

# 第十三届中国软件杯大赛

## A8-基于国产操作系统的应用融合软件

——Any2Remote 队伍：以上队伍全部无效

### 摘 要

Any2Remote 项目是为了解决国产操作系统与 Windows 系统之间的应用融合问题而开发的一套跨平台兼容软件。随着国内 IT 产业的蓬勃发展，自主可控的开放生态正在逐步形成，全球 IT 生态格局正从过去的“一极”向“两极”演变。在这一背景下，两种生态的应用程序能够在同一桌面系统中融合使用的需求日益增长，Any2Remote 应运而生。这款软件允许用户在国产操作系统上无缝使用 Windows 应用，有效解决了部分业务软件在国产系统中的“水土不服”问题。Any2Remote 采用了前沿技术，在 Windows 端使用 .NET 架构和 WinUI3 设计，提供原生应用级的使用体验；在国产操作系统端则采用 Electron 配合 Vue 框架进行开发，确保了强大的跨平台兼容性。通过重新制作编译适用于国产操作系统的 FreeRDP 通信技术，实现了软件的核心功能包括远程应用发布、实时通信、安全认证、应用生命周期管理以及外接存储透传等。通过 ASP.NET Core SignalR 实现的实时通信机制，确保了远程应用的实时管理和同步。在安全性方面，Any2Remote 使用自签名证书和 HTTPS 协议，保障了通信过程的加密和安全。此外，软件还提供了全面的应用生命周期管理，从安装、发布、使用到卸载的全过程都可以在国产系统上完成。通过精心设计的用户界面和流畅的操作体验，Any2Remote 不仅解决了兼容性问题，还为用户提供了一个统一、高效的工作环境。该项目的开发代表了对国产化环境大趋势的积极响应，为推动国产操作系统的广泛应用和 IT 生态的协调发展做出了重要贡献，同时也为未来更多创新性跨平台解决方案的出现铺平了道路。

# 目录

1 项目背景.....	4
2 任务描述.....	4
3 创意描述.....	4
3.1 整体架构选型.....	4
3.2 WINDOWS 端 .....	5
3.3 国产操作系统端.....	5
4 项目实施的技术方案及原理.....	6
4.1 WINDOWS 端功能设计 .....	6
4.1.1 Windows 端后台服务 .....	6
4.1.1.1 自签名证书部署与初始化.....	6
4.1.1.2 实时通信.....	7
4.1.1.3 Restful API.....	7
4.1.2 Remote App 管理 .....	8
4.1.3 Any2Remote 系统服务与高级功能 .....	10
4.1.3.1 Any2Remote 增强模式.....	10
4.1.3.2 重启所有服务.....	11
4.1.3.3 重置.....	11
4.1.4 客户端连接管理与用户.....	12
4.1.4.1 查看相关会话连接信息.....	12
4.1.4.2 断开指定连接.....	13
4.1.4.3 注销指定用户会话.....	13
4.1.5 UI 设计 .....	14
4.1.5.1 颜色与排版.....	14
4.1.5.2 组件与界面布局.....	14
4.2 国产操作系统端功能设计 .....	15

4.2.1 连接 Windows .....	15
4.2.1.1 联通性测试.....	15
4.2.1.2 连接至远程 Windows 设备 .....	15
4.2.2 管理应用.....	16
4.2.2.2 添加、删除发布应用.....	17
4.2.3 应用发布至桌面.....	18
4.2.4 应用远程安装.....	19
4.2.5 应用卸载.....	19
4.2.6 外接介质透传.....	21
4.2.7 UI 设计.....	22
4.2.7.1 颜色与排版.....	22
4.2.7.2 组件与界面布局.....	23
4.2.7.3 设计工具与资源.....	24
4.3 双端通信设计.....	24
4.3.1 FreeRDP.....	24
4.4 项目结构图.....	错误!未定义书签。
<b>5 运行环境.....</b>	<b>28</b>
5.1 开发环境 .....	28
5.2 项目工程可运行环境.....	28
<b>6 结语 .....</b>	<b>28</b>

# 1. 项目背景

近二十多年来，主流 IT 底层标准、架构、生态等大多都由国际 IT 大厂制定，随着近几年国内软硬件 IT 厂商蓬勃发展，正在逐步形成自主可控的开放生态，全球 IT 生态格局将由过去“一极”向“两极”演变，两种生态的应用程序可以在一个桌面系统中融合使用的需求将越来越普遍。

随着国产 IT 软硬件在办公场景下被快速推广使用，原有的 Intel 和 Windows 计算机体系已经开始逐渐迁移到国产计算机生态体系，而新的计算机环境（国产操作系统）下软件生态的完善、兼容和替代还需要长期持续发展。而对于用户使用而言，原有的大量 Windows 应用不兼容，导致正常业务不可用或不稳定。比如有些行业软件在国产系统中还没有适配，无法使用，如 IE 浏览器只有 Windows 版，无基于 Linux 系列操作系统的版本，有些医疗行业医院管理信息系统 HIS 软件用到了 IE 浏览器插件，必须在使用 IE 浏览器，但是一些日常的 Office 软件（如 WPS）在国产系统中已经较成熟。

我们希望设计一个能让已在国产系统中适配和未适配的 2 类软件可以在同一台国产 PC 环境上正常使用并且不会影响原国产系统的软件。

## 2. 任务描述

本任务为实现一套跨平台兼容软件，跨 Windows 和国产化 2 类操作系统，将未适配国产系统的应用通过我们开发的软件可以在国产操作系统中使用，满足国产化生态下的办公业务需求。团队需要利用分别部署了国产操作系统和 Windows 操作系统的 PC 进行开发软件的客户端和服务端，实现该融合软件的安装部署和应用的融合的基础功能以及应用数据管理和应用生命周期管理的高级拓展功能。

项目工程实现了融合软件客户端的启动时间在 10s 以内，融合软件客户端运行之后不会影响原国产系统的应用软件的正常运行，Windows 应用运行时使用流畅，不会出现明显的卡顿，可以在国产系统中同时开 Windows 应用和国产系统的应用窗口，安装国产操作系统为麒麟桌面版 V10 SP1 2403，确保开发的融合软件符合要求。

## 3. 创意描述

### 3.1 整体架构选型

针对 Windows 端和国产系统端运行环境的不同，我们在 Windows 端采用了 .NET 架构进行开发，原生适配 Windows 底层操作，与 Windows 深度集成，可以方便地调用 Win32 API，充分利用 Windows 系统的各种特性和服务。在国产系统端，考虑到系统兼容性等因素，我们采用了 Electron 配合 Vue 框架进行开发，适配各种不同的运行内核与环境，具备强大的跨平台兼容性。得益于 Web 技术，我们可以创建更加现代化的、交互性好的用户界面。

## 3.2 Windows 端

在设计上，Windows 采用 Fluent Design 设计，使用随 Windows 应用 SDK 提供的 WinUI3 从头构建的新式 Windows 桌面应用，打造原生应用级的使用体验。

通过在 ASP.NET 托管的 SignalR Core 服务，结合 WebSocket 与 长轮询，实现了实时的应用推送机制，将 Windows 端发布的应用实时推送到国产操作系统上

结合 Powershell 内置的自签名生成命令和 Windows 自身证书服务 SDK，实现了自签名证书颁布并在 Windows 端部署，在 Https 通信过程实现了加密，保障了安全性。

通过调用 Windows COM 组件与相关 Win32 API 对打开的应用快捷方式或 exe 可执行文件进行解析，获取应用名称，图标，路径地址，卸载路径等详细信息。

通过对系统注册表信息的解析实现了对电脑已安装应用的检索，用户可选取应用一键进行应用发布。

内置的 RDP Sharp 部署系统，通过 DLL 注入等手段修改系统关键文件与服务解除了 Windows 只能对单个应用只能开启一个会话，Windows 家庭版不可使用 RDP 服务的限制，在国产操作系统上实现了 Windows 应用多开，在单台 Windows 主机上实现了多用户同时访问。

## 3.3 国产操作系统端

我们选取了 RDP 作为国产系统端的传输协议，并考虑使用开源实现 FreeRDP 完成对 RDP 客户端连接支持。由于系统内自带的 FreeRDP 版本过于老旧，对远程应用传输的支持不佳，我们重新制作编译了适用于国产操作系统的专门版本。通本地缓存技术，我们可以实现在不同 windows 服务端之间的快速切换。

在通信安全方面，双端通信采用了 Https 技术，相比于 Http 通信，Https 通过 CA 证书认证以及 SSL/TLS 协议进行加密，有效防止了信息被第三方窃取，在需要运行国产操作系统的环境下提供了更加安全的保障，防止信息泄露。

通过对电脑接入的外部介质检测，我们实现了将电脑的外接 U 盘等介质进行透传。在使用 windows 应用时可直接将文件保存在本地介质上。用户还可选择性地透传他们认为需要用到的设备，而对其他设备无需进行传输，实现了更高的安全性。

在应用生命周期管理方面，我们提供了从应用安装、发布、使用到卸载的全生命周期管理。用户只需在国产系统上即可完整对应用的全部操作。

在软件使用体验上，我们采用了 Material Design 进行整体设计。考虑到用户可能在不同分辨率上使用我们的软件，我们针对高分辨率进行了适配优化。在应用部署上，我们对需要使用的依赖程序包进行了一键初始化，用户无需进行多次应用安装，一次点击即可完成安装。

## 4. 项目实现的技术方案及原理

整个项目的工程化过程分为以下几个模块：

- (1) Windows 端功能设计
- (2) 国产操作系统端功能设计
- (3) 双端通信

### 4.1 Windows 端功能设计

#### 4.1.1 Windows 端后台服务

##### 4.1.1.1 自签名证书部署与初始化

为了充分考虑网络安全，Any2Remote 使用自部署的 HTTPS 为相关通信服务提供支持。

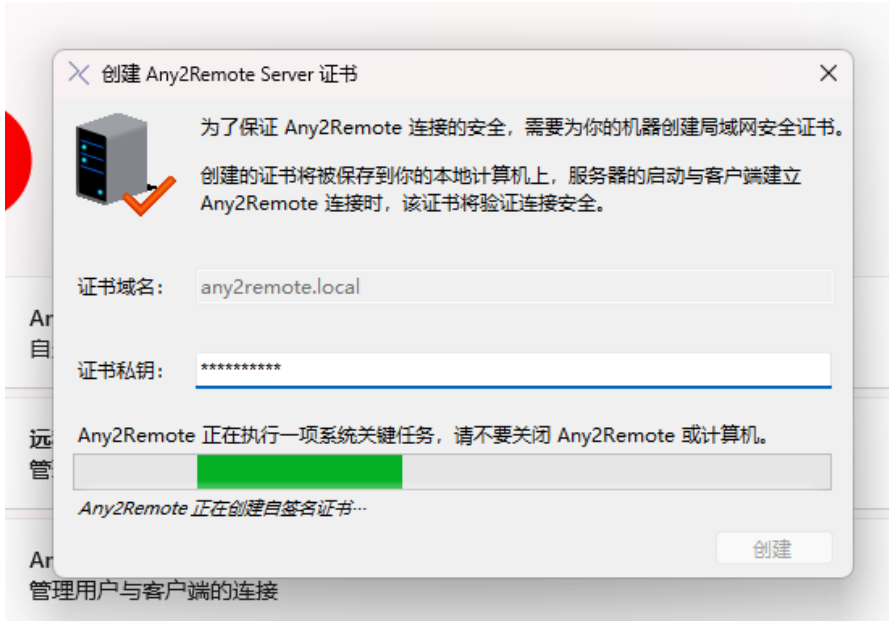


Figure 1 为自签名证书设置私钥，确保连接安全性

结合 Powershell 内置的自签名生成命令和 Windows 自身证书服务 SDK，实现了自签名证书颁布并在 Windows 端部署，在 https 通信过程实现了加密，保障了安全性。

