PASANTÍA EMPRESARIAL

Monografía de pasantía empresarial en HV Multiplay

Sistema de automatización de registro de direcciones IP para la empresa HV Multiplay

Por:

Estefanía González Díaz

FACULTAD DE INGENIERÍA DE TELECOMUNICACIONES

UNIVERSIDAD SANTO TOMÁS

BOGOTÁ D.C

2025

PASANTIA EMPRESARIAL

Monografía de pasantía empresarial en HV Televisión

Sistema de automatización de registro de direcciones IP para la empresa HV Multiplay

Por:

Estefanía González Díaz

Dirigido por:

Ing. Carlos Enrique Montenegro Narváez

FACULTAD DE INGENIERÍA DE TELECOMUNICACIONES

UNIVERSIDAD SANTO TOMÁS

BOGOTÁ D.C

2025

Nota de aceptación

_____
_____
_____
_____

_____
Decano

_____
Tutor Universidad

_____
Evaluador

DEDICATORIA

A mi padre, Gustavo González Giraldo y a mi madre, Zohe Gudiela Díaz de Gonzalez, por ser mi mayor inspiración, por el amor, las enseñanzas y dedicación que me han brindado a lo largo de mi vida. Por sus palabras de aliento y su constante insistencia para poder culminar con mis estudios y poder culminar con mis estudios. A esas personas que no se rindieron y me exigieron, que estuvieron en los momentos más importantes y a la vez difíciles para que pudiera cumplir con éxito todo lo que me he propuesto hasta el momento.

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

TABLA

ANEXOS

Introducción

Actualmente en esta era digital, la seguridad informática se ha vuelto relevante en todos los entornos tecnológicos y se ha convertido en un pilar fundamental para todas las empresas, cuya seguridad es prioridad para ellas. Teniendo en cuenta esto, cuando se habla de direcciones IP se debe tener una identificación correcta del origen de la misma con el fin de reducir en su mayoría todos los ataques informáticos que se puedan presentar por las diferentes actividades realizadas en dichas empresas, siendo como una de sus prioridades el reducir los riesgos de pérdidas de datos informáticos. En los últimos tiempos se ha visto en una mayor cantidad estos ataques cibernéticos contra grandes empresas a nivel mundial y a nivel local, se han vuelto una prioridad el defenderse contra vulneraciones de seguridad en los sistemas internos de las empresas. El avance de la tecnología ha permitido que estos ataques sean más sofisticados y difíciles de contrarrestar.

Esta monografía tiene como objetivo el desarrollar un sistema de automatización en Python para la detectar y guardar el origen de una o más direcciones en un archivo plano. Como solución se planteó una aplicación en la cual se pueda colocar el archivo del cual se desea saber el origen de las direcciones IP y posteriormente guardar en el equipo de la persona. Como resultado el empleado puede tomar varias direcciones IP en cualquiera de sus versiones, en archivos ya sea Excel o texto y posteriormente guardar los resultados en su equipo, para el ingreso de esta aplicación se debe de tener credenciales que son creadas con anterioridad, facilitando la labor de cada empleado y mitigando la pérdida de tiempo aprovechándolo en otras acciones o deberes de mayor importancia beneficiando a la empresa.

En el capítulo 1 se describe la problemática que tiene la empresa a raíz de una llegada masiva de direcciones IP, podemos ver la justificación donde se explica la idea del proyecto que se tiene y se muestra algunos casos reales en cuanto a seguridad informática, también se hace énfasis en las políticas que tiene Colombia para ayudar a mitigar este tipo de inconvenientes de seguridad y los objetivos principales de la monografía.

En el capítulo 2 podemos ver, a grandes rasgos, la definición de lo que significa una dirección IP y como están conformadas para su correcto funcionamiento. Se resalta como las empresas han gestionado diferentes métodos para mitigar en lo posible estos ataques cibernéticos dentro de su infraestructura.

En el capítulo 3 plantea la solución del problema paso a paso y se logra desarrollar una herramienta para el usuario con la cual contrarrestar y solucionar este problema de seguridad informática.

En el capítulo 4 muestra la conclusión que da solución a los objetivos que se plantearon al principio del proyecto, mostrando resultados a cada uno de los problemas planteados y como esta aplican fue de utilidad para los trabajadores de la empresa.

1. SISTEMA DE AUTOMATIZACIÓN DE REGISTRO DE DIRECCIONES IP PARA LA EMPRESA HV MULTIPLAY

1.1 Planteamiento Del Problema

En la era actual, la seguridad de la información en las conexiones es algo importante, así como es fundamental cumplir con las normas nacionales e internacionales de protección de datos sensibles, y trabajar en la prevención de actividades maliciosas los cuales son una prioridad en los entornos tecnológicos; esto para que los clientes y empleados de una empresa estén más protegidos. Si se proporciona la correcta identificación de la fuente de las direcciones IP se puede saber el origen, el proveedor de servicio y otros datos que serían de gran utilidad en la mitigación de riesgos informáticos.

Se ha observado que en la empresa HV Multiplay, desde hace varios meses se han estado presentando numerosas solicitudes provenientes de direcciones IP desconocidas hacia su servidor. Esto ha generado una alerta, motivada por la necesidad de identificar el origen de dichas direcciones. Sin embargo, actualmente este proceso se realiza de forma manual, lo que ha dificultado el análisis y ha incrementado la cantidad de solicitudes cuya procedencia es incierta. Este escenario representa un riesgo para posibles ataques cibernéticos, interrupciones en el servicio, conectividad, tiempos de inactividad en el servicio, surgiendo como respuesta a una necesidad manifiesta por parte de unos clientes. La necesidad manifiesta por parte de un cliente, el cual desea saber el origen de estas direcciones IP que han llegado al servidor. Esta actividad en la actualidad se realiza de forma manual, las personas que se encargan de esto son las áreas

encargadas ya sea el NOC o el área de aprovisionamiento de datos; pero este trabajo se completa en aproximadamente 2 meses ya que en muchos casos son más de 200.000 direcciones IP ya que la consulta se realiza una a una y se encuentran bloqueos por parte de las páginas donde se está realizando el trabajo, generando que el empleado se desenfoque de sus tareas.

Esta situación se produce porque no existe un sistema de monitoreo que permita identificar las direcciones IP, e identificar si la solicitud es una petición real o es una que quiera hacer un ataque. la ausencia de otra aplicación para detección de direcciones IP; permite que no sean monitoreadas de una manera apropiada para saber su origen. la organización no cuenta con un sistema de monitoreo de origen de las direcciones IP, esto puede ser causado por diversos elementos como son el uso de proxy o VPNs, esto porque los usuarios o sistemas están utilizando una red privada virtual (VPN) o un servidor proxy, otro factor que se puede ver son las IP compartidas o dinámicas , son las que su dirección IP puede estar en múltiples dispositivos, ataques cibernéticos como el escaneo de puertos, ataques de fuerza bruta que es el intento de acceder a servidores por medio de combinaciones de usuarios y contraseña, o también un caso muy conocido el DDoS (Denegación de servicio distribuida) que es la llegada masiva de tráfico de manera intencional para saturar un servicio. Los casos mencionados con anterioridad son las causas o condiciones más probables para este tiempo de problema.

A continuación, veremos varias soluciones en cuanto a ciberseguridad en el área jurídica que han ocurrido en Colombia en los últimos años:

ley 1273 de 2009 “Por medio de la cual se modifica el Código Penal, se crea un

nuevo bien jurídico tutelado - denominado "de la protección de la información y de los datos"- y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones" (Funcion Publica, 2025), Esto quiere decir que establece medidas para proteger los sistemas informáticos, las telecomunicaciones y la información , en contra de delitos informáticos , estableciendo así sanciones para quienes cometan este delito con el fin de asegurar la integridad de los datos y la seguridad en las redes digitales. Las sanciones que se generen al momento que se incumpla alguna de las condiciones escritas en la ley.

Ley 1581 de 2012 de protección de datos personales: es una ley que complementa la regulación vigente para la protección del derecho fundamental que tienen todas las personas naturales a autorizar la información personal que es almacenada en bases de datos o archivo, así como su posterior actualización y rectificación. Esta ley se aplica a las bases de datos o archivos que contengan datos personales de personas naturales. (Imsaslud, 2025)

Las anteriores dos leyes nos permiten tener un margen de privacidad con los datos que están en la web, esto quiere decir que las empresas tienen el deber de salvaguardar todo tipo de información que de una u otra forma puede afectar a alguna persona y entidad.

¿Cuál es el sistema de monitoreo adecuado para realizar una identificación del origen de las direcciones IP?

1.2 Justificación

El presente proyecto surge, así como el cliente anterior otros clientes se pueden hacer la misma pregunta y los empleados de la empresa requieren saber de dónde proviene el ingreso masivo de direcciones IP que llegan a su servidor, esta actividad se puede automatizar con el desarrollo de una aplicación que ayudará a disminuir el tiempo de búsqueda notablemente a la hora al ver el origen de dichas direcciones. Este proyecto permitirá automatizar de forma práctica y funcional dicho proceso, enfocando que el empleado coloque mayor prioridad en otros asuntos de mayor importancia para la empresa

Como casos de éxito en cuanto a la implementación en la implementación de aplicaciones de monitoreo podemos observar las siguientes empresas que vieron la ciberseguridad como una necesidad para mantener a sus clientes cubiertos frente a cualquier eventualidad.

