TRABAJO DE GRADO
Opción Investigación o Proyecto de Grado

NEXORY: Sistema de Gestión Integral para Proveedores de Internet (ISP)

Corporación Universitaria Remington

Facultad de Ingeniería

Programa de Ingeniería de Sistemas

Angie Vanesa Sánchez Cañaveral

Giovanny Alberto Flórez Osorio

Proyecto de Grado

2026

Tabla de contenido

Lista de tablas 5
Lista de Figuras 5
Dedicatoria 7
Agradecimientos 8
Resumen 9
Palabras clave 11
Introducción y Marco Teórico 12
Sistemas de información para la gestión de ISPs 12
Autenticación y seguridad en aplicaciones web 13
WhatsApp como canal empresarial 13
Computación en la nube con Microsoft Azure 14
Gestión de red con MikroTik y RouterOS 14
Planteamiento del Problema 15
Justificación 15
Objetivos 17
Objetivo General 17
Objetivos Específicos 17
Metodología 19
Tipo de investigación 19
Enfoque metodológico 19
Diseño metodológico 19
Población y muestra 19
Técnicas de recolección de información 20
Link de forms encuesta 21
Validación e Instrumentos 21
Levantamiento de requisitos 22
Diseño de la arquitectura 23
Desarrollo del backend 23
Integración con MikroTik RouterOS 24
Desarrollo del frontend 25

Pruebas y despliegue en Azure 25
Tecnologías utilizadas 27
Resultados y Discusión 28
Sistema de autenticación con doble factor (2FA) 28
Dashboard principal 31
Módulo de Clientes 32
Módulo de Soporte Técnico 35
Módulo de Cortes Automáticos de Servicio 36
Módulo de Comunicados Masivos 37
Panel de Notificaciones 40
Discusión y Análisis del Sistema 41
Impacto del sistema 41
Ventajas competitivas 41
Limitaciones identificadas 41
Comparación con soluciones comerciales 41
Aportes principales 42
Alcance del proyecto 42
Integración MikroTik: alcance técnico 43
Desafíos técnicos durante la implementación 44
Conclusiones 45
Referencias 46
Anexos 49
Anexo A. Arquitectura del sistema en Azure (producción) 49
Anexo B. Operaciones MikroTik implementadas 49
Anexo C. URLs de producción 50
Anexo D. Video demostrativo del sistema Nexory 50

Lista de tablas

Tabla 1 Tecnologías utilizadas en el proyecto Nexory. Elaboración propia, 2026. 24
Tabla 2 Operaciones MikroTik de Nexory. Elaboración propia, 2026. 46
Tabla 3 URLs del sistema Nexory en producción. Elaboración propia, 2026. 47

Lista de Figuras

Ilustración 1 Pantalla inicio formulario.....18
Ilustración 2 Formulario de encuesta 19
Ilustración 3 Pantalla de inicio de sesión de Nexory.....26
Ilustración 4 Pantalla de espera tras el envío del código OTP de seis dígitos 26
Ilustración 5 Código OTP recibido en WhatsApp.....26
Ilustración 6 Código OTP en correo 26
Ilustración 7 Vista del correo electrónico con el código OTP.....27
Ilustración 8 Ingreso del código correcto y acceso al dashboard 27
Ilustración 9 Alerta de inicio de sesión whatsapp 28
Ilustración 10 Alerta de inicio de sesión correo.....28
Ilustración 11 Dashboard principal de Nexory con KPIs del ISP 28
Ilustración 12 Dashboard vista extendida con indicadores adicionales.....29
Ilustración 13 Dashboard en modo oscuro.....29
Ilustración 14 Módulo de clientes sin registros al iniciar el sistema 30
Ilustración 15 . Formulario de creación: datos básicos 30
Ilustración 16 información de contacto.....30
Ilustración 17 Datos del plan y red 31
Ilustración 18 Configuración MikroTik PPPoE.....31
Ilustración 19 Cuenta creada en MikroTik.....31
Ilustración 20 Detalle del cliente RouterOS.....31
Ilustración 21 Perfil completo del cliente en Nexory con todos sus detalles 32
Ilustración 22 Diagnóstico de conectividad del cliente: cuenta, sesión, tráfico e IP asignada 32
Ilustración 23 Lista de tickets de soporte técnico con filtros 33
Ilustración 24 Detalle de un ticket con información completa.....33
Ilustración 25 . Ventana para agregar una nota interna al ticket 33
Ilustración 26 Nota registrada en el historial del ticket.....33
Ilustración 27 Módulo de cortes automáticos: cortes programados, ejecutados y restaurados.....34
Ilustración 28 Módulo de comunicados sin registros.....35
Ilustración 29 Paso 1: datos generales del comunicado 35

Ilustración 30 Paso 2: redacción del mensaje 35
Ilustración 31 Paso 3: vista previa del mensaje en WhatsApp.....35
Ilustración 32 Paso 4: selección de canales (WhatsApp / correo).....35
Ilustración 33 Paso 5: selección de destinatarios (todos, por plan o por etiqueta) 36
Ilustración 34 Comunicado publicado en la lista con detalle de envíos.....36
Ilustración 35 Comunicado recibido en el correo electrónico del cliente 37
Ilustración 36 Comunicado recibido por WhatsApp.....37
Ilustración 37 Panel de notificaciones del sistema.....37

Dedicatoria

Este trabajo está dedicado a los pequeños proveedores de internet que operan en municipios y barrios de Colombia donde la conectividad todavía es una promesa a media. A ellos, que con pocos recursos sostienen la conexión de familias enteras. También a mi familia, por su paciencia, apoyo y comprensión durante todo el desarrollo de este proyecto y por motivarme.

Agradecimientos

Se agradece a la Corporación Universitaria Remington por todos los conocimientos y aprendizajes brindados durante la formación académica. También al tutor, por el acompañamiento, la orientación y la paciencia a lo largo de este proceso. De igual manera, se reconoce el aporte de las comunidades y herramientas de código abierto como Node.js, React, Prisma, PostgreSQL y WAHA, que fueron fundamentales para el desarrollo del proyecto. Finalmente, se agradece al ISP que permitió conocer de cerca las necesidades y dificultades reales que se buscaban solucionar.

Resumen

Nexory es una plataforma web diseñada para que los proveedores de internet de pequeña y mediana escala en Colombia puedan administrar su operación desde un solo lugar. El proyecto se originó a partir de la identificación de que muchos de estos negocios todavía gestionan su información mediante hojas de cálculo, grupos de WhatsApp y registros en papel, lo que genera errores y pérdida de información importante.

El proyecto se desarrolló mediante una metodología ágil basada en iteraciones, realizando ajustes continuos según las necesidades identificadas durante las pruebas con usuarios reales de un ISP. Para la recolección de información se realizaron entrevistas con personal administrativo y técnico, permitiendo identificar los procesos que requerían mayor optimización.

El alcance incluyó el diseño, desarrollo, despliegue y pruebas funcionales de una plataforma para la gestión de clientes, tickets de soporte, cortes y reconexiones automáticas, autenticación de doble factor y envío de comunicados masivos, con integración exclusiva a equipos MikroTik mediante la API de RouterOS.

Para el backend se utilizó Node.js 22 con Express, para el frontend React 19 con Vite y Tailwind CSS, y PostgreSQL como base de datos. Para las notificaciones y el doble factor de autenticación se utilizó WAHA, que permite enviar mensajes de WhatsApp mediante una API REST. Una de las principales características de Nexory consiste en la comunicación directa con routers MikroTik, desde donde es posible gestionar conexiones PPPoE, consultar el estado de los clientes y monitorear los equipos sin necesidad de acceder directamente a RouterOS.

El despliegue se realizó en Microsoft Azure utilizando servicios de bajo costo. Los resultados obtenidos muestran que la plataforma centraliza procesos que anteriormente requerían varias horas

de trabajo y los reduce a minutos. Como resultado del desarrollo realizado, se obtuvo un sistema funcional que facilita la gestión operativa de los ISP y permite ejecutar tareas técnicas y administrativas desde una única plataforma.

Palabras clave

Gestión de clientes, plataforma web, gestión empresarial, proveedor de internet, MikroTik.

Introducción y Marco Teórico

Colombia tiene más de 1.200 proveedores de internet registrados ante el MinTIC, y la gran mayoría de ellos son microempresas que operan en municipios pequeños o en barrios periféricos de las ciudades principales. Aunque estos ISP juegan un papel importante en la conectividad del país, su gestión interna suele ser precaria: los datos de los clientes se guardan en hojas de Excel, los cobros se anotan en cuadernos y los reportes de fallas llegan por grupos de WhatsApp sin ningún tipo de seguimiento.

El presente proyecto surge de la observación directa de esa realidad en un ISP de Colombia. El objetivo fue construir una herramienta que se ajustara a sus necesidades: sin licencias costosas, sin infraestructura propia, y que usara los canales de comunicación que el equipo ya conoce, principalmente WhatsApp.

Sistemas de información para la gestión de ISPs

Un sistema de información, en su definición más directa, es un conjunto de componentes que trabajan juntos para recopilar, procesar y distribuir datos que apoyan la toma de decisiones dentro de una organización (Laudon & Laudon, 2020). Para un ISP, esto se traduce en poder saber en tiempo real cuántos clientes están conectados, cuáles tienen pagos pendientes o cuántos tickets de soporte llevan más de 24 horas sin resolverse.

Existen soluciones en el mercado como SONAR, POWERCODE o ISPConfig que cubren muchas de estas necesidades, pero sus costos de licencia rondan los cientos o miles de dólares al año, lo cual está fuera del alcance de operadores pequeños. Nexory busca ocupar ese espacio intermedio.

Autenticación y seguridad en aplicaciones web

La seguridad fue una prioridad desde el diseño. Para la autenticación se utilizó JSON Web Tokens (JWT), un estándar abierto que transmite información verificada entre el frontend y el backend sin necesidad de guardar sesiones en el servidor (Jones et al., 2015).