具体而言，Any2Remote 将使用 PowerShell 命令 `New-SelfSignedCertificate` 在本地计算机的证书存储区创建自签名证书。该证书的有效期为 100 年。之后，使用 PowerShell 脚本导出创建的自签名证书（导出路径为 `%APPDATA%\Any2Remote\Certificates`）。导出过程包括生成 `.pfx` 文件和 `.cer` 文件：

`.pfx` 文件包含证书和私钥，受密码保护。

.cer 文件仅包含证书的公钥部分。

导出后，导出的 .cer 文件将被导入到本地计算机的根证书存储区。并在 hosts 文件中添加 DNS 记录（any2remote.local）以便在本地或局域网解析，最后，刷新 DNS 缓存并将证书的相关信息（如 DNS 名称、密码、证书路径）导出到一个 JSON 文件中。

#### 4.1.1.2 实时通信

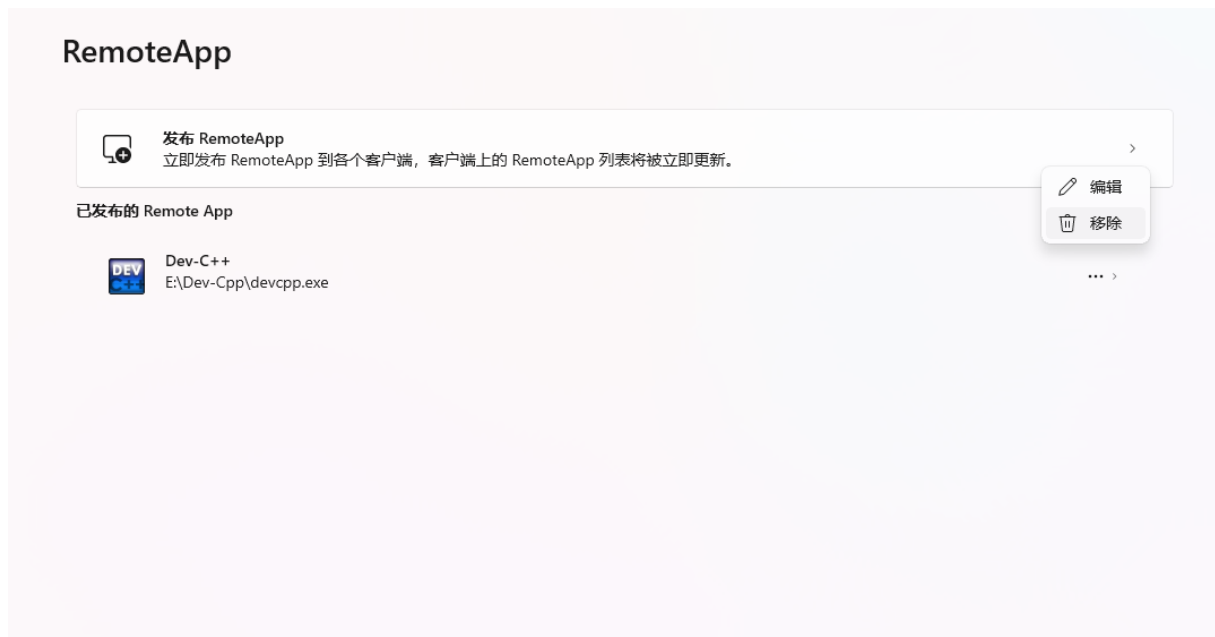


Figure 2 已发布远程应用展示页，可对应用进行编辑或删除

ASP.NET Core SignalR 是一个开放源代码库，可用于简化向应用添加实时 Web 功能。实时 Web 功能使服务器端代码能够将内容推送到客户端。通过 SignalR Core 集成的 WebSocket 与长轮询等功能，管理员可以对国产 Linux 客户端的 Remote App 列表进行实时管理。更改，删除，添加都可以实时在各个已连接到的国产操作系统客户端同步。

具体而言，首先，客户端通过发送 HTTP 请求来与服务器建立连接。SignalR 协议选择机制会确定使用哪种传输协议。连接建立后，客户端和服务器可以通过调用 Hub 方法进行通信。服务器可以调用连接的客户端上的方法，发送消息给客户端。SignalR 会确保消息到达指定的客户端。任何对 Remote App 的写操作都会使得 Any2Remote 后台服务器触发 RefreshRequired 事件，该事件将被传播到所有连接到 Windows 端的客户端强制刷新相关数据。

#### 4.1.1.3 Restful API

Any2Remote 后台含有一系列供国产操作系统调用的 HTTP Restful API 以在局域网范围内为国产操作系统端提供包括 Remote App，上传文件，安装卸载等相关服务。

<b>App</b>		^
GET	/api/app/certificate	▼
<b>File</b>		^
POST	/api/file	▼
POST	/api/file/stream	▼
<b>RemoteApp</b>		^
GET	/api/remoteapps	▼
POST	/api/remoteapps	▼
GET	/api/remoteapps/{appId}/icon	▼
DELETE	/api/remoteapps/{appId}	▼

Figure 3 与国产操作系统客户端的通信接口

Any2Remote 为客户端提供了以下 API:

- HTTP GET /api/remoteapps: 获取所有远程应用程序列表。
- HTTP GET /api/remoteapps/{appId}/icon: 获取指定远程应用程序的图标。
- HTTP POST /api/remoteapps: 添加一个新的远程应用程序。
- HTTP DELETE /api/remoteapps/{appId}: 删除指定的远程应用程序。
- HTTP POST /api/file/uploadfile: 上传一个小型文件到

%APPDATA%\Any2Remote\UploadFiles, 并返回服务器为其创建的临时文件名。

- HTTP GET /api/app/certificate: 下载存储在

%APPDATA%\Any2Remote\Certificates 下的 any2remote.cer 证书文件。

#### 4.1.2 Remote App 管理

Any2Remote 支持 Windows 管理员在本机上发布、修改、移除、自定义 Remote App。



### 4.1.2.1 Remote App 发布与解析原理



Figure 4 应用发布页面

**Any2Remote** 支持拖动、从电脑已安装应用、快捷方式、乃至无安装信息的原始可执行文件中解析出应用程序相关执行与安装信息并完成发布。

#### 4.1.2.1.1 拖动发布



Figure 5 在 Windows 系统中将.exe 或快捷方式拖拽进行发布

**Any2Remote** 会自动识别拖入的文件类型，并委托其它三种方式完成解析。

#### 4.1.2.1.2 通过快捷方式 (.lnk) 发布

通过调用 Windows COM 组件对打开的应用快捷方式进行解析，获取应用名称，图标，路径地址，卸载路径等详细信息。

#### 4.1.2.1.3 通过快捷方式 (.exe) 发布

通过 P/Invoke 调用非托管 Win32 API 对 exe 可执行文件进行解析，获取应用名称，图标，路径地址，卸载路径等详细信息。

#### 4.1.2.1.3 从已发布的应用程序发布

通过读取系统关键注册表内的信息，综合开始菜单中应用程序信息分析应用名称，图标，路径地址，卸载路径等详细信息。

4.1.2.2 Remote App 与系统应用

我们对部分系统应用，例如 Internet Explorer，进行了特殊处理。用户可以在 Windows 端发布 Internet Explorer 并在 Linux 客户端启动 Internet Explorer。

