E.T. Nº 36
ALMIRANTE GUILLERMO BROWN

Redes

Escáner de red - Documentación

Año: 5º	División: 1º	Turno: Tarde	
Nombre: Alejo	Apellido : Guerra		

Profesor: Oscar A. Obregón

Fecha de entrega: 19/08/2025

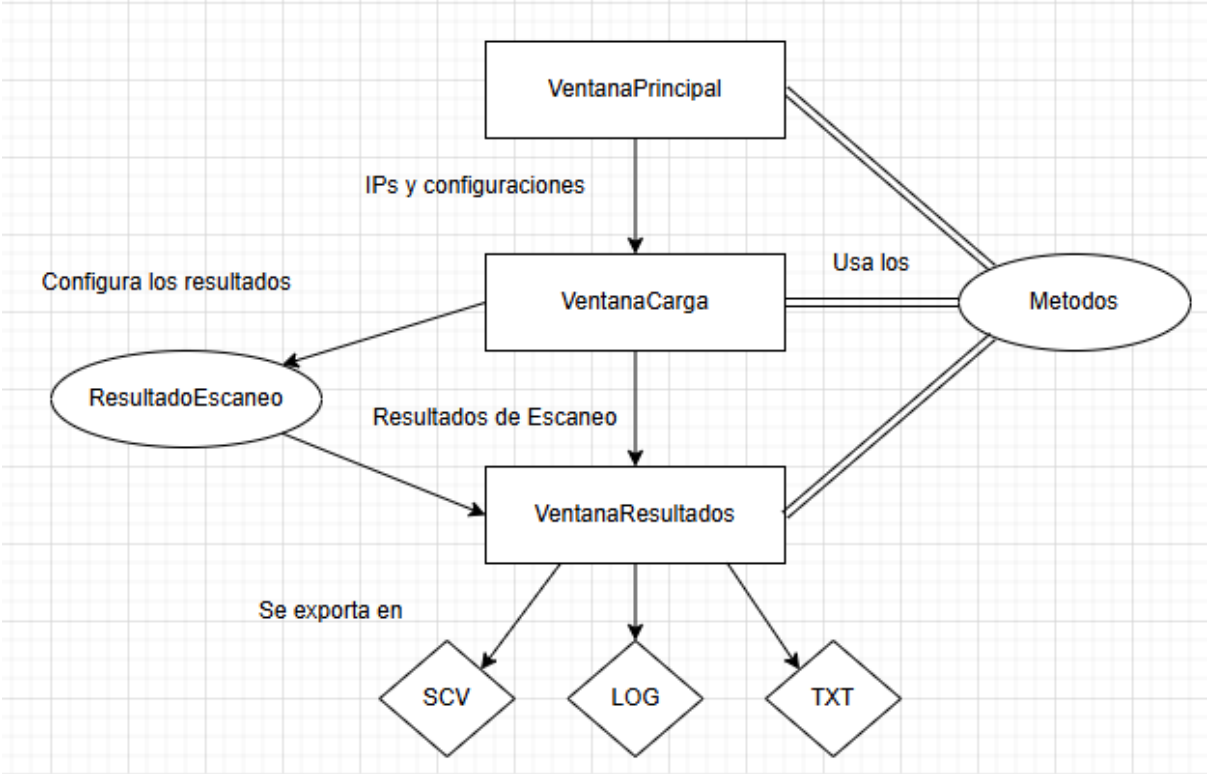
Scanner de red - Documentación:

Descripción:

Este programa fue desarrollado con el objetivo de simplificar una tarea compleja: el escaneo de redes. Problemas como la falta de visibilidad sobre los dispositivos conectados, la dificultad para identificar rápidamente problemas de conectividad o detectar equipos no autorizados, y el tiempo excesivo requerido para realizar tareas manuales, pueden resolverse de manera eficiente mediante la implementación de esta herramienta.

El software permite escanear un rango de direcciones IP, desde una IP inicial hasta una IP final, proporcionando información clara y organizada sobre el estado de cada equipo. El propósito del proyecto fue crear una solución que simplifique significativamente estas tareas, combinando una interfaz intuitiva con la capacidad de manejar análisis de red más complejos de manera efectiva.

Diagrama del código:



Metodologías del trabajo: ¿Cuáles fueron los métodos utilizados?

El primer paso consistió en definir claramente el objetivo del proyecto y las especificaciones que debía cumplir. A continuación, diseñé y desarrollé las ventanas y clases necesarias para la interfaz visual. Una vez finalizada la parte gráfica, investigué el funcionamiento del recorrido de direcciones IP y los comandos esenciales para garantizar que el programa operara correctamente.

Para conceptualizar la lógica principal, primero la modelé en Python, dado que me resultaba más sencillo entender y probar el flujo, y posteriormente la trasladé a Java. Con el apoyo de inteligencia artificial y mis conocimientos previos, desarrollé toda la lógica del programa, para luego enfocarme en optimizar la interfaz, el código y la experiencia del usuario.

Durante todo el proceso, se realizaron mejoras continuas en términos de eficiencia, corrección de errores, velocidad de ejecución y estética general del programa.

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

javax.swing
Uso Swing para toda la interfaz gráfica: ventanas (JFrame), paneles (JPanel), botones (JButton), etiquetas