下面是Controller\login.php代码

<?php  
namespace app\admin\controller;  
  
use think\facade\Db;  
use think\facade\Request;  
  
class Login  
{  
 public function login()  
 {  
 try {  
 // 基础参数校验  
 $data = Request::post();  
 if (empty($data['username']) || empty($data['password'])) {  
 throw new \Exception('请输入用户名和密码', 400);  
 }  
  
 // 模拟登录验证（实际应验证密码）  
 $user = Db::*name*('users')  
 ->alias('u')  
 ->leftJoin('departments d', 'u.dept\_id = d.dept\_id')  
 ->leftJoin('party\_branches p', 'u.branch\_id = p.branch\_id')  
 ->field('u.user\_id,u.real\_name,u.avatar,u.role\_id,u.dept\_id,u.branch\_id,d.dept\_name,p.branch\_name')  
 ->where('u.username', $data['username'])  
 ->find();  
  
 if (!$user) {  
 throw new \Exception('用户不存在', 404);  
 }  
  
 // 获取动态菜单  
 $menuData = $this->buildMenu($user['role\_id']);  
  
 // 生成访问令牌  
 $token = md5('hospital\_token\_' . date('YmdHis'));  
  
 return json([  
 'code' => 200,  
 'data' => [  
 'userInfo' => [  
 'userId' => $user['user\_id'],  
 'realName' => $user['real\_name'],  
 'avatar' => $user['avatar'] ?? '/static/avatar/default.png',  
 'department' => $user['dept\_name'] ?? '未分配科室',  
 'partyBranch'=> $user['branch\_name'] ?? '未分配党组织'  
 ],  
 'menuList' => $menuData,  
 'token' => $token,  
 'expire' => time() + 7200  
 ],  
 'message' => '登录成功'  
 ]);  
  
 } catch (\Exception $e) {  
 return json([  
 'code' => $e->getCode() ?: 500,  
 'message' => $e->getMessage(),  
 'tip' => '系统代码：HOSP\_' . date('His') . uniqid()  
 ]);  
 }  
 }  
  
 */\*\*  
 \* 构建动态菜单结构  
 \*/* private function buildMenu($roleId)  
 {  
 $role = Db::*name*('role')->where('id', $roleId)->value('menus');  
 $menuIds = json\_decode($role, true) ?: [];  
  
 $menus = Db::*name*('menu')  
 ->whereIn('id', $menuIds)  
 ->order('sort ASC')  
 ->select()  
 ->toArray();  
  
 return $this->buildTree($menus);  
 }  
  
  
 */\*\*  
 \* 递归构建菜单树  
 \*/* private function buildTree($menus, $parentId = 0)  
 {  
 $tree = [];  
 // 提取当前层级菜单  
 $currentLevel = array\_filter($menus, function($m) use ($parentId) {  
 return $m['parent\_id'] == $parentId;  
 });  
  
 // 纯sort升序排序（核心修改）  
 usort($currentLevel, function($a, $b) {  
 return $a['sort'] - $b['sort']; // 直接比较sort值  
 });  
  
 // 构建树形结构  
 foreach ($currentLevel as $menu) {  
 $children = $this->buildTree($menus, $menu['id']);  
 $node = [  
 'path' => $menu['path'],  
 'name' => str\_replace('/', '', $menu['path']),  
 'label' => $menu['title'],  
 'icon' => $menu['icon'],  
 'url' => ucfirst(explode('/', $menu['path'])[1] ?? 'Home')  
 ];  
 if (!empty($children)) {  
 $node['children'] = $children;  
 }  
 $tree[] = $node;  
 }  
 return $tree;  
 }  
}

下面是vite项目中login.vue的代码

<template>

    <div class="body-login">

        <el-form :model="loginForm" class="login-container">

            <h3>欢迎登录</h3>

            <el-form-item>

                <el-input type="input" placeholder="请输入账户" v-model="loginForm.username"></el-input>

            </el-form-item>

            <el-form-item>

                <el-input type="password" placeholder="请输入密码" v-model="loginForm.password"></el-input>

            </el-form-item>

            <el-form-item>

                <el-button type="primary" @click="login"> 登录 </el-button>

            </el-form-item>

        </el-form>

    </div>

</template>

<script setup>

import {reactive,getCurrentInstance} from 'vue'

import { useAllDataStore } from '@/stores/index.js'

import { useRouter } from 'vue-router'

const { proxy } = getCurrentInstance()

const store=useAllDataStore()

const router=useRouter()

const loginForm = reactive({

  username: 'admin',

  password: 'admin',

});

const login=async ()=>{

    const res = await proxy.$api.getMenu(loginForm);

    if(res){

        // 新增本地存储（需同步更新store）

        localStorage.setItem('access\_token',  res.token);

        store.updateMenuList(res.menuList)

        //在这里执行添加路由方法,并传入router

        store.addMenu(router)

        store.state.token=res.token

        store.state.userInfo=res.userInfo

        // 强制刷新路由（避免动态路由加载问题）

        //window.location.reload();

        console.log(res)

        router.push("/home")

    }

}

</script>

<style scoped lang="less">

.body-login{

    position: fixed;

    top: 0;

    left: 0;

    width: 100vw;

    height: 100vh;

    background-image: url("../assets/images/background.png");

    background-size: cover;

    background-position: center;

    background-repeat: no-repeat;

    display: flex;

    align-items: center;

    justify-content: center;

}

.login-container {

  width: 350px;

  background-color: #fff;

  border: 1px solid #eaeaea;

  border-radius: 15px;

  padding: 35px 35px 15px 35px;

  box-shadow: 0 0 25px #cacaca;

  margin: 180px auto;

  h3 {

    text-align: center;

    margin-bottom: 20px;

    color: #505450;

  }

  :deep(.el-form-item\_\_content) {

    justify-content: center;

  }

}

</style>

下面是request.js的代码

import axios from "axios";

import { ElMessage } from "element-plus";

import config from "@/config"

const service = axios.create();

//const NETWORK\_ERROR = "网络错误……";

//添加请求拦截器

service.interceptors.request.use(

    function(config){

        const accessToken = localStorage.getItem('access\_token');

        if (accessToken) {

            config.headers.Authorization  = `Bearer ${accessToken}`;

            config.headers['X-CSRF-TOKEN']  = getCSRFToken(); // 可选安全增强

        }

        return config;

    },

    function(error){

        return Promise.reject(error)

    }

);

//添加相应拦截器

service.interceptors.response.use((res)=>{

    const {code,data,msg}=res.data;

    if(code===200){

        //console.log("正常返回");

        //console.log(data)

        return data;

    }else{

        console.log("异常");

        const NETWORK\_ERROR = "网络错误…";

        ElMessage.error(msg || NETWORK\_ERROR);

        return Promise.reject(msg || NETWORK\_ERROR)

    }

});

function request(options){

    options.method = options.method || "get";

    //统一post和get请求参数传递方式

    if(options.method.toLowerCase()==="get"){

        options.params = options.data;

    }

    //对mock的开关进行处理

    let isMock=config.mock;

    if(typeof options.mock!=="undefined"){

        isMock=options.mock;

    }

    //针对环境做一个处理

    if(config.env==="prod"){

        //不能用mock

        service.defaults.baseURL=config.baseApi;

    }else{

        service.defaults.baseURL=isMock ? config.mockApi : config.baseApi;

    }

    return service(options);

}

export default request;