项目概述：本项目，是一个商品后台管理系统。采用前后端分离开发，主要用于给内部运营人员管理商品用的。主要有登录模块(登录注册忘记密码等页面),用户管理模块(管理内部登录用户的),权限管理(包含角色列表,权限列表[不同人有不同权限]),商品管理(管理商品的增删改查)，数据统计(领导看商品销售数据，主要用到echarts)等模块

本地接口baseurl根地址http://localhost:8888/api/private/v1/

线上接口baseurl根地址-https://www.liulongbin.top:8888/api/private/v1/

线上演示地址-http://gl.timemeetyou.com/#/login 用户名:admin 密码:123456

01-登录业务流程介绍

一..登录总思路 -baseurl/login

1.前端验证规则-使用elmentui组件

2.验证后-请求axios接口(按照接口要求提供参数)

3.根据响应数据-写逻辑:提醒用户登录成功，存储token

二.接口工具-postman介绍，测试登录接口

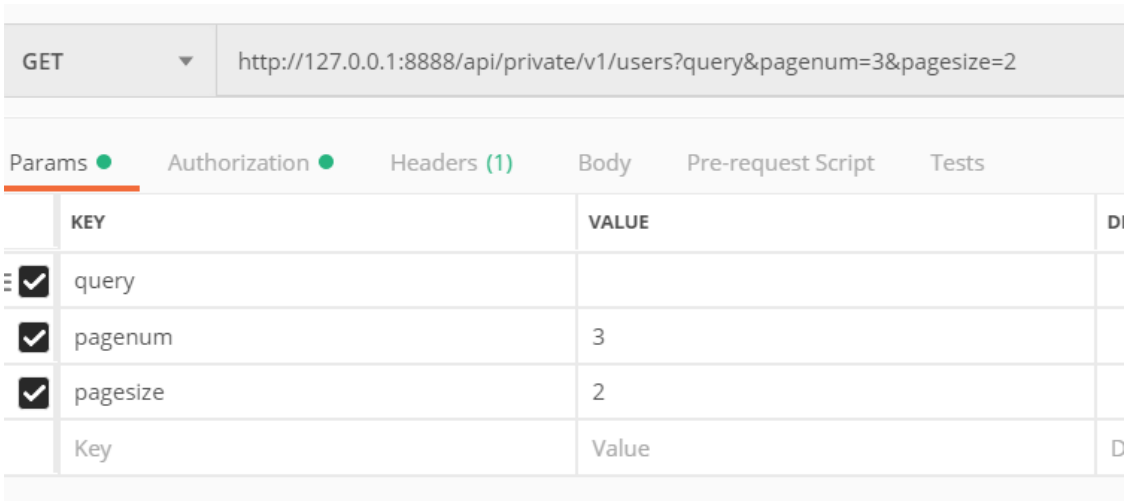
1. get与post:区别

get请求-参数是在url里,一般是不敏感的数据用get

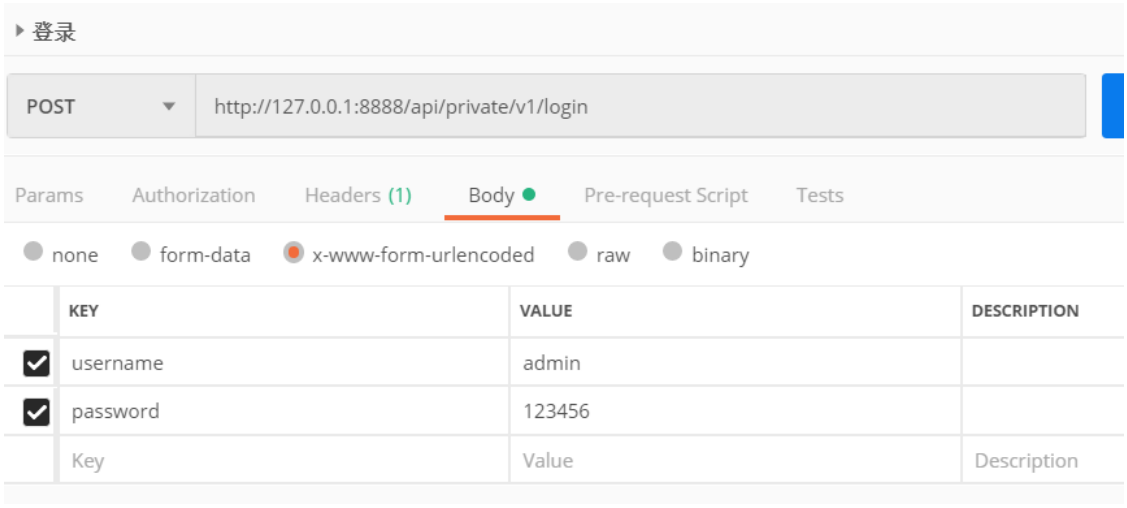
post请求-参数通过协议头里body传输，参数不在url里。一般安全数据用post

2. postman接口工具里-如何设置get与post请求

get请求设置:



post请求:



三.登录常见问题

1.传参问题-要与接口要求的入参一致

2.判断状态-要的是后台响应参数的状态

```
// 1.用elment验证通过
this.$refs[formName].validate((valid) => {
  if (valid) {
      //2.请求接口
      let username=this.ruleForm.age
      let password=this.ruleForm.pass

      this.$http.post('login',{username,password}).then(res=>{
        console.log(res); //数据请求成功
        //2.1 提醒用户登录成功
      if(res.data.meta.status==200){
          this.$message({
            message: res.data.meta.msg,
            type: 'success'
          });
          //3.根据响应数据，做提醒，跳转
          this.$router.push('/home')
      }
```

02退出功能,左侧权限列表等

一.退出实现

思考问题:

1.点击退出后，如果没有登录不能在进入，如果登录了，可以在进入？我怎么知道登录了？

路由导航守卫,登录后存储token值

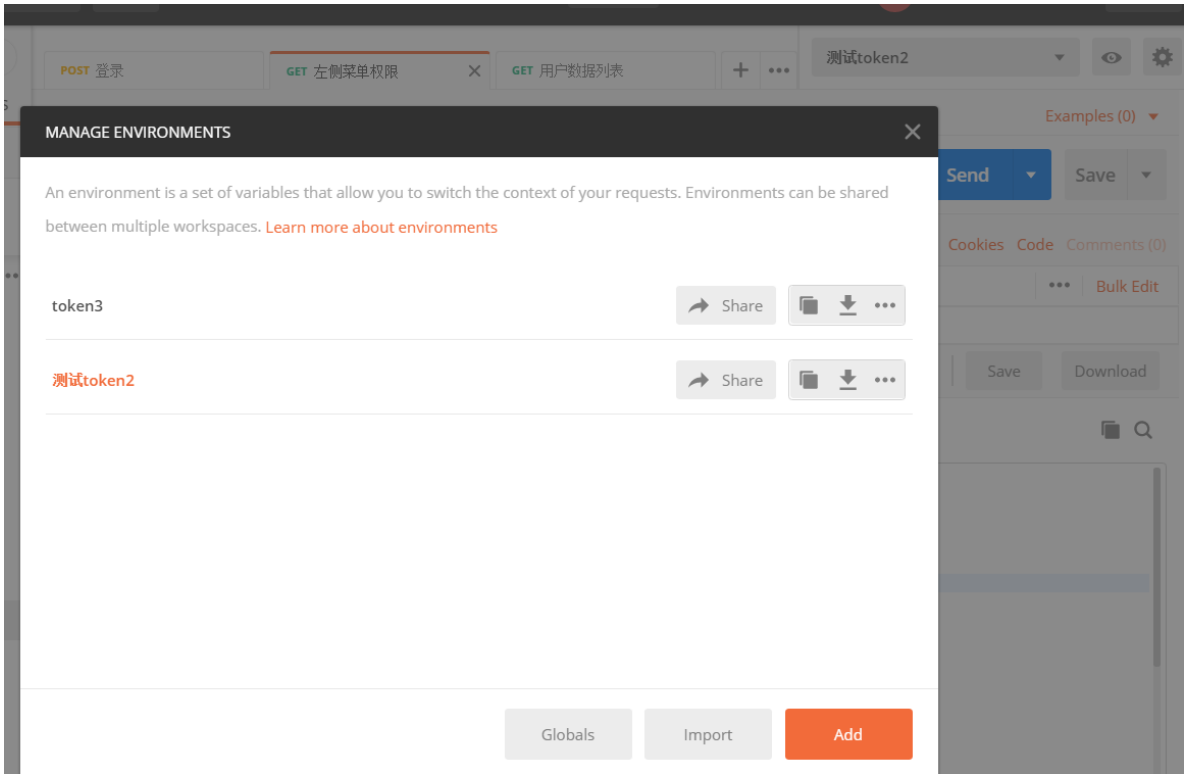
2.登录后的，所有接口数据都是需要token值的，否则获取不了，如何实现?

每次请求数据，协议头里，带着token,才能拿到数据

二.进入后台获取数据问题

1.后台数据,每次拿数据都需要token值

postman接口设置，每次携带token值问题，如何设置?



页面代码怎么解决这个，每次携带token问题？-请求拦截器里，带上头协议

```
//请求拦截里，带着头协议，才能拿到数据-好处，不用每个数据请求里都写协议头了
axios.interceptors.request.use(opt=>{
  opt.headers.Authorization=sessionStorage.getItem('token')
  return opt
})
```

三.左侧权限列表

1.用导航组件实现基本布局-navmenu

2.路由点击跳转-需要加入属性router,配合 获取数据里的路径

{{item2.authName}}

3.默认激活打开-属性 :default-active="/users"

点击时，存储路径，页面加载，获取路径

4.点击展开与收缩属性 :collapse="true"

5.图标问题-从element里找图标, 通过class绑定，每个图标不一样，可以定义数组

03.用户模块

一搜索用户思路- 1.传入搜索值，请求用户列表接口，注意传入的参数-当前页数pagenum,每页条数pagesize

二.添加用户--思路：1.调用dialog对话框组件显示 2.表单双向数据绑定:model="addForm", 验证规则绑定:rules="addFormRules" 3.提交表单时,获取该表单名字 ref="addFormRef"，验证表单规则,可自定义规则。注意prop属性绑定的验证规则里的属性 4.提交接口，在次调用渲染列表

三. 修改用户-1.弹出修改对话框,获取当前行信息2.提前存储当前行用户信息。为什么要存储每行信息?因为点击确定操作时，当前行用户信息不知道。3.因为有多列都涉及到点击操作,所以要用作用域插槽 scope 有属性$index，代表索引，row-当前行所有信息

四.更新用户-验证规则通过后，请求！！！注意接口,id是必传项(id获取时,是变量)，其它参数要带着,内容可以为空。

五.删除用户-调用确认组件this.$confirm,确认后调删除接口

六.分配角色-1.调用角色对话框，存储角色信息，点击更新时用到 2.获取所有角色接口，存储数据，更新时用到 3.下拉列表选择时-v-model绑定数据id，属性:value="item.id" 配合获取下拉选择项4.点击确定时，提交用户id,角色id更新数据

```
<p>
  分配新角色:
  <el-select  placeholder="请选择" v-model="selectedRoleId"
  @change="getRoleId">
    <el-option  v-for="item in roleList" :value="item.id" :key="item.
    id"  :label="item.roleName">
      {{item.roleName}}
    </el-option>
  </el-select>
</p>
```

七.更新角色-1.获取角色信息，下拉选择时的角色id(注意是绑定的角色id数据selectedRoleId) 2.调用接口更新

```
//更新角色-1.获取角色信息，下拉选择时的角色id(注意是绑定的角色id数据selectedRoleId)
2.传递接口-当前用户信息,角色id
async  updateRole(){
    //！！！注意,判断不存在时，弹出错误的同时,后边要带retrun不让执行。否则会有错误
    if(!this.selectedRoleId){
     return  this.$message.error('请选择要分配的角色！')
    }
    //更新接口-传递用户id,下拉选择角色id
     const {data:res}=await this.$http.put(`users/${this.roleInfo.id}/role`,
     {rid:this.selectedRoleId})
     console.log(188,res);
     if (res.meta.status == 200) {
              this.$message({
                showClose: true,
                message: res.meta.msg,
                type: 'success',
              })
              this.roleFormVisible = false
              this.getUserList()

          }
```

八.切换用户状态-获取当前行信息，请求接口，传入动态参数

```
<el-table-column label="状态" >
  <!-- 状态切换 -->
  <template slot-scope="scope">
        <el-switch
        v-model="scope.row.mg_state"
        active-color="#13ce66"
        inactive-color="#ff4949" @change="switchState(scope.row)
        ">
      </el-switch>
  </template>
```

九.分页思路:本接口主要是后台分页：后台要求传入当前页数pagenum,每页条数pagesize。后台会返回总条数total,当前页数pagesize。使用分页插件实现分页样式，当条数改变时传入条数提交接口，当页数改变时传入页数提交接口

