# 接口文档

一.商品展示模块，这里需要获得商品数据的api，get请求

所需要的参数是：

0.商品idgoodsId：Long;

1.在购买模块中的商品图片image1：url

2.商品名称name：string

3.商品价格price：number

4.商品描述description：string

在商品详情模块中

5.商品展示图片image2：url

6.商品详情details：{

  brand:string

  model:string

  color:string

  size:string

  price:number

  description:string

}

二.进行前后端接口交互

1.第一步为了避免跨域问题我们要在vite.config.json对于api根域进行封装

如下：

 server:

  {

    proxy: {

      '/api': {

        target: 'http://localhost:8080/SuperShop',

        changeOrigin: true,

        rewrite: (path) => path.replace(/^\/apis/, ''),

      },

  }},

  2.第二步：封装api,在utils文件夹下创建一个文件http3.js，使用axios进行基础封装

  2.1

  如下：

    import axios from "axios";

    import { ElMessage } from 'element-plus'//这个用于请求失败时的错误情况弹出

    const  httpInstance = axios.create({

    baseURL: '/apis',

    timeout: 5000

    })

  2.2

  如下：

    // axios请求拦截器

      httpInstance.interceptors.request.use(config => {

      return config

      }, e => Promise.reject(e))

  2.2响应式拦截与捕捉错误信息

  如下：

    // axios响应式拦截器

      httpInstance.interceptors.response.use(res => res.data, e => {

   /// 确保 e.response 和 e.response.data 存在

      const message = e.response?.data?.message || '请求出错，请稍后再试';

   // 统一错误提示

     ElMessage({

     type: 'warning',

     message: message

   });

     return Promise.reject(e)

   })

    export default httpInstance //一定要记得导出

 3.请求接口获取对象数据

 3.1请在api文件夹下创建一个文件Goods.js，进行接口的进一步封装

 3.2具体内容如下：

   export const  getGoodsAPI = () => {

    return httpInstance({

      url:'/goods/list'

    })

  }//httpInstance这个是之前封装根域导出的内容记得引入

  3.3在组件中引入接口请求获取数据

  如下：

  import { ref, onMounted } from 'vue'

  import { getGoodsAPI } from '@/apis/goods';

  //获取数据列表

  const goodProduct = ref([])

  const getGoods = async () => {

  const res = await getGoodsAPI()

  goodProduct.value = res.result  //依据后端返回的数据属性值来看

  }

  onMounted(()=>getGoods())//V中的一个生命周期钩子

  接下来你就可以调用goodProduct来获取数据了

  例如 <image src="{{goodProduct.image1}}"/>

三.当客户具有购买商品，填写购买信息点击购买按钮时，需要调用后端接口将购买者相关信息存储在Customers数据库中的buyers表中

所需参数：

        name: String,

        phone: String,

        wechat: String,

        goodsId: Long,

四.利用接口进行交互

上面1-2步骤做过一次之后就不需要做了，因为接口的根域是相同的，现在直接开始第三部

3.在apis文件下创建一个文件Costumes.js，进行接口的进一步封装(以后所有关于Costumer的接口都封装在这里)

  export const postbuyersAPI = (buyerInfo, goodsId) => {

  return httpInstance({

    url: '/goods/buy',

    method: 'POST',

    data: {

      ...buyerInfo, // 这里展开 buyerInfo 对象

      goods: {

        goodsId, // 直接使用传入的 goodsId

      },

    },

4.使用接口进行业务处理

  methods: {

    onPurchase() {

      this.purchaseDialogVisible = true;

    },

    submitPurchase() {

      this.$refs.purchaseForm.validate((valid) => {

        if (valid) {

          postbuyersAPI(this.buyerInfo, this.goodsproduct.goodsId).then(response => {//这里使用了api接口，请看表单绑定的数据对象为：buyerInfo

            this.$message.success(`成功购买了 ${this.product.name}`);

            this.purchaseDialogVisible = false;

            this.resetForm();

          }).catch(error => {

            this.$message.error('购买失败，请重试！');

          });

        } else {

          this.$message.error('请完善购买信息！');

        }

      });

    },

    resetForm() {

      this.buyerInfo = {

        name: '',

        phone: '',

        wechat: ''

      };

      this.$refs.purchaseForm.resetFields();