Además, se agregó un segundo factor de autenticación (2FA) basado en códigos OTP de seis dígitos que se envían al teléfono del usuario por WhatsApp. Las contraseñas PPPoE de los clientes se almacenan cifradas con AES-256, de modo que incluso si alguien accediera a los datos en crudo, no podría leer las credenciales de red (NIST SP 800-63B, 2017).

WhatsApp como canal empresarial

WhatsApp cuenta con más de 2.000 millones de usuarios en el mundo y en América Latina es el canal de comunicación empresarial más utilizado. Para un ISP pequeño, esto significa que tanto el personal como los clientes ya se encuentran en esta plataforma.

WAHA (WhatsApp HTTP API) es una solución de código abierto que se ejecuta en un contenedor Docker y expone una API REST para enviar y recibir mensajes. A diferencia de la API oficial de Meta, que requiere aprobación empresarial y tiene costos por mensaje, WAHA funciona conectando directamente un número de WhatsApp al sistema, lo que lo hace accesible para proyectos con presupuesto limitado (DevLikeAPro, 2024).

Computación en la nube con Microsoft Azure

Desplegar la aplicación en servidores propios habría requerido una inversión inicial importante. La nube resuelve ese problema: se paga únicamente por lo utilizado, la infraestructura es gestionada por el proveedor y el sistema queda accesible desde cualquier lugar con conexión a internet.

Microsoft Azure fue la plataforma elegida por su integración con las herramientas de desarrollo utilizadas y porque ofrece planes de nivel básico (B1, B1ms) que permiten mantener un sistema funcional con costos manejables (Microsoft Azure, 2024).

Gestión de red con MikroTik y RouterOS

MikroTik es una empresa de origen letón cuyos routers son ampliamente utilizados por ISPs en Latinoamérica, principalmente por su relación costo-beneficio y la potencia de RouterOS. Uno de sus aspectos más útiles es que expone una API propietaria en el puerto 8728 que permite automatizar prácticamente cualquier tarea de administración de red desde una aplicación externa (MikroTik, 2023).

Para los ISP que basan su servicio en conexiones PPPoE, esta integración es clave: sin necesidad de salir de Nexory, el operador puede crear cuentas, cortar el servicio, restaurarlo, reiniciar sesiones y consultar métricas del equipo. Esto elimina la necesidad de que el personal técnico acceda directamente a la consola de RouterOS para tareas rutinarias.

Planteamiento del Problema

La gestión de un ISP pequeño en Colombia implica el cobro de mensualidades, la atención de fallas de conectividad, el seguimiento de clientes en mora y la coordinación con el equipo técnico en campo. Sin herramientas adecuadas, estas tareas recaen en pocas personas que atienden múltiples urgencias de forma continua, sin una visión clara del estado del negocio.

El problema se agrava debido a los altos costos de licencia o la complejidad de las soluciones disponibles, que en muchos casos requieren apoyo especializado para su implementación. Por ello, muchos ISP locales continúan utilizando Excel y WhatsApp como principales herramientas de gestión.

En este contexto, el proyecto se orienta a responder la siguiente pregunta: ¿es posible desarrollar un sistema de gestión integral para proveedores de internet que sea económicamente viable, fácil de usar para personal sin conocimientos técnicos avanzados y que se integre con las herramientas y equipos existentes?

Justificación

Técnicamente, las herramientas actuales de desarrollo web permiten construir sistemas robustos con equipos pequeños y costos bajos. La cantidad de librerías de código abierto disponibles para Node.js, React y PostgreSQL hace que el tiempo de desarrollo sea manejable para equipos pequeños o desarrolladores individuales.

Desde el punto de vista económico, el modelo de nube elimina la inversión inicial en servidores. Con Azure es posible mantener en producción un sistema como Nexory por un costo mensual razonable para una pequeña empresa.

Socialmente, mejorar la eficiencia de los ISP locales tiene un impacto directo en la calidad del servicio que reciben sus clientes. Una empresa que puede identificar rápidamente una falla y comunicarla por WhatsApp opera de manera más eficiente y genera mayor confianza.

Objetivos

Objetivo General

Desarrollar una plataforma web para la gestión de proveedores de internet que centralice los procesos administrativos y operativos, incluyendo la administración de clientes, soporte técnico, pagos, comunicaciones y gestión de red con routers MikroTik, con despliegue en la nube de Microsoft Azure.

Objetivos Específicos

- Diseñar e implementar un backend RESTful con Node.js, Express y TypeScript, conectado a PostgreSQL mediante Prisma ORM, que gestione todos los módulos del sistema.
- Construir una interfaz web (SPA) con React 19, Vite y Tailwind CSS que permita operar el sistema sin conocimientos técnicos especializados.
- Integrar autenticación con JWT y doble factor (2FA) mediante OTP enviado por WhatsApp a través de WAHA, con cifrado AES-256 para credenciales sensibles.
- Implementar integración completa con RouterOS de MikroTik: creación de cuentas PPPoE, habilitación, deshabilitación, reinicio de sesiones, diagnóstico y monitoreo de métricas.

- Desplegar el sistema en Microsoft Azure usando App Service, Static Web Apps, Container Instances y PostgreSQL Flexible Server, optimizando costos con planes básicos.
- Desarrollar un módulo de comunicados masivos por WhatsApp para notificar a los clientes sobre cortes, pagos y avisos generales de forma personalizada.

Metodología

Nexory no se construyó en orden de arriba hacia abajo. Se trabajó de manera iterativa, avanzando por módulos y realizando ajustes sobre la marcha según lo que se iba descubriendo al probar con usuarios reales del ISP. Cada semana surgían correcciones o funcionalidades que terminaban siendo diferentes a como se habían planeado.

Tipo de investigación

Se desarrolló una investigación de tipo aplicada, orientada a la resolución de problemas prácticos y operativos dentro de la gestión de proveedores de internet de pequeña escala en Colombia.

Enfoque metodológico

Se utilizó un enfoque mixto (cualitativo y cuantitativo). Cualitativamente, se analizaron las necesidades y flujos de trabajo del personal. Cuantitativamente, se evaluó la eficiencia del sistema midiendo tiempos de procesamiento técnico y automatización de alertas.

Diseño metodológico

Se implementó una metodología ágil de desarrollo por bloques o sprints iterativos, permitiendo separar el desarrollo en capas (frontend, backend, base de datos) y realizar entregas funcionales semanales ajustadas según la retroalimentación de los usuarios.

Población y muestra

La población de estudio abarca los pequeños proveedores de servicios de internet en Colombia. La muestra específica para la validación del prototipo estuvo constituida por el equipo humano de un ISP local en funcionamiento, incluyendo su administrador general, personal técnico de campo y gestores de cobro.

Técnicas de recolección de información

Se utilizaron entrevistas semiestructuradas con el personal del ISP para identificar la problemática inicial. Adicionalmente, como instrumento de validación formal, se aplicó un formulario estructurado digital para recopilar métricas de usabilidad y feedback de experiencia de usuario.



Ilustración 1 Pantalla inicio formulario.

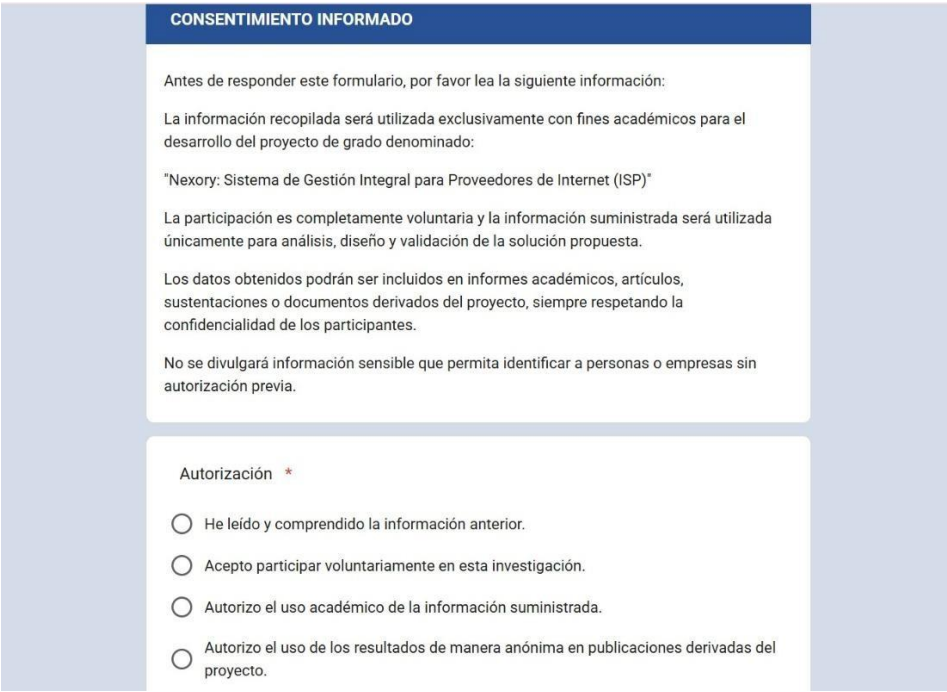


Ilustración 2 Formulario de encuesta

Link de forms encuesta

https://forms.gle/pcVH5Ee7R2rePxYx8

Validación e Instrumentos

Los módulos fueron validados mediante pruebas funcionales de caja negra y de integración con enrutadores físicos MikroTik, controlando escenarios críticos de desconexión y latencia para garantizar la estabilidad del servicio bajo RouterOS 7.x y versiones previas.

Levantamiento de requisitos

Para identificar los requerimientos del sistema se realizaron entrevistas con el personal que haría uso de la plataforma: el administrador del ISP, los técnicos en campo y la persona encargada de cobros. Cada uno tenía una perspectiva diferente del problema. El administrador requería visualizar reportes y controlar el acceso de su equipo. Los técnicos necesitaban poder consultar el estado de un cliente desde el celular. La persona encargada de cobros buscaba automatizar los recordatorios de pago por WhatsApp.

