场查询业务接口 |
| List<Type> getTypeList(); |
| List<Map<String,Object>> getTypeNumber(); |

2.9 IForecastPriceService接口设计

详细描述：

|  |
| --- |
| IForecastPriceService价格预测业务接口 |
| List<Map<String,Object>> getForecastPriceByMarketAndProduct(String marketID, String productid); |

2.10 IUserService接口设计

详细描述：

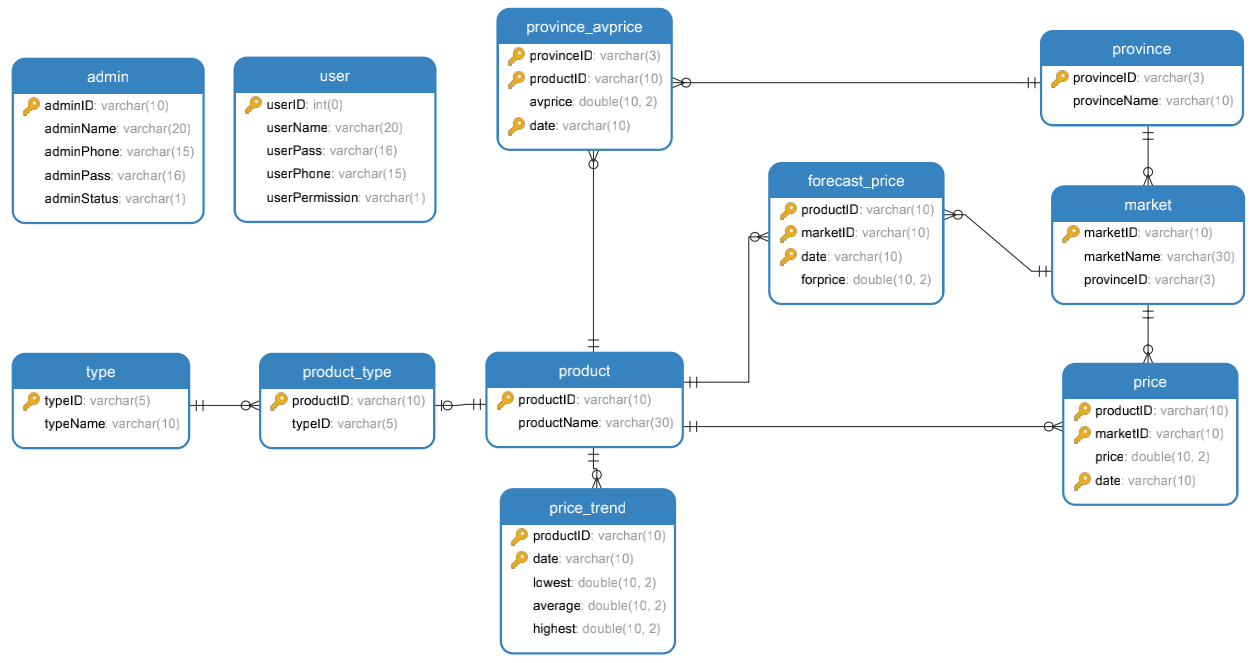
|  |
| --- |
| IUserService市场查询业务接口 |
| User loginCheck(String userName,String userPass); |
| int regAdd(String userName,String userPass); |
| User getUser(String userName); |
| int resetPass(String userName,String userPass,String newUserPass); |
| int resetUserInfo(String userName,String newName,String userPhone); |

2.11 业务接口实现类汇总

|  |  |
| --- | --- |
| 接口 | 实现类 |
| IMarketService | MarketServiceImpl |
| IPriceService | PriceServiceImpl |
| IPriceTrendService | PriceTrendServiceImpl |
| IProductService | ProductServiceImpl |
| IProductTypeService | ProductTypeServiceImpl |
| IProvinceService | ProvinceServiceImpl |
| IProvinceAvpriceService | ProvinceAvpriceServiceImpl |
| ITypeService | TypeServiceImpl |
| IUserService | UserServiceImpl |