Evolutio aspira a ser el guardián de la ciberseguridad (El pais, 2025):

Evolutio, la cual es una empresa dedicada a servicios de tecnología, ha robustecido su posición después de la compra de empresas como lo son: Warpcom, Dagram, Plusnet y Securnet. Esto ha dado la oportunidad a la empresa de impulsar las habilidades de tráfico, seguridad en la red y defensa para Ip’s corporativas. Las mejoras mencionadas con anterioridad han dado pie para mejorar otros servicios como lo es la detección de amenazas y la mejora de su infraestructura para Europa.

Artículo 1 DECRETO 338 de 2022

“Artículo 2.2.21.1.3.9. Puestos de Mando Unificado de Seguridad Digital. Instancia

de colaboración y coordinación interinstitucional que tiene como objetivo articular y facilitar la toma de decisiones estratégicas y operaciones necesarias, para prevenir o gestionar incidentes cibernéticos sobre las infraestructuras críticas los servicios esenciales, y que permiten la garantía de los derechos ciudadanos cuando actúan en el ciberespacio. Las autoridades que intervengan en los puestos de mando unificado lo harán para el cumplimiento coordinado de las funciones que señala la constitución, la ley, y bajo el respeto y protección de los derechos humanos." (Sistema único de información normativa, 2025)

Artículo 2.2.21.1.4.3. Obligaciones de seguridad de las autoridades titulares de infraestructura crítica, o que presten servicios esenciales. Las autoridades, definidos como titulares de infraestructura crítica o que presten servicios esenciales, propenderán por contar con un plan de seguridad digital, protección de las redes, las infraestructuras críticas cibernéticas, los servicios esenciales y los sistemas de información en el ciberespacio y deberán hacer periódicamente una evaluación del riesgo de seguridad digital. Para lo anterior, deben contar con normas, políticas, procedimientos, recursos técnicos, administrativos y humanos necesarios para gestionar efectivamente el riesgo, y en cumplimiento de las mejores prácticas y estándares que le sean exigibles.

Artículo 2.2.21.1.4.4. Afectación significativa. Para los efectos del presente Título, se entenderá por afectación significativa, aquella que se ocasiona a las Infraestructuras críticas cibernéticas, servicios esenciales e intereses nacionales para la seguridad digital, protección de las redes, de las infraestructuras, y los sistemas de información en el ciberespacio. (Sistema Único de información normativa, 2015)

El Ministerio de Tecnologías de la información y las Comunicaciones determinará los umbrales y variables cualitativas o cuantitativas de una afectación significativa, teniendo en cuenta los siguientes factores:

- **RESOLUCIÓN NÚMERO 00500 DE MARZO 10 DE 2021** (Sistema único de información normativa, 2025)**:**

ARTÍCULO 6. La gestión de la seguridad de la información, seguridad digital y la gestión de riesgos de la entidad. Los sujetos obligados deben determinar e implementar controles para mitigar los riesgos que pudieran afectar la seguridad digital y física de acuerdo con el resultado del análisis y evaluación de riesgos y cumplir con las siguientes características y responsabilidades:

- Estar al tanto de las nuevas modalidades de ciberataques que pudieran llegar a afectar a la entidad, según las políticas que establezca la entidad de acuerdo con su evaluación de riesgo y atendiendo criterios de razonabilidad.
- Establecer las capacitaciones que recibirán los funcionarios de la entidad en temas relacionados con seguridad digital y mantenerlos actualizados sobre las nuevas amenazas cibernéticas

ARTÍCULO 17. Etapas generales de la gestión de incidentes de seguridad digital. Los sujetos obligados deben incluir en su estrategia de seguridad digital las actividades a realizar en las etapas de prevención; protección y detección; respuesta y comunicación; recuperación y aprendizaje, como mínimo deberán incorporar.

Los anteriores artículos resaltan la importancia de darle una solución a la problemática de la ciberseguridad, ya que esto podría generar inconvenientes tales como la inconformidad

de los clientes. Las personas involucradas deberán estar en constante actualización de datos e información de las nuevas modalidades de ataques que podría presentar la empresa para que de esta forma puedan tener una respuesta correcta y rápida frente a incidentes.

1.3 Objetivos

Objetivos Generales

- Desarrollar un sistema de automatización en Python para detectar y guardar el origen de una o más direcciones IP en un archivo plano, con el fin de minimizar o mitigar los riesgos de ataques cibernéticos para diversas empresas.

Objetivos Específicos

- Identificar los requerimientos que debe tener la aplicación de acuerdo con las necesidades del cliente para generar una solución al problema
- Realizar un diagrama de casos de uso con las características que se identificaron para ofrecer una solución de automatización.
- Implementar la aplicación por medio de la interfaz gráfica de Python, para generar y escanear los archivos e identificar las direcciones IP.
- Desarrollar pruebas para ver la confiabilidad de la aplicación y que esta cumpla con los requisitos.

2. Marco Referencial

2.1 Marco Conceptual

2.1.1 Direccionamiento IP (Marino, 2025):

Es un método el cual permite asignarle una dirección IP a unos o varios equipos que se encuentren dentro de una misma red, este tipo de direcciones son únicas y se asignan a los dispositivos de la red a la cual pertenecen, se encuentran dos tipos de direcciones IP como son la IPv4 e IPv6 ambas tienen parámetros diferentes para su correcto funcionamiento.

2.1.2 Notación para IPv4 (Guevara, 2025):

Este tipo de notación se trabaja con números binarios o decimales. Los bits que se encuentran en el fragmento de red deberán coincidir con los que están en cada uno de los dispositivos de la red a la cual va a pertenecer estos deben mantener una secuencia de 32 bits. Como se puede observar en la tabla 1 la notación decimal siempre tiene el formato dependiendo de cual clase se este manejando en la red, los bloques van a estar separados por puntos y dentro de cada antes de cada punto los valores pueden variar de 1 – 255

Notación decimal (la más conocida)	192.168.3.120
Notación binaria	11000000.10101000.00000011.01111000
Notación hexadecimal	C0 A8 03 78

Tabla 1 Notación (Castillo, 2019)

2.1.3 Notación para IPv6 (Byspel, 2025):

Este tipo de direcciones se manejan principalmente de forma hexadecimal esto quiere decir que maneja números y letras, de la siguiente manera números de 0 al 9 y letras de la A a la F, esto permite que se puedan manejar grandes volúmenes de valores de forma mas compacta. Las direcciones IPv6 manejan 128 bits los cuales son divididos en 8 segmentos donde cada uno de estos tiene 16 bits. Este tipo de notación se esta utilizando con mas frecuencia en la actualidad ya que las direcciones IPv4 están llegando a su limite de utilización. Como se puede observar en la tabla dos, esta seria la forma correcta de realizar una dirección IPv6,

Notación	**2010:DB92:AC32:FA10:00AA:1254:A03D:CC49**

Tabla 2 Notación IPv6 (Castillo, 2019)

2.1.4 Clases de direcciones IP

Clase	Rango de direcciones	Enmascaramiento de subred	IP de ejemplo	Bits principales	Número máximo de redes	Solicitud
IP Clase A	1 a 127	255.0.0.0	1.1.1.1	8	128	Se utiliza para una gran cantidad de hosts.
IP Clase B	128 a 191	255.255.0.0	128.1.1.1	16	16384	Utilizado para redes de tamaño mediano.
IP Clase C	192 a 223	255.255.255.0	192.1.11.	24	2097157	Utilizado para red de área local.
IP Clase D	224 a 239	NA	NA	NA	NA	Reserva para multitarea.
IP Clase E	240 a 254	NA	NA	NA	NA	Esta clase está reservada para fines de investigación y desarrollo.

Ilustración 1 Clases de direcciones en IPV4 (Leo, 2025)

- **Clase A (Meridianoutpost, 2025)**

Este tipo de direcciones están implementadas para grandes infraestructuras de red con muchos equipos, esta clase permite la utilización de 126 las cuales utilizan el primer octeto de palabras de izquierda a derecho los cuales él es octeto de red ID y los siete bits restantes serán los hosts.

- **Clase B (cecomart, 2025):**

Este tipo de direcciones tienen los dos primeros octetes como de red, mientras que los dos últimos sean disponibles para ser configurados a un equipo, son redes de tamaño mediano, se utilizan para universidades o pequeñas organizaciones del gobierno.

- **Clase C (Larus, 2025) :**

Las direcciones de este tipo son mas que todo utilizadas para redes pequeñas o locales, como se puede observar en la ilustración 1, estas redes tienen la posibilidad de convertirse en subredes más pequeñas esto es ideal para mejorar la seguridad y optimizar el trafico dentro de la red. Dentro de sus múltiples redes separadas pueden tener 254 dispositivos conectados.

- **Clase D (vpn unlimited., 2025):**

Se utilizan para comunicaciones multicast, esta clase de direcciones son frecuentemente utilizadas para transmisión de audio y video, no posee mascara de red ya que no son utilizadas en redes tradicionales.

- **Clase E:**

Como se puede observar en la ilustración 1, estas direcciones son utilizadas exclusivamente para investigaciones o para desarrollo.

2.1.5 Diferencias entre IPv4 e IPV6

2.1.5.1 *Tamaño de la dirección IP:*

IPV4: 32 bits.

IPv6: 128 bits.

2.1.5.2 Direccionamiento:

IPv4: Número y bits binarios separados por punto.

IPv6: Alfanuméricos y con bits binarios separados por dos puntos (:).

2.1.5.3 Clases de direcciones IP:

IPv4: Cinco clases de direcciones IP diferentes.

