Vinculación Universidad-Industria: Desarrollo de sistema de administración remota por la UNS, SIA Interactive y Fundación Sadosky
Juan Bajo, Gonzalo Silveira, Bruno Pazos, Teo Vogel, and Claudio Delrieux
SADIO Electronic Journal of Informatics and Operations Research (EJS) Vol. 24 No. 2 (2025) e-ISSN 1514-6774
https://doi.org/10.24215/15146774e078 | https://revistas.unlp.edu.ar/ejs
Sociedad Argentina de Informática e Investigación Operativa | Universidad Nacional de La Plata | Buenos Aires | Argentina

Vinculación Universidad-Industria: Desarrollo de sistema de administración remota por la UNS, SIA Interactive y Fundación Sadosky

Juan Bajo[1], Gonzalo Silveira[2], Bruno Pazos[3], Teo Vogel[4], and Claudio Delrieux[1]

[1] DIEC-ICIC Universidad Nacional del Sur, Bahía Blanca, Argentina
{juan.bajo,cad}@uns.edu.ar
[2] SIA Interactive, Buenos Aires, Argentina gonzalos@siainteractive.com
[3] Departamento de Informática Trelew, Universidad Nacional de la Patagonia San Juan Bosco, Trelew, Argentina
pazosbruno@gmail.com
[4] DCIC, Universidad Nacional del Sur, Bahía Blanca, Argentina
teovogel97@gmail.com

Abstract. Los sistemas de escritorio remoto permiten a los usuarios acceder y controlar un sistema informático desde otro dispositivo a distancia, generalmente a través de Internet. Facilitan tareas como la administración de sistemas, el acceso a archivos y la colaboración, siendo útiles tanto para entornos empresariales como para la asistencia técnica y la capacitación a distancia. Si bien existen protocolos de red abiertos para este fin, hasta el momento no existen soluciones completas de escritorio remoto gratuitas y de código abierto. En este trabajo se presenta un proyecto en desarrollo de un sistema de administración y control remoto, diseñado para satisfacer las necesidades de empresas que requieran esta herramienta, para soporte técnico o control de sus dispositivos. El proyecto es llevado a cabo por integrantes de la Universidad Nacional del Sur en colaboración con la empresa SIA Interactive, y cuenta con financiamiento parcial de la Fundación Sadosky.

Keywords: Escritorio Remoto · Vinculación Tecnológica · Software Libre

Abstract. Remote desktop systems allow users to access and control a computer system from another device remotely, usually via the Internet. These systems facilitate tasks such as system administration, file transfer, and collaboration, being useful for both business environments and technical support or remote training. Although there are open protocols freely available, there are currently no complete free and open-source remote desktop solutions. This paper presents an ongoing project for a remote management and control system, designed to meet the needs of companies that require this tool for technical support or device control. The project is being carried out by members of the National University of the South in collaboration with SIA Interactive, and is partially funded by the Sadosky Foundation.

Received May 2025; Accepted June 2025; Published July 2025

Keywords: Remote Desktop · Tecnology Transfer · Open Source

1 Introducción

Los sistemas de escritorio remoto permiten a los usuarios acceder y controlar una computadora desde otra ubicación a través de una conexión de red, ya sea internet o local. Estos sistemas facilitan la colaboración, el soporte técnico, la educación [5] y el acceso a recursos computacionales desde cualquier lugar con conexión a la red [6]. El funcionamiento de este tipo de sistemas consiste en la instalación de un software cliente en el dispositivo desde el cual se quiere acceder y un software de servidor en la computadora remota. Actualmente existen protocolos de red diseñados para resolver esta problemática como Remote Desktop Protocol (RDP) de Microsoft y Virtual Network Computing (VNC) aunque su utilización requiere de conocimientos avanzados sobre conceptos de redes y sistemas operativos. A partir de estas limitaciones para usuarios no técnicos, en los últimos años se implementaron diversos sistemas comerciales populares que simplifican la operatoria como TeamViewer [4] y AnyDesk [1]. Estas plataformas comerciales, además del control remoto, ofrecen funciones como transferencia de archivos, impresión remota, acceso a dispositivos locales y seguridad mediante cifrado de datos y autenticación de usuarios. En la actualidad estos sistemas se utilizan en una gran variedad de contextos, desde entornos empresariales hasta uso personal para acceder a computadoras domésticas de forma remota [9].