4.1.3 Any2Remote 系统服务与高级功能

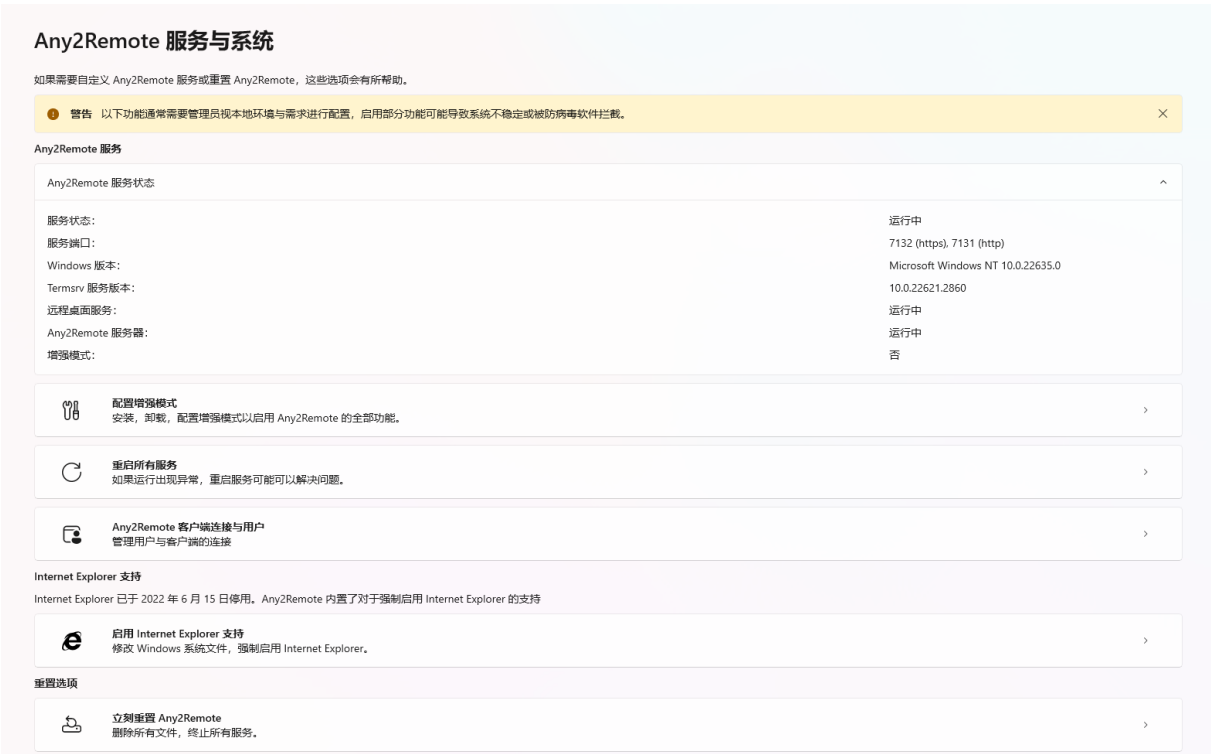


Figure 6 Any2Remote 的高级功能，包括增强模式，重启服务，管理所有远程会话

4.1.3.1 Any2Remote 增强模式

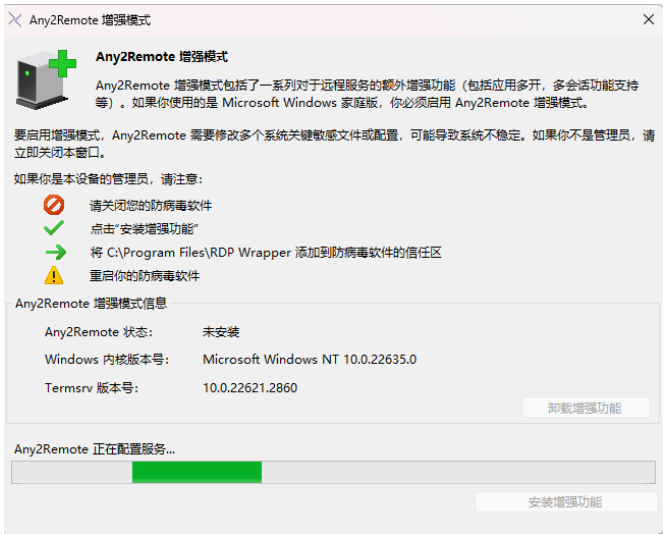


Figure 7 增强模式安装界面

内置的增强模式配置可以通过 DLL 注入、获取系统特权、禁用 WOW64 子系统、篡改防火墙策略等手段修改系统关键文件与服务以启用 Any2Remote 增强模式。Any2Remote 增强模式包括了一系列对于远程服务的额外增强功能（包括应用多开，多会话功能支持等）。如果用户使用的是 Microsoft Windows 家庭版，则必须启用 Any2Remote 增强模式。


安装增强模式时，Any2Remote 将会检查该机器恶的操作系统与 CPU 架构是否符合安装增强模式的要求，随后，Any2Remote 获取 SeDebugPrivilege 特权以终止原生的远程桌面服务（Termsrv）。为了确保我们修改的系统文件是正确的，在我们禁用 WOW64 文件重定向子系统后，我们将支持 Any2Remote 增强模式与远程连接的关键文件写入系统文件夹（system32），并在注册表中将负责远程桌面连接的服务启动信息替换为支持 Any2Remote 增强模式的替代，最后，我们重启远程桌面服务，并恢复 WOW64 文件重定向子系统。

同理，卸载增强模式时，Any2Remote 获取 SeDebugPrivilege 特权以终止远程桌面服务，并在注册表中将负责远程桌面连接的服务启动信息恢复为 Windows 原生信息。最后，我们重启远程桌面服务，并恢复 WOW64 文件重定向子系统。

Windows 端客户端用户界面本身仅支持 Windows 10 以上版本的运行。但是 Any2Remote 增强模式支持版本的范围如下：

操作系统	ProductMajorPart	ProductMinorPart	支持状态	注释
Windows XP	5	1	不支持	Windows XP is not supported.
Windows Server 2003 或 XP 64-bit Edition	5	2	不支持	Windows Server 2003 or XP 64-bit Edition is not supported.
Windows Vista	6	0	不完全支持	Windows Vista is not fully supported.
Windows 7	6	1	不完全支持	Windows 7 is not fully supported.
Windows 8 或更高版本	其他	其他	支持	Success

### 4.1.3.2 重启所有服务



**重启所有服务**  
如果运行出现异常，重启服务可能可以解决问题。

如果运行出现异常，重启服务可能可以解决问题。

Any2Remote 将会重新启动以下服务：

- Any2Remote 后台服务器
- Termsrv 的相关依赖服务
- Termsrv （远程桌面连接服务）

### 4.1.3.3 重置

重置选项



**立刻重置 Any2Remote**  
删除所有文件，终止所有服务。

点击“重置”按钮时，Any2Remote 将会：

- (1) 停止服务器：停止当前运行的服务器。
- (2) 更新服务器注册表键：将服务器的注册表键设置为对应的值。
- (3) 清理项目

缓存清理：删除 %APPDATA%\Any2Remote\ 目录下的所有文件，包括证书文件，客户端上传的文件。

注册表清理：删除 SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\TSAppAllowList\Applications 下所有项，本机发布的 Remote App 将被全部清理。

证书清理：使用 PowerShell 脚本删除 Cert:\LocalMachine\Root 路径下指定 DNS 名称的证书。

增强模式插件清理：如果增强模式插件已安装且正在运行，则停止 termsrv 服务，卸载插件，并重启 termsrv 服务。

4.1.4 客户端连接管理与用户



Any2Remote 可以管理所有已连接到客户端的连接会话，包括查看相关连接信息，中断连接，注销会话等功能。

4.1.4.1 查看相关会话连接信息

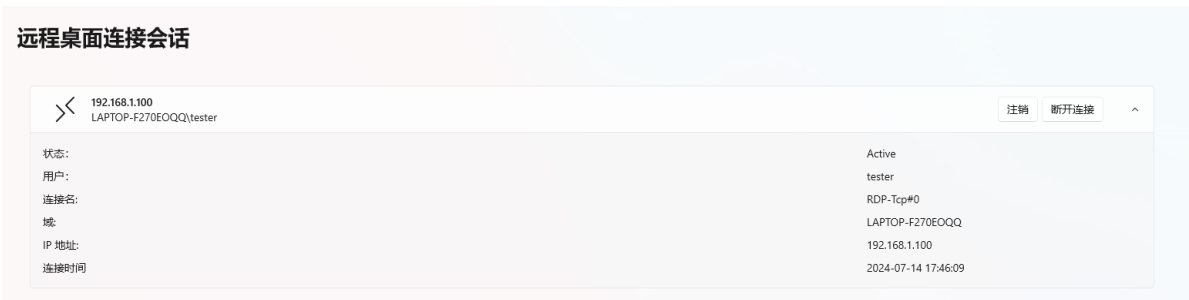


Figure 8 管理远程连接页面

通常而言，一个远程会话含有以下信息：

字段名	类型	含义
SessionId	long	会话的唯一标识符。
WinStationName	string	会话的窗口站名称。
Status	SessionConnectStatus	会话的连接状态，枚举类型，表示会话的当前连接状态。
UserName	string	当前会话的用户名。
Domain	string	当前会话用户的域名。
Address	string	当前会话客户端的 IP 地址。
ConnectTime	DateTime	当前会话的连接时间，表示会话何时开始。

各个状态解释如下：

枚举值	含义
Active	会话处于活动状态。
Connected	会话已连接。
ConnectQuery	会话正在查询连接状态。
Shadow	会话处于影子模式，正在被监视。
Disconnected	会话已断开连接。
Idle	会话处于空闲状态。
Listen	会话正在监听新的连接。
Reset	会话正在重置。
Down	会话已关闭。
Init	会话正在初始化。

#### 4.1.4.2 断开指定连接

断开来自客户端的连接将会使得连接到该会话的客户端上的远程连接与 Remote App 失去画面。断开已登录用户与指定的远程桌面服务会话的连接，而不关闭会话，如果用户随后登录到同一远程桌面会话主机（RD 会话主机）服务器，则用户将重新连接到同一会话并恢复他的工作。

#### 4.1.4.3 注销指定用户会话

注销来自客户端的连接将会使得连接到该会话的客户端上的远程连接与 Remote App 立刻终止。指定的远程桌面服务会话将被立即注销，如果还有正在运行的工作，相关数据将会丢失。

## 4.1.5 UI 设计

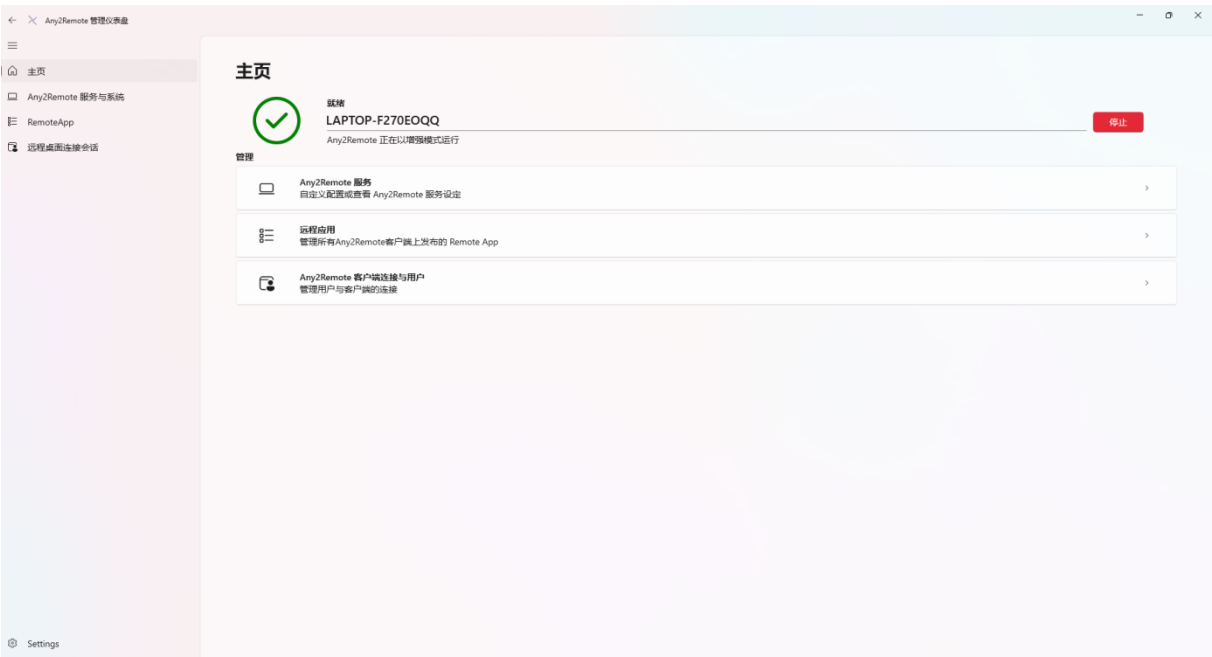


Figure 9 整体 UI 展示，遵循 Windows11 设计语言

在设计上，Windows 端采用 Fluent Design 设计，使用随 Windows 应用 SDK 提供的 WinUI3 从头构建的新式 Windows 桌面应用，打造原生 Windows 11 UI 的使用体验。