IPv6: número ilimitado de direcciones IP incluyendo las de rango privado.

2.1.5.4 *Configuración:*

IPv4: Cada sistema deberá ser configurado para poder conectarse con otros, la red. Se podría configurar con DHCP o de manera manual.

IPv6: Configuración opcional según lo que se necesite y también soporta la autoconfiguración entre equipos que tengan IPV6.

2.1.5.5 Interoperabilidad:

IPv4: Topologías, capacidad y movilidad limitada**s.**

IPv6: Interoperabilidad y movilidad incluidas en los dispositivos de red.

2.1.6 Qué es una subred (Coudflare, 2025)

Son redes dentro de otras redes, lo cual hacen que al reducir su tamaño en porciones mas pequeñas mejoren el trafico y pueda dar un beneficio que es el de evitar los router que se encuentre la red y proceda a realizar recorridos innecesarios.

2.2 PRINCIPALES PÁGINAS PARA PODER VER EL ORIGEN DE LAS DIRECCIONES IP SEGÚN SU UBICACIÓN GEOGRÁFICA

“Las direcciones IPv4 e IPv6 generalmente se asignan de manera jerárquica. A los usuarios se les asignan direcciones IP mediante: Proveedores de servicios de Internet (ISP). Los ISP tienen asignaciones de direcciones IP de un registro de Internet (LIR) local o Registro Nacional de Internet (NIR), o de su Registro Regional de Internet (RIR) correspondiente.**”** (IANA, 2025)

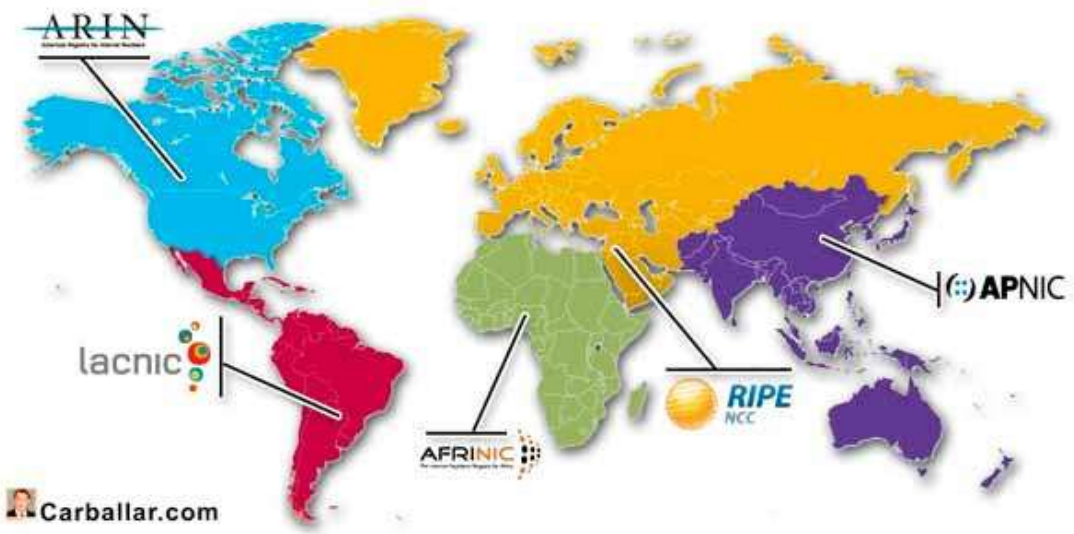


Ilustración 2 Organización de recursos numéricos (carballar, 2025)

La función principal que tiene la IANA es la de asignar grupos a direcciones que no se hicieron en el Registro Regional de direcciones de internet, pero todo bajo los lineamientos de la política global, esto dependiendo del sitio donde se encuentre encuentra

o se va a asignar la dirección como se puede ver en la imagen anterior.

2.2.1 Prácticas de buenas configuraciones

Al momento de tener grandes volúmenes de direcciones IP y las subredes que la componen, es recomendable que se realicen las siguientes pautas para poder saber cómo funciona la red general de una empresa y de esta manera los ingenieros que administran dichas direcciones tengan claridad a la hora de asignar a los clientes sus propios segmentos de red para sus servicios:

- **Planificación de direcciones IP.** En este caso es recomendable tener un total o parcial conocimiento sobre las direcciones y segmentos de red que se le tiene a cada clientes para que de esta forma no se presenten conflictos, garantizar
- **Documentación y monitoreo:** es importante que a la empresa se le haga un mantenimiento continuo y por ende realizar toda la documentación correspondiente a los cambios que se realice en la infraestructura de la empresa, para que todos los empleados del área correspondiente tengan conocimiento de los equipos que hacen parte de la empresa
- **Medidas de seguridad:** es de vital importancia que en las empresas se creen un esquema de medidas de seguridad, para poder protegerse de ataques como: ataques Man-in-the-middle, escaneo de puertos o ataques a los servidores y redes de infraestructura, entre otros

2.2.2 Marco Referencial

2.2.2.1 Actividades Realizadas

A continuación, se describen las actividades que se realizaron en el área de aprovisionamiento de datos de la empresa HV Multiplay.

Actividad	Descripción de Actividad
Monitoreo de sistema Cactus, Zabbix	Se monitorea cada uno de los sectores donde se encuentra el servicio que proporciona la empresa. Se observaban las caídas, los SNR en los puntos troncales
Activación de tv (Sitel)	A través de las direcciones ip se ingresan con sus respectivas credenciales y posteriormente se da la activación, esto por medio de la página del proveedor Sitel, los pasos serian de la siguiente forma 1. Call center recibe la llamada del cliente para la activación 2. Se guarda en un archivo en la nube para de estar forma sepan cual es el servicio que se está activando y cual no.

Actualización ONUS	Se recibieron varios equipos para poder actualizar por medio de una USB booteable, de esta forma se procede a clasificar cuales están en buen estado, o cuales; para dependiendo de su condición enviar al laboratorio o al sitio donde se encuentra el técnico de la empresa.
Actualización plataformas diagramas	Se entra a la aplicación lucid para plasmar la red que se encuentra en el papel, para que de esta forma cualquier persona de la empresa tenga acceso a la red (ya sea el área de aprovisionamiento o al de NOC).
Identificación de direcciones IP	Se realizó un programa en Python por medio de una interfaz gráfica para saber a quién pertenece una dirección IP ya sea en versión IPv4 o IPc6. Mas adelante se explicará como es el funcionamiento del programa-

Tabla 3 Actividades para Hv Multiplay

3. Funcionamiento De La Aplicación

Esta aplicación tiene como objetivo la búsqueda del origen de direcciones IP que se generan en un archivo de texto desde el servidor, ordena la información por columnas y almacena dichas direcciones en un archivo plano.

Para este proyecto se buscó solucionar una actividad que, en la actualidad, la empresa

está haciendo de manera manual por medio de la automatización, con el fin de acabar la tediosa tarea de tener que registrar las IP de manera manual en el archivo de texto original que se descarga desde el servidor, ya que al ingresar cierta cantidad de direcciones esta se bloquea, al tener demasiadas solicitudes desde una misma IP a las páginas web tales como Lacnic, Apnic, WHOIS etc, llegando al límite de recolección de datos diarios desde una misma IP y esto genera un retraso en las tareas cotidianas de la empresa, posibles riesgo en cuanto a la caída de un nodos y pérdida del servicio en una o más troncales. Esta aplicación busca disminuir el tiempo de recopilación de datos, saber el origen de la dirección IP con exactitud y de esta manera cubrir una necesidad básica tanto de la empresa como de los empleados.