<!-- 分页插件 @size-change(条数改变时触发)

@current-change(页数发生改变时触发)

:current-page(设置当前页码)

:page-size(设置每页的数据条数) layout是布局样式

:total(设置总页数) -->

```
      <el-pagination
        @size-change="handleSizeChange"
        @current-change="handleCurrentChange"
        :current-page="userInfo.pagenum"
        :page-sizes="[1, 2, 5, 10]"
        :page-size="userInfo.pagesize"
        layout="total, sizes, prev, pager, next, jumper"
        :total="total">
      </el-pagination>
</div>
```

04 权限模块

一.用户,角色,权限之间关系

1.多个用户可隶属于1个角色，1个角色可拥有多个权限。每个用户登录进到看到的权限不一样，对应的左侧菜单列表不一样

二.角色列表渲染

1.渲染权限列表思路-使用三重嵌套for循环生成权限下拉列表

#调用el-table里的展开列数据格式

#布局要合理-因为页面要的格式是3层,数据格式也是3层,如何取出每1层下对应的数据？table嵌套外层1行2列，第　2列里又嵌套1行2列。这样总共就有了3列。每列渲染对应的数据

```
<!-- 角色列表区域 -->
<el-table :data="rolelist" border stripe>
  <!-- 展开列 -->
  <el-table-column type="expand">
    <template slot-scope="scope">
      <el-row :class="['bdbottom', i1 === 0 ? 'bdtop' : '', 'vcenter']" v-for="(item1, i1) in scope.row.children"
      :key="item1.id">
        <!-- 渲染一级权限 -->
        <el-col :span="5">
          <el-tag closable @close="removeRightById(scope.row, item1.id)">{{item1.authName}}</el-tag>
          <i class="el-icon-caret-right"></i>
        </el-col>
        <!-- 渲染二级和三级权限 -->
        <el-col :span="19">
          <!-- 通过 for 循环 嵌套渲染二级权限 -->
          <el-row :class="[i2 === 0 ? '' : 'bdtop', 'vcenter']" v-for="(item2, i2) in item1.children" :key="item2.id">
            <el-col :span="6">
              <el-tag type="success" closable @close="removeRightById(scope.row, item2.id)">{{item2.authName}}</el-tag>
              <i class="el-icon-caret-right"></i>
            </el-col>
            <el-col :span="18">
              <el-tag type="warning" v-for="item3 in item2.children" :key="item3.id" closable @close="removeRightById
              (scope.row, item3.id)">{{item3.authName}}</el-tag>
            </el-col>
          </el-row>
        </el-col>
      </el-row>

      <!-- <pre>
        {{scope.row}}
      </pre> -->
    </template>
  </el-table-column>
  <!-- 索引列 -->
  <el-table-column type="index"></el-table-column>
  <el-table-column label="角色名称" prop="roleName"></el-table-column>
```

2.点击分配权限展示树形列表权限

#1.布局调用el-tree组件,先把组件搞明白

```
<!-- 分配权限的对话框 -->
<!-- el-tree树形组件-
show-checkbox:显示复选框
node-key:设置选中节点对应的值
default-expand-all:是否默认展开所有节点
:default-checked-keys 设置默认选中项的数组
ref:设置引用 -->
<el-dialog title="分配权限" :visible.sync="setRightDialogVisible" width="50%" @close="setRightDialogClosed">
  <!-- 树形控件 -->
  <el-tree :data="rightslist" :props="treeProps" show-checkbox node-key="id" default-expand-all
  :default-checked-keys="defKeys" ref="treeRef"></el-tree>
  <span slot="footer" class="dialog-footer">
    <el-button @click="setRightDialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="allotRights">确 定</el-button>
  </span>
</el-dialog>
```

#2.获取所有权限的数据 *const* { data: res } = await this.$http.get('rights/tree') ,渲染绑定

#3.默认选中实现？通过递归的形式，获取当前角色下所有三级权限的id，并保存到 默认defKeys 选中数组中 。递归会不断的遍历当前角色下childen推入默认选中数组

#4.提交选中更新权限-点击为角色分配权限-注意这里调用element两个方法来记录你选中了那些：1个是全选中方法getCheckedKeys()，1个是半选中方法getHalfCheckedKeys,在把它们合并成数组[105,116,117,150,101,104]。注意接口提交的时候要的是字符串

05 商品管理模块 ***

概述：商品分类功能是为某一类商品填加大类(包含1,2,3级分类)，分类参数是给某个3级分类填加参数或者属性说明的，商品列表是为具体某个商品上传详情信息的,比如图片，描述等。

录属关系:商品列表隶属于某个分类参数,分类参数又隶属于某个商品分类

一.商品分类列表-用于展示商品的大分类

0.商品分类数据接口-type参数可设置要几层数据

const { data: res } = await this.$http.get('categories', {

params: {type: 3, pagenum: 1,pagesize: 5 })

1.树状菜单-第3方tree组件 详情地址-https://gitee.com/vm_liu/vue-table-with-tree-grid tree组件

里面可放置不同模板。通过slot给模板名字,配置数据源时 起同样名字

```
<!-- 表格 :data(设置数据源) :columns(设置表格中列配置信息) :selection-type(是否有复选框)
:expand-type(是否展开数据) show-index(是否设置索引列) index-text(设置索引列头)
rder(是否添加纵向边框) :show-row-hover(是否鼠标悬停高亮)  详情地址-https://gitee.com/
_liu/vue-table-with-tree-grid  tree组件-->
<tree-table class="treeTable" :data="catelist" :columns="columns"
:selection-type="false" :expand-type="false" show-index index-text="#" border
:show-row-hover="false">
    <!-- 是否有效 -->
  <template slot="isok" slot-scope="scope">
    <i class="el-icon-success" v-if="scope.row.cat_deleted === false" style="color:
    lightgreen;"></i>
    <i class="el-icon-error" v-else style="color: red;"></i>
  </template>
```

columns: [

{

label: '是否有效',

// 表示，将当前列定义为模板列

type: 'template',

// 表示当前这一列使用模板名称

template: 'isok'

},

二.填加分类

1. 级联菜单使用-

<!-- expandTrigger='hover'(鼠标悬停触发级联) v-model(设置级联菜单绑定数据) :options(指定级联菜单数据源) :props(用来配置数据显示的规则)

clearable(提供“X”号完成删除文本功能) change-on-select(是否可以选中任意一级的菜单) @change="parentCateChanged" 选择项改变时触发事件-->

<el-cascader expand-trigger="hover" :options="parentCateList" :props="cascaderProps" v-model="selectedKeys" @change="parentCateChanged" clearable change-on-select>

2.点击确定提交分类

#接口需要入参-分类名称,父分类id,当前分类层级

const { data: res } = await this.$http.post('categories', {cat_name: '', cat_pid: 0,cat_level: 0})

#分类层级，要根据父分类层级提交参数,你怎么知道父级要类选择了几级?