### 4.1.5.1 颜色与排版

Any2Remote Windows 端的颜色体系强调使用柔和的色调和自然的渐变，辅以前 Windows 当前主题色，以创造出一种平滑和优雅的视觉体验。Fluent Design 中的“材料”概念（Materials）包括了“光”（Light）、“深度”（Depth）、“运动”（Motion）、“材质”（Material）和“规模”（Scale），这些元素通过颜色的应用得到了具体表现。例如，通过“光”来模拟现实世界中的光影效果，使得界面更具层次感和立体感。

Any2Remote 的排版设计强调简洁、清晰和一致性。它采用了一套标准的字体（如 Segoe UI）和排版规范，这些规范确保了文本在不同设备和屏幕尺寸上的一致性和可读性。同时还注重层次结构，通过不同的字体大小、字重和间距来引导用户的注意力。例如，标题和副标题使用较大的字体和粗体字，内容文本则使用较小的字体和常规字重，这样可以帮助用户快速理解信息的层次和重要性。

### 4.1.5.2 组件与界面布局

Any2Remote 每个组件都遵循 Fluent Design 的原则，这意味着它们在视觉上是统一的，并且在交互上是直观的。例如，按钮组件在用户悬停、点击和禁用状态下会有不同的视觉反馈，这些反馈通过颜色、阴影和动画来表现，使得用户的操作更加自然和流畅。

Any2Remote 支持响应式设计，可以自动适应不同屏幕尺寸和设备类型。通过网格布局（Grid）和堆叠布局（StackPanel）等布局控件，使得布局可以根据视口的变化自动调整。这在创建响应式应用时尤为重要，确保了界面在不同设备上的一致性和用户体验。

## 4.2 国产操作系统端功能设计

### 4.2.1 连接 Windows

#### 4.2.1.1 联通性测试

用户根据提示输入待连接的 Windows 端对应的 IP 地址，账户名称以及账户密码，点击“测试连接”后软件通过我们自行编译的 hfreerdp 工具尝试访问远程桌面协议（RDP）服务器。在验证是否通信的时候我们会跳过对服务器的 RDP 证书验证，减少用户确认次数。由于不需要对 RDP 服务器创建真正的连接通道，我们仅进行身份验证，而不启动完整的远程桌面会话。用于测试凭据是否正确。在验证过程中，我们采用指定使用网络级身份验证（Network Level Authentication, NLA）来进行连接。NLA 是一种增强的身份验证方法，可以在 RDP 会话完全建立之前对用户进行身份验证。若能顺利连接成功，软件会将对应的用户名，IP 和密码进行本地存储，用户每次打开应用的时候均可一键回连至曾经连接过的 IP 设备。

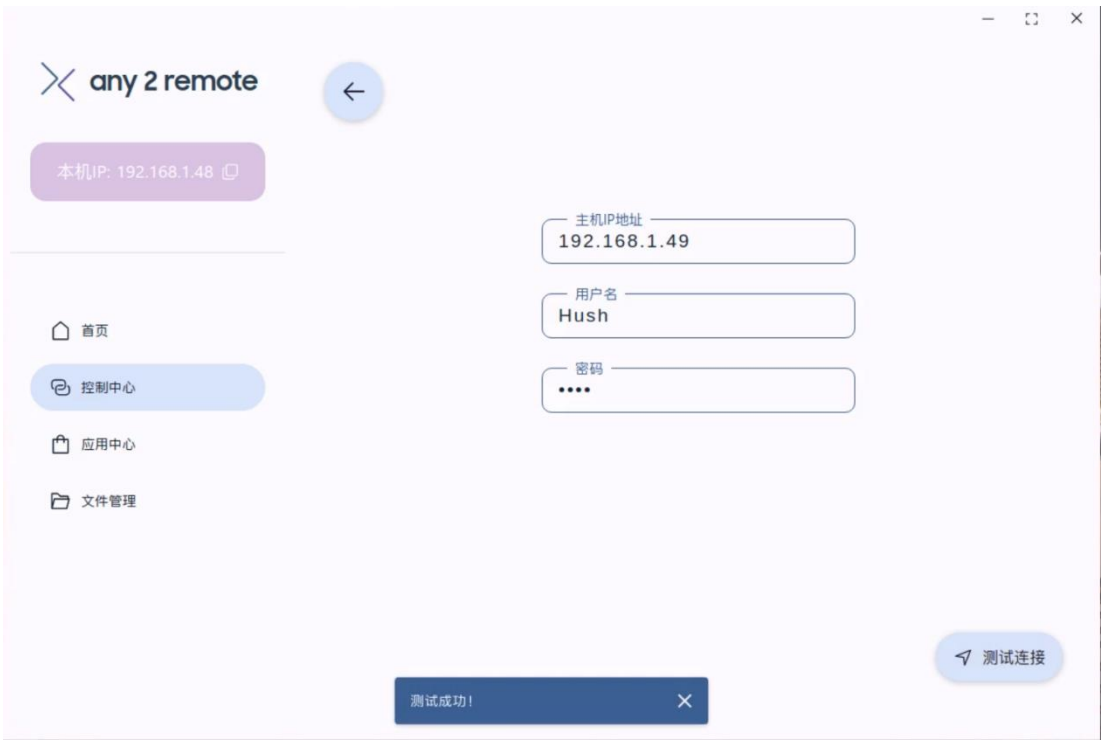


Figure 10 测试 Windows 联通界面

#### 4.2.1.2 连接至远程 Windows 设备

在用户点击连接之后，我们依旧通过 hfreerdp 工具测试连通性，确认联通后系统自动从远程 Windows 上运行的服务端软件下载自签名的 Https 证书文件。在本地得到“any2remote.cer”证书之后，软件通过调用 OpenSSL 系统命令，将文件

“any2remote.cer” 转换为 “any2remote.crt” 格式。随后，软件会提示用户输入系统密码，此时软件通过执行在 “/usr/share/ca-certificates/” 目录下创建 “any2remote/” 文件夹，用于存放我们自己的证书文件。随后复制转换后的证书文件 “any2remote.crt” 到该目录。使用 “update-ca-certificates” 系统命令更新系统的证书存储，使新添加的证书生效。将 IP 地址和主机名对添加到 “/etc/hosts” 文件，确保系统能识别远程主机名。若以上步骤均顺利执行，系统会在当前缓存中存放新连接的用户信息，便于软件进一步与远程主机进行通信，例如查看应用，发布应用等功能。



Figure 11 下载并安装证书

## 4.2.2 管理应用

### 4.2.2.1 查看所有已发布应用

在用户打开“应用中心”界面时，软件在后台会创建一个 SignalR 连接对象，指定使用 WebSocket 作为传输协议，与 windows 端上运行的服务端进行通信，并且启用自动重连功能，以便在连接断开时自动重新连接。通过 WebSocket 连接，软件可以监听服务器发送的刷新通知。在每次收到通知时，软件都会刷新通知时主动请求更新远程应用列表，实现了应用发布的实时通信。软件可以获取所有发布应用的应用名称、执行文件路径、工作目录、应用图标，以及可解析应用能够获取它的卸载路径等高级信息。



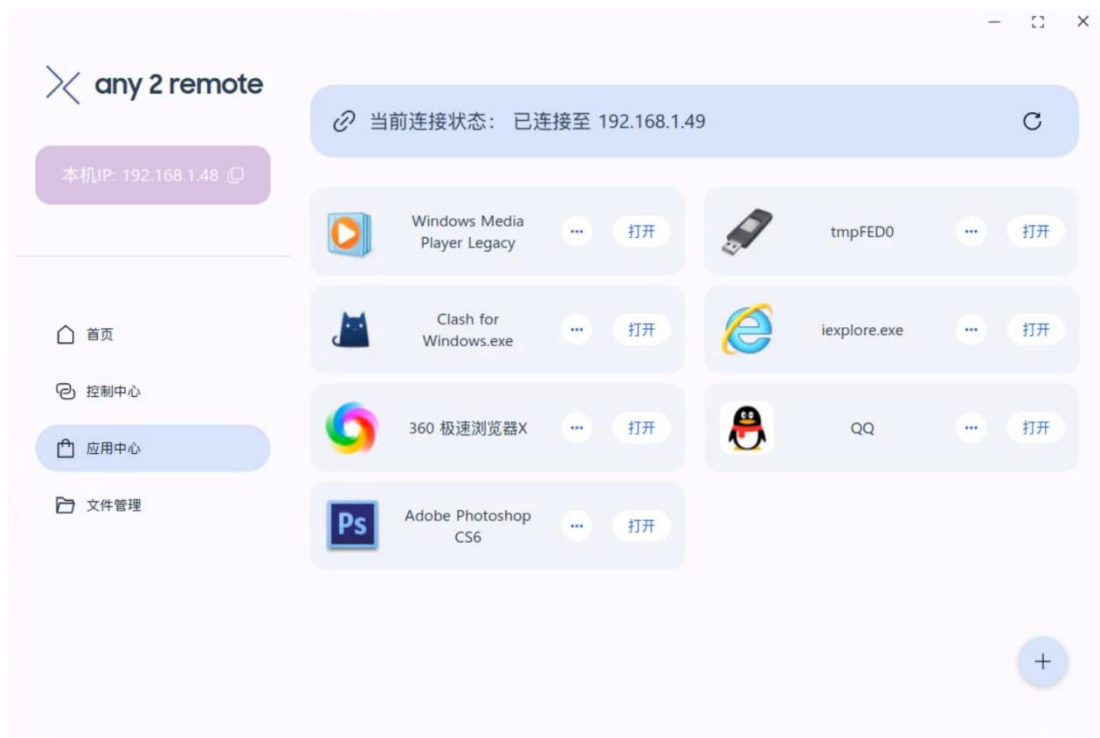


Figure 12 查看所有已发布应用

#### 4.2.2.2 添加、删除发布应用

点击右下角的加号后，系统会将 Windows 端管理页面进行投射，用户可以选择快捷方式或应用程序进行发布。选定后，服务端将对其进行解析，尝试获取卸载地址等高级信息。用户也可再此界面移除不想使用的应用。操作完成后，Windows 服务端会自动向国产系统客户端更新信息推送，无需手动刷新。