第七部分 数据库设计

一、数据库整体结构图

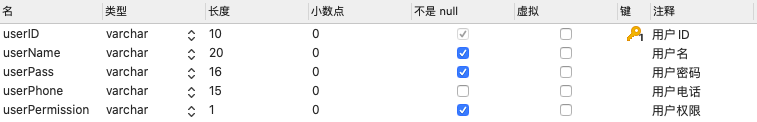


二、用户管理

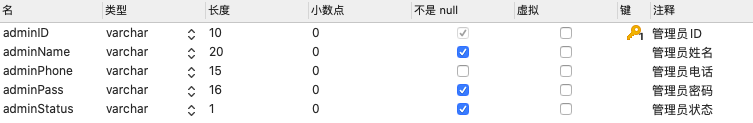
用户管理表格

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 注释 |
| 1 | user | 用户表 |
| 2 | admin | 管理员表 |

1、user表结构



2、admin表结构



三、市场信息管理

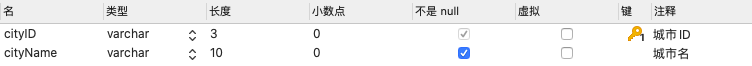
市场信息管理表格

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 注释 |
| 1 | province | 省表 |
| 2 | city | 市表 |
| 3 | province\_city | 省市表 |
| 4 | market | 市场表 |

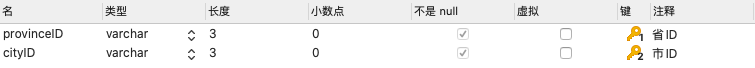
1、province表结构

图片

2、city表结构



3、province\_city表结构



province\_city表外键



4、market表结构



market表外键

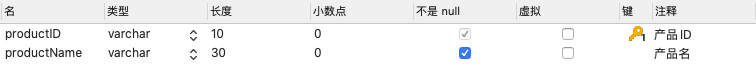


四、农产品信息管理

农产品信息管理表格

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 注释 |
| 1 | product | 产品表 |
| 2 | type | 种类表 |
| 3 | product\_type | 产品种类表 |
| 4 | price | 价格表 |

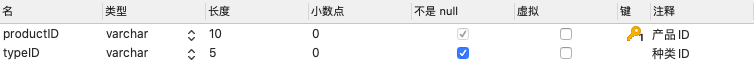
1、product表结构



2、type表结构



3、product\_type表结构



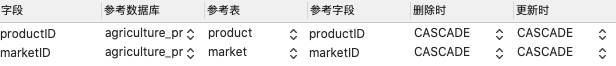
product\_type外键



4、price表结构



price表外键



五、查询业务管理

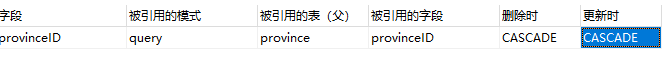
查询业务管理表格

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 注释 |
| 1 | province\_avprice | 省平均价格表 |
| 2 | forecast\_price | 预测价格表 |
| 3 | price | 价格表 |

1.province\_avprice表结构



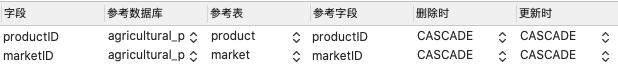
province\_avprice表外键



2.forecast\_price表结构



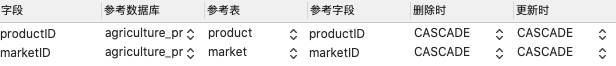
forecast\_price表外键



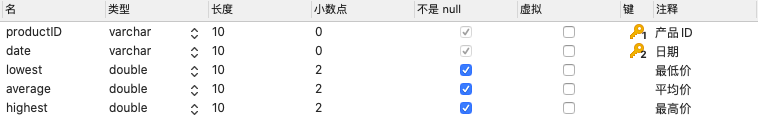
3.price表



price表外键



4.price\_trend表



price\_trend表外键

图片

第八部分 补充设计和说明

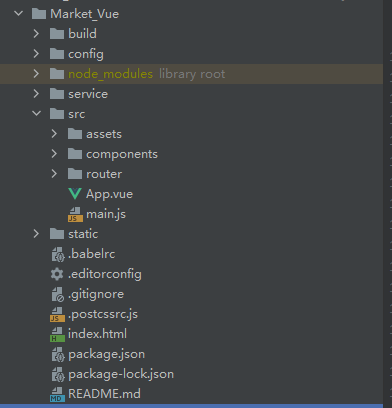
一、配置环境设计





二、目录结构设计

2.1、前端目录结构



2.2、后端目录结构