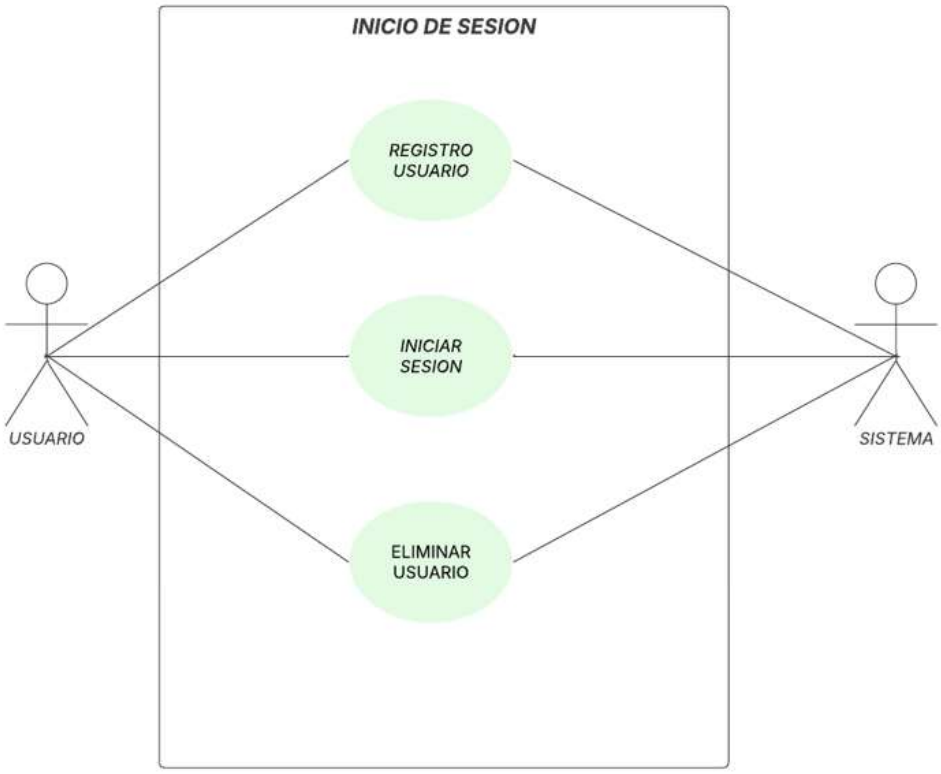


Ilustración 3 Diagramas de caso de uso inicio de sesión

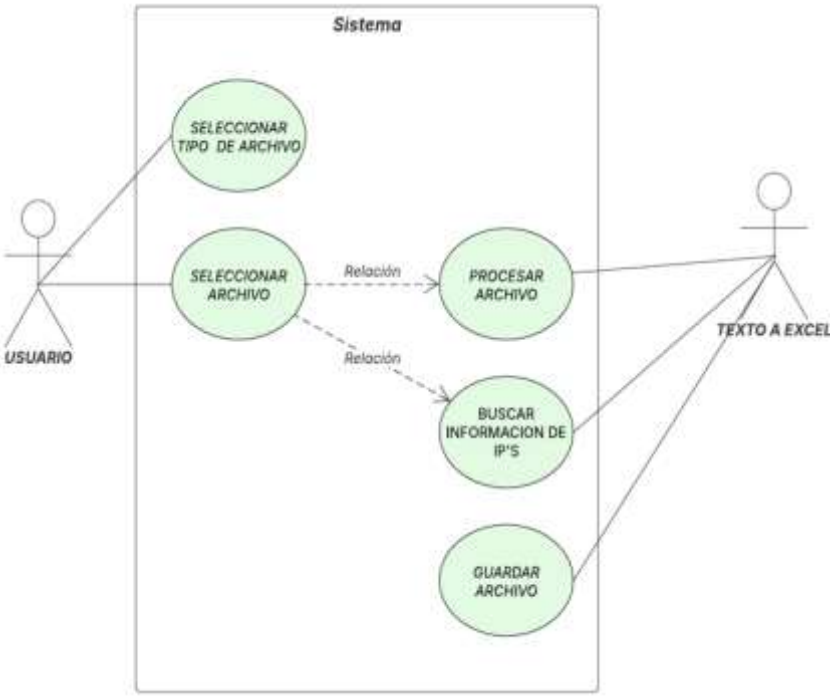


Ilustración 4 Diagramas de caso de uso aplicación principal

En la ilustración 3 se puede observar la interacción que va tener el usuario por medio de la interfaz gráfica en donde se puede registrar o iniciar sesión y si es necesario eliminar un usuario que ya no se encuentre en la empresa, la información se guarda en una base de datos en el computador, mientras que en la ilustración 4 nos presenta como se puede seleccionar el tipo de archivo, archivo del cual se quiere saber el origen de las direcciones IP, en la parte derecha muestra cómo se procesa el archivo de las IP y del archivo que se selecciona se busca la información, como último paso se guarda el archivo en la ruta de elección del usuario final. Esta aplicación tiene dos módulos principales los cuales se muestran a continuación:

3.1.1 Módulo de registro:

En la siguiente imagen, se puede observar las distintas interacciones que puede

hacer el usuario, como digitación del usuario, contraseña, inicio de sesión y eliminación de usuario, todo esto almacenados en una base de datos como se observa en la ilustración 5.

Ilustración 5 Ventana inicio

Al intentar acceder sin tener un usuario previamente registrado en el sistema, la aplicación permitirá la opción de realizar un nuevo registro. En la interfaz se observa una ventana de apoyo con la cual el usuario podrá realizar el registro como podemos ver en la ilustración 6. El nuevo usuario quedará guardado en la base de datos.

Ilustración 6 Ventana aviso de" USUARIO NO ENCONTRADO"

En la ilustración 7 podemos evidenciar la forma en la que el empleado puede realizar el registro, el cual consiste en dos label que permitirá generar un usuario y una contraseña. Esta información será agregada a la base de datos, con el fin de dar ingreso al programa inicial.

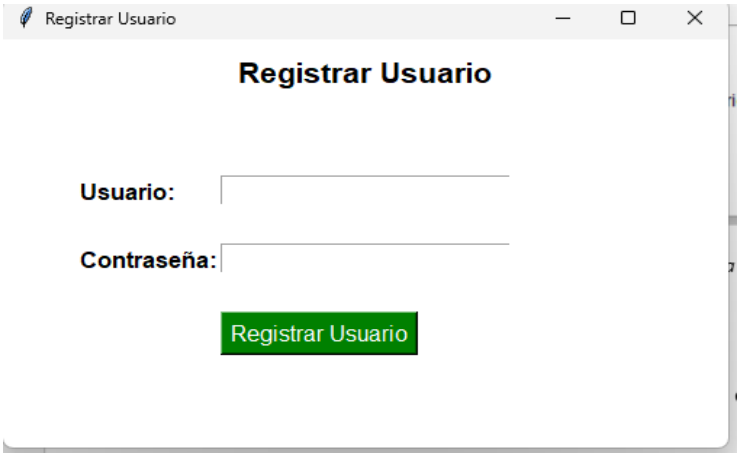


Ilustración 7 Ventana aviso de" Registro de usuario"

En la imagen a continuación se explica como el módulo permitirá gestionar las operaciones de creación, eliminación e inicio de sesión del usuario, detallando el paso a paso que la persona debe realizar, tal y como se muestra en la Diagrama de flujo de ventana inicial (**Anexo A**).

3.1.2 EXPLICACION ANEXO A (ver Anexo A)

Se empieza a ejecutar el programa, mientras tanto se conecta a la base de datos que se crea automáticamente cuando se registra el primer usuario, se muestra la ventana principal y después se espera la interacción que va a realizar el usuario.

Cuando el usuario inicia sesión la ventana muestra que necesita ingresar el usuario y contraseña si deja campos vacíos esta mostrara un error que no se llenó ningún dato, si

no deja espacios en blancos, primero se busca si el usuario este o no en la base de datos,

si el usuario no se encuentra dentro de la base de datos saldrá una ventana emergente que dirá si desea registrarse si desea registrarse se abrirá una nueva ventana que le preguntara sus credenciales para que en ese o en otro momento pueda ingresar a la aplicación, si acepta el registro del nuevo usuario se enviaran los datos a la base de datos y validara si tiene un usuario o si no, de lo contrario cerrara .si está dentro de la base de datos que se creó se dará el mensaje de bienvenida ,cuando se realiza el ingreso de la persona se cerrara la ventana y se procederá a ejecutar el programa principal.

3.1.3 Módulo de ejecución:

3.1.3.1 Módulo de inicio y selección de archivo.

En este primer módulo se quiere dar a conocer la interfaz gráfica, la cual va a ser visible para el usuario. En dicha interfaz será posible seleccionar el formato del archivo que se quiere buscar ya sea Excel, texto o CSV, posterior a esto se selecciona el archivo, luego se debe escribir el nombre de la columna en la que se encuentran los datos del mismo y finalmente activar el motor de búsqueda para las direcciones IP señalado en el botón que dice “procesar archivo “, como podemos ver en la ilustración 8.

En la ilustración 9 muestra cómo se verá dicha acción usando un explorador de Windows que podemos encontrar en el computador.

Ilustración 8 Interfaz de usuario

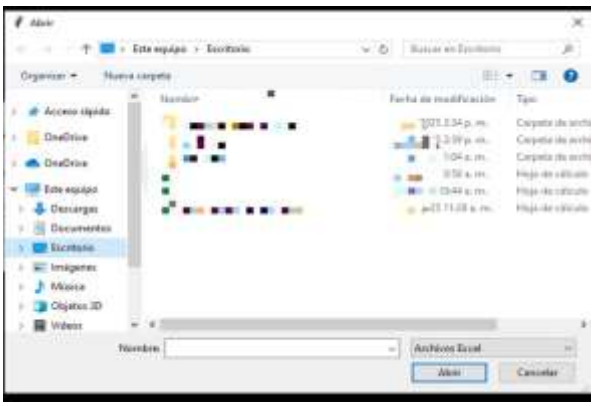


Ilustración 9 Explorador de Windows

En esta ilustración se muestra el funcionamiento de la aplicación, paso a paso se muestra las acciones e interacción que realiza el usuario con esta aplicación, desde el inicio de la interfaz, continuando con el archivo seleccionado y como el aplicativo reacciona ante la búsqueda de este. También podemos ver que la aplicación asimila la existencia de estos archivos dentro del computador continuando con el proceso de identificación reconociendo el tipo de archivo o por lo contrario, si el archivo es inexistente por lo tanto dará un error como lo muestra. Para comprender el proceso de ejecución del módulo de inicio y selección de archivo (ver el **Anexo B).**

3.1.3.2 Módulo ejecutar:

Como se puede observar en la ilustración 14 vemos el procesamiento interno que realiza el sistema luego de la selección del archivo y de colocar el nombre de la columna, a continuación, la aplicación reconoce los registros de las direcciones IP seleccionándolos en grupos no mayores a 10.000 y convirtiéndolos como consecuencia en

archivos temporales, esto se realiza para que el documento pueda tener un poco más de control para las consultas que realiza, facilitando su automatización. Esta aplicación buscara las diferentes direcciones IP en cualquiera de sus versiones, y esto se realiza mediante la búsqueda por medio de una librería que se utiliza en el archivo la cual es IPWhois.