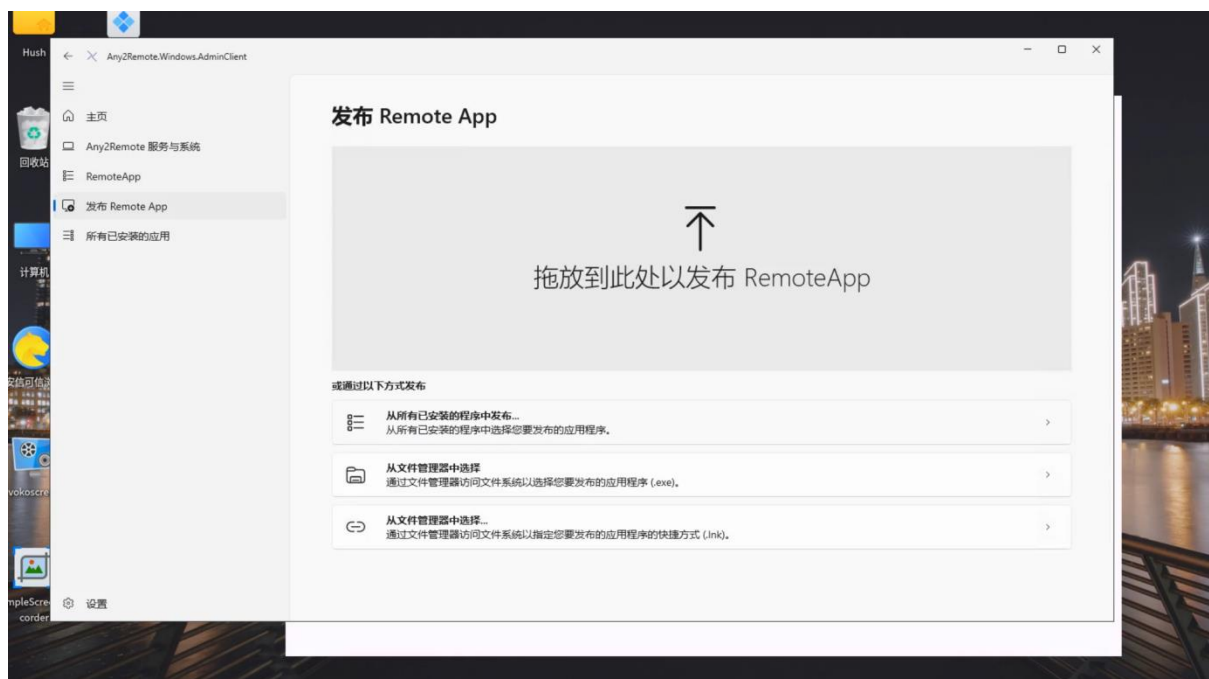


Figure 13 远程发布 Windows 应用

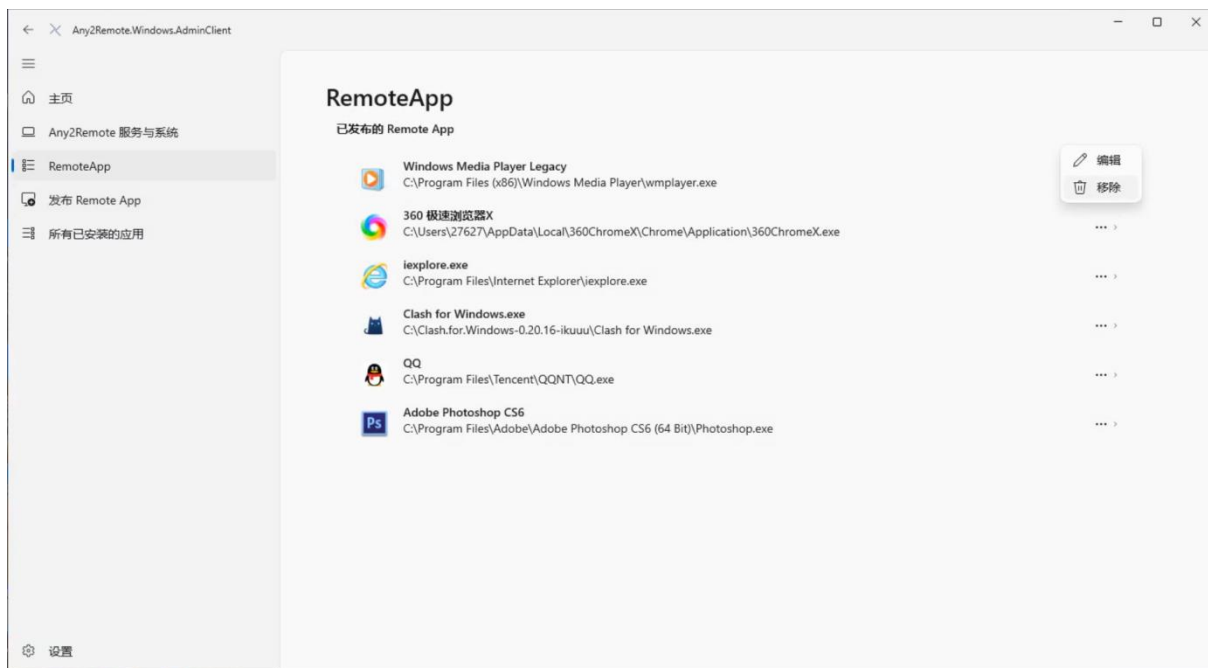
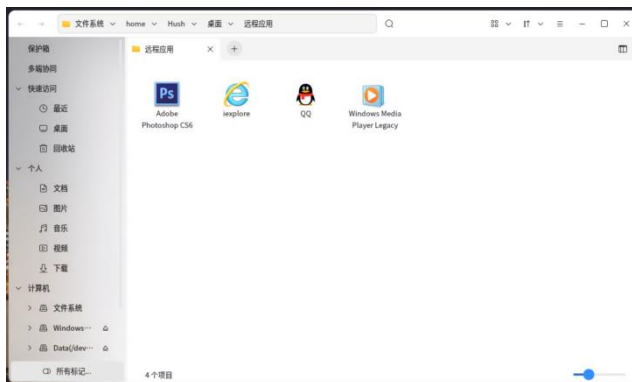


Figure 14 编辑或删除已发布的应用

### 4.2.3 应用发布至桌面

用户选择一个应用并点击将其发布至桌面之后，应用会检测桌面是否已经存在“远程应用”文件夹，若没有则创建一个。随后，程序会创建并打开一个写入流，使得能将 Windows 端的应用图标保存至国产操作系统本地。之后从文件中读取证书，并创建一个自定义的 https.Agent 实例，用于处理安全的 HTTPS 请求。在成功连接至 Windows 服务端之后，使用 HTTPS 协议下载应用的图标，并将其保存到指定的文件路径。图标我们采用 jpg!800 格式，使其可以在国产操作系统上正常展示。为了能够顺利在桌面创建快捷方式文件，软件此时从渲染进程转到主进程，实现与系统深层次的交互。主进程首先解析出上一步下载图标的路径，同样打开一个写入流，写入快捷方式文件内容，包括应用的显示名称、执行命令、图标路径等信息。在最后设置快捷方式文件的权限，使其可执行。通过这些过程，保证了远程应用快捷方式的创建是自动化和用户透明的。



4.2.4 应用远程安装

我们提供了两种方式安装本地应用，用户可选择拖拽 exe 文件上传，或者点击打开文件管理器手动进行浏览选择。在选择文件后，软件首先会确保缓存中有已经连接上的远程 Windows 客户端，否则将不会进行上传。之后，软件会对文件的类型进行一次检查，若不是以 exe 为后缀的文件，软件会弹出一个提示，并且中止后续的操作，确保用户上传的安装文件格式正确。经过验证后创建一个新的 FormData 对象 formData。将文件 file 添加到 formData 中。随后，我们使用 axios.post 发送 HTTP POST 请求，将 formData 上传到指定的 Windows 服务器地址设置请求头 Content-Type 为 multipart/form-data，表明请求体中包含文件数据。随后待 Windows 返回上传结果，若顺利上传，Windows 服务端将返回文件的保存地址，通过调用系统终端命令使用 hfreerdp 通过 RDP 连接到之前连接的 Windows 服务器并运行上传的应用程序。确保了用户选择的文件能够安全地上传到服务器，并在上传成功后自动执行相应的远程操作。

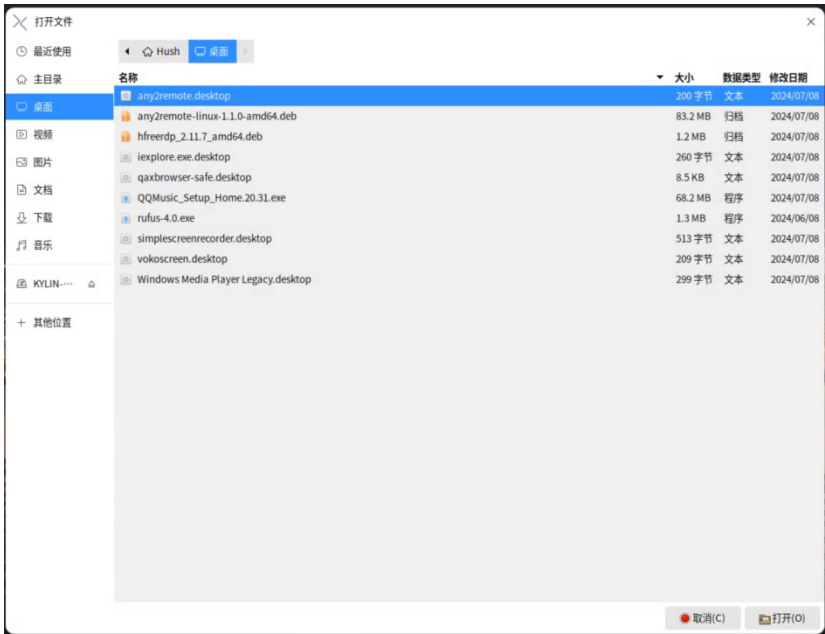


Figure 15 选中要安装的应用文件



4.2.5 应用卸载

若用户在发布应用时选择了可以正确解析卸载路径的应用（例如可以在 Windows

添加或删除应用程序中卸载的应用），国产操作系统软件端会从 Windows 服务端接收到对应应用的卸载路径。由于卸载应用往往需要系统的管理员权限，而这会触发 Windows 用户账户控制（UAC）提示。经过测试，若使用 freerdp3 及以后的版本则无法顺利传输 UAC 权限确认界面。因此我们采用重新编译后的 freerdp2 作为底层的通信协议。由此我们实现了传输 UAC 界面以及应用卸载功能。点击卸载后则会弹出对应应用的卸载界面，完成卸载后对应的应用发布界面也会通过 SignalR 服务器自动进行更新，移除已经卸载的应用，无需用户手动进行移除。