Las soluciones comerciales existentes representan, en general, un costo extremadamente alto para empresas medianas y chicas que requieren proveer soporte a clientes con múltiples dispositivos. Debido a esto, es claro que se necesitan soluciones de bajo costo y extensibles para dar soporte al acceso remoto para pequeñas y medianas empresas. Además, la posibilidad de integración con sistemas propios es nula, dificultando su operación y duplicando recursos. Las ventajas de poseer software de acceso remoto son transversales a cualquier tipo de organización, entre otras podemos identificar la optimización de recursos humanos, la minimización de tiempos de espera y fácil acceso a terminales dedicadas, por ejemplo en cartelería digital.

En el presente trabajo se describe el progreso del desarrollo de un sistema remoto de administración y control de sistemas basado en la tecnología presente en el software de código abierto RustDesk [3]. Este trabajo fue desarrollado por la Universidad Nacional del Sur junto con la empresa SIA Interactive, financiado parcialmente por la Fundación Sadosky.

2 Organización del Equipo

El trabajo fue llevado a cabo por un equipo mixto compuesto por desarrolladores de la Universidad Nacional del Sur y de la empresa SIA. El seguimiento general del proyecto junto con la coordinación de las diferentes etapas fue realizado por la fundación Sadosky. Desde la universidad se aportó un grupo de 3 desarrolladores calificados formado por un especialista en la plataforma Flutter Android,

y dos desarrolladores web Python avanzados. Desde la empresa SIA Interactive, se asignaron desarrolladores front-end, diseñadores gráficos y de experiencia de usuario y *testers*. La participación de SIA Interactive en el proyecto fue particularmente útil debido a su experiencia en el desarrollo y desplegado de soluciones interactivas a gran escala. Además de los desarrolladores, ambos equipos poseen *project managers* internos que organizaban las tareas. La comunicación fue fluida a través de plataformas de comunicaciones para trabajo y se programaron reuniones de avance semanales. Ambos grupos colaboraron estrechamente para asegurar el éxito y la calidad del trabajo, combinando sus habilidades y experiencias para alcanzar los objetivos establecidos.

El trabajo se dividió en 4 etapas de 3 meses de duración cada una alineadas con la entrega de informes del equipo de trabajo hacia la Fundación Sadosky. En la primera etapa se investigaron las diferentes alternativas tanto comerciales como de software libre de aplicaciones de escritorio remoto. La recopilación de datos generada a partir de esa investigación permitió conocer el estado del arte respecto a esas soluciones y detectar posibles soluciones abiertas para ser utilizadas en el proyecto a implementar. Dentro de las alternativas analizadas, además de RustDesk, se estudiaron: TeamViewer Remote, AnyDesk, RealVNC, GoTo Resolve, EV Reach, Apache Guacamole y MeshCentral. De todas las alternativas estudiadas, se eligió RustDesk como proyecto origen por presentar características adecuadas para el proyecto, por el soporte a diferentes codecs de video y protocolos de transmisión, facilidades de instalación para clientes y administradores, propiedad de los datos, seguridad, arquitectura de software, etc. Por otra parte, RustDesk es el proyecto que más plataformas soporta, teniendo alternativas para Android, iOS, macOS, Linux y Windows junto a un cliente web. Durante esta etapa, también se hizo una prueba de la plataforma desplegada en servidores propios para ver limitaciones de ancho de banda, cantidad de dispositivos conectados en simultáneo y funcionamiento con los dispositivos específicos de la empresa SIA. Teniendo elegido el proyecto a partir del cuál se iba a implementar la solución, se procedió a diseñar un modelo de datos para el almacenamiento de los dispositivos, clientes (*tenants*) y usuarios.