//给级联菜单绑定一个数组,selectedKeys，[1]，你选择的项会存储到数组里

#选择的分类层级是哪个?

// 级联如果 selectedKeys 数组中的 length 大于0，证明选中的父级分类。反之，就说明没有选中任何父级分类

```
// 选择项发生变化触发这个函数
parentCateChanged() {
  console.log(this.selectedKeys)
  // 如果 selectedKeys 数组中的 length 大于0，证明选中的父级分类
  // 反之，就说明没有选中任何父级分类
  if (this.selectedKeys.length > 0) {
    // 父级分类的Id
    this.addCateForm.cat_pid = this.selectedKeys[this.selectedKeys.length - 1]
    // 为当前分类的等级赋值
    this.addCateForm.cat_level = this.selectedKeys.length
  } else {
    // 父级分类的Id
    this.addCateForm.cat_pid = 0
    // 为当前分类的等级赋值
    this.addCateForm.cat_level = 0
  }
},
```

三.分类参数

1. 概述：商品参数用于显示商品固定的特征信息，可以通过电商平台商品详情页面直观的看到

2.布局-通过el-tab进行切换，通过v-model绑定激活项

```
<!-- tab 页签区域 -->
<el-tabs v-model="activeName" @tab-click="handleTabClick">
  <!-- 添加动态参数的面板 -->
  <el-tab-pane label="动态参数" name="many">
    <!-- 添加参数的按钮 -->
    <el-button type="primary" size="mini" :disabled="isBtnDisabled" @click="addDialogVisible=true">添加参数</el-button>
    <!-- 动态参数表格 -->
    <el-table :data="manyTableData" border stripe>
      <!-- 展开行 -->
      <el-table-column type="expand">
        <template slot-scope="scope">
          <!-- 循环渲染Tag标签 -->
          <el-tag v-for="(item, i) in scope.row.attr_vals" :key="i" closable @close="handleClose(i, scope.row)">{{item}}</el-tag>
          <!-- 输入的文本框 -->
          <el-input class="input-new-tag" v-if="scope.row.inputVisible" v-model="scope.row.inputValue" ref="saveTagInput" size="small"
          @keyup.enter.native="handleInputConfirm(scope.row)" @blur="handleInputConfirm(scope.row)">
          </el-input>
          <!-- 添加按钮 -->
          <el-button v-else class="button-new-tag" size="small" @click="showInput(scope.row)">+ New Tag</el-button>
        </template>
      </el-table-column>
      <!-- 索引列 -->
      <el-table-column type="index"></el-table-column>
      <el-table-column label="参数名称" prop="attr_name"></el-table-column>
      <el-table-column label="操作">
        <template slot-scope="scope">
          <el-button size="mini" type="primary" icon="el-icon-edit" @click="showEditDialog(scope.row.attr_id)">编辑</el-button>
          <el-button size="mini" type="danger" icon="el-icon-delete" @click="removeParams(scope.row.attr_id)">删除</el-button>
        </template>
      </el-table-column>
    </el-table>
  </el-tab-pane>
  <!-- 添加静态属性的面板 -->
  <el-tab-pane label="静态属性" name="only">···
  </el-tab-pane>
```

3.1逻辑实现-注意这里只允许为第三级分类添加相关参数!

#用户选中前两级怎么办？设置动态参数数据，静态属性数据为空，这样就不会渲染数据

#如果选中是第三级？-根据所选分类的cateId(分类id是根据你所选的第几个大类,动态获取的)，和当前所处的面板，获取对应的参数

const { data: res } = await this.$http.get(`categories/${this.cateId}/attributes`, {params: { sel: this.activeName } })

#三级分类id如何获取？-级联菜单绑定数组selectedCateKeys,级联选择改变时，数组里会有3个值，获取下

<el-cascader expand-trigger="hover" :options="catelist" :props="cateProps" v-model="selectedCateKeys" @change="handleChange">

//可放在计算属性里,因为cateId是动态获取的

```
computed: {
    // 当前选中的三级分类的Id
  cateId() {
      if (this.selectedCateKeys.length === 3) {
          return this.selectedCateKeys[2]
      }
   return null
  }
}
```

3.2 封装一个获取参数列表数据getParamsData() ,方便级联选中变化时调用,tab切换时也会调用

```
async getParamsData() {
  // 证明选中的不是三级分类
  if (this.selectedCateKeys.length !== 3) {
    this.selectedCateKeys = []
    this.manyTableData = []
    this.onlyTableData = []
    return
  }

  // 证明选中的是三级分类
  console.log(this.selectedCateKeys)
  // 根据所选分类的Id，和当前所处的面板，获取对应的参数
  const { data: res } = await this.$http.get(
    `categories/${this.cateId}/attributes`,
    {
      params: { sel: this.activeName }
    }
  )

  if (res.meta.status !== 200) {
    return this.$message.error('获取参数列表失败！')
  }
  console.log(res.data);
  //注意标签数据，后台返回的是字符串。这里处理成数组，方便使用
  res.data.forEach(item => {
    item.attr_vals = item.attr_vals ? item.attr_vals.split(' ') : []
    // 控制文本框的显示与隐藏
    item.inputVisible = false
    // 文本框中输入的值
    item.inputValue = ''
  })

  //根据tab激活名字，调用动态参数与静态属性数据
  console.log(res.data)
  if (this.activeName === 'many') {
    this.manyTableData = res.data
  } else {
    this.onlyTableData = res.data
```

3.3 每行里tag标签-通过组件实现——tag删除回车输入内容时触发编辑提交参数接口 put提交 categories/:id/attributes/:attrId