    },

  },

/\*

登录模块这里也是需要一个登录的api接口，然后调用这个接口进行登录验证。

一.首先这个是一个post请求，不需要接收数据比较方便，在这前两步都相同就是第三步不同

export const loginAPI=({username, password})=>{//利用解构赋值 直接传俩参数，利于维护，他们说如果不去解构的话，都不知道要post什么数据

    return httpInstance.post(

        '/admin/login',

        {

            username,

            password

        },

       { headers: {

            'Content-Type': 'application/json'

          }

        }

    )

}

二.直接调用接口，如下内容

//二.获取form实例做统一校验

const formRef = ref(null)

const router = useRouter()

//const {username,password}=form.value//获取表单的两个字段

const doLogin = ()=>{

//调用实例方法

formRef.value.validate(async(valid) => {

     if(valid){

         // TODO LOGIN

     const res =await loginAPI(form.value)//调用api验证账户是否正确,

     console.log(res)

      // 1. 提示用户

      ElMessage({ type: 'success', message: '登录成功' })

     //2.利用cookies存储用户信息，但是现在只有一个用户先不写了

     //3.验证成功后跳转到相关页面

      router.replace({ path: '/back' })

     }else{

      console.log('登录失败')

      return false

     }

  })

}

/\*

一.这里上传商品需要个接口

1.接口封装

export const addProductAPI = (productData) => {

return httpInstance({

  url: '/goods/list', // 假设这个 URL 是商品上传的接口

  method: 'POST',

  data: { ...productData }, // 这样使用展开运算符

});

};

2.使用接口

async onSubmit() {

    this.$refs.productForm.validate(async (valid) => {

      if (valid) {

        try {

          const response = await addProductAPI(this.product);

          this.$message.success('商品添加成功！');

          // 根据需要重置表单或处理后续逻辑

          this.onReset();

        } catch (error) {

          this.$message.error('商品添加失败: ' + error.message);

        }

      } else {

        this.$message.error('请完善商品信息！');

        return false;

      }

    });

  },

/\*

一.查看交易历史也需要用到接口

1.

export const getTransactionHistoryAPI = () => {

  return httpInstance({

    url: '/transactions/history', // 假设这是获取交易历史的接口 URL

    method: 'GET',

  });

};

/\*

其实可以使用pinia来管理商品数据，这样只用放送一次商品的数据请求，但是项目刚开始，所以就先不进行优化了

假设product是使用api获取得到的数据

一.在显示购买方的时间需要接口获取买方列表

1.接口封装

export const getbuyersAPI = () => {

  return httpInstance({

    url: '/goods/buy', // 假设这个 URL 返回所有购买者的列表

    method: 'GET',

  });

};

2.利用接口获取的列表数据

  import { ref, onMounted } from 'vue';

  import { getbuyersAPI } from '@/apis/buyers'; // 请替换为实际的 API 文件路径

  // 获取购买者列表

  const buyers = ref([]);

  const getBuyers = async () => {

    try {

      const res = await getbuyersAPI();

      buyers.value = res.data; // 根据后端返回的数据属性值

    } catch (error) {

      console.error('获取购买者列表失败:', error);

    }

  };

  onMounted(() => getBuyers()); // 在组件挂载时调用获取购买者列表

3.将获取的数据用于渲染

<div class="buyer-list">

    <el-card v-for="buyer in buyers" :key="buyer.id" class="buyer-card" @click="selectBuyer(buyer)" >

      <p>购买者: {{ buyer.name }}</p>

      <p>联系电话: {{ buyer.contact }}</p>

      <p>联系微信: {{ buyer.contact }}</p>

    </el-card>

  </div>

二.这里有一个冻结商品的业务，当商品库存stock为0时，

商品将不允许购买者购买，并将商品信息从数据库的删除

而且还可以将交易者的信息上传到Costumes数据库中trader表中。

1.接口封装

export const freezeProductAPI = (productId, traderId，Nowtime) => {//这里traderId是buyerId

  return httpInstance({

    url: '/goods/freeze', // 假设这个 URL 是冻结商品的接口

    method: 'POST',

    data: {

      productId,

      traderId,

      Nowtime

    },

  });

};

2.调用接口

  methods: {

    selectBuyer(buyer) {

      this.selectedBuyer = buyer;

      this.freezeDialogVisible = true;

    },

    confirmFreeze() {

      // 调用冻结商品接口

      freezeProductAPI(this.product.id, this.selectedBuyer.id)

        .then(response => {

          this.$message.success(`商品已冻结给购买者 ${this.selectedBuyer.name}`);

          this.freezeDialogVisible = false;

        })

        .catch(error => {

          this.$message.error('冻结商品失败: ' + error.message);

        });

    }

  }

};

\*/