En la segunda etapa se comenzó con la implementación del sistema backend a partir del diseño y las especificaciones planteadas en la etapa anterior y se comenzó con el diseño de las interfaces de usuario del sistema.

Durante la tercera y cuarta etapa se trabajó sobre la adaptación de la aplicación móvil Android para dar soporte al nuevo producto junto con la implementación del frontend de la plataforma previo diseño de la experiencia de usuario.

3 Arquitectura de la solución

La solución implementada utiliza parte del desarrollo del software de código abierto RustDesk (ver Figura 3).

3.1 RustDesk

RustDesk es una solución de software libre, desarrollado en lenguaje de programación Rust junto con el framework para desarrollo multi-plataforma Flutter, para acceso y control remoto que permite operar con dispositivos de manera remota. El cliente está disponible en diversos sistemas operativos como Microsoft Windows, Apple macOS, Apple iOS, Android y Linux, además de un cliente web. El diseño de RustDesk permite su utilización sin la necesidad de contar con herramientas adicionales como VPNs o direccionamiento de puertos. Funciona, además, a través de firewalls o NATs. Estas características permiten gran facilidad de uso por parte de usuarios finales sin conocimientos técnicos avanzados, como así también su adopción en entornos seguros. Entre sus características distintivas se incluyen: acceso remoto multiplataforma, encriptación end-to-end, posibilidad de desplegar servidor propio, transferencia de archivos, y TCP tunneling. Por el momento, la solución libre ofrece el código para los distintos sistemas operativos junto con el código fuente de los servidores involucrados para desplegar un servidor propio en la nube. Es importante aclarar que la solución brindada por RustDesk no incluye la administración de dispositivos, lo cual es de vital importancia en empresas que tengan acceso a gran cantidad de ellos. El objetivo del desarrollo del software es agregar la capacidad de la gestión automática de los dispositivos remotos junto con un sistema de control de acceso flexible para soporte técnico. Si bien la solución fue desarrollada junto con SIA Interactive para ser utilizada en sus sistema de cartelería digital, el proyecto se mantiene libre para su adopción por parte de otros actores.

3.2 Protocolo Rendezvous

El protocolo Rendezvous en redes es un método utilizado para permitir que dispositivos o nodos en una red se comuniquen entre sí sin necesidad de tener información previa sobre las direcciones IP o ubicaciones de los otros nodos. Este protocolo se utiliza comúnmente en sistemas de comunicación peer-to-peer (P2P) y en redes de sensores [8]. En RustDesk, la implementación de este protocolo consiste en los siguientes pasos (ver Figura 2 y 6).

- Registro inicial: Cuando se lanza la aplicación RustDesk en algún dispositivo conectado a la red, se registra en un servidor Rendezvous centralizado (signaling server). Durante este registro, el nodo proporciona información sobre sus capacidades y servicios disponibles, mientras que el servidor responde con las credenciales para su acceso.
- Solicitud de conexión: Cuando un nodo necesita comunicarse con otro nodo en la red, envía una solicitud al servidor Rendezvous especificando las credenciales del dispositivo remoto a conectarse.
- Búsqueda: El servidor Rendezvous busca en su base de datos de nodos registrados las credenciales provistas y retorna los datos de conexión del dispositivo remoto para un intento de conexión a través de la técnica de *Punch Hole* [7].

