E.T. Nº 36
ALMIRANTE GUILLERMO BROWN

Redes

Escáner de red - Documentación

Año: 5º	División: 1º	Turno: Tarde

Autor	Alejo Guerra

Profesor: Oscar A. Obregón

Fecha de entrega: 19/08/2025

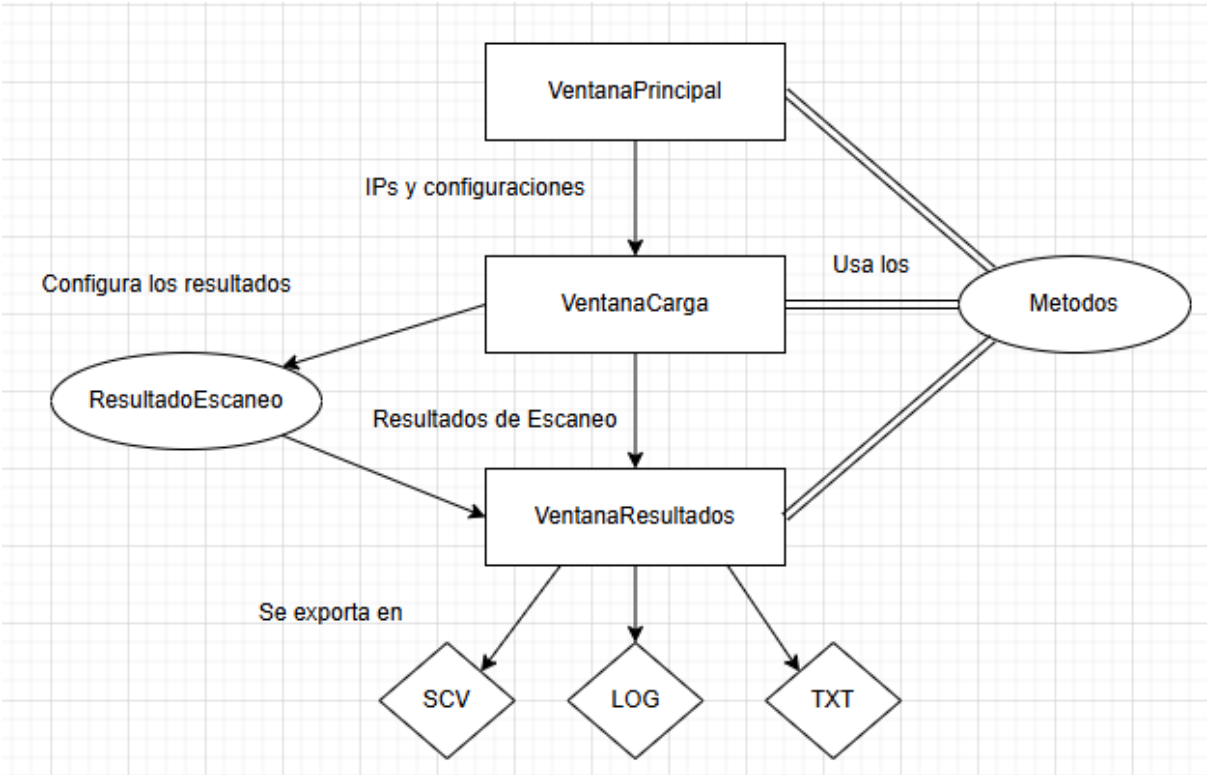
Scanner de red - Documentación:

Descripción:

Este programa se formó para simplificar una tarea complicada, el escaneo de redes. La falta de visibilidad clara sobre los dispositivos conectados, la dificultad para identificar rápidamente problemas de conectividad o dispositivos no autorizados, y la inversión de tiempo excesivo en tareas manuales son problemas que se pueden resolver simplemente implementando este programa.

El programa sirve para escanear dos IPs desde un determinado rango, empezando por una IP Inicial y terminando en una IP final. La idea de este proyecto fue simplificar una tarea y construir una herramienta con una interfaz simple pero que puede ser usada para trabajos complejos.

Diagrama del código:



Metodologías del trabajo: ¿Cuáles fueron los métodos utilizados?

Lo primero que hice fue averiguar cuál iba a ser el objetivo y las especificaciones que tenía que cumplir, después empecé a crear las ventanas y clases que iban a ser necesarias . Cuando terminé la parte visual, empecé a estudiar cómo funcionaba el recorrido de ips y los comandos que iban a ser necesarios para que este programa pueda funcionar bien. Después, pensé el funcionamiento principal del programa en Python, porque entendía mejor, y después lo pasé a java. Por último y con ayuda de la inteligencia artificial y los conocimientos que averigüe, empecé a crear toda la parte lógica para finalmente enfocarme más en las mejoras visuales y de código.