1.下拉选择时，获取参数列表数据，对数据里的 "attr_vals": "a b c" 给的是字符串进行转换成数组操作

2.bug问题?在tag里输入内容时，同时会操作多个行里的内容问题解决办法？给每行的tag组件外面加入作用域插槽，渲染时只渲染当前行数据(scope.row.attr_vals),输入内容时只操作当前行数据 绑定v-model时，也只绑定当前行

```
<!-- 作用域插槽 -只拿当前行的数据 -->
<template  slot-scope="scope">
        <el-tag
        :key="tag"
        v-for="(tag,i) in scope.row.attr_vals"
        closable
        :disable-transitions="false"
        @close="handleClose(scope.row,i)">
        {{tag}}
      </el-tag>
      <el-input
        class="input-new-tag"
        v-if="scope.row.inputVisible"
        v-model="scope.row.inputValue"
        ref="saveTagInput"
        size="small"
        @keyup.enter.native="handleInputConfirm(scope.row)"
        @blur="handleInputConfirm(scope.row)"
      >
      </el-input>
      <el-button v-else class="button-new-tag" size="small"
      @click="showInput(scope.row)">+ New Tag</el-button>
</template>
```

3.tag输入内容时,第1次输入不上，会合并行的问题? 在遍历attr_vals数据转换成数组时，就给每个对象当前行绑定属性

```
 const { data: res } = await this.$http.get(`categories/${catId}/attributes?
 sel=${sel}`)
 console.log(res);
 if (res.meta.status !== 200) {
   return this.$message.error('获取参数列表失败！')
 }

//2.把数据里tag字符串标签变成数组
res.data.forEach(item=>{
        //attr_vals="a b c"
        console.log(item.attr_vals);
       //  this.dynamicTags=item.attr_vals.split(' ')   //如果定义统一的数组，会有
       覆盖问题。怎么只修改当前行信息里的数据？
       //转数组
       item.attr_vals= item.attr_vals?item.attr_vals.split(' '):[]
       //给当前行增加属性-否则第一次输入不了内容，或者报错
       item.inputVisible=false
       item.inputValue=''

})
```

四.商品列表

0.布局-用到步骤条el-step组件,每个tab组件里套着表单组件。步骤条与tab如何联动？，可给tab声明1个激活索引v-model="activeIndex"同时每个tab子项给到name="索引"，步骤条调用该索引

```
<!-- 步骤条区域 -->
<el-steps :space="200" :active="activeIndex - 0" finish-status="success"
align-center>
  <el-step title="基本信息"></el-step>
  <el-step title="商品参数"></el-step>
  <el-step title="商品属性"></el-step>
  <el-step title="商品图片"></el-step>
  <el-step title="商品内容"></el-step>
  <el-step title="完成"></el-step>
</el-steps>

<!-- tab栏区域 -->

<el-form :model="addForm" :rules="addFormRules" ref="addFormRef"
label-width="100px" label-position="top">
  <el-tabs v-model="activeIndex" :tab-position="'left'"
  :before-leave="beforeTabLeave" @tab-click="tabClicked">
    <el-tab-pane label="基本信息" name="0">···
    </el-tab-pane>
    <el-tab-pane label="商品参数" name="1">···
    </el-tab-pane>
    <el-tab-pane label="商品属性" name="2">···
    </el-tab-pane>
    <el-tab-pane label="商品图片" name="3">
```

1.tab切换验证：切换下一个,得验证上一个是否选择？可用离开之前属性-:before-leave 调用方法，判断用户是否选择了下拉的3级列表,

<el-tabs v-model="activeIndex" :tab-position="'left'" :before-leave="beforeTabLeave" @tab-click="tabClicked">

```
beforeTabLeave(activeName, oldActiveName) {
  // console.log('即将离开的标签页名字是：' + oldActiveName)
  // console.log('即将进入的标签页名字是：' + activeName)
  // return false
  if (oldActiveName === '0' && this.addForm.goods_cat.length !== 3) {
    this.$message.error('请先选择商品分类！')
    return false
  }
},
```

2.点击商品属性或商品参数面板，要根据上次选择的3级分类id请求对应数据。你怎么知道它点击的是商品属性，还是商品参数？点击时@tab-click="tabClicked"通过判断当前激活索引是多少，做相应的接口请求

```
async tabClicked() {
  // console.log(this.activeIndex)
  // 证明访问的是动态参数面板
  if (this.activeIndex === '1') {
    const { data: res } = await this.$http.get(
      `categories/${this.cateId}/attributes`,
      {
        params: { sel: 'many' }
      }
    )

    if (res.meta.status !== 200) {
      return this.$message.error('获取动态参数列表失败！')
    }

    console.log(res.data)
    res.data.forEach(item => {
      item.attr_vals =
        item.attr_vals.length === 0 ? [] : item.attr_vals.split(' ')
    })
    this.manyTableData = res.data
  } else if (this.activeIndex === '2') {…
  }
},
```

3.图片上传:

使用upload组件完成图片上传
在element.js中引入upload组件，并注册
因为upload组件进行图片上传的时候并不是使用axios发送请求
所以，我们需要手动为上传图片的请求添加token，即为upload组件添加headers属性

```
        <el-tab-pane label="商品图片" name="3">
          <!-- el-upload商品图片上传
action:指定图片上传api接口
:on-preview ：  当点击图片时会触发该事件进行预览操作,处理图片预览
:on-remove ： 当用户点击图片右上角的X号时触发执行
:on-success：当用户点击上传图片并成功上传时触发
list-type ： 设置预览图片的方式
:headers ： 设置上传图片的请求头 -->
          <el-upload :action="uploadURL" :on-preview="handlePreview"
          :on-remove="handleRemove" list-type="picture" :headers="headerObj"
          :on-success="handleSuccess">
            <el-button size="small" type="primary">点击上传</el-button>
          </el-upload>
        </el-tab-pane>
        <el-tab-pane label="商品内容" name="4">
          <!-- 富文本编辑器组件 -->
          <quill-editor v-model="addForm.goods_introduce"></quill-editor>
          <!-- 添加商品的按钮 -->
          <el-button type="primary" class="btnAdd" @click="add">添加商品</
          el-button>
        </el-tab-pane>
```

3.2图片预览,移除某个图片,图片上传成功等功能处理

图片预览：1.通过存在内存中的文件流，获取后台会返回1个正式路径地址，你把此路径写入到图片中即可

移除某个图片: 0.在移除的时候删除的是临时路径 1. 如何获取将要删除的图片的临时路径 ?通过存在内存中的文件流， 从 提交表单图片pics 数组中，找到这个图片对应的索引值。从该数组中splice切除掉