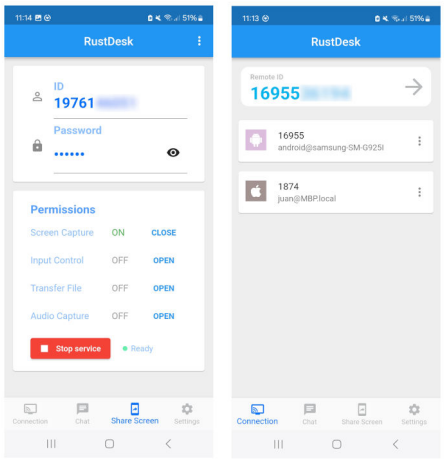

Fig. 1. Capturas de la aplicación de RustDesk para Android. La aplicación funciona tanto como cliente, para controlar un dispositivo remoto, como servidor para ser controlado.

– Comunicación directa: En caso que el *Punch Hole* funcione se establece la conexión con el stream de video correspondiente con el escritorio remoto. En caso que falle, se utiliza un segundo servidor rendezvous, llamado Relay Server el cual es accesible por los dispositivos cliente y servidor.

4 Implementación

Si bien la solución de RustDesk provee una implementación para la conexión punto a punto y el envío de comandos del cliente y el video del servidor, no provee soporte para la administración de los dispositivos a controlar. Esta funcionalidad es un valor agregado de las soluciones comerciales y es fundamental para empresas con gran cantidad de dispositivos para controlar, como lo es típicamente una empresa de cartelería digital. En la operación propia de una empresa de estas características es necesario contar con las nociones de cliente (*tenant*), dispositivo, carpeta de dispositivos, además de un esquema flexible de permisos pensado para el soporte técnico. Para lograr un sistema completo de administración de dispositivos se implementó un panel de administración web para los dispositivos y se modificó la aplicación cliente del proyecto RustDesk para interactuar con el mencionado panel.

4.1 Aplicación cliente

La aplicación cliente del proyecto RustDesk es el punto de entrada a la solución. Consiste en un ejecutable disponible para los sistemas operativos más utilizados tanto de escritorio como móviles. Si bien se la denomina aplicación cliente debido

a la arquitectura *rendezvous* mencionada, desde el punto de vista lógico funciona como cliente y servidor al mismo tiempo. Su utilización es similar a las soluciones comerciales: al momento de ejecutarla provee las credenciales necesarias para conectarse a ese dispositivo a partir de un número único a nivel sistema y una contraseña. También existe la posibilidad de usar ese mismo aplicativo para conectarse a otro dispositivo ingresando las credenciales del sistema remoto al cuál se quiere conectar. Ver imagen 1

La aplicación está desarrollada utilizando el framework Flutter [2] para desarrollo multiplataforma,ma, mientras que el núcleo está implementado utilizando el lenguaje Rust. De esta manera, se tienen clientes en diferentes sistemas operativos de manera rápida, sin necesidad de reescribir la lógica de comunicación.

Esta aplicación, parte del proyecto RustDesk, fue modificada para ser parte de la solución propia. En primer lugar, se implementó la lógica para que la aplicación se comunique con el panel de administración, a través de una autenticación. Al momento de instalar un nuevo dispositivo, la persona técnica encargada ingresa un usuario y una contraseña y la aplicación quedará asociada con el panel de seguimiento. Posteriormente, cada vez que la aplicación rote sus credenciales, se subirán actualizadas al panel. Esto es necesario para que el usuario final pueda acceder directamente desde el panel de seguimiento, sin necesidad de recordar las claves de acceso o de otro técnico frente al dispositivo.

4.2 Panel de administración

El objetivo general del trabajo en desarrollo consiste en implementar un sistema completo de administración de dispositivos remotos. En este sentido es necesario diseñar un sistema que permita, entre otras cuestiones, el manejo de los dispositivos por parte de los usuarios del sistema y los respectivos permisos, así como también la organización de los dispositivos de manera lógica.

El panel de seguimiento se implementó como una aplicación web RESTful cliente-servidor (Ver imagen 5). El backend del panel de administración fue implementado utilizando Python y el framework FastAPI, junto con una base de datos relacional.