De esas conversaciones surgió la lista de módulos del sistema y la prioridad de cada uno. También se identificó que la conexión con MikroTik era indispensable, ya que el ISP opera completamente con esos routers y cualquier herramienta que no se conectara con ellos quedaría incompleta.

Diseño de la arquitectura

Se decidió separar el sistema en tres capas independientes: el frontend como una aplicación de página única (SPA) que el usuario visualiza en el navegador, el backend como una API REST que centraliza la lógica del negocio y la base de datos PostgreSQL gestionada con Prisma. Esta separación permitió trabajar en paralelo.

También se planificó la tabla de logs de auditoría desde el inicio, ya que conocer quién realizó cada acción dentro del sistema era un requisito considerado no negociable por el administrador.

Desarrollo del backend

Se escribió el backend con Node.js 22 y TypeScript. El uso de TypeScript obligó a definir tipos para todo desde el principio, lo cual al comienzo se percibía como más trabajo, pero posteriormente resultó beneficioso. Cada módulo cuenta con su propio controlador, sus rutas y sus validaciones con Zod antes de que cualquier dato llegue a la base de datos.

El sistema de autenticación fue uno de los primeros en desarrollarse, ya que todos los demás módulos dependen de él. El flujo de 2FA con WAHA requirió varias pruebas hasta lograr que el código OTP llegara de manera confiable.

Integración con MikroTik RouterOS

La integración con RouterOS representó uno de los principales retos técnicos durante el desarrollo del proyecto. El desafío no fue solo conectarse al router, sino manejar los casos donde el router no responde, o donde RouterOS

7.x devuelve respuestas vacías con un formato diferente al de versiones anteriores.

Para ese último problema se desarrollo una función auxiliar llamada safeWrite que captura el error UNKNOWNREPLY y devuelve un arreglo vacío.

Las operaciones implementadas cubren el ciclo de vida completo de un cliente de red: crear su cuenta PPPoE, cortar el servicio por mora, restaurarlo cuando paga, reiniciar una sesión con problemas y hacer diagnóstico completo con cuatro verificaciones.

Desarrollo del frontend

El frontend quedó como una SPA hecha con React 19 y Vite. Para el enrutamiento usamos React Router v7, para el manejo de peticiones TanStack Query, y Zustand para el estado global. El diseño visual usa Tailwind CSS v4 con componentes de Radix UI.

Un aspecto al que se le dio importancia fue que cualquier miembro del equipo del ISP, sin importar qué tan familiarizado estuviera con software de gestión, pudiera usar la plataforma con poca o ninguna capacitación.

Pruebas y despliegue en Azure

El despliegue en Azure trajo varios problemas que no habían aparecido en el entorno local. El más importante fue la incompatibilidad del binario nativo de Prisma entre macOS ARM y el Linux Debian de Azure App Service. La solución fue declarar el target "debian-openssl-3.0.x" en el schema de Prisma y generar el binario correcto antes de empaquetar el despliegue.

También encontro bloqueo del DNS de npmjs.org en redes corporativas, y el firewall de PostgreSQL en Azure que por defecto no permite conexiones desde App Service.

Tecnologías utilizadas

La siguiente tabla resume las herramientas y servicios usados en el proyecto:

Tabla 1 Tecnologías utilizadas en el proyecto Nexory. Elaboración propia, 2026.

Componente	Tecnología	Versión
Runtime backend	Node.js	22 LTS
Framework backend	Express	4.18
Lenguaje	TypeScript	5.3 / 5.9
ORM	Prisma	6.19
Autenticación	jsonwebtoken + bcryptjs	9.0 / 2.4
Cifrado credenciales	crypto-js (AES-256)	4.2
Validación	Zod	3.22
Integración MikroTik	node-routeros	1.6
Tareas programadas	node-cron	4.2
Framework frontend	React	19.2
Build tool	Vite	7.2
Estilos	Tailwind CSS	v4
Componentes UI	Radix UI + shadcn/ui	Último
Estado global	Zustand	5.0
Peticiones HTTP	TanStack Query	5.90
Animaciones	Framer Motion	12.24
Base de datos	PostgreSQL	16
WhatsApp API	WAHA (devlikeapro/waha)	noweb
Contenedorización	Docker	Latest
Backend en la nube	Azure App Service (Linux B1)	Node 22-lts
Frontend en la nube	Azure Static Web Apps	Free tier
WhatsApp en la nube	Azure Container Instances	1 core / 1.5 GB
Base de datos en la nube	Azure PostgreSQL Flexible	Standard_B1ms
Registro de imágenes	Azure Container Registry	Basic
Editor	Visual Studio Code	Último
Control de versiones	Git	Último

Resultados y Discusión

Al finalizar el desarrollo, Nexory quedó en producción con ocho módulos funcionales accesibles desde internet. A continuación, se presenta cada módulo con capturas de pantalla del sistema en funcionamiento real.

Sistema de autenticación con doble factor (2FA)

El acceso a Nexory requiere dos pasos. Primero el usuario ingresa su nombre de usuario y contraseña en la pantalla de login. Si las credenciales son correctas, el sistema genera un código OTP de seis dígitos, lo guarda en la base de datos con expiración de diez minutos y lo envía por WhatsApp al teléfono registrado. El usuario ingresa ese código y el sistema le emite un token JWT con 24 horas de vigencia. Si alguien roba las credenciales de un empleado, no puede entrar sin tener acceso físico al teléfono de esa persona.

Adicionalmente, cada inicio de sesión exitoso dispara una alerta de seguridad que llega tanto por WhatsApp como por correo electrónico, informando hora y dispositivo desde el que se accedió.