图片上传成功：0.点击上传成功后 ,后台会返回临时路径和正式地址 1. 拼接得到一个图片信息对象 *const* picInfo = { pic: response.data.tmp_path } ！ ！ 注意，因为后端接口需要传pic字段，所以拼接下 2. 将图片信息对象picInfo，push 到表单图片pics数组中。

```
},
// 处理图片预览效果
handlePreview(file) {
  console.log(file)
  this.previewPath = file.response.data.url
  this.previewVisible = true
},
// 处理移除图片的操作
handleRemove(file) {
  // console.log(file)
  // 1. 获取将要删除的图片的临时路径
  const filePath = file.response.data.tmp_path
  // 2. 从 pics 数组中，找到这个图片对应的索引值
  const i = this.addForm.pics.findIndex(x => x.pic === filePath)
  // 3. 调用数组的 splice 方法，把图片信息对象，从 pics 数组中移除
  this.addForm.pics.splice(i, 1)
  console.log(this.addForm)
},
// 监听图片上传成功的事件
handleSuccess(response) {
  console.log(response)
  // 1. 拼接得到一个图片信息对象
  const picInfo = { pic: response.data.tmp_path }
  // 2. 将图片信息对象，push 到pics数组中
  this.addForm.pics.push(picInfo)
  console.log(this.addForm)
},
```

4.使用富文本编辑器插件

想要使用富文本插件vue-quill-editor，就必须先从依赖安装该插件
引入并注册vue-quill-editor，打开main.js，编写如下代码

```
//导入vue-quill-editor（富文本编辑器）
import VueQuillEditor from 'vue-quill-editor'
//导入vue-quill-editor的样式
import 'quill/dist/quill.core.css'
import 'quill/dist/quill.snow.css'
import 'quill/dist/quill.bubble.css'
......
//全局注册组件
Vue.component('tree-table', TreeTable)
//全局注册富文本组件
Vue.use(VueQuillEditor)
```

选择语言

使用富文本插件vue-quill-editor

```
<!-- 富文本编辑器组件 -->
<el-tab-pane label="商品内容" name="4">
  <!-- 富文本编辑器组件 -->
  <quill-editor v-model="addForm.goods_introduce"></quill-editor>
  <!-- 添加商品按钮 -->
  <el-button type="primary" class="btnAdd">添加商品</el-button>
</el-tab-pane>
```

5.点击提交填加商品-需要将已有数据，转换成接口要求的格式。最后提交整个表单

#将goods_cat从数组转换为"1,2,3"字符串形式

#接口要的attrs数组，需要包含商品的动态参数和静态属性。所以需要单独处理将manyTableData（动态参数）和 onlyTableData（静态属性)数据处理添加到attrs

```
// 添加商品
add() {
  this.$refs.addFormRef.validate(async valid => {
    if (!valid) {
      return this.$message.error('请填写必要的表单项！')
    }

    // 调用lodash插件-深拷贝方法cloneDeep(obj),复制1份对象
    //将addForm进行深拷贝，避免goods_cat数组转换字符串之后导致级联选择器报错。因为级联要的是数组，但是接口要的是字符串
    const form = _.cloneDeep(this.addForm)
    //将goods_cat从数组转换为"1,2,3"字符串形式。
    form.goods_cat = form.goods_cat.join(',')
    // 处理动态参数
    this.manyTableData.forEach(item => {
      const newInfo = {
        attr_id: item.attr_id,
        attr_value: item.attr_vals.join(' ')
      }
      this.addForm.attrs.push(newInfo)
    })
    // 处理静态属性
    this.onlyTableData.forEach(item => {
      const newInfo = { attr_id: item.attr_id, attr_value: item.attr_vals }
      this.addForm.attrs.push(newInfo)
    })
    form.attrs = this.addForm.attrs
    console.log(form)

    // 发起请求添加商品
    // 商品的名称，必须是唯一的
    const { data: res } = await this.$http.post('goods', form)

    if (res.meta.status !== 201) {
      return this.$message.error('添加商品失败！')
    }
```

06数据统计-百度echarts

1. 导入 echarts-导入 echarts-由于版本问题，直接导入import echarts from 'echarts'可能会报init错误，可把安装的包卸载 npm uninstall echarts 。重新安装4以下低版本的echarts cnpm i echarts@4

echarts官网找例子：https://echarts.apache.org/examples/zh/index.html

2.调用接口，改变那些需要修改的数据

```
<!-- 2. 为ECharts准备一个具备大小（宽高）的Dom -->
<div id="main" style="width: 750px;height:400px;"></div>
```

```
// 此时，页面上的元素，已经被渲染完毕了！
async mounted() {
  // 3. 基于准备好的dom，初始化echarts实例
  var myChart = echarts.init(document.getElementById('main'))

  const { data: res } = await this.$http.get('reports/type/1')
  if (res.meta.status !== 200) {
    return this.$message.error('获取折线图数据失败！')
  }

  // 4. 准备数据和配置项-对象合并方法
  const result = _.merge(res.data, this.options)

  // 5. 展示数据
  myChart.setOption(result)
},
```

07-项目上线优化

首页优化文章参考-https://blog.csdn.net/weixin_43638968/article/details/109093199

名词解释:

开发环境: npm run dev/serve 在本地运行的，对包的体积基本没要求，就是本地玩的

生产环境:npm run build 生成1个dist文件夹，需要对这个文件进行体积的优化，然后上传到线上运行的

我怎么知道你是在开发或者生产运行的？process.env.NODE_ENV === 'production' 生产环境

开发与上线都会运行index.html文件,这个文件越小,首页加载越快,公司越省钱(大公司)

一.打包介绍:

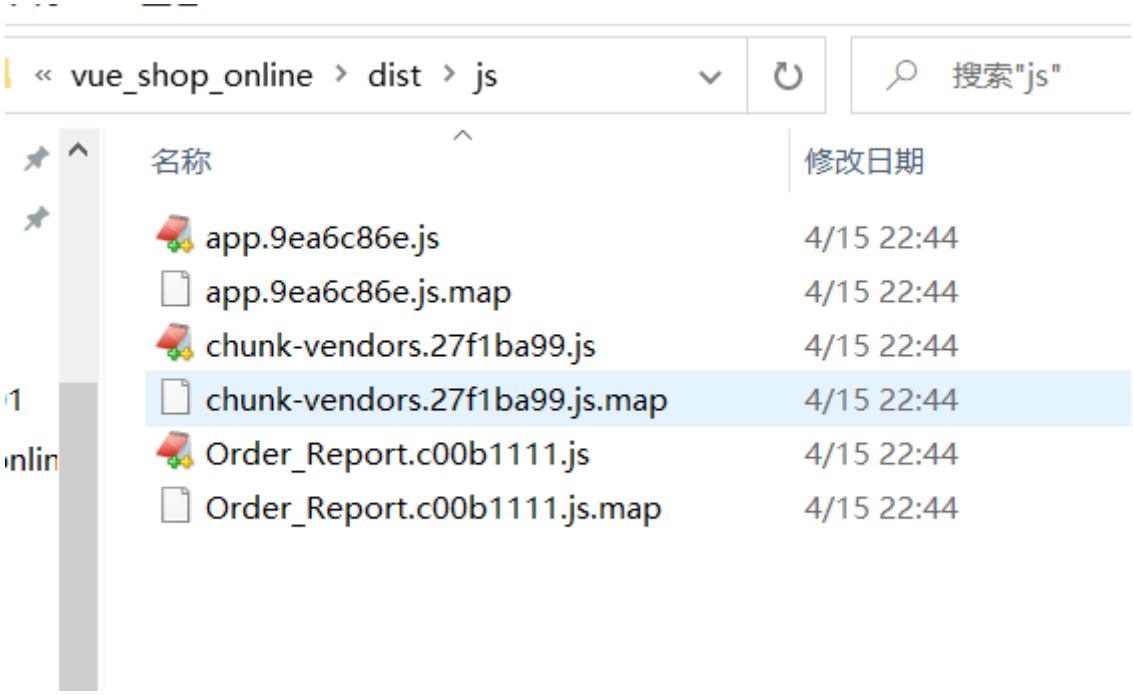
npm run build 生成打包dist文件。dist文件就是要上线的项目

app.js 一般是打包的main.js里的引入

chunk-vendors一般是打包的第三方插件,框架

Order_Report 一般是路由按需加载的名字

.map文件 是.js文件的映射镜像



二.文件配置

默认情况下，依赖项的所有第三方包都会被打包到js/chunk-vendors.**.js文件中，导致该js文件过大，首页加载时太慢。要对其优化,针对开发环境与生产环境单独配置引入文件。或者生成打包报告 npm run build -- *--report* 看打包报告进行优化分析

0.实际工作中,开发环境与生产环境需要经常切换。而它们加载的东西不一样，可通过vue.config.js配置些不同的入口路径,输出路径等

1.要判断是生产环境，还是开发环境？process.env.NODE_ENV === 'production'或者 process.env.NODE_ENV === 'development'

2. 如果是生产就进入生产环境的配置main-prod.js，如果是开发就进入开发的配置main-dev.js环境

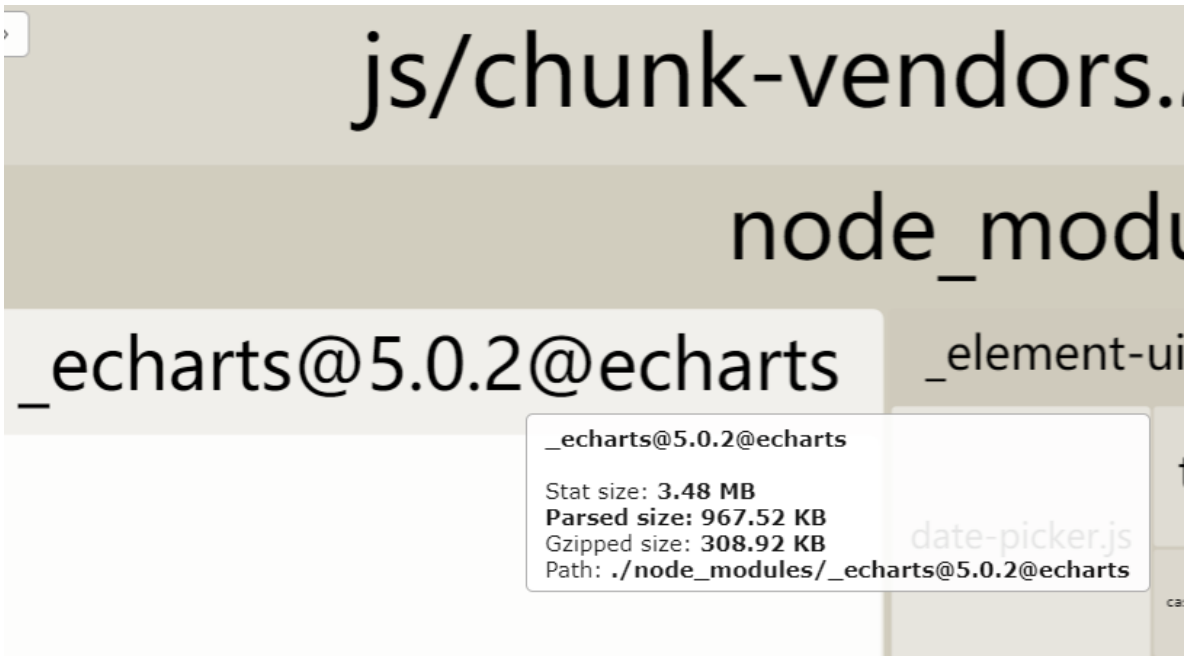
3. 生产环境-一般把第三方插件,用cdn方式引入，超链接src引入。console.log统一移除

三.优化打包体积- 可用npm run build -- *--report* 看打包报告——进行优化分析

stat size:代表原始包大小

parsed size:打包后大小

Gzipped size:如果开启gzip压缩后,大小。后端用nginx开户



四.修改vue.config.js文件——配置webpack去找不同的入口文件，目的是针对生产环境进行代码抽离优化

```
module.exports = {
    //配置输出路径
    publicPath: './',
        //引导webpack去找入口文件，即不同的main.js
    chainWebpack: config => {
        config.when(process.env.NODE_ENV === 'production', config => {
            config.entry('app').clear().add('./src/main-prod.js')
     //生产模式下打包时,把第3方插件排除在外不打包,这样可减少chunk文件的体积，提升性能。可把第3方插件放在打包后的index.html里，通过链接方式引入。不过使用externals属性要注意的是，虽然可以优化首屏加载速度，但是由于静态资源分离，也会增加http请求数量。所以如果是小项目，最好就不要用externals属性，因为小项目打包的出来的vender.js体积不大，建议项目体量较大的项目再用比较合适。

     //说一下这里的键值对配置，key名vue是包名,这个值是import from 'vue'时用的名称。value值是别名,是你在项目中用的别名
     //参考文章  https://www.jianshu.com/p/f6b3f097a56d
            config.set('externals', {
                'vue': 'Vue',
                'element-ui': 'ELEMENT',  //注意这个名字，不能叫其它的
                'echarts': 'echarts',
            })
        config.plugin('html').tap(args => {
            return args
        })
     })
     config.when(process.env.NODE_ENV === 'development', config => {
        config.entry('app').clear().add('./src/main.js')
        config.plugin('html').tap(args => {
            return args
        })
     })

},
    //去除生产环境的代码镜像-productionSourceMap
    productionSourceMap: false,
}
```

