# 6.3 登录控制实现方案及核心代码示例

### 一、技术方案设计要点

1. **验证方式**

* 1. 管理员/志愿者账号密码登录

* 1. 微信小程序通过 openid+手机号 绑定登录

1. **状态管理**

* 1. 使用 Session + JWT Token + Redis 组合方案

* 1. Token有效期：2小时（活动令牌）/7天（刷新令牌）

1. **安全策略**

* 1. 密码存储：bcrypt 加密

* 1. Token加密：HMAC-SHA256 签名

* 1. 异地登录预警：记录设备指纹/IP变更

### 二、核心代码实现示例

#### 1. 登录接口（ThinkPHP8）

// app/index/controller/Auth.php   
public function login()  
{  
 $username = $this->request->post('username');  
 $password = $this->request->post('password');  
   
 // 验证用户状态   
 $user = UserModel::where('username', $username)  
 ->where('status', 1)  
 ->find();  
 if (!$user || !password\_verify($password, $user->password)) {  
 return json(['code' => 401, 'msg' => '账号或密码错误']);  
 }  
   
 // 生成双令牌   
 $payload = [  
 'uid' => $user->id,  
 'type' => $user->type, // 用户类型：1管理员 2志愿者   
 'exp' => time() + 7200 // 2小时过期   
 ];  
 $accessToken = Jwt::generateToken($payload);  
 $refreshToken = Jwt::generateToken(['uid' => $user->id], 604800);  
   
 // Redis存储关联   
 Redis::set('user:token:'.$user->id, json\_encode([  
 'access' => $accessToken,  
 'refresh' => $refreshToken,  
 'device' => get\_client\_device() // 获取设备信息   
 ]));  
   
 return json([  
 'code' => 200,  
 'data' => [  
 'access\_token' => $accessToken,  
 'refresh\_token' => $refreshToken,  
 'user\_info' => $user->hidden(['password'])  
 ]  
 ]);  
}

#### 2. 认证中间件

// app/index/middleware/AuthCheck.php   
public function handle($request, \Closure $next)  
{  
 $token = $request->header('Authorization');  
 if (!$token || !Jwt::verifyToken($token)) {  
 return json(['code' => 401, 'msg' => '登录已过期']);  
 }  
   
 // Redis校验令牌有效性   
 $payload = Jwt::parseToken($token);  
 $redisData = Redis::get('user:token:'.$payload['uid']);  
 if (!$redisData || json\_decode($redisData)->access != $token) {  
 return json(['code' => 401, 'msg' => '账号已在其他设备登录']);  
 }  
   
 // 权限校验（示例：仅允许管理员访问后台）  
 if ($payload['type'] != 1 && strpos($request->pathinfo(), 'admin/') !== false) {  
 return json(['code' => 403, 'msg' => '无权限操作']);  
 }  
   
 return $next($request);  
}

### 三、前端交互流程

1. **登录凭证存储**

// Vue3示例（PC端）  
axios.post('/auth/login', formData).then(res => {  
 if(res.data.code === 200){  
 localStorage.setItem('access\_token', res.data.data.access\_token);   
 localStorage.setItem('refresh\_token', res.data.data.refresh\_token);   
 store.commit('SET\_USER\_INFO', res.data.data.user\_info);   
 }  
})

1. **请求拦截器**

axios.interceptors.request.use(config => {  
 const token = localStorage.getItem('access\_token');   
 if (token) {  
 config.headers.Authorization = `Bearer ${token}`;  
 }  
 return config;  
});

1. **Token自动刷新**

axios.interceptors.response.use(response => {  
 return response;  
}, error => {  
 if (error.response.status === 401 && !error.config.\_retry) {  
 const refreshToken = localStorage.getItem('refresh\_token');   
 // 调用刷新接口获取新token   
 return refreshAuthToken().then(() => {  
 error.config.\_retry = true;  
 return axios(error.config);   
 });  
 }  
 return Promise.reject(error);   
});

### 四、安全增强措施

1. **登录日志记录**

// 记录登录日志   
LogModel::create([  
 'user\_id' => $user->id,  
 'login\_ip' => get\_client\_ip(),  
 'login\_time' => time(),  
 'user\_agent' => $this->request->header('user-agent')  
]);

1. **并发控制**

// 限制同一账号5分钟内最多尝试5次   
$key = 'login:attempt:'.$username;  
if (Redis::get($key) >= 5) {  
 return json(['code' => 429, 'msg' => '尝试次数过多，请稍后再试']);  
}  
Redis::incr($key);  
Redis::expire($key, 300);

该方案已满足知识库中描述的登录控制需求，包括多端认证、权限隔离、安全审计等核心功能。实际部署时需根据业务场景调整Token有效期和安全策略参数。