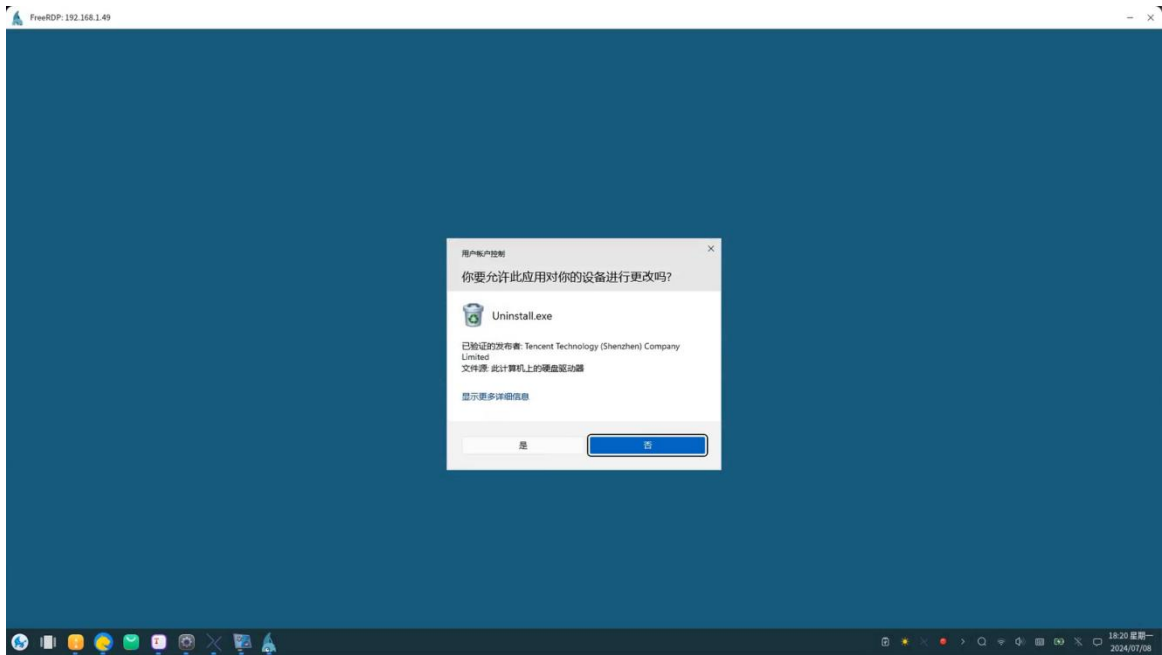


Figure 16 弹出管理员权限申请窗口

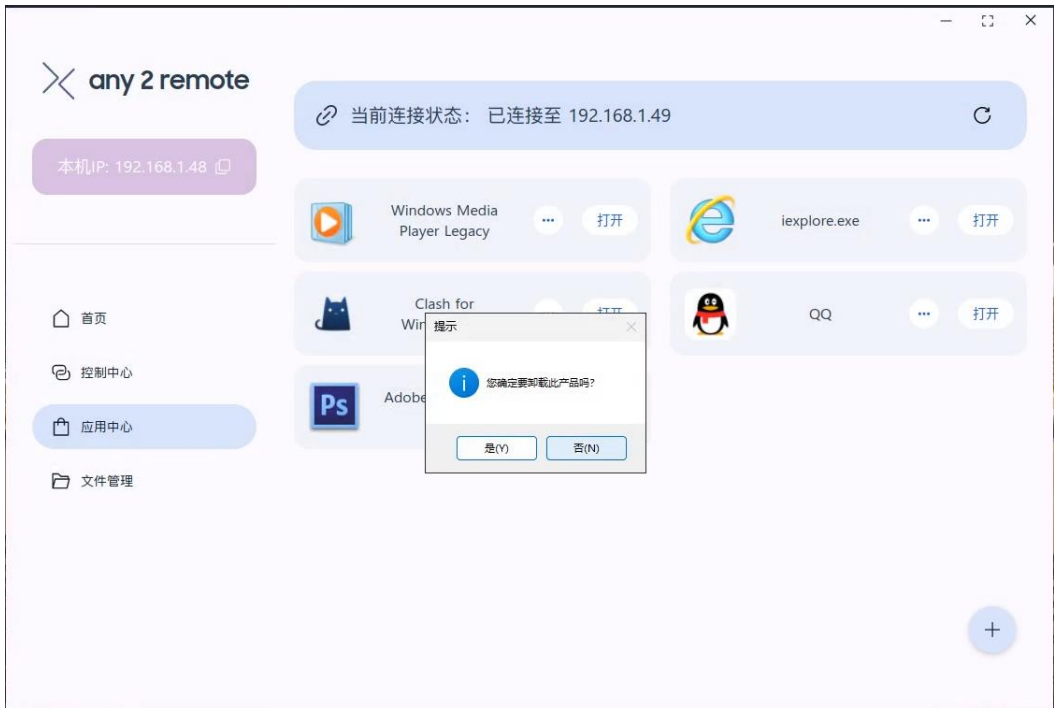


Figure 17 卸载页面

## 4.2.6 外接介质透传

在打开设备管理界面时，软件会使用 Node.js 的 `child_process` 模块中的 `exec` 方法执行 `lsblk` 命令，并且以 JSON 格式获取系统的块设备信息。从获取的所有块设备信息中过滤出所有表示为 SCSI 的磁盘设备。打印过滤后的设备列表 `sdbDevices`。由于部分外接磁盘可能有多个分区，遍历每个 `sdbDevices` 设备。如果设备有子设备（即分区），则进一步遍历每个分区。如果分区有挂载点 `mountpoint`，将其添加到设备列表中。上述操作即可查询出当前国产操作系统中所接入的所有外接设备。随后，软件从缓存中获取之前连接的所有 USB 选项，并与当前设备挂载点进行对比。更新设备状态，确保系统显示当前连接的所有 USB 存储设备。在使用融合应用时，设备会随之映射至 Windows 端上，用户产生的文件（如下载文件，WPS 产生的 word 进行保存等）都可以直接保存到国产操作系统端。此外，用户可以在接入的多个 USB 存储设备中选择性地透传选中的设备，而将敏感设备保存在本地不进行网络传输，进一步实现了系统安全性。