五.在public文件里找到index.html进行cdn引入

```
index.html
public > index.html > html > head
OCTYPE html>
ml lang="en">
head>
 <meta charset="utf-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width,initial-scale=1.0">
 <link rel="icon" href="<%= BASE_URL %>favicon.ico">
 <title>test</title>
    <!-- 如果是生产环境才走cdn。！！！注意得加判断，否则会有控制台错误 -->
    <% if (process.env.NODE_ENV==='production' ) { %>
      <!-- 引入vue的cdn，因为elment依赖vue -->
      <script src=https://cdn.staticfile.org/vue/2.5.22/vue.min.js></script>

      <!-- 引入elment样式 -->
      <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
      <!-- 引入elment组件库 -->
      <script src="https://unpkg.com/element-ui/lib/index.js"></script>
      <!-- 引入echa 关注链接 (ctrl + 单击)
      <script src="https://cdn.staticfile.org/echarts/4.1.0/echarts.min.js"></script>

      <% } %>
/head>
body>
```

六.优化其它项

0.去除sourcemap镜像文件,map文件比较占体积

1.打包输出目录，要变成./，否则加载index.html文件时，找不到其它js,css文件

2.开发环境配置,一般可配置跨域

```
//打包输出目录
publicPath:'./',
//开发环境-配置
devServer: {
 open: true, // 自动启动浏览器
 host: '0.0.0.0', // localhost
 port: 6060, // 端口号
 hotOnly: false, // 热更新

 overlay: {···
  },
  proxy: {
    //配置跨域-只能用于开发环境,上线环境不起作用
    '/api': {
      target: 'http://ustbhuangyi.com/sell/api/seller', // 接口的域名 在页面调用时用 /
      api/
      // ws: true, // 是否启用websockets
      changOrigin: true, // 开启代理，在本地创建一个虚拟服务端
      pathRewrite: {
        '^/api': '/'
      }
    }
  }
},
//去除生产环境的代码镜像-productionSourceMap
productionSourceMap: false,
```

4.配置上线console.log输出，需要配合插件，在babel.config.js里配置下

```
// 项目上线时-去除console.log 。注意需要安装cnpm i babel-plugin-transform-remove-console -D
const prodPlugins = []
if (process.env.NODE_ENV === 'production') {
  prodPlugins.push('transform-remove-console')
}

module.exports = {
  'presets': [
    '@vue/app'
  ],
  //此处为按需引入的elment,你们用全局引入的不用管
  'plugins': [
    [
      'component',
      {
        'libraryName': 'element-ui',
        'styleLibraryName': 'theme-chalk'
      }
    ],
  //别忘记引入去除console插件
    ...prodPlugins,
  ]
}
```

08-相关面试题

项目概述：本项目，是一个商品后台管理系统。采用前后端分离开发，主要用于给内部运营人员管理商品用的。主要有登录模块(登录注册忘记密码等页面),用户管理模块(管理内部登录用户的),权限管理(包含角色列表,权限列表[不同人有不同权限]),商品管理(管理商品的增删改查)，数据统计(领导看商品销售数据，主要用到echarts)等模块

一.登录模块:

1.登录鉴权如何做的?

1.前台传入用户名和密码，后台返回token。把token存到本地或vuex 2.通过router的全局守卫beforeEach进行鉴权处理,判断是否有token 3.进入到后台，请求其它数据要带着token,在axios请求拦截器,可统一设置请求头协议header加token值

2.登录后你们的token有没有过期时间?过期了怎么处理?

有过期时间，一般是一天,后台返回错误码（一般是4001，5001）提示我过期了或者无效token，进行判断提醒用户跳转重新登录

3.登录验证码怎么实现的?登录密码有没有加密传给后台?

后端提供了一个接口，返回了图片路径，前端渲染就行了

有加密，通过引入md5插件加密下传给后台，后台解析下就可以了

二.用户模块-管理网站不同登录用户的

1.你们分页，搜索，分配角色是怎么实现的?

分页功能 :分为前端分页和后端分页，大部分都是让后端进行分页.

我们分页是后端进行的分页，前端通过UI框架提供的分页组件实现,需要后台返回总页数total,当前第几页pagenum

2.分配角色:需要通过作用域插件传过来当前行信息，下拉时获取选择的角色id传给后台

三.权限管理模块

1.你们的权限是如何实现的？根据前台登录不同用户，后台返回该用户的权限路径,然后前端在路由表里，把所有的路径都写上。这样其实就是静态路由

2.你是静态路由，还是动态路由？你有没有听说过动态路由？

所谓动态路由就是通过后台返回的路由路径，动态填加的，router.addRoutes()

3.分配权限—

后台返回的是树结构数据,结合绑定tree组件,进行渲染就行了。

打开弹窗时，有默认选中如何实现？给它绑定一个数组，通过递归遍历，拿到该角色下所有权限id，给它推入到该默认选择数组中。

提交更新权限-传入角色id,和所有选中的权限id给后台

四.商品管理模块-对所有商品进行增删改查操作

图片上传如何实现的？1.通过upload组件进行上传，前端给它url上传路径和头协议token值 2.上传成功后，后台返回临时路径(前台有删除操作,从file文件里删除的临时路径)和正式地址(做预览操作)

如果用原生实现上传，说下思路？1.通过input标签,type=file 2.文件协议是通过form-data传输二进制流 3.提交接口路径

五.echarts图表-主要给领导看一些销售数据用的

通过引入echarts插件，找到相关图表，复制里面代码,修改options配置里data数据

六.首屏加载如何优化?

0.首先要区分开发环境和生产环境,杂区分？创建2个文件，里面定义了环境变量。通过process.env.NODE_ENV === 'production' 生产环境，process.env.NODE_ENV === 'development' 开发环境

1.把大的第三方插件-echarts,elmentui抽离出去。通过webpack插件chainWebpack，配置链式调用找到生产环境的main,单独配置externals抽离。在index.html通过cdn引入链接

2.SourceMap镜像文件配置成false

3.路由懒加载

4.把console.log 移除

5.文件压缩(后端开户gzip压缩)