La aplicación cuenta con ABM de usuarios y sistema de login. La organización de los dispositivos es multi-usuario (*multi-tenant*). Dentro de cada *tenant* se almacenan los dispositivos organizados lógicamente en carpetas. En sistemas de cartelería digital, esto permite organizar lógicamente los dispositivos emulando su organización física, por ejemplo, en cadenas de negocios que poseen múltiples sucursales.

5 Resultados y discusiones

Actualmente, el proyecto se encuentra en la etapa final de desarrollo. La elección de RustDesk como origen al desarrollo presentado fue producto de una etapa de pruebas de todas las opciones viables. En esta etapa se probaron múltiples soluciones, tanto comerciales como libres, algunas de ellas fueron: TeamViewer

Remote, AnyDesk, RealVNC, ConnectWise ScreenConnect, GoTo Resolve, EV Reach, Apache Guacamole y MeshCentral. La ventaja de RustDesk es su buen funcionamiento, su disponibilidad en diferentes sistemas operativos y la posibilidad de *self-hosting* de su arquitectura completa. Las pruebas fueron realizadas desplegando la arquitectura en sistemas de infraestructura como servicio (IaaS) Amazon Web Services y Microsoft Azure.

El equipo de trabajo está compuesto por dos líderes de proyecto, uno por cada institución involucrada en el desarrollo informático: UNS y SIA Interactive. Además, el equipo posee 4 desarrolladores de software entre ambas instituciones.

Una vez terminado, el proyecto será de dominio público y podrá ser usado y desplegado por cualquier organización que requiera resolver el problema del acceso remoto.

6 Conclusión

En este trabajo se presentó un caso de desarrollo de software llevado a cabo entre la Universidad Nacional del Sur, la empresa SIA Interactive por medio del financiamiento parcial de la fundación Sadosky. El desarrollo de este proyecto en colaboración entre la Universidad Nacional del Sur y la empresa SIA Interactive permitió enriquecer tanto el proceso de investigación como su aplicación práctica. En primer lugar, la universidad aportó conocimiento teórico, y una base académica sólida, lo que asegura que el proyecto esté respaldado por los últimos avances en el campo. Por otro lado, la empresa aportó experiencia práctica al haber desarrollado sistemas con restricciones similares, comprensión de las necesidades del mercado y capacidad para implementar soluciones de manera eficiente y rentable. Los avances realizados a partir de esta vinculación prometen dar soluciones a la comunidad sobre un problema cuya solución de código libre aún no existe y proponer un caso de éxito entre las dos instituciones.

References

1. Anydesk website. Available at http:// https://www.anydesk.com/ (10/4/2024)
2. Flutter website. Available at http://flutter.dev/ (10/4/2024)
3. Rustdesk website. Available at http:// rustdesk.com/ (17/4/2024)
4. Teamviewer website. Available at http://www.teamviewer.com/ (10/4/2024)
5. Hutchison, D., Bekkering, E.: Using remote desktop applications in education
6. Jiang, M., Gou, G., Shi, J., Xiong, G.: I know what you are doing with remote desktop. In: 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC). pp. 1–7 (2019). https://doi.org/10.1109/IPCCC47392.2019.8958721
7. Maier, D., Haase, O., Wäsch, J., Waldvogel, M.: Nat hole punching revisited. In: 2011 IEEE 36th Conference on Local Computer Networks. pp. 147–150 (2011). https://doi.org/10.1109/LCN.2011.6115173
8. Pelc, A.: Deterministic rendezvous in networks: A comprehensive survey. Networks **59**(3), 331–347 (2012). https://doi.org/https://doi.org/10.1002/net.21453, https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21453