IPWhois se basa en RDAP (permite consulta de dominios, direcciones IP y sistemas autónomos) para recuperar y analizar datos de las diferentes direcciones y nos ayuda a obtener información de la propiedad de la IP , estas consultas se realizan de una en una y se van organizando de manera continua en el documento final, para que el usuario pueda ver el origen de la dirección IP que está buscando y de esta forma poder organizarlo en un archivo Excel para que se vea más ordenado y comprensible para el usuario.

3.1.4 EXPLICACION ANEXO B (ver Anexo B)

Al momento de ser enviado a la interfaz del proyecto final, saldrá una ventana que tiene 4 ítems los cuales son mostrar la opción de que tipo de archivo quiere procesar (Excel, texto o csv) y para saber cuál es el archivo (esto lo selecciona el usuario), también se pedirá el encabezado de la columna que quiere que se extraiga, si se coloca un archivo que no es mostrara un error, de lo contrario verificara la existencia del archivo, si el archivo es encontrado se convertirá en un archivo Excel para que sea más fácil la lectura, se generara una doble columna en el archivo, si la conversión del archivo es fallida se procesara cerrar el programa. Cuando el programa tiene más de 1000 registros se dividen en archivos temporales de lo contrario se buscará la columna que se escribió desde un inicio, se duplicara de nuevo la información del origen de las direcciones y al final del documento agregar estas dos columnas que es “origen_con_mascara” o “origen_sin_masacara”.

Después de finalizar la opción anterior, pedirá el nombre para guardar el archivo Excel y procederá a cerrar la ventana.

4. CONCLUSIONES

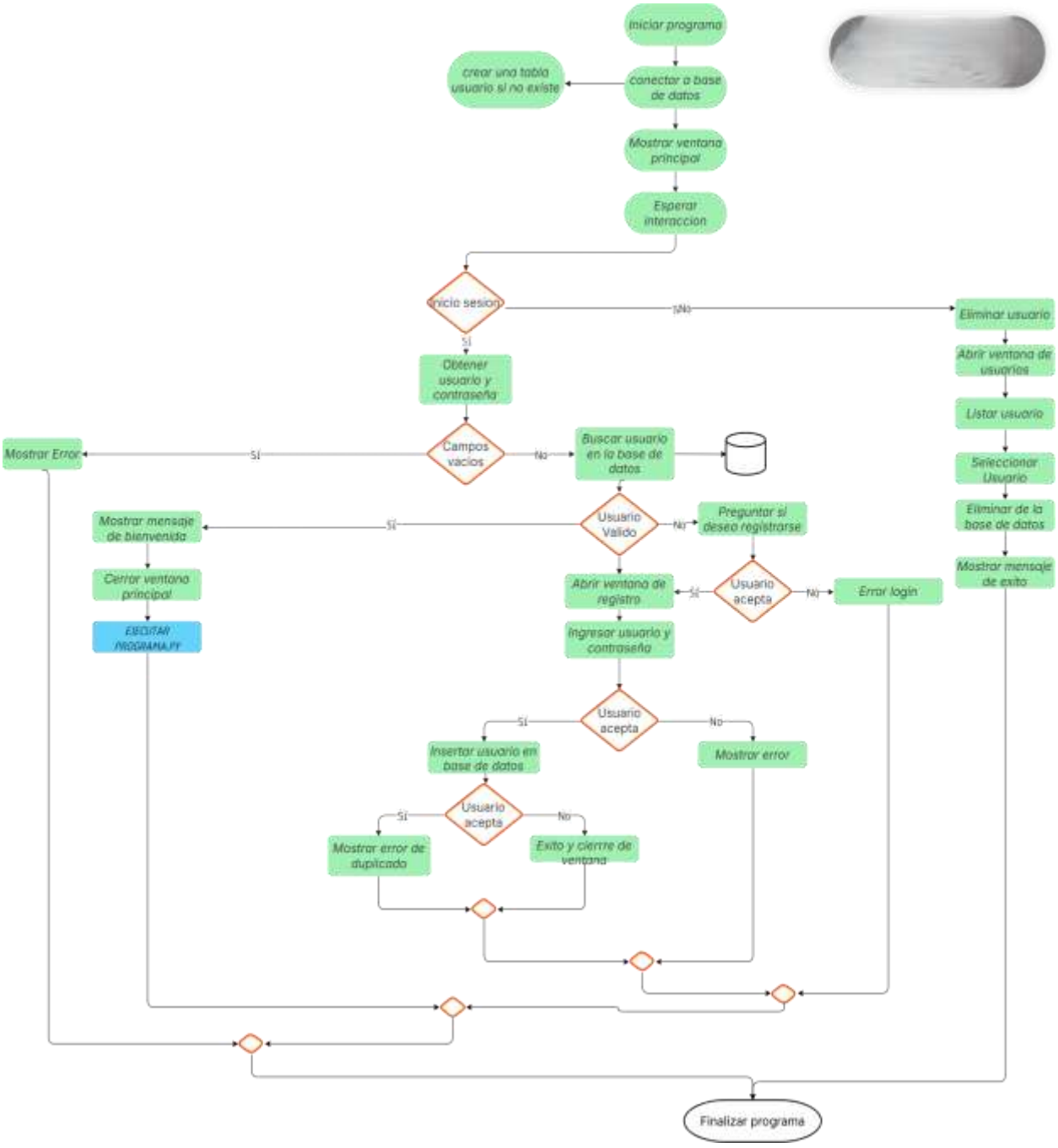
Esta aplicación logro ser una herramienta para la empresa HV Multiplay, permitiendo reducir el tiempo que los ingenieros invierten en la búsqueda de direcciones IP de forma manual, logrando una automatización de la búsqueda diaria de estas direcciones en diferentes páginas web, también los usuarios consiguieron determinar de donde provienen con exactitud y pudieron organizar de una manera más avanzada los archivos de texto o de Excel para una base de datos interna de la empresa. Hablando puntualmente del tiempo que se consiguió reducir al implementar este aplicativo, se vio que el tiempo anteriormente destinado para esta actividad era de más de 3 meses, posterior a la implementación de este aplicativo, los trabajadores ya no destinaban la misma cantidad de horas para este proceso, reduciendo el tiempo exponencialmente hasta llegar a las 2 o 3 semanas haciendo más efectivo la búsqueda de las direcciones IP.

La búsqueda de las direcciones desde un mismo equipo, en un principio se generaba un bloqueo en las paginas donde se realizan las consultas, el hecho de no saber de dónde provienen las direcciones puede causar una vulnerabilidad en el sistema, este problema quedo solucionado al implementar este aplicativo puesto que este les dio la oportunidad tener una ubicación clara de cada dirección IP y esto lograba que fuera mas seguro, evitando las filtraciones de seguridad en los servidores de la empresa.

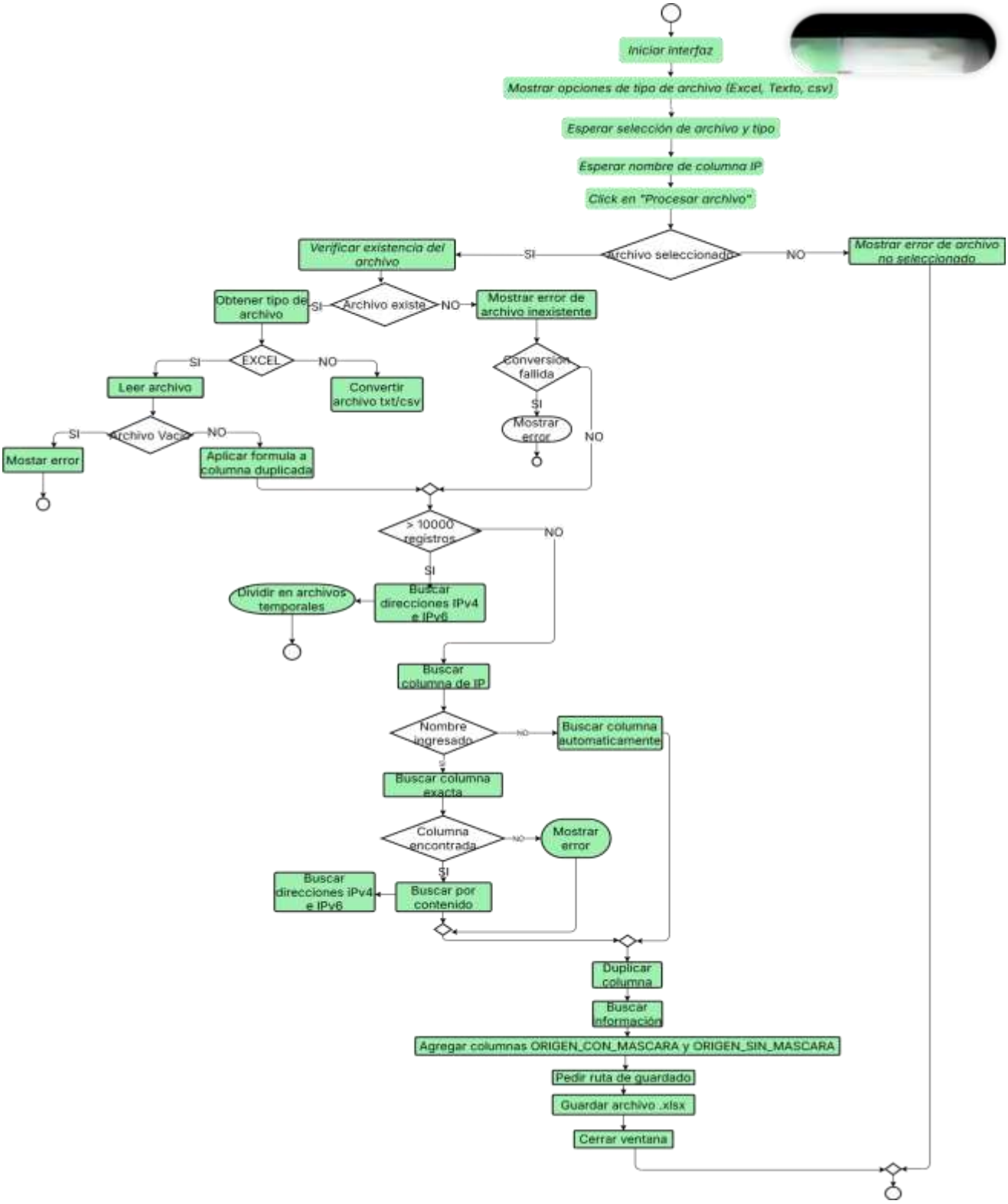
Para los trabajadores fue de mayor utilidad la búsqueda de direcciones IP dado que pueden dejar el aplicativo funcionando y esto ayuda a que ellos puedan desarrollar otras actividades que requieren más atención dentro de su horario laboral, haciendo más eficiente el desarrollo de la empresa. Para este proceso que consta de identificar las diferentes direcciones IP que llegan a un servidor, era una prioridad demasiado importante para el objetivo de este proyecto lograr una automatización de este proceso que anteriormente se realizaba de forma manual, permitiendo un mejoramiento en el rendimiento y el desempeño en el área de aprovisionamiento de datos.