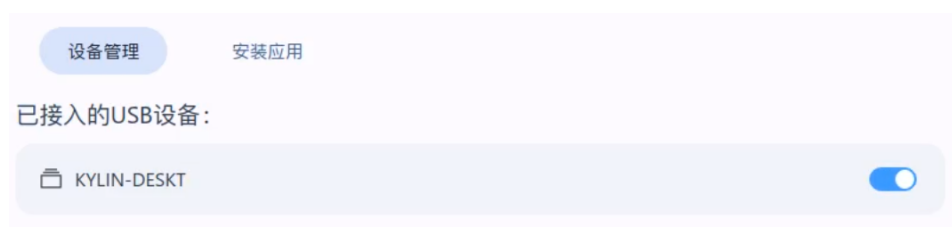


Figure 18 选择外接 USB 设备

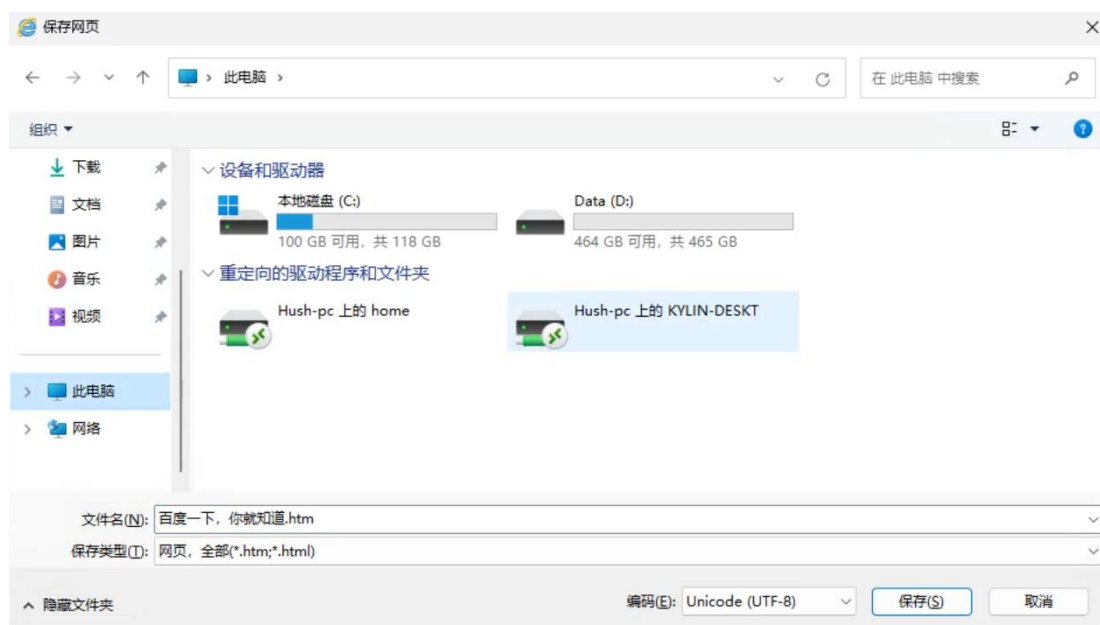


Figure 19 将文件直接保存至本地存储设备中

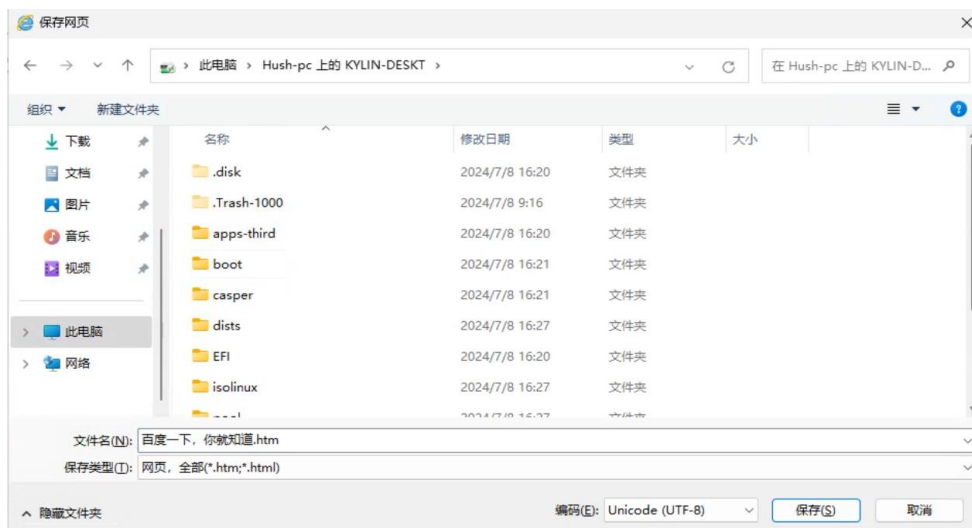


Figure 20 浏览本地存储设备

## 4.2.7 UI 设计

### 4.2.7.1 颜色与排版

国产操作系统客户端整体采用 **Material Design** 设计语言进行设计。采用了以紫色为主的色彩方案，以浅蓝色作为强调色，创造出现代感和深度。界面元素巧妙地运用了卡片式设计，配合微妙的阴影效果，营造出清晰的信息层次。简洁的线性 **Material** 图标和清晰明了的文字排版进一步强化了设计的清晰度和可读性。字体选择上整体采用了苹方字体，进一步提高可读性与设计感。整体布局充分利用了空白空间，避免了视觉上的拥挤感。颜色调配上，我们采用低饱和色系进行设计。正文字体采用了深蓝色，使用 浅蓝色 作为背景提示色，提示色为 深灰色，错误强调色则为 洋红色。在下面我们展示了所使颜色的配色版，整体配色遵循了 **Material Design** 的设计语言。



Figure 21 整体界面展示，遵循 Material Design 3 设计语言



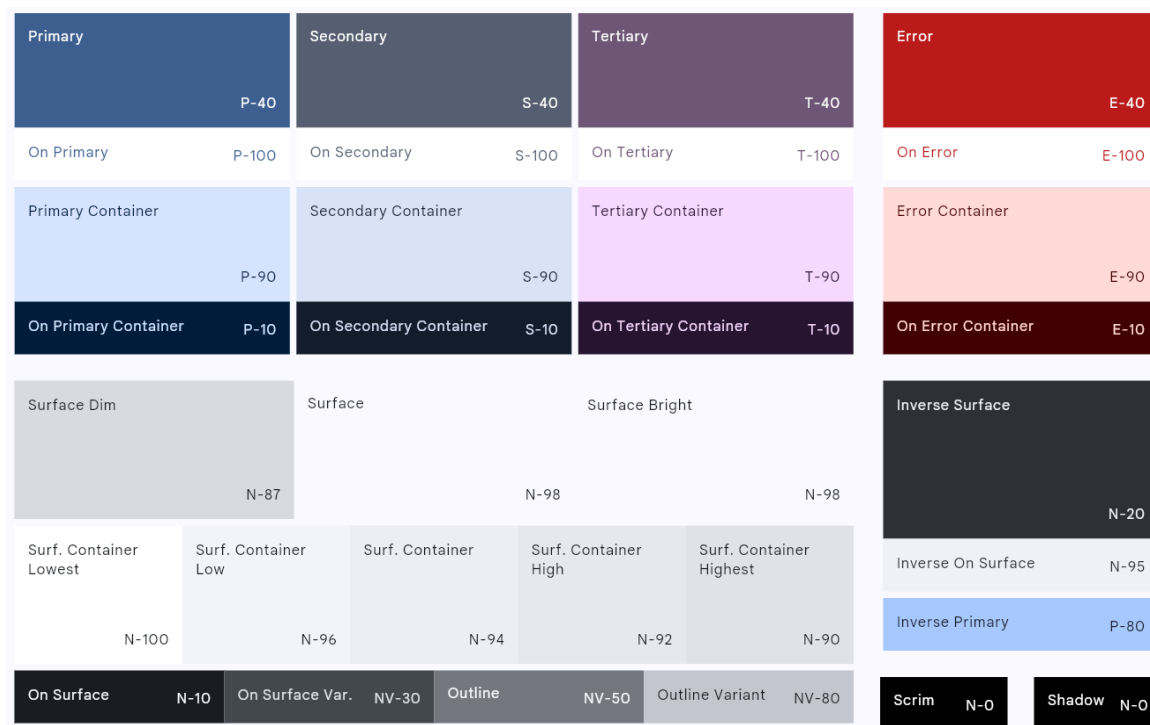


Figure 22 整体配色方案，更现代，更简洁

#### 4.2.7.2 组件与界面布局

界面布局遵循 **Material Design** 的网格系统，利用卡片组件和适当的留白来组织信息，增强了整体的清晰度和可读性。左侧的导航菜单采用了 **Material Design** 常用的侧边导航模式，图标和文字组合清晰明了，突出了当前所在的页面。应用列表采用了卡片式布局。界面中的主要按钮均采用了 **Material Design** 浮动操作按钮，通过精心设计的阴影效果，带来纵深悬浮感。点击按钮之后具有涟漪波纹，带来具有设计感的动画效果。

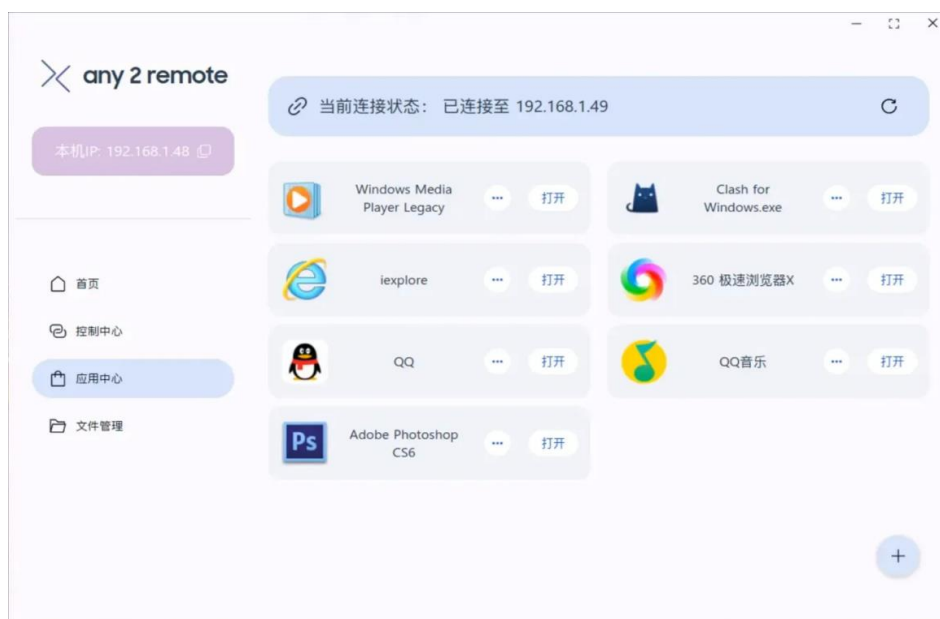


Figure 23 网格化布局，清晰展示所有内容

### 4.2.7.3 设计工具与资源

在设计过程中，我们使用了以下工具和资源来实现 Material Design：

设计工具：使用 Material Builder 和 Adobe XD 等设计工具，结合 Material Design 的设计套件（Material Design Kit）进行界面设计。

开发资源：在开发过程中，我们使用了 Material Design 提供的前端框架和库，例如 MDUI、Element Plus（Vue），确保设计的一致性和实现的高效性。

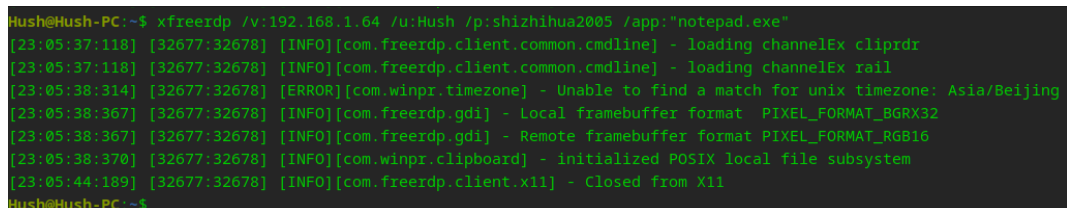
## 4.3 双端通信设计

### 4.3.1 FreeRDP

FreeRDP 是一个开源的远程桌面协议（RDP）客户端，允许用户通过网络远程访问和控制计算机。它实现了微软的 RDP 协议，使用户能够连接到运行 Windows 操作系统的远程服务器。工作原理上，FreeRDP 首先通过指定的端口与 RDP 服务器建立 TCP 连接，然后进行 TLS/SSL 加密，确保数据传输的安全性。连接建立后，服务器会要求用户提供凭据进行身份验证，通过验证后，客户端与服务端之间的会话开始。

在会话过程中，服务器会将图形数据传输到客户端。服务器发送图形命令和位图数据，客户端解析这些命令并在本地绘制图形。为了优化带宽使用，我们将图像的压缩开启至低，并且开启了位图缓存功能。用户在客户端上的键盘和鼠标输入会被捕捉并传输到服务器，服务器则根据这些输入更新图形数据并发送回客户端。此外，我们开启了 FreeRDP 的剪贴板共享、设备重定向和多媒体重定向功能，实现了声音的远程传输，允许用户在远程会话中使用本地设备和资源。为了提高性能，我们使用了 NSCodec 编码器。NSCodec 是一种高效的图像压缩技术，它被设计用于优化网络传输中的图像数据。通过使用先进的压缩算法，能够在保持图像质量的同时，显著减少所需的传输数据量，从而加快图像的加载速度并降低传输成本。此外，我们采用了渐进式渲染，提高了传输画面的响应速度，为用户带来了更加平滑的视觉体验。

由于国产操作系统整体位于软件发行下游版本，且默认安装源中提供提供的 FreeRDP 版本老旧，不支持打开远程应用。因此我们对 FreeRDP 的特定版本进行了手动编译以及移植。



```
Hush@Hush-PC:~$ xfreerdp /v:192.168.1.64 /u:Hush /p:shizhihua2005 /app:"notepad.exe"
[23:05:37:118] [32677:32678] [INFO][com.freerdp.client.common.cmdline] - loading channelEx clipdr
[23:05:37:118] [32677:32678] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rail
[23:05:38:314] [32677:32678] [ERROR][com.winpr.timezone] - Unable to find a match for unix timezone: Asia/Beijing
[23:05:38:367] [32677:32678] [INFO][com.freerdp.gdi] - Local framebuffer format PIXEL_FORMAT_BGRX32
[23:05:38:367] [32677:32678] [INFO][com.freerdp.gdi] - Remote framebuffer format PIXEL_FORMAT_RGB16
[23:05:38:370] [32677:32678] [INFO][com.winpr.clipboard] - initialized POSIX local file subsystem
[23:05:44:189] [32677:32678] [INFO][com.freerdp.client.x11] - Closed from X11
Hush@Hush-PC:~$
```