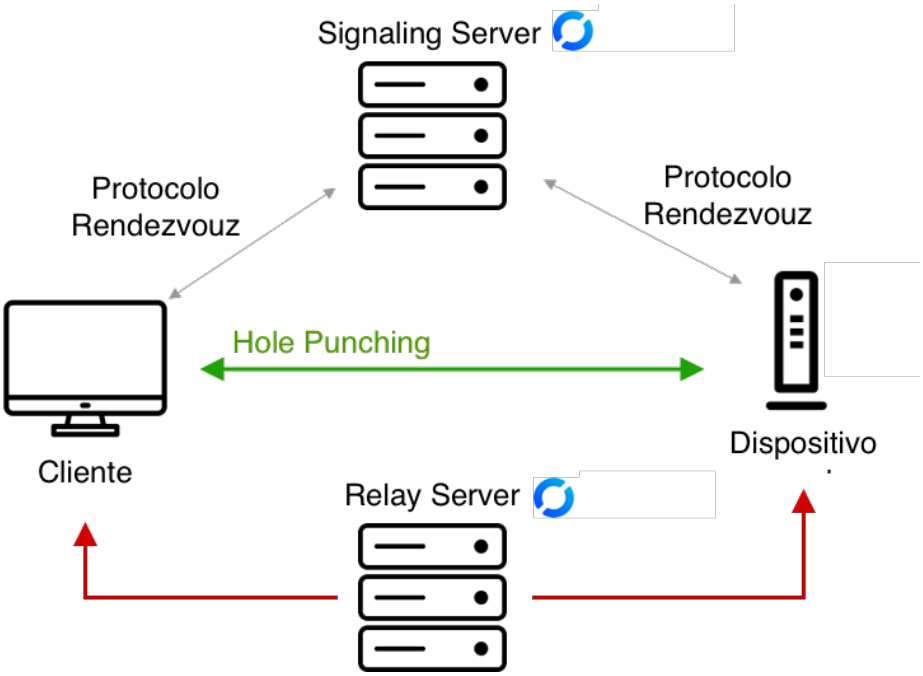

Fig. 2. El Protocolo Rendezvouz implementado permite que dos dispositivos puedan accederse y comunicarse en entornos de redes privadas. Al momento de iniciar la aplicación, tanto el cliente como el servidor se reportan al *signaling server* obteniendo un id único. Si la técnica de Hole Punching es exitosa se genera una conexión punto a punto, caso contrario la comunicación se realiza a través del *relay server*.

9. Song, T., Wang, J., Wu, J., Ma, R., Liang, A., Gu, T., Qi, Z.: Fastdesk: A remote desktop virtualization system for multi-tenant. Future Generation Computer Systems **81**, 478–491 (2018). https://doi.org/https://doi.org/10.1016/j.future.2017.07.001, https://www.sciencedirect.com/science/article/pii/S0167739X17304776