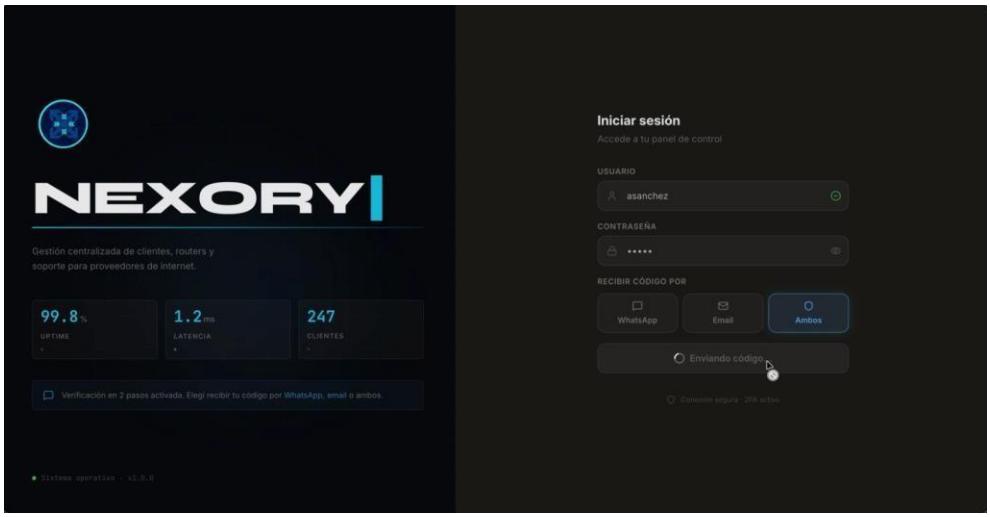


Ilustración 3 Pantalla de inicio de sesión de Nexory

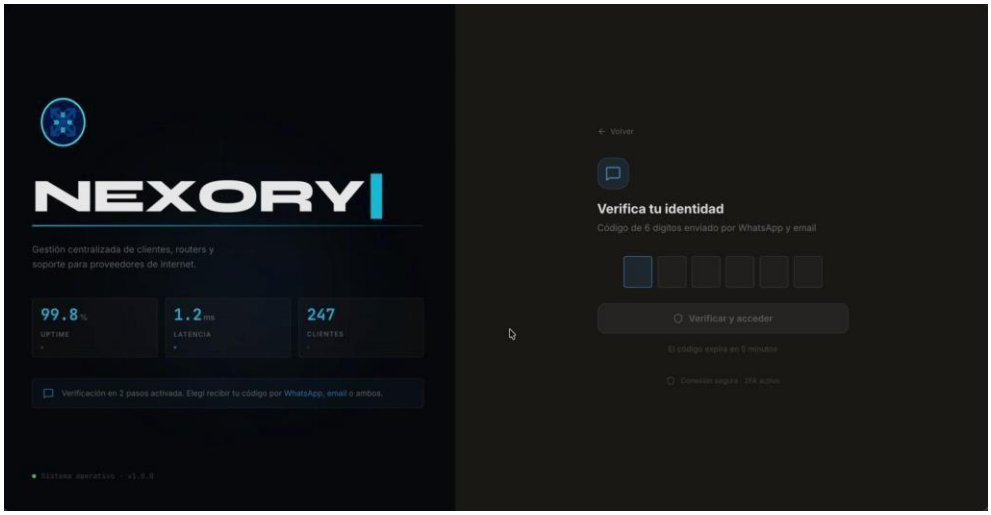


Ilustración 4 Pantalla de espera tras el envío del código OTP de seis dígitos



Ilustración 5 Código OTP recibido en WhatsApp



Ilustración 6 Código OTP en correo

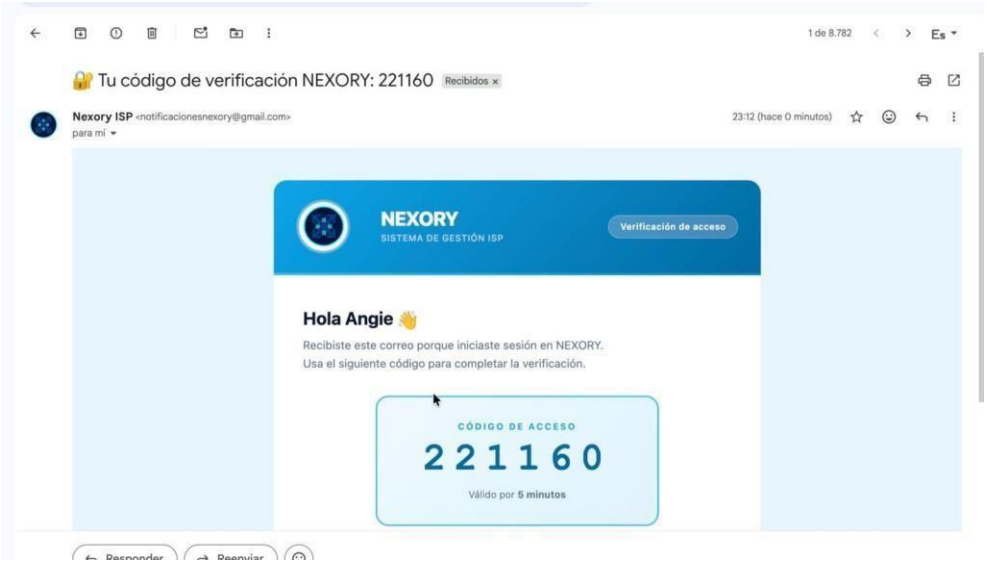


Ilustración 7 Vista del correo electrónico con el código OTP

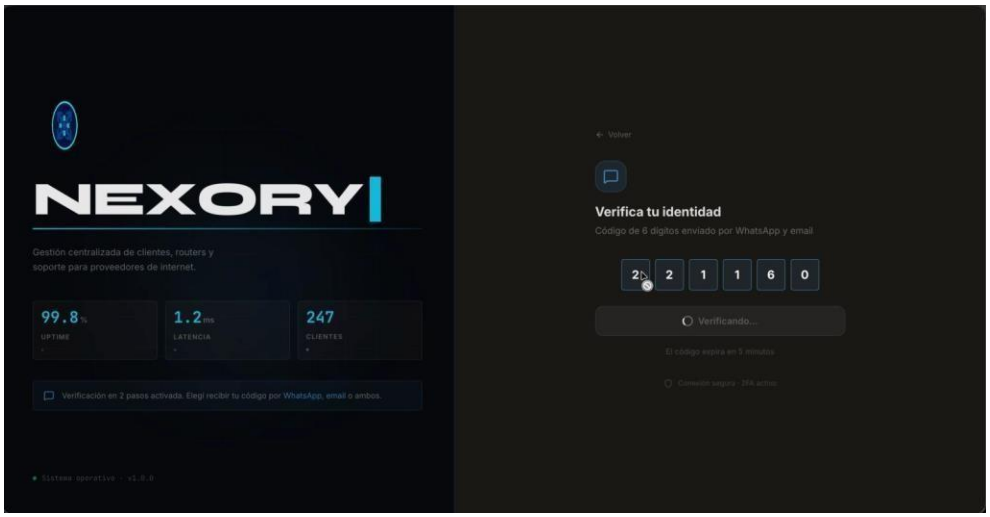


Ilustración 8 Ingreso del código correcto y acceso al dashboard



Ilustración 9 Alerta de inicio de sesión whatsapp

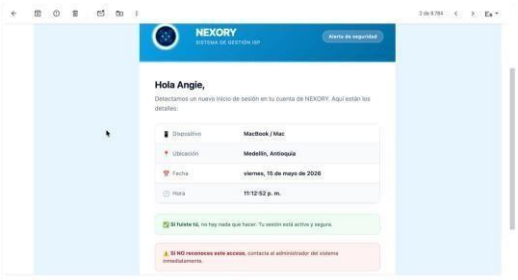


Ilustración 10 Alerta de inicio de sesión correo

Dashboard principal

El dashboard es la primera pantalla que aparece después de iniciar sesión. Muestra los indicadores más importantes del ISP de un vistazo: clientes activos, tickets abiertos, uptime de red y pagos pendientes. Esa pantalla es la que más usan los administradores porque les permite arrancar el día sabiendo qué es lo más urgente. También cuenta con modo oscuro, que el usuario puede activar desde cualquier pantalla.

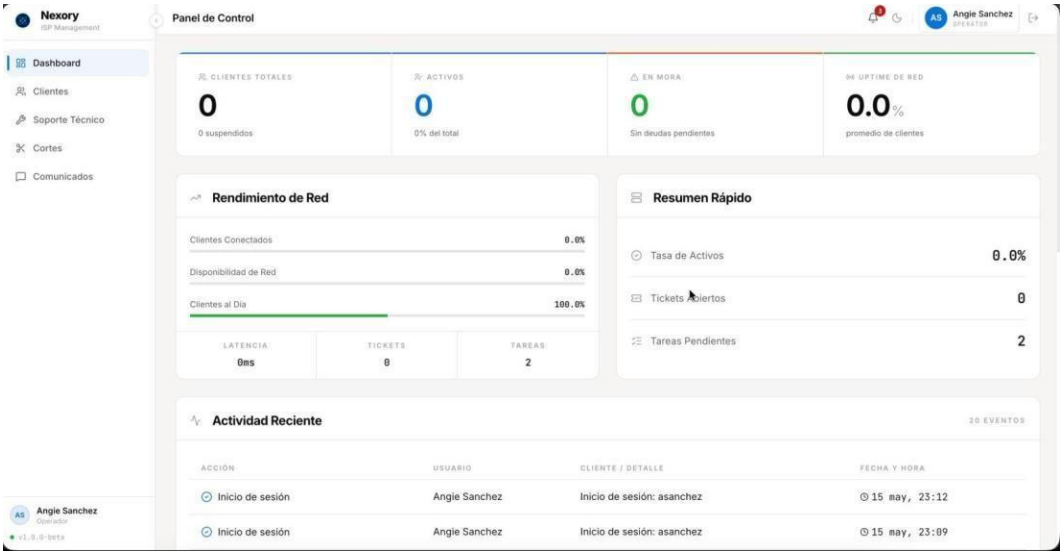


Ilustración 11 Dashboard principal de Nexory con KPIs del ISP

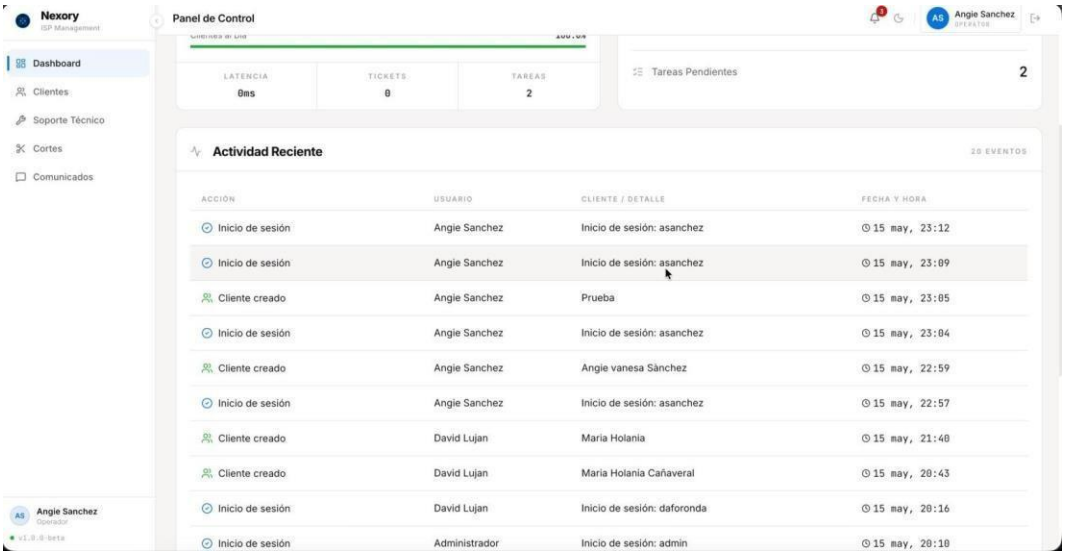


Ilustración 12 Dashboard vista extendida con indicadores adicionales

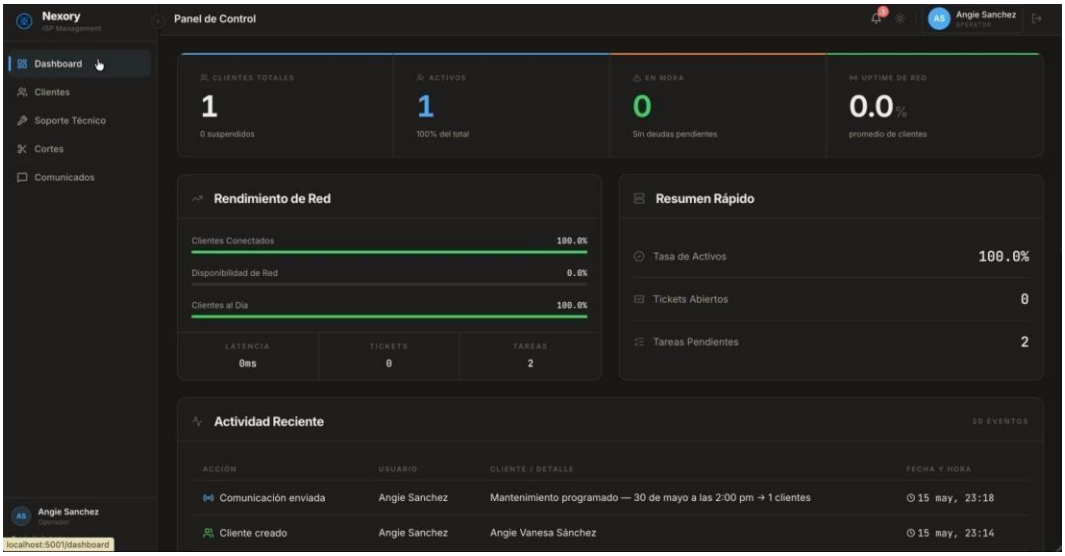


Ilustración 13 Dashboard en modo oscuro

Módulo de Clientes

El módulo de clientes centraliza toda la información de cada suscriptor: datos de contacto, plan contratado, velocidad, día de pago, historial de conexiones y las credenciales PPPoE del router. Al registrar un cliente nuevo, el sistema ofrece crearlo simultáneamente

en MikroTik, verificando que el nombre de usuario no exista previamente para evitar duplicados.