ANEXOS

Anexo A Módulo de registro



Anexo B Módulo de ejecución



Anexo C Código para ventana de inicio de sesión

```
import os
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk
import sqlite3
import subprocess
conexion = sqlite3.connect("usuarios.db")
cursor = conexion.cursor()
cursor.execute("""
CREATE TABLE IF NOT EXISTS usuarios (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    usuario TEXT UNIQUE NOT NULL,
    contrasena TEXT NOT NULL
)
""")
conexion.commit()
def abrir_registro():
    ventana_registro = tk.Toplevel(ventana_principal)
    ventana_registro.title("Registrar Usuario")
    ventana_registro.geometry("500x300")
    ventana_registro.configure(bg='white')
    tk.Label(ventana_registro, text="Registrar Usuario", font=("Arial", 16, "bold"), bg='white').pack(pady=10)
    tk.Label(ventana_registro, text="Usuario:", bg='white', font=("Arial", 12, "bold")).place(x=50, y=100)
    entrada_nuevo_usuario = tk.Entry(ventana_registro, font=("Arial", 12))
    entrada_nuevo_usuario.place(x=150, y=100, width=200)
    tk.Label(ventana_registro, text="Contraseña:", bg='white', font=("Arial", 12, "bold")).place(x=50, y=150)
    entrada_nueva_contrasena = tk.Entry(ventana_registro, show="*", font=("Arial", 12))
    entrada_nueva_contrasena.place(x=150, y=150, width=200)
    def registrar_usuario():
        nuevo_usuario = entrada_nuevo_usuario.get()
        nueva_contrasena = entrada_nueva_contrasena.get()
        if nuevo_usuario and nueva_contrasena:
            try:
                cursor.execute("INSERT INTO usuarios (usuario, contrasena) VALUES (?, ?)", (nuevo_usuario, nueva_contrasena))
                conexion.commit()
                messagebox.showinfo("Registro exitoso", "Usuario registrado correctamente")
                ventana_registro.destroy()
            except sqlite3.IntegrityError:
                messagebox.showerror("Error", "El usuario ya existe")
        else:
            messagebox.showerror("Error", "Por favor, completa todos los campos")
    tk.Button(ventana_registro, text="Registrar Usuario", command=registrar_usuario, font=("Arial", 12), bg="green", fg="white").place(x=150, y=200)
def eliminar_usuario():
    ventana_eliminar = tk.Toplevel()
    ventana_eliminar.title("Eliminar Usuario")
    ventana_eliminar.geometry("400x300")
    ventana_eliminar.configure(bg='white')
    def confirmar_eliminacion():
        usuario_a_eliminar = lista_usuarios.get(tk.ACTIVE)
        if usuario_a_eliminar:
            cursor.execute("DELETE FROM usuarios WHERE usuario = ?", (usuario_a_eliminar,))
            conexion.commit()
            messagebox.showinfo("Usuario eliminado", f"El usuario '{usuario_a_eliminar}' fue eliminado correctamente.")
            ventana_eliminar.destroy()
        else:
            messagebox.showerror("Error", "Por favor, selecciona un usuario para eliminar.")
    tk.Label(ventana_eliminar, text="Selecciona un usuario para eliminar:", bg='white', font=("Arial", 12, "bold")).pack(pady=10)
    lista_usuarios = tk.Listbox(ventana_eliminar, font=("Arial", 12))
    cursor.execute("SELECT usuario FROM usuarios")
    usuarios = cursor.fetchall()
    for usuario in usuarios:
        lista_usuarios.insert(tk.END, usuario[0])
    lista_usuarios.pack(pady=10)
    tk.Button(ventana_eliminar, text="Eliminar", command=confirmar_eliminacion, font=("Arial", 12), bg="black", fg="white").pack(pady=10)
```

```
def realizar_login():
    usuario = entrada_usuario.get()
    contrasena = entrada_contrasena.get()
    if not usuario or not contrasena:
        messagebox.showerror("Error", "Por favor, completa todos los campos.")
        return
    cursor.execute("SELECT * FROM usuarios WHERE usuario = ? AND contrasena = ?", (usuario, contrasena))
    if cursor.fetchone():
        messagebox.showinfo("Login exitoso", f"¡Bienvenido {usuario}!")
        ventana_principal.destroy()
        codigo_path = r"C:\Users\Usuario\OneDrive\Escritorio\proyecto-final-3-04-25\proyecto-final-3-04-25\codigo-base.py"
        if os.path.exists(codigo_path):
            try:
                subprocess.run(["python", codigo_path])
            except Exception as e:
                messagebox.showerror("Error", f"No se pudo ejecutar codigo-base.py: {str(e)}")
        else:
            messagebox.showerror("Error", f"No se encontró el archivo 'codigo-base.py' en la ruta especificada:\n{codigo_path}")
    else:
        respuesta = messagebox.askquestion("Usuario no encontrado", "El usuario no existe. ¿Deseas registrarte?")
        if respuesta == "yes":
            abrir_registro()
        else:
            messagebox.showerror("Error de login", "Usuario o contraseña incorrectos.")
ruta_proyecto = os.path.dirname(os.path.abspath(__file__))
ruta_imagen = os.path.join(ruta_proyecto, "logohvplay-43_poster_.png")
ventana_principal = tk.Tk()
ventana_principal.title("Menú Principal")
ventana_principal.geometry("500x300")
ventana_principal.configure(bg='white')
try:
    imagen_original = Image.open(ruta_imagen)
    nuevo_ancho = 300
    nueva_altura = 70
    imagen_redimensionada = imagen_original.resize((nuevo_ancho, nueva_altura), Image.LANCZOS)
    imagen_tk = ImageTk.PhotoImage(imagen_redimensionada)
    tk.Label(ventana_principal, image=imagen_tk, bg='white').place(relx=0.5, rely=0.1, anchor="center")
except FileNotFoundError:
    messagebox.showerror("Error", f"No se encontró la imagen en {ruta_imagen}. Verifica la ubicación del archivo.")

tk.Label(ventana_principal, text="Usuario:", bg='white', font=("Arial", 12, "bold")).place(x=50, y=120)
entrada_usuario = tk.Entry(ventana_principal, font=("Arial", 12))
entrada_usuario.place(x=150, y=120, width=200)
tk.Label(ventana_principal, text="Contraseña:", bg='white', font=("Arial", 12, "bold")).place(x=50, y=160)
entrada_contrasena = tk.Entry(ventana_principal, show="*", font=("Arial", 12))
entrada_contrasena.place(x=150, y=160, width=200)
tk.Button(ventana_principal, text="Iniciar Sesión", command=realizar_login, font=("Arial", 12), bg="blue", fg="white").place(x=120, y=200)
label_eliminar = tk.Label(ventana_principal, text="Eliminar Usuario", fg="black", font=("Arial", 9), cursor="hand2", bg='white')
label_eliminar.place(x=250, y=205)
label_eliminar.bind("<Button-1>", lambda e: eliminar_usuario())
ventana_principal.mainloop()
conexion.close()
```