Figure 24 系统安装源提供的 FreeRDP 不支持远程应用打开

在我们的测试中，我们发现 2.11.7 较为稳定，且支持绝大部分 Any2Remote 需要的功能。我们在手动编译时，同时还移除了不必要的功能，将非必需依赖作为静态链接以提高 FreeRDP 的效率。我们接下来的所有说明都基于 [Release 2.11.7 · FreeRDP/FreeRDP \(github.com\)](#) 上源码并假设解压为 FreeRDP 目录。注意：



## Any2Remote 国产操作系统客户端已附带打包完毕的 FreeRDP 2.11.7

### FreeRDP 编译

考虑到部分国产操作系统的 GCC 以及 libc 等基础设施的版本不符合编译环境，要编译 FreeRDP 执行：

```
cmake -GNinja \  
    -B freerdp-build \  
    -S freerdp \  
    -DBUILD_SHARED_LIBS=OFF \  
    -DCMAKE_BUILD_TYPE=Release \  
    -DCMAKE_SKIP_INSTALL_ALL_DEPENDENCY=ON \  
    -DWITH_SERVER=OFF \  
    -DWITH_SAMPLE=OFF \  
    -DWITH_PLATFORM_SERVER=OFF \  
    -DUSE_UNWIND=OFF \  
    -DWITH_SWSCALE=OFF \  
    -DWITH_FFMPEG=OFF \  
    -DWITH_WEBVIEW=OFF \  
    -DWITH_CLIENT_SDL=OFF -DWITH_PROXY_MODULES=OFF \  
cmake --build freerdp-build
```

### FreeRDP 打包

我们将编译好的可执行文件 “./freerdp-build/bin/client/X11/xfreerdp” 根据 [dpkg\(1\)-Linux manual page \(man7.org\)](#) 所述的方式放入打包目录，

```
|—DEBIAN  
|   control  
|  
|—usr  
|   bin  
|   hfreerdp-x11
```

其中 control 为 APT 系发行版的描述依赖关系文件，内容如下：

```
Package: hfreerdp
Version: 2.11.7
Section: devel

Depends: ocl-icd-opencl-dev, libmp3lame0, libopus0, libsoxr-lsr0, libsoxr0,
libpam0g, libssl1.1, openssl, libxkbfile1, libx11-6, libxrandr2, libxi6, libxrender1,
libxext6, libxinerama1, libxfixes3, libxcursor1, libxv1, libxdamage1, libxtst6,
libcups2, libpcsclite1, libasound2, libasound2-data, libasound2-plugins, libpulse-dev,
libgsm1, libuuid1, libxml2, libxml2-utils, libfaad2, libfaac0, libSDL2-2.0-0, libSDL2-
ttf-2.0-0, libpkcs11-helper1, liburiparser1, libkrb5-dev, libsystemd0, libfuse3-3,
libswscale5, libcairo2, libavutil56, libavcodec58, libswresample3, libwebkit2gtk-4.0-
37, libpkcs11-helper1

Priority: optional

Architecture: amd64

Maintainer: HimuQAQ <himu.official@gmail.com>

Homepage: https://www.freerdp.com/

Installed-Size: 4069

Provides: hfreerdp

Conflicts: hfreerdp, freerdp, freerdp2-x11, freerdp2-wayland

Replaces: hfreerdp

Description: HFreeRDP is an XFreeRDP-X11 Nightly client pre compiled by
HimuQAQ, suitable for Debain 10 based systems
```

之后，我们使用 `dkpg -u` 将其打包，用户便可以直接使用 `apt` 安装软件包且自动处理依赖关系了。

4.4 项目结构图

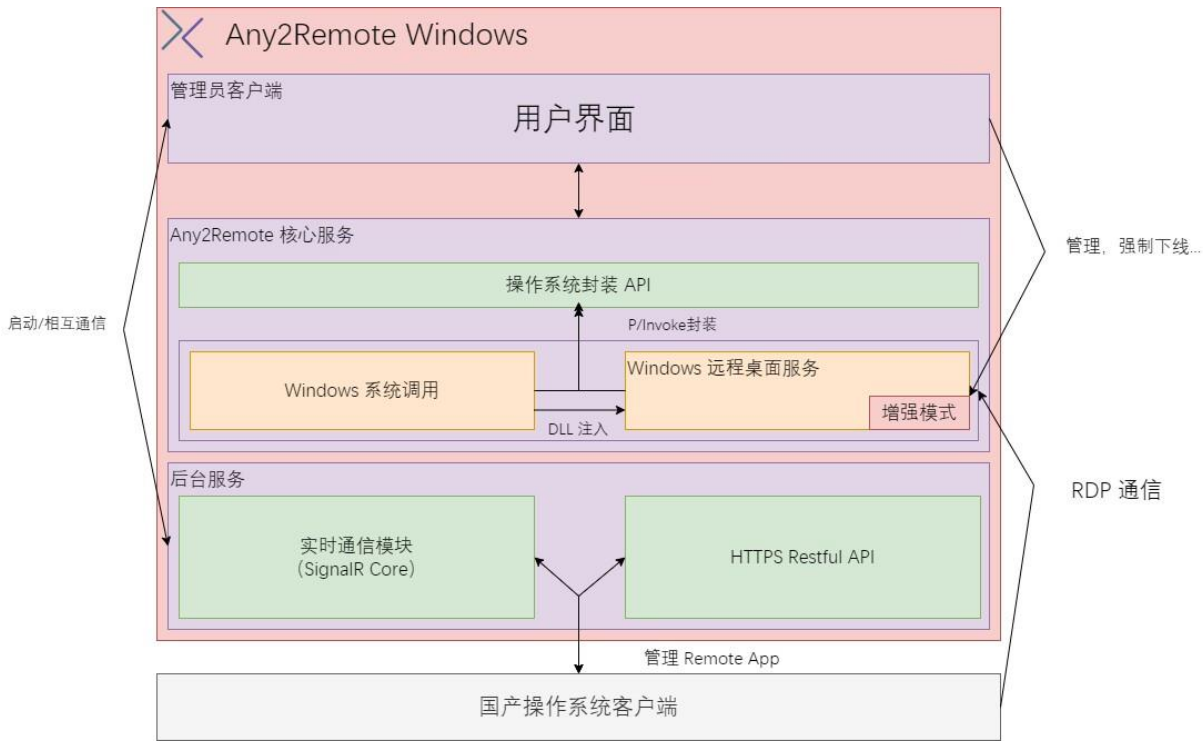


Figure 25 Windows 服务端系统架构图

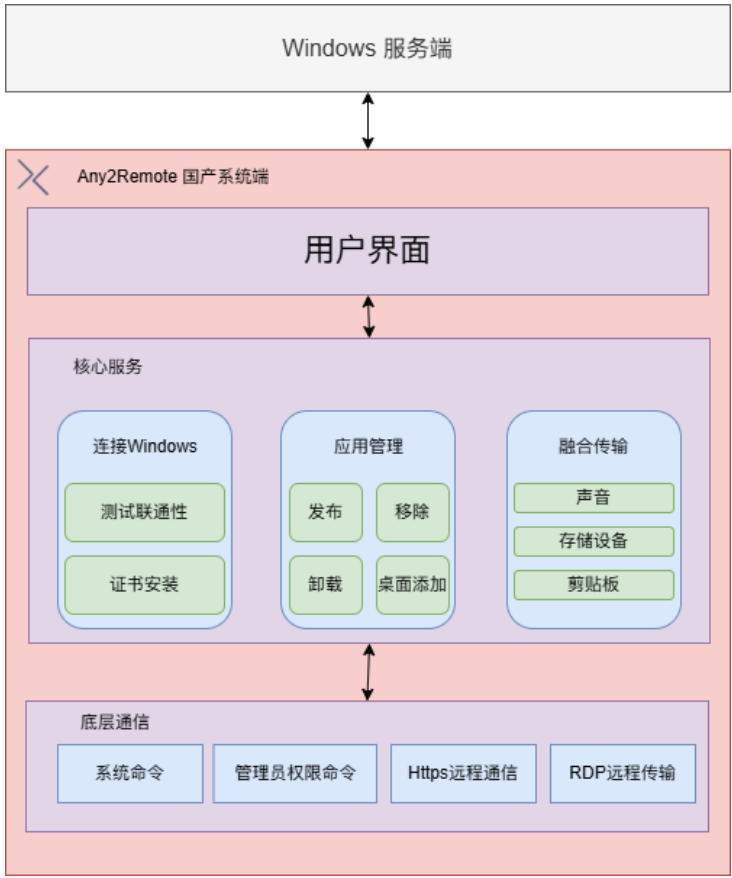


Figure 26 国产操作系统客户端系统架构图

## 5. 运行环境

### 5.1 开发环境

我们在 Kylin 桌面版 V10 SP1 2403 版本上进行开发，Windows 版本采用了 Windows 11 23H2 beta (build 22625.3930)。在开发过程中，Windows 服务端使用 Visual Studio 2022, .NET 6 SDK. 进行开发。国产操作系统客户端使用了 JetBrains WebStorm 2023.1, Electron 21.4.4, Vue 3.2.33 进行开发。

### 5.2 项目工程可运行环境

经测试，服务端兼容从 Windows10 到 Windows11 的多种操作环境。国产操作系统端在统信 UOS V20 1060 以及麒麟桌面版 V10 SP1 2403 进行测试，主要功能均可顺利使用。硬件方面，在英特尔酷睿™ i5 6400 桌面版与 i5 12500H 移动版上顺利运行 Windows 程序。在 AMD 锐龙™ 5800H 移动版上顺利运行国产操作系统程序。我们在 32GB, 16GB, 8GB 的系统内存上都取得了流畅的运行结果。建议至少具有 4GB 运行内存。

## 6. 结语

近二十多年来，主流 IT 底层标准、架构、生态等大多都由国际 IT 大厂制定，随着近几年国内软硬件 IT 厂商蓬勃发展，正在逐步形成自主可控的开放生态，全球 IT 生态格局将由过去“一极”向“两极”演变，两种生态的应用程序可以在一个桌面系统中融合使用的需求将越来越普遍。我们开发的 Any2Remote 软件不但适应国产化环境的大趋势，也在一定程度上解决了部分业务软件“水土不服”的问题。