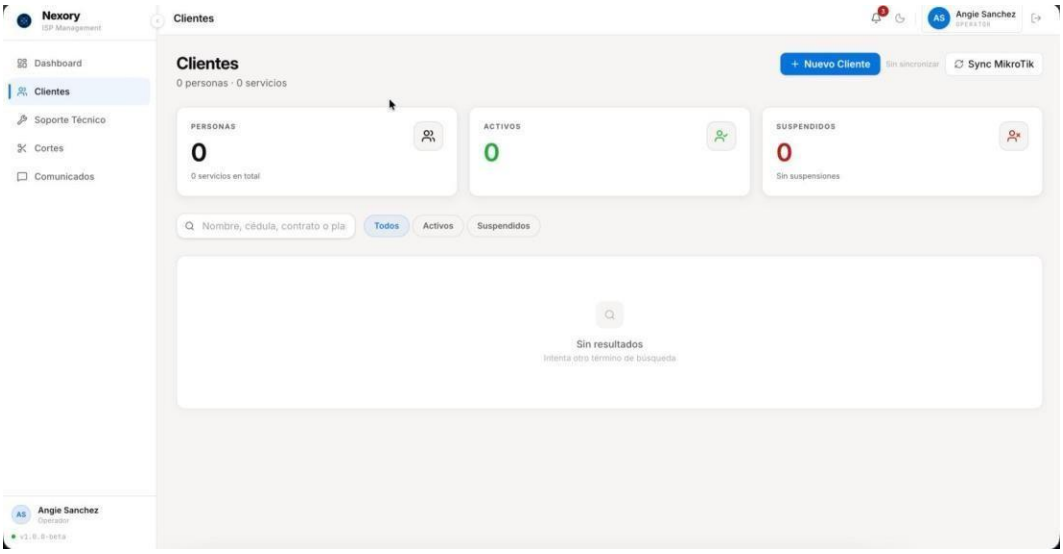


Ilustración 14 Módulo de clientes sin registros al iniciar el sistema

El formulario de creación de cliente se divide en varios pasos para no abrumar al operario con todos los campos a la vez:

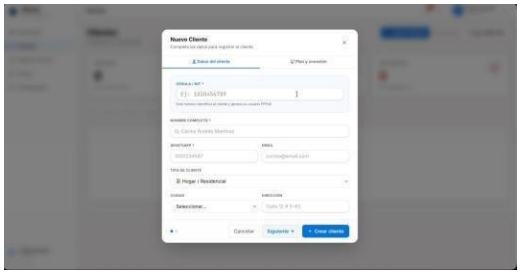


Ilustración 15 . Formulario de creación: datos básicos

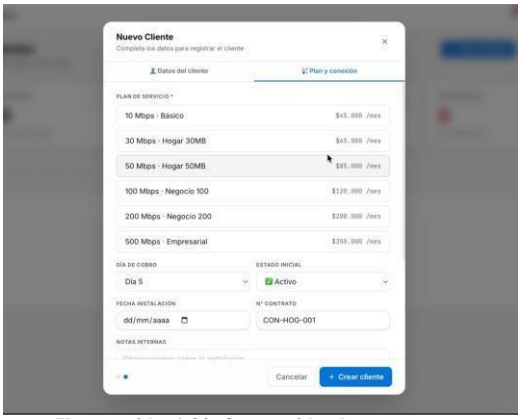


Ilustración 16 información de contacto

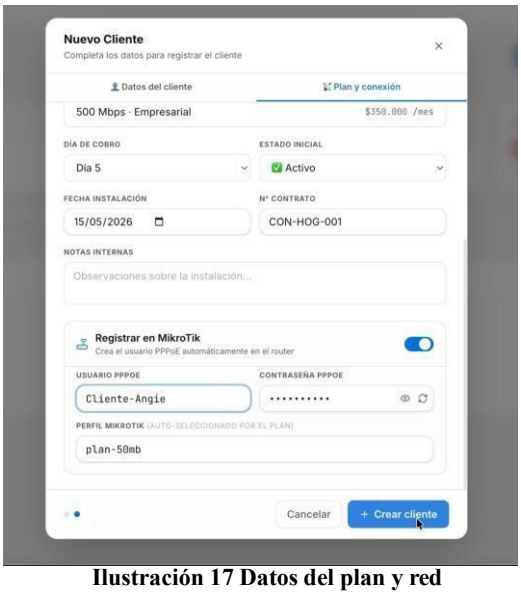


Ilustración 17 Datos del plan y red

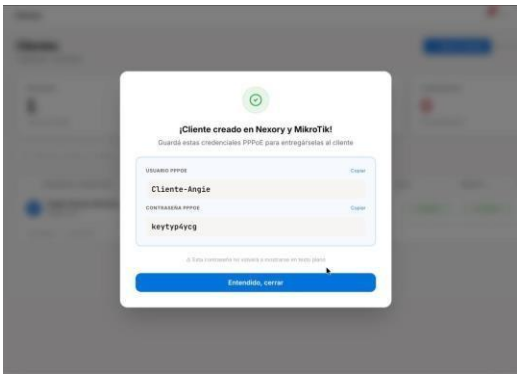


Ilustración 18 Configuración MikroTik PPPoE

Una vez creado el cliente en Nexory, la cuenta PPPoE queda registrada automáticamente en el router MikroTik. Las siguientes imágenes muestran cómo aparece ese registro directamente en RouterOS:



Ilustración 19 Cuenta creada en MikroTik



Ilustración 20 Detalle del cliente RouterOS

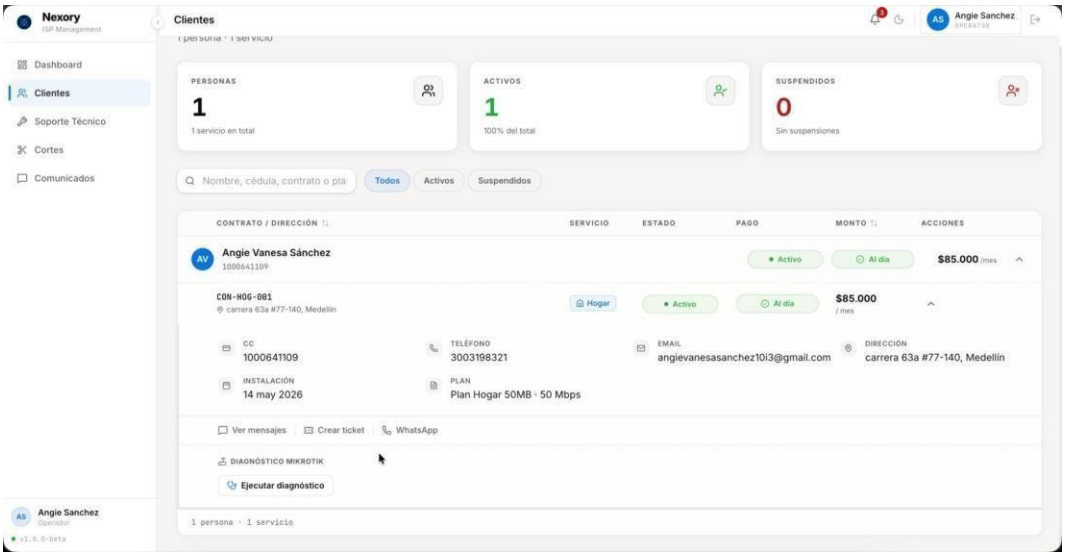


Ilustración 21 Perfil completo del cliente en Nexory con todos sus detalles

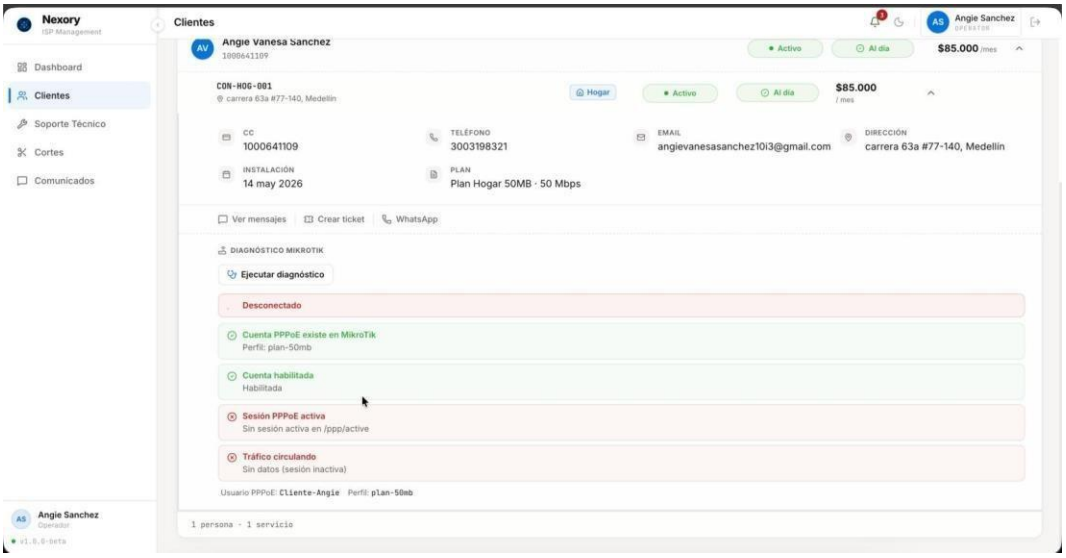


Ilustración 22 Diagnóstico de conectividad del cliente: cuenta, sesión, tráfico e IP asignada

Módulo de Soporte Técnico

El módulo de soporte gestiona los tickets con cuatro niveles de prioridad (BAJA, MEDIA, ALTA, URGENTE) y cinco estados posibles. Cada ticket puede tener notas internas para que el técnico deje registro de lo que encontró durante la atención de la falla.