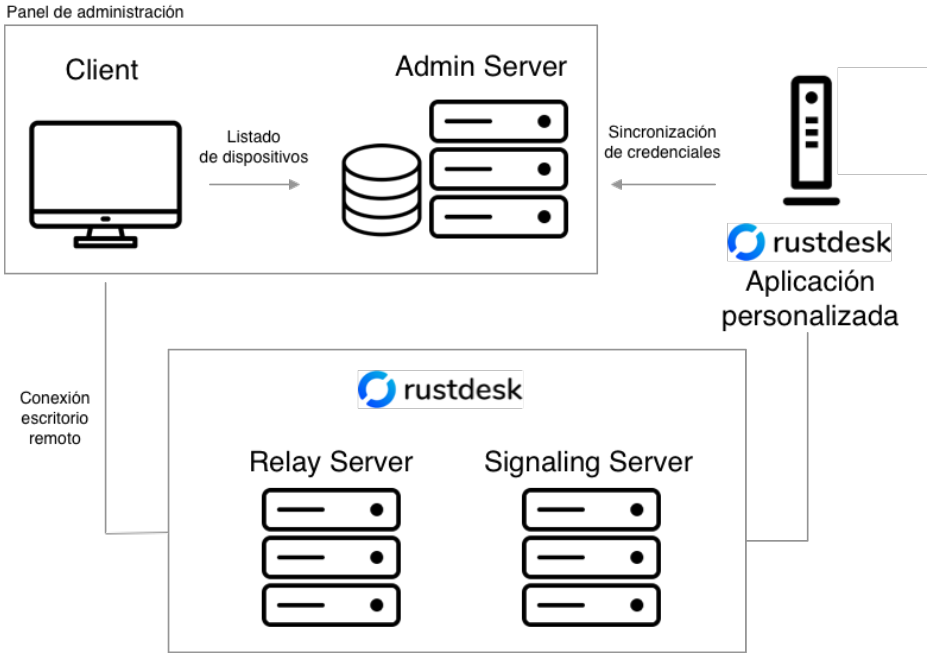

Fig. 3. Arquitectura del sistema desarrollado. A partir de las soluciones del Relay y Signaling Server, provistas por el proyecto RustDesk, se implemento una aplicación cliente-servidor web que permite la administración de dispositivos.

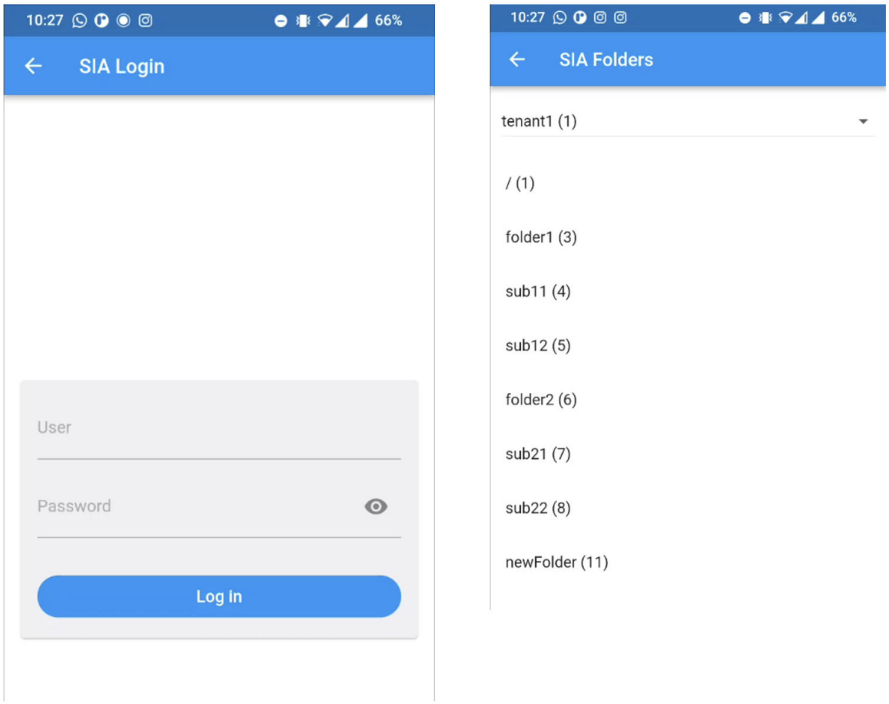

Fig. 4. La aplicación del proyecto original fue modificada para ser funcional a la nueva arquietectura del sistema planteada.