Anexo D Código Programa base

```
import pandas as pd
import tkinter as tk
from tkinter import filedialog, messagebox
from ipwhois import IPWhois
from openpyxl.styles import Font
from openpyxl.utils import get_column_letter
from openpyxl import load_workbook
import re
import os
def get_column_case_insensitive(df, col_input):
    col_input_clean = col_input.strip().lower()
    time_date_pattern = r'^\d{1,2}[:/-]\d{1,2}[:/-]\d{1,2}$'
    for col in df.columns:
        col_name = col.strip()
        col_lower = col_name.lower()
        if col_lower == col_input_clean and not re.match(time_date_pattern, col_name):
            return col
    return None
def buscar_columna_por_valor(df, valor):
    valor_lower = valor.strip().lower()
    for col in df.columns:
        for celda in df[col].head(10).astype(str):
            if valor_lower in celda.strip().lower():
                return col
    return None
def buscar_columna_ip_general(df):
    patron_ipv4 = r'\b(?:\d{1,3}\.){3}\d{1,3}(?:/(3[0]|[12]?[0-9]))?\b'
    patron_ipv6 = r'\b(?:[A-Fa-f0-9]{1,4}:){1,7}[A-Fa-f0-9]{1,4}(?:/(12[0-8]|[1-9]?\d))?\b'
    patron = re.compile(patron_ipv4 + "|" + patron_ipv6)
    for col in df.columns:
        resultados = df[col].head(10).astype(str).apply(lambda x: patron.search(x))
        valor_encontrado = resultados.dropna().apply(lambda x: x.group(0))
        valores_validos = valor_encontrado[valor_encontrado.str.match(patron_ipv4, na=False)]
        if valores_validos.notna().any():
            print(f" Columna detectada con direcciones IP: {col}")
            return col
    print(" No se encontró ninguna columna con direcciones IP válidas en las primeras 10 filas.")
    return None
def ajustar_celdas(df):
    new_cols = []
    for col in df.columns:
        if isinstance(col, str) and col.strip():
            new_cols.append(col.strip())
        else:
            serie = df.iloc[:, df.columns.get_loc(col)].astype(str)
            if any(valor.strip() != "" for valor in serie):
                new_cols.append("  ")
            else:
                new_cols.append("-")
    df.columns = new_cols
    df = df.fillna("  ")
    df = df.replace(r"^\s*$", "  ", regex=True)
    return df
```

```
def buscar_informacion_ip(ip):
    if "/" in ip:
        pure_ip = ip.split("/")[0].strip()
    else:
        pure_ip = ip.strip()
    try:
        obj = IPWhois(pure_ip)
        resultados = obj.lookup_rdap(asn_methods=["dns", "whois", "http"])
        organizacion = resultados.get("asn_description", "No Data")
        return organizacion if organizacion not in (None, "", "Error", "No Data") else "  "
    except Exception as e:
        print(f"Error en RDAP para {pure_ip}: {e}")
        return "  "
def seleccionar_archivo():
    tipo_seleccionado = tipo_var.get()
    tipos_archivo = {
        "Excel": [("Archivos Excel", "*.xlsx")],
        "Texto": [("Archivos de texto", "*.txt")],
        "CSV": [("Archivos CSV", "*.csv")]
    }
    archivo = filedialog.askopenfilename(filetypes=tipos_archivo.get(tipo_seleccionado, [("Todos los archivos", "*.*")]))
    if archivo:
        ruta_archivo.set(archivo)
        print(f"Archivo seleccionado: {archivo}")
    else:
        messagebox.showerror("Error", "No se seleccionó ningún archivo.")
def split_line(line):
    return re.split(r'\s{2,}', line.rstrip("\n"))
```

```python
def convertir_archivo_txt(archivo_entrada, buscar_encabezado=""):
    try:
        with open(archivo_entrada, "r", encoding="utf-8") as file:
            lines = file.readlines()
        if not lines:
            raise ValueError("El archivo TXT está vacío.")
        def split_line(line):
            return re.split(r'\s{2,}', line.rstrip("\n"))
        header_index = None
        if buscar_encabezado.strip() != "":
            token_busqueda = buscar_encabezado.strip().lower()
            for i in range(min(10, len(lines))):
                tokens_temp = split_line(lines[i])
                if any(token.strip().lower() == token_busqueda for token in tokens_temp):
                    header_index = i
                    break
        if header_index is None:
            header_index = 0
        header_tokens = split_line(lines[header_index])
        header_tokens = [token.strip() if token.strip() != "" else "-" for token in header_tokens]
        data_lines = lines[header_index+1:]
        data_tokens = [split_line(line) for line in data_lines]
        max_columns = max(len(header_tokens), max((len(tokens) for tokens in data_tokens), default=0))
        if len(header_tokens) < max_columns:
            header_tokens.extend([f"Extra_{i+1}" for i in range(max_columns - len(header_tokens))])
        elif len(header_tokens) > max_columns:
            header_tokens = header_tokens[:max_columns]
        processed_data = []
        for tokens in data_tokens:
            if len(tokens) < max_columns:
                tokens.extend([""] * (max_columns - len(tokens)))
            elif len(tokens) > max_columns:
                tokens = tokens[:max_columns]
            processed_data.append(tokens)
        df = pd.DataFrame(processed_data, columns=header_tokens)
        df = ajustar_celdas(df)
        archivo_salida = archivo_entrada.replace(".txt", "_procesado.xlsx")
        df.to_excel(archivo_salida, index=False, engine="openpyxl")
        print(f"Archivo TXT convertido exitosamente: {archivo_salida}")
        return df
    except Exception as e:
        messagebox.showerror("Error", f"No se pudo convertir el archivo TXT: {str(e)}")
        return None
def convertir_archivo_csv(archivo_entrada):
    try:
        df = pd.read_csv(archivo_entrada, keep_default_na=False)
        return ajustar_celdas(df)
    except Exception as e:
        messagebox.showerror("Error", f"No se pudo procesar el archivo CSV: {e}")
        return None
```

```python
def aplicar_formula_en_columna_duplicada(excel_file, nombre_columna_ip):
    try:
        wb = load_workbook(excel_file)
        ws = wb.active
        encabezados = [cell.value for cell in ws[1] if cell.value is not None]
        columna_duplicada = f"{nombre_columna_ip}_DUPLICADO"
        if columna_duplicada not in encabezados:
            print(f"No se encontró la columna '{columna_duplicada}'. Verifica que esté creada antes.")
            return
        col_index_original = encabezados.index(nombre_columna_ip) + 1
        col_index_duplicada = encabezados.index(columna_duplicada) + 1
        col_letter_original = get_column_letter(col_index_original)
        col_letter_duplicada = get_column_letter(col_index_duplicada)
        for row in range(2, ws.max_row + 1):
            ws[f"{col_letter_duplicada}{row}"].value = f'=TEXTO({col_letter_original}{row},"###.###.###.###")'
        wb.save(excel_file)
        print(f"Se aplicó correctamente la fórmula en la columna '{columna_duplicada}'.")
    except Exception as e:
        print(f"Error al procesar el archivo Excel: {e}")
def convertir_archivo(archivo_entrada, tipo_archivo, header_busqueda=""):
    try:
        if tipo_archivo == "Texto" and archivo_entrada.lower().endswith(".txt"):
            df = convertir_archivo_txt(archivo_entrada, header_busqueda)
        elif tipo_archivo == "CSV" and archivo_entrada.lower().endswith(".csv"):
            df = convertir_archivo_csv(archivo_entrada)
        elif tipo_archivo == "Excel" and archivo_entrada.lower().endswith(".xlsx"):
            df = pd.read_excel(archivo_entrada, engine="openpyxl", dtype=str)
        else:
            messagebox.showerror("Error", "El tipo de archivo seleccionado no coincide con su extensión.")
            return None
        if df is None or df.empty:
            messagebox.showerror("Error", f"El archivo {tipo_archivo} no tiene datos válidos.")
            return None
        return df
    except Exception as e:
        messagebox.showerror("Error", f"No se pudo leer el archivo: {str(e)}")
        return None
```

```python
def formatear_numero_como_ip(numero_str, tipo_archivo):
    if tipo_archivo == "Excel" and len(numero_str) == 12:
        return f"{numero_str[:3]}.{numero_str[3:6]}.{numero_str[6:9]}.{numero_str[9:]}"
    return numero_str
def convertir_ipv6(hex_str, tipo_archivo):
    if tipo_archivo == "Excel" and len(hex_str) == 32:
        grupos = [hex_str[i:i+4] for i in range(0, 32, 4)]
        return ":".join(grupos)
    return hex_str
```

```python
def procesar_columna_ip(df, columna):
    ips = df[columna].apply(lambda x: extraer_ip(x))
    primera_ip_index = None
    for idx, ip in enumerate(ips):
        if ip != "-":
            primera_ip_index = idx
            break
    if primera_ip_index is not None and primera_ip_index > 0:
        print(f"Primera dirección IP detectada en la fila {primera_ip_index + 1}")
        encabezados = df.iloc[primera_ip_index - 1]
        df.columns = encabezados
        df = df.iloc[primera_ip_index:].reset_index(drop=True)
        origen_con = []
        origen_sin = []
        for ip in ips[primera_ip_index:]:
            if "/" in ip:
                organizacion = buscar_informacion_ip(ip.split("/")[0])
                origen_con.append(organizacion)
                origen_sin.append("  ")
            else:
                organizacion = buscar_informacion_ip(ip)
                origen_sin.append(organizacion)
                origen_con.append("  ")
        ultima_columna_index = len(df.columns)
        df.insert(ultima_columna_index, "ORIGEN_CON_MASCARA", origen_con)
        df.insert(ultima_columna_index + 1, "ORIGEN_SIN_MASCARA", origen_sin)
        print("Encabezados y columnas organizados correctamente.")
        guardar_archivo(df)
    else:
        print("No se pudo detectar una IP válida para definir la fila de encabezados.")
```

```python
def organizar_encabezados_y_columnas(df, ws, nombre_columna_ip):
    try:
        primera_fila_encabezados = None
        for row_num, row in enumerate(ws.iter_rows(values_only=True), start=1):
            if nombre_columna_ip in row:
                primera_fila_encabezados = row_num
                break
        if primera_fila_encabezados:
            print(f"Encabezados encontrados en la fila {primera_fila_encabezados}")
            ws.delete_rows(1, primera_fila_encabezados - 1)
            for cell in ws[primera_fila_encabezados]:
                if cell.value == nombre_columna_ip:
                    cell.font = Font(bold=True)
            ultima_columna_index = len(df.columns)
            df.insert(ultima_columna_index, "ORIGEN_CON_MASCARA", [" "]*df.shape[0])
            df.insert(ultima_columna_index + 1, "ORIGEN_SIN_MASCARA", [" "]*df.shape[0])
            print("Encabezados y columnas organizados correctamente.")
        else:
            print("No se encontró la fila de encabezados correctamente.")
        return df
    except Exception as e:
        print(f"Error al organizar los encabezados y columnas: {e}")
        return df
```

```python
def extraer_ip(valor):
    valor = str(valor)
    patron_ipv4 = r'\b(?:\d{1,3}\.){3}\d{1,3}(?:/(3[0]|[12]?[0-9]))?\b'
    patron_ipv6 = (
        r'\b(?:[A-Fa-f0-9]{1,4}:){1,7}[A-Fa-f0-9]{1,4}(?:/(12[0-8]|[1-9]?\d))?\b'
    )
    patron = re.compile(patron_ipv4 + "|" + patron_ipv6)
    match = patron.search(valor)
    if match:
        ip_encontrada = match.group(0)
        if "." in ip_encontrada and ip_encontrada.count(".") == 3:
            return ip_encontrada
        elif ":" in ip_encontrada:  # IPv6, conserva estructura
            return ip_encontrada
    return "-"
```

```python
def procesar_archivo_segun_modo():
    archivo = ruta_archivo.get()
    if not archivo:
        messagebox.showerror("Error", "No se seleccionó ningún archivo.")
        return
    tipo_seleccionado = tipo_var.get()
    nombre_columna_entrante = columna_ip.get().strip()
    print(f"Valor ingresado en columna_ip: '{nombre_columna_entrante}'")
    if not os.path.exists(archivo):
        messagebox.showerror("Error", "El archivo no existe o no se puede acceder.")
        return
    if tipo_seleccionado == "Excel":
        df = pd.read_excel(archivo, engine="openpyxl", dtype=str)
        if df.empty:
            messagebox.showerror("Error", "El archivo Excel está vacío o no contiene datos válidos.")
            return
        aplicar_formula_en_columna_duplicada(archivo, nombre_columna_entrante)
    else:
        df = convertir_archivo(archivo, tipo_seleccionado, header_busqueda=nombre_columna_entrante)
        if df is None or df.empty:
            messagebox.showerror("Error", "Error en la conversión del archivo TXT/CSV a Excel.")
            return
    print(f"Columnas detectadas tras conversión: {df.columns}")
    if nombre_columna_entrante:
        chosen_column = get_column_case_insensitive(df, nombre_columna_entrante)
        if not chosen_column:
            chosen_column = buscar_columna_por_valor(df, nombre_columna_entrante)
            if chosen_column:
                print(f"Se encontró la columna '{chosen_column}' a partir de la búsqueda en el contenido.")
            else:
                print("No se encontró la columna ingresada por el usuario.")
    else:
        chosen_column = buscar_columna_ip_general(df)
    if not chosen_column:
        messagebox.showerror("Error", "No se encontró una columna válida para direcciones IP.")
        return
    if f"{chosen_column}_DUPLICADO" not in df.columns:
        df[f"{chosen_column}_DUPLICADO"] = df[chosen_column]
    df[f"{chosen_column}_DUPLICADO"] = df[f"{chosen_column}_DUPLICADO"].astype(str).apply(
        lambda x: formatear_numero_como_ip(x, tipo_seleccionado) if re.match(r"^\d{12}$", x) else
        (convertir_ipv6(x, tipo_seleccionado) if re.match(r"^[A-Fa-f0-9]{32}$", x) else x)
    )
    df["ORIGEN_CON_MASCARA"] = df[f"{chosen_column}_DUPLICADO"].astype(str).apply(
        lambda x: buscar_informacion_ip(x) if "/" in x else " "
    )
    df["ORIGEN_SIN_MASCARA"] = df[f"{chosen_column}_DUPLICADO"].astype(str).apply(
        lambda x: buscar_informacion_ip(x) if "/" not in x and x != "-" else " "
    )
```

```
if tipo_seleccionado != "Excel":
    df["ORIGEN_CON_MASCARA"] = df[chosen_column].apply(lambda ip: buscar_informacion_ip(ip) if "/" in str(ip) else "  ")
    df["ORIGEN_SIN_MASCARA"] = df[chosen_column].apply(lambda ip: buscar_informacion_ip(ip.split("/")[0]) if "/" in str(ip) else buscar_informacion_ip(ip))
guardar_archivo(df)
print(f"Archivo procesado correctamente: {archivo}")
if __name__ == '__main__':
    ventana = tk.Tk()
    ventana.title("BUSCADOR DE DIRECCIONES IP")
    ventana.geometry("400x300")
    ruta_archivo = tk.StringVar()
    columna_ip = tk.StringVar()
    tipo_var = tk.StringVar(value="Excel")
    tk.Label(ventana, text="Seleccionar tipo de archivo:", font=("Helvetica", 10, "bold")).pack()
    tk.Radiobutton(ventana, text="Excel", variable=tipo_var, value="Excel").pack(anchor=tk.W)
    tk.Radiobutton(ventana, text="Texto", variable=tipo_var, value="Texto").pack(anchor=tk.W)
    tk.Radiobutton(ventana, text="CSV", variable=tipo_var, value="CSV").pack(anchor=tk.W)
    tk.Button(ventana, text="Buscar archivo", command=seleccionar_archivo).pack(pady=5)
    tk.Label(ventana, text="Nombre de la columna de IPs (opcional):", font=("Helvetica", 10)).pack()
    tk.Entry(ventana, textvariable=columna_ip, width=30).pack(pady=5)
    tk.Button(ventana, text="Procesar archivo", command=procesar_archivo_segun_modo).pack(pady=10)
    ventana.mainloop()
```