Durante todo el proceso fui optimizando el programa, mejorando la estética, arreglando todos los errores que surgían, aumentando la velocidad, etc.

Tecnologías utilizadas: ¿Cuáles son las tecnologías utilizadas? ¿Por qué?

javax.swing
Uso Swing para toda la interfaz gráfica: ventanas (`JFrame`), paneles, botones, etiquetas, tablas (`JTable`), diálogos (`JOptionPane`, `JFileChooser`), barras de progreso, comboboxes y toolbars. Es la base para construir y manejar la GUI.

java.awt
Uso AWT para los layouts y utilidades visuales: `BorderLayout`, `GridBagLayout`, `FlowLayout`, `Dimension`, `Color`, `Font`, `Cursor` y los eventos básicos. Complementa a Swing para posicionar y estilizar componentes.

java.awt.event / javax.swing.event
Manejo eventos de usuario: `ActionListener` para botones, `MouseAdapter`/`MouseEvent` para doble clic en la tabla, y `DocumentListener` para validación en tiempo real de campos de texto.

java.io / java.nio.charset
Lectura/escritura de archivos (TXT, CSV, LOG) y lectura de salida de procesos (`BufferedReader`, `OutputStreamWriter`, `BufferedWriter`, `File`, `FileOutputStream`). Uso `StandardCharsets.UTF_8` para asegurar la codificación correcta al guardar archivos.

java.net.InetAddress / ProcessBuilder
Realizo operaciones de red básicas con `InetAddress` (`isReachable`, `getHostName`) para pings, y ejecuto procesos externos como `nslookup` con `ProcessBuilder` para obtener información adicional (host server, DNS).

java.util
Uso colecciones (`ArrayList`), ordenamientos (`Comparator`), utilidades de fecha (`Date`) y otras utilidades estándar para almacenar y procesar resultados.

java.text.SimpleDateFormat
Formateo timestamps para encabezados de informes y logs (por ejemplo `yyyy-MM-dd HH:mm:ss`).

java.util.concurrent *(opcional / cuando se usa multihilo)*
Si activo paralelismo, uso `ExecutorService` / `Executors` para acelerar pings concurrentes y no bloquear la interfaz.

Clases internas del proyecto

- `scanner.modelo.ResultadoEscaneo` — modelo que representa cada resultado (ip, estado, host, tiempo, ttl, hostServer, servidorDNS, mensajeError).
- `scanner.modelo.Metodos` — utilidades: validación de IPs, conversión `ipToLong`, parseo de tiempos, y métodos trasladados de red (`hacerPing`, `hacerNslookup`).

A medida que iba haciendo el proyecto, surgieron estos problemas:

- *Problemas con exportación en excel:* Estuve mucho tiempo tratando de que se exportara directamente a formato excel, porque tenia que importar un montón de módulos manualmente o pasar a proyecto Maven. Esto se solucionó cambiando el tipo de archivo a CSV, que era mucho más simple.
- *Problemas con los componentes de JSwing:* Muchos componentes no se adaptaban bien a la interfaz que planteaba, por lo que tuve que estar mucho tiempo para lograr que queden correctamente.
- *Falta de conocimiento:* Java es demasiado extenso, y tiene muchas funcionalidades. Pasé mucho tiempo entendiendo todo lo que yo pensaba añadir a mi proyecto, además del funcionamiento de las IPs y los comandos de consola
- *MultiThreading :* Buscando acelerar mi proyecto, recurrí a usar hilos, esto hizo que muchas funciones de mi programa en ese momento dejaran de funcionar correctamente. Hacer el traslado costó mucho tiempo.

Escalabilidad: ¿Qué se podría mejorar en el futuro?

El programa está hecho con la idea de poder ampliarse a futuro, por lo que tiene una base que en caso de querer expandir no se tendrá que modificar ni rediseñar.

Mejorar la velocidad: Si el programa requiere usarse con fines profesionales se necesitaría aumentar la velocidad del escaneo.

Adaptabilidad a otros S.O: El programa no es compatible con otro sistema operativo diferente a Windows por los comandos, entonces si el programa llega a ser una herramienta fundamental se puede extender para tener más alcance a otros dispositivos.

Disponibilidad para todas las versiones de Java: Mi programa se desarrolló en java 1.8, por lo que muchas funciones no funcionan en versiones más recientes, y si necesitas usarlo, tendrias que adaptarte a la versión.