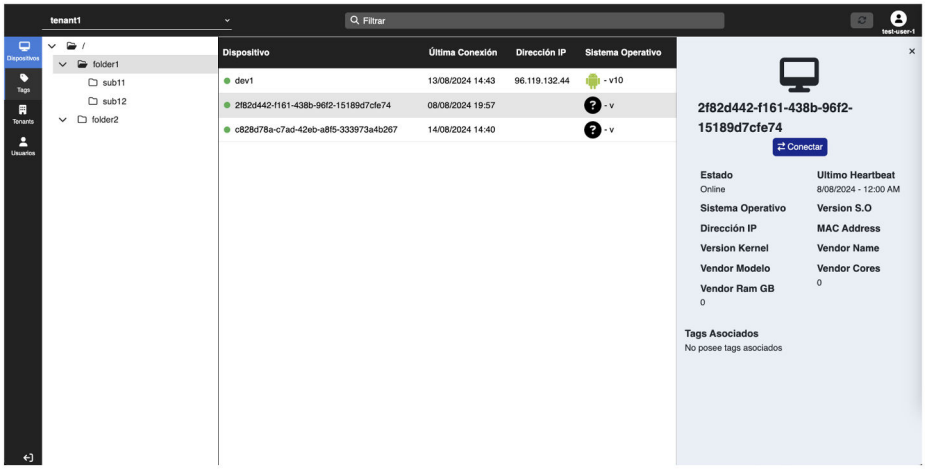

Fig. 5. Captura del panel web del administrador de dispositivos. Los dispositivos son organizados en tenants y carpetas

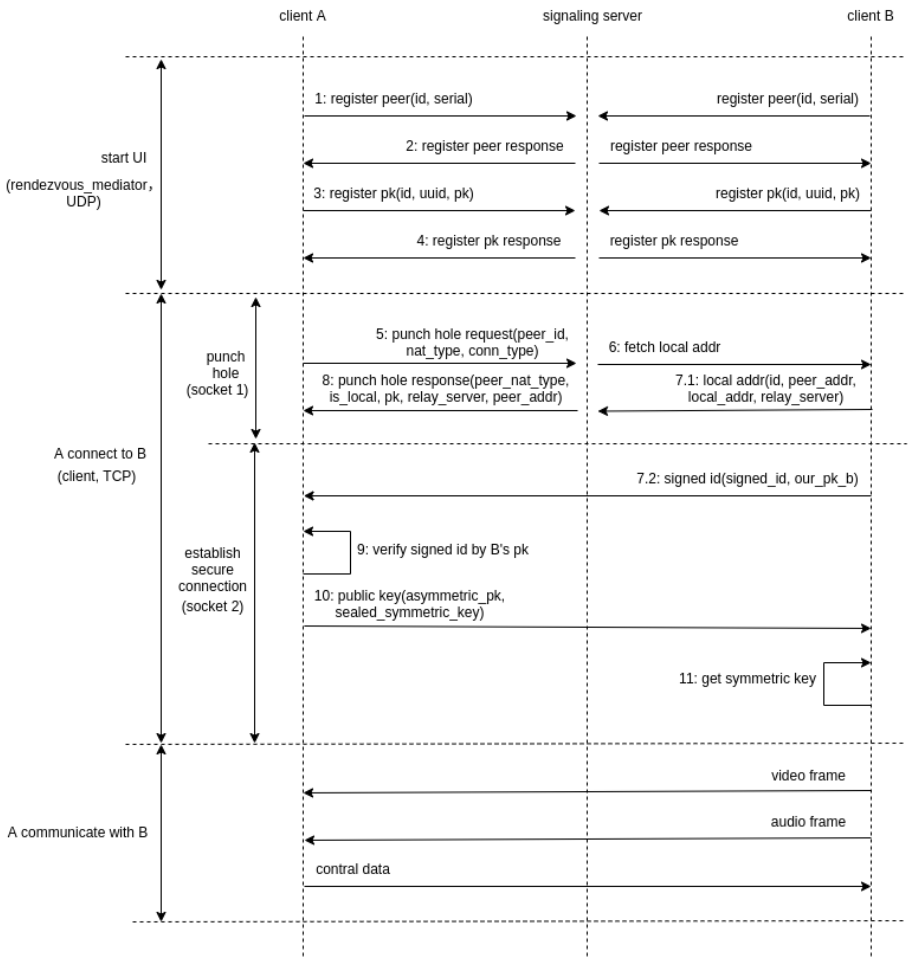

Fig. 6. Detalle de la implementación del protocolo de rendezvous. Imagen extraída de la documentación pública de RustDesk [3]