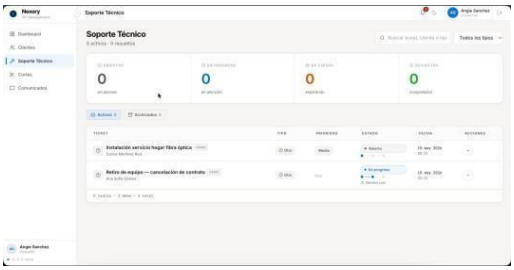


Ilustración 23 Lista de tickets de soporte técnico con filtros

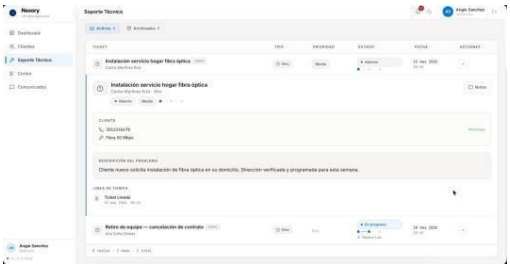


Ilustración 24 Detalle de un ticket con información completa

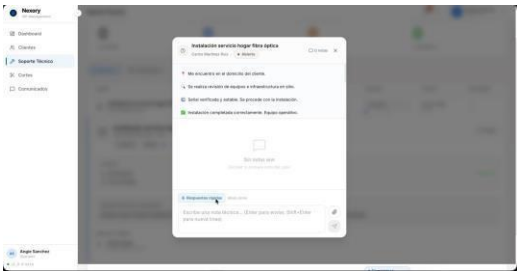


Ilustración 25 . Ventana para agregar una nota interna al ticket

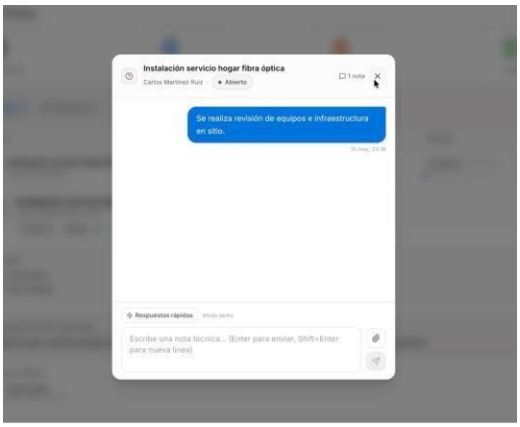


Ilustración 26 Nota* registrada *en el historial del ticket

Módulo de Cortes Automáticos de Servicio

El módulo de cortes permite programar suspensiones de servicio y ejecutarlas directamente sobre el router MikroTik. Registra la fecha de programación, la de ejecución y la de restauración del servicio. Incluye un job automático para cortes mensuales por mora.

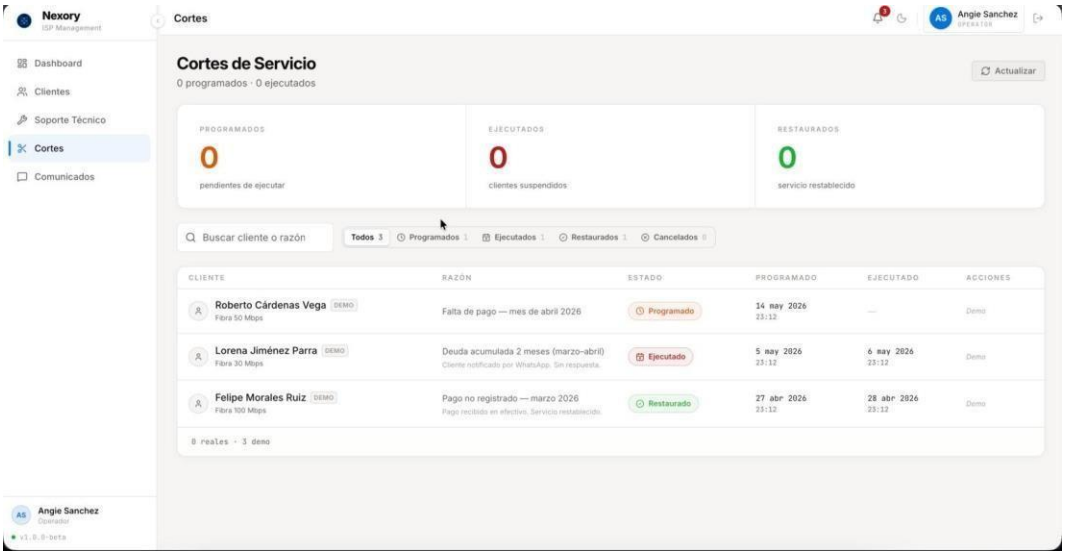


Ilustración 27 Módulo de cortes automáticos: cortes programados, ejecutados y restaurados

Módulo de Comunicados Masivos

El módulo de comunicados permite enviar mensajes masivos por WhatsApp y/o correo electrónico a grupos de clientes filtrados por estado, plan o etiquetas. El sistema personaliza cada mensaje con el nombre del destinatario antes de enviarlo y registra el resultado de cada envío.

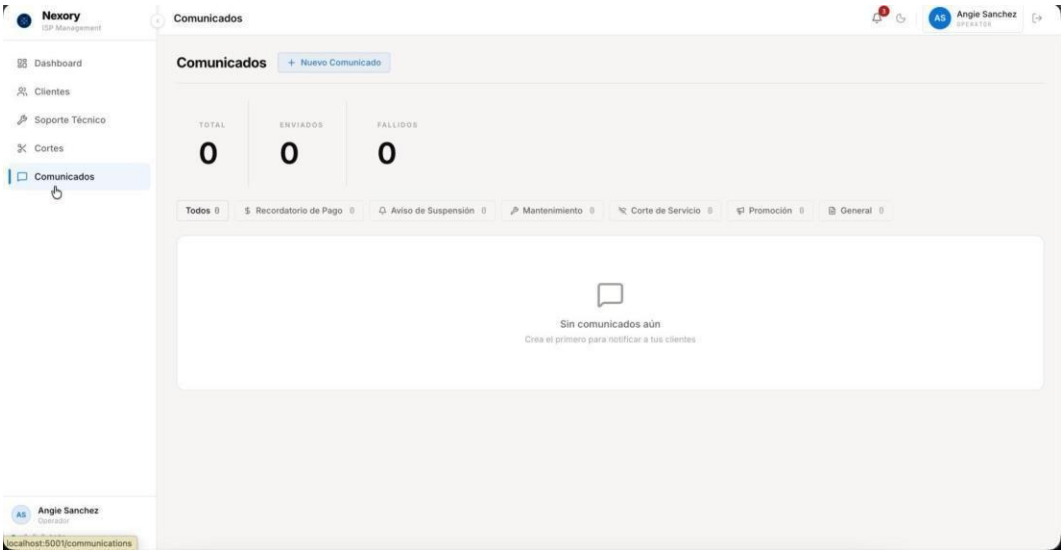


Ilustración 28 Módulo de comunicados sin registros

La creación de un comunicado guía al operario por cinco pasos: datos generales, redacción del mensaje, vista previa en WhatsApp, selección de canales de envío y selección de destinatarios:

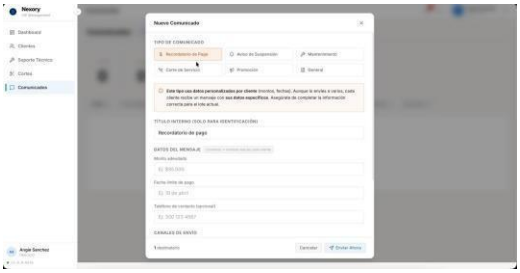


Ilustración 29 Paso 1: datos generales del comunicado



Ilustración 30 Paso 2: redacción del mensaje



Ilustración 31 Paso 3: vista previa del mensaje en WhatsApp

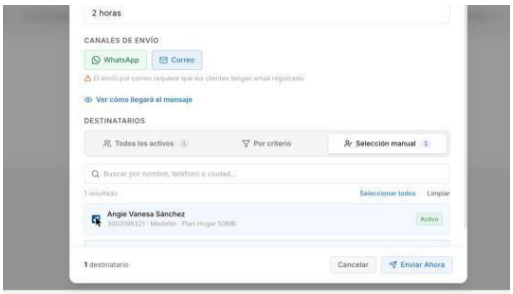


Ilustración 32 Paso 4: selección de canales (WhatsApp / correo)

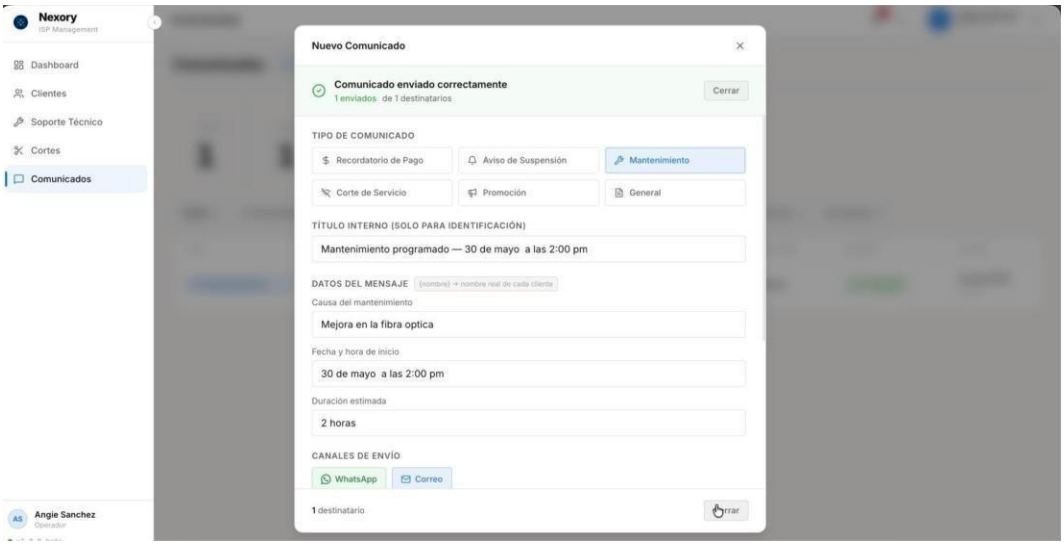


Ilustración 33 Paso 5: selección de destinatarios (todos, por plan o por etiqueta)