Bibliografía

Byspel. (06 de 03 de 2025). *Direcciones IPV6 Notación y representación*. Obtenido de Direcciones IPV6 Notación y representación: https://byspel.com/direcciones-ipv6-notacion-y-representacion/

carballar. (28 de 05 de 2025). *Quién controla Internet. Organizaciones que dirigen la red global*. Obtenido de Quién controla Internet. Organizaciones que dirigen la red global: https://carballar.com/quien-controla-internet-organizaciones-que-dirigen-la-red-global

cecomart. (06 de 03 de 2025). *Tipos de IP's y clases: A, B, C, D y E*. Obtenido de Tipos de IP's y clases: A, B, C, D y E: https://cecomart.com/tipos-ip-que-son-ventajas-clases/

Coudflare. (06 de 03 de 2025). *¿Qué es una subred? | Cómo funciona una subred*. Obtenido de ¿Qué es una subred? | Cómo funciona una subred: https://www.cloudflare.com/es-es/learning/network-layer/what-is-a-subnet/

El pais. (6 de 03 de 2025). *Evolutio aspira a ser el guardián de la ciberseguridad*. Obtenido de Evolutio aspira a ser el guardián de la ciberseguridad: https://elpais.com/economia/negocios/2024-10-01/evolutio-aspira-a-ser-el-guardian-de-la-ciberseguridad.html

Funcion Publica. (6 de marzo de 2025). *Ley 1273 de 2009*. Obtenido de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=34492

Guevara, R. (06 de 03 de 2025). *Direccionamiento IPv4 e IPv6*. Obtenido de Direccionamiento IPv4 e IPv6: https://guevara3117.wordpress.com/wp-content/uploads/2019/05/ipv4-ipv6.pdf

IANA. (07 de 03 de 2025). *Number Resources*. Obtenido de Number Resources: https://www.iana.org/numbers

Imsaslud. (6 de 03 de 2025). *ABC Ley 1581 de 2012 Protección de Datos Personales*. Obtenido de https://www.imsalud.gov.co/web/sin-categoria/abc-ley-1581-de-2012-proteccion-de-datos-personales/#:~:text=La%20ley%20de%20protecci%C3%B3n%20de,como%20su%20posterior%20actualizaci%C3%B3n%20y

Larus. (06 de 03 de 2025). *¿Qué es una dirección IP de clase C?* Obtenido de https://larus.net/blog/what-is-a-class-c-ip-address/

Marino, N. (06 de 03 de 2025). *Direccionamiento IP: la base de la comunicación en redes*. Obtenido de Direccionamiento IP: la base de la comunicación en redes: https://www.tokioschool.com/noticias/direccionamiento-ip/

Meridianoutpost. (6 de 03 de 2025). *The Five IPv4 Classes - Quick Reference*. Obtenido de The Five IPv4 Classes - Quick Reference: https://www.meridianoutpost.com/resources/articles/IP-classes.php

Sistema único de información normativa. (06 de 03 de 2025). *DECRETO 1078 DE 2015*. Obtenido de https://www.suin- juriscol.gov.co/viewDocument.asp?id=30019521

Sistema único de información normativa. (06 de 03 de 2025). *RESOLUCIÓN 500 DE 2021*. Obtenido de https://www.suin-juriscol.gov.co/viewDocument.asp?id=30044822

vpn unlimited. (06 de 03 de 2025). *Class D IP address*. Obtenido de Class D IP address: https://www.vpnunlimited.com/help/cybersecurity/class-d-ip-address