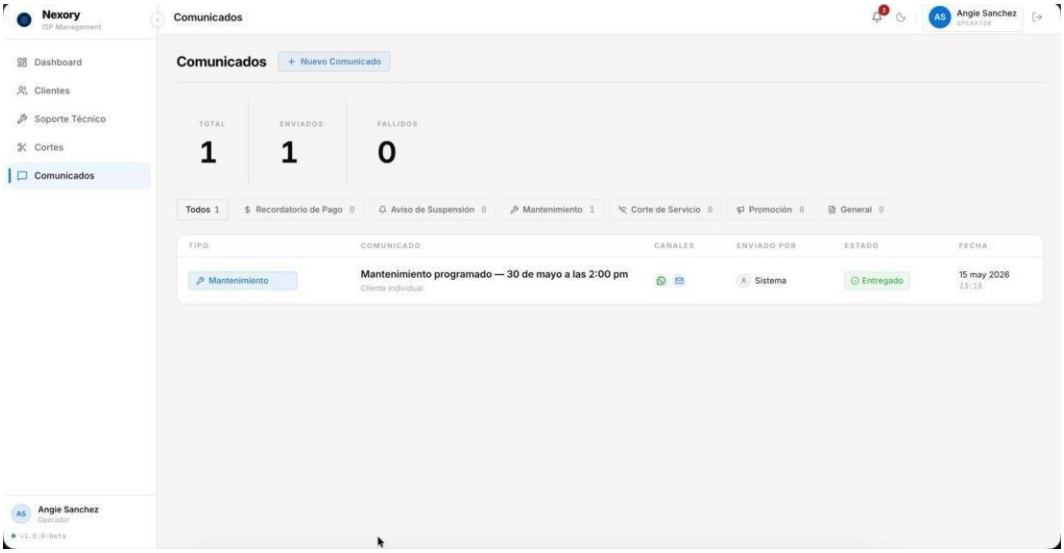


Ilustración 34 Comunicado publicado en la lista con detalle de envíos

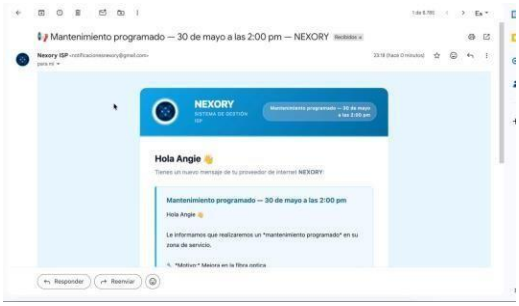


Ilustración 35 Comunicado recibido en el correo electrónico del cliente

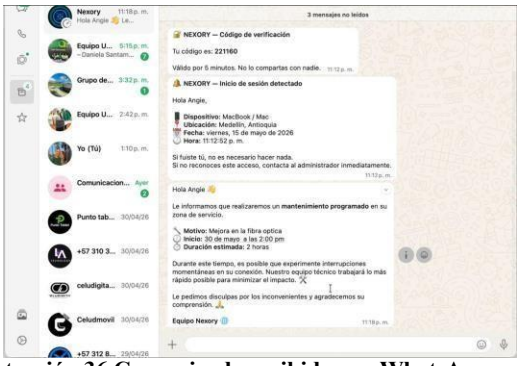


Ilustración 36 Comunicado recibido por WhatsApp

Panel de Notificaciones

Nexory incluye un panel de notificaciones en la barra superior que consolida todos los eventos recientes del sistema: nuevos tickets, cortes ejecutados, pagos registrados y comunicados enviados. Esto permite al administrador tener visibilidad de lo que está pasando sin tener que revisar cada módulo por separado.

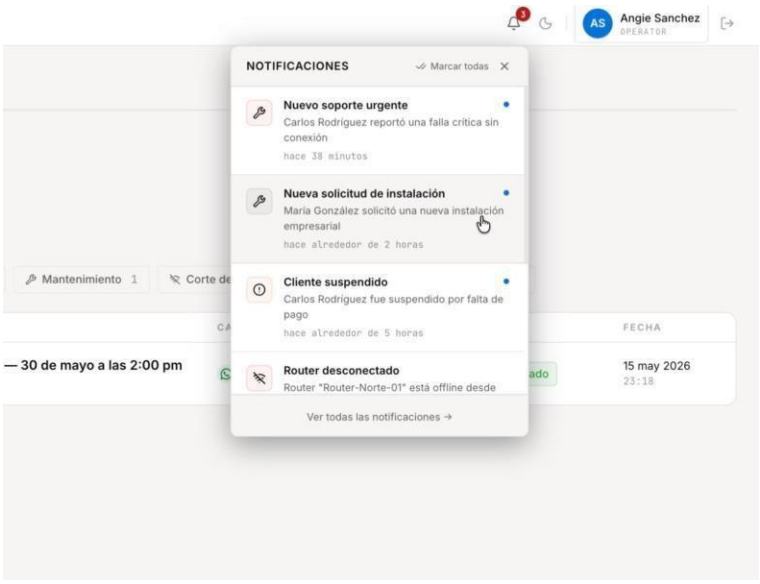


Ilustración 37 Panel de notificaciones del sistema

Discusión y Análisis del Sistema

Impacto del sistema

La implementación de Nexory generó un impacto altamente positivo en la operatividad de los micro ISPs. Los tiempos administrativos se redujeron exponencialmente, centralizando procesos dispersos que tomaban horas en conciliaciones manuales a tan solo unos clics de distancia.

Ventajas competitivas

El software elimina de raíz la dependencia de registros fragmentados en Excel y papel. Al basarse en componentes Open Source (como WAHA) y una arquitectura en la nube de bajo costo con planes básicos de Azure, reduce drásticamente las barreras de costo iniciales y operativas, evitando tarifas de las APIs corporativas oficiales de Meta.

Limitaciones identificadas

El sistema depende críticamente de una conexión continua de red hacia la nube y los routers MikroTik locales. Asimismo, existen desafíos de compatibilidad en RouterOS 7.x, debido al manejo de respuestas vacías estructuradas que requirieron el desarrollo de funciones personalizadas envolventes (como safeWrite) para evitar interrupciones de ejecución.

Comparación con soluciones comerciales

Frente a software propietario como SONAR, POWERCODE o ISPConfig, cuyos costos de licenciamiento anual son prohibitivos para microempresarios, Nexory ofrece las funciones esenciales necesarias (gestión de red PPPoE, soporte técnico, alertas automatizadas) con nulo

costo por licencia, adaptándose a la realidad económica local.

Aportes principales

El mayor aporte es la democratización y simplificación de la gestión técnica de red. Los empleados del ISP sin conocimientos avanzados en telecomunicaciones pueden realizar diagnósticos, dar de alta clientes y programar suspensiones sin tener que manipular directamente Winbox o comandos SSH.

Alcance del proyecto

El alcance de este proyecto comprende el diseño, desarrollo, despliegue en la nube y pruebas funcionales de Nexory, un sistema web centralizado para la administración de Proveedores de Servicios de Internet (ISP). El sistema abarca la implementación de módulos para la autenticación de doble factor (2FA), gestión integral de clientes, administración de tickets de soporte técnico, ejecución de cortes y reconexiones automáticas, y el envío de comunicados masivos. El alcance técnico se limita a la integración exclusiva con enrutadores de la marca MikroTik mediante su API de RouterOS. No contempla la configuración de hardware físico, tendido de redes ni la integración con equipos de enrutamiento de otros fabricantes. El sistema se entregará desplegado y funcional en la infraestructura de Microsoft Azure.

Integración MikroTik: alcance técnico

La integración con MikroTik fue una de las funcionalidades que generó mayor utilidad dentro de la operación del ISP. Antes de tener Nexory, ejecutar un corte de servicio por mora requería que el técnico abriera Winbox, buscara al cliente en la lista de PPPoE secrets, lo deshabilitara, revisara si tenía sesión activa y la terminara manualmente. Con Nexory ese proceso se reduce a un botón en la pantalla del cliente.

El diagnóstico completo de un cliente fue particularmente útil. Cuando un cliente llama diciendo que no tiene internet, el técnico puede abrir Nexory desde el celular y en segundos ver si la cuenta PPPoE existe, si está habilitada, si hay sesión activa y si está circulando tráfico. Esos datos aparecen en pantalla con indicadores de estado claros: IP asignada, tiempo conectado y consumo en KB, MB o GB según corresponda.

La compatibilidad con RouterOS 7 fue un reto específico. Esa versión devuelve una respuesta llamada !empty cuando una consulta no tiene resultados, que versiones anteriores simplemente devolvían vacío. La librería node-routeros lanzaba un error en ese caso, así que desarrollamos una función envolvente que detecta ese error específico y devuelve un arreglo vacío, de modo que el resto del sistema no necesita saber con qué versión de RouterOS está tratando.

También se agregó una consulta de métricas del router: carga de CPU, memoria disponible, tiempo encendido y versión de RouterOS instalada. Con eso, el administrador puede darse cuenta rápidamente si el equipo está al límite sin tener que abrir otro programa.

Desafíos técnicos durante la implementación

El despliegue en Azure enseñó más cosas de las que se esperaba. El problema de los binarios de Prisma fue el primero en aparecer: el binario que se genera en macOS con procesador Apple Silicon es completamente diferente al que necesita un servidor Linux en Azure, y eso solo se descubre cuando el servidor arranca y no encuentra el archivo correcto.

También aprendió que los entornos de desarrollo corporativos pueden bloquear el acceso al registro de npm, lo que complica flujos de trabajo que se dan por sentados. Y que Azure PostgreSQL, aunque tiene opción de pausa, se reinicia automáticamente después de siete días sin importar si está en pausa o no.

Conclusiones

Nexory demuestra que no es necesario un gran presupuesto ni un equipo amplio para desarrollar un sistema de gestión funcional. Con herramientas de código abierto y servicios básicos de Azure, el sistema quedó en producción y operativo.

La gestión de servicios como corte, restauración, diagnóstico y reinicio de sesiones desde Nexory elimina la necesidad de acceder a herramientas como Winbox o SSH, lo que permite a los técnicos resolver incidentes en menor tiempo y mejorar la atención a los usuarios finales.

El 2FA mediante WhatsApp facilitó su adopción, ya que el personal del ISP ya utiliza esta plataforma de forma habitual, evitando la necesidad de aplicaciones adicionales. Lo mismo ocurre con los comunicados masivos, que al estar integrados en el sistema permiten ahorrar tiempo y mantener un registro centralizado.

El despliegue en Azure representó una curva de aprendizaje importante, debido a problemas no documentados claramente o que solo aparecían bajo condiciones específicas. La resolución de estos inconvenientes fortaleció la comprensión del funcionamiento real de los entornos de producción en la nube.

Como trabajos futuros, se propone incorporar reportes con gráficas de ingresos y clientes por mes para mejorar el dashboard, así como desarrollar una versión móvil que facilite el uso del sistema por parte de los técnicos en campo.

Referencias

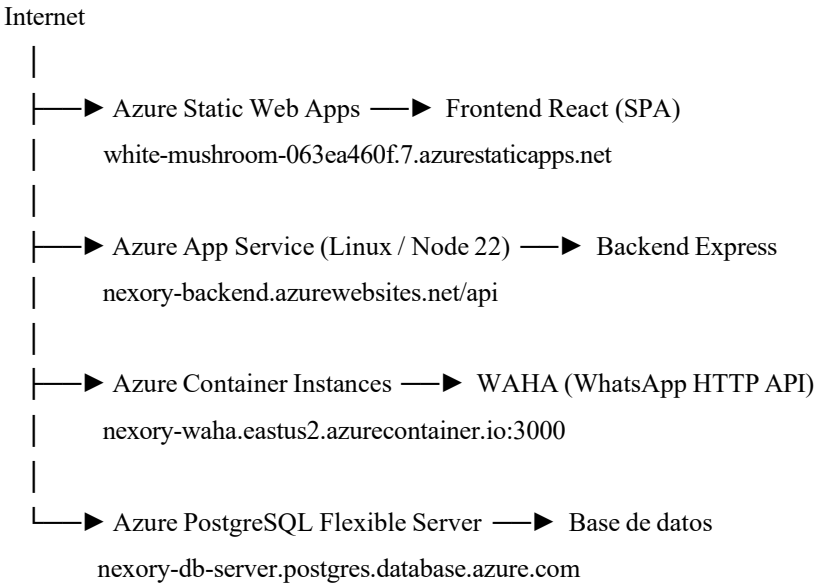
- Bucko, A., Vishi, K., Krasniqi, B., & Rexha, B. (2023). *Enhancing JWT authentication and authorization in web applications based on user behavior history*. Computers, 12(4), 78. https://doi.org/10.3390/computers12040078
- Muszyński, T., & Koziel, G. (2020). *A security analysis of authentication and authorization implemented in web applications based on the REST architecture*. Journal of Computer Sciences Institute, 16, 252–260. https://doi.org/10.35784/jcsi.1925
- Radivilova, T., Kirichenko, L., Pantelieiev, V., Mazepa, A., & Bilodid, V. (2024). *Analysis of authentication methods for full-stack applications and implementation of a web application with an integrated authentication system*. Innovative Technologies and Scientific Solutions for Industries, 3, 76–86. https://doi.org/10.30837/2522-9818.2024.3.076
- Gbenle, T. P., Abayomi, A. A., Uzoka, A. C., et al. (2022). *Applying OAuth2 and JWT protocols in securing distributed API gateways: Best practices and case review*. International Journal of Multidisciplinary Research and Growth Evaluation, 3(5), 628–634. https://doi.org/10.54660/IJMRGE.2022.3.5.628-634
- Shingala, K. (2019). *JSON Web Token (JWT) based client authentication in MQTT*. arXiv. https://arxiv.org/abs/1903.02895
- Ethelbert, O., Moghaddam, F. F., Wieder, P., & Yahyapour, R. (2017). *A JSON token-based authentication and access management schema for cloud SaaS applications*. arXiv. https://arxiv.org/abs/1710.08281
- Leila, M., Zitouni, A., & Djoudi, M. (2019). *A new framework of authentication over cloud computing*. arXiv. https://arxiv.org/abs/1902.00542

- Kovacevic, I., Stojkov, M., & Simic, M. (2024). *Authentication and identity management based on zero trust security model in micro-cloud environment*. arXiv. https://arxiv.org/abs/2410.21870
- Armbrust, M., et al. (2010). *A view of cloud computing*. Communications of the ACM, 53(4), 50–58. https://doi.org/10.1145/1721654.1721672
- Gholami, A., & Laure, E. (2016). *Security and privacy of sensitive data in cloud computing: A survey of recent developments*. arXiv. https://arxiv.org/abs/1601.01498
- Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.
- Laudon, K. C., & Laudon, J. P. (2020). *Management information systems: Managing the digital firm*. Pearson.
- Richardson, C. (2018). *Microservices patterns: With examples in Java*. Manning Publications.
- IETF. (2015). *RFC 7519: JSON Web Token (JWT)*. https://www.rfc-editor.org/rfc/rfc7519
- OWASP Foundation. (2021). *OWASP Top 10 Web Application Security Risks*. https://owasp.org/www-project-top-ten/
- Microsoft. (2024). *Azure App Service documentation*. https://learn.microsoft.com/en-us/azure/app-service/
- Microsoft. (2024). *Azure Database for PostgreSQL — Flexible Server documentation*. https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/
- Microsoft. (2024). *Azure Static Web Apps documentation*. https://learn.microsoft.com/en-us/azure/static-web-apps/
- MikroTik. (2023). *RouterOS API documentation*. https://help.mikrotik.com/docs/spaces/ROS/pages/47579160/API
- Prisma. (2024). *Prisma ORM documentation*. https://www.prisma.io/docs

- React. (2024). *React documentation*. https://react.dev/
- Vite. (2024). *Vite documentation*. https://vitejs.dev/
- Node.js. (2024). *Node.js documentation*. https://nodejs.org/en/docs
- TanStack. (2024). *TanStack Query documentation*. https://tanstack.com/query
- Zustand. (2024). *Zustand documentation*. https://zustand-demo.pmnd.rs
- WAHA. (2024). *WAHA — WhatsApp HTTP API*. https://waha.devlike.pro
- Ministerio de Tecnologías de la Información y las Comunicaciones [MinTIC]. (2024). *Reporte* de conectividad e inclusión digital — Colombia. https://www.mintic.gov.co/portal/inicio/
- Link de forms encuesta → https://forms.gle/pcVH5Ee7R2rePxYx8
- Link video demostrativo del proyecto https://canva.link/pxnwbk1f67h4kvb

Anexos

Anexo A. Arquitectura del sistema en Azure (producción)



Anexo B. Operaciones MikroTik implementadas

La siguiente tabla describe las funciones del servicio de integración RouterOS:

Tabla 2 Operaciones MikroTik de Nexory. Elaboración propia, 2026.

Función	Descripción
testConnection	Verifica conectividad con el router y retorna la versión de RouterOS.
createPppoeClient	Crea un usuario PPPoE en /ppp/secret, valida duplicados y cifra la contraseña con AES-256.
disableClient	Deshabilita la cuenta PPPoE y termina la sesión activa (ejecuta el corte).
enableClient	Habilita nuevamente la cuenta PPPoE (restaura el servicio).
listPppoeClients	Lista todos los usuarios PPPoE configurados con su perfil y estado.
listActiveSessions	Lista sesiones activas con IP, tiempo de conexión y bytes transferidos.
diagnoseClient	Diagnóstico simple: sesión activa, IP, uptime y consumo.
fullDiagnoseClient	Diagnóstico completo: cuenta existente, habilitada, sesión activa y tráfico circulante.
restartPppoeSession	Termina la sesión activa sin deshabilitar la cuenta (fuerza reconexión).

getRouterMetrics	Consulta CPU, memoria, uptime y versión de RouterOS del equipo.

Anexo C. URLs de producción

Tabla 3 URLs del sistema Nexory en producción. Elaboración propia, 2026.

Recurso	URL / Valor
Frontend	https://white-mushroom-063ea460f.7.azurestaticapps.net
Backend API	https://nexory-backend.azurewebsites.net/api
WAHA Dashboard	http://nexory-waha.eastus2.azurecontainer.io:3000/dashboard
PostgreSQL Host	nexory-db-server.postgres.database.azure.com
Container Registry	nexoryacrdev.azurecr.io
Grupo de recursos	nexory-rg (East US 2)

Anexo D. Video demostrativo del sistema Nexory

Con el fin de complementar la presentación del proyecto, se dispone de un video demostrativo en el que se explica la arquitectura, el funcionamiento y los principales módulos implementados en el sistema Nexory.

Video del proyecto:

https://canva.link/pxnwbk1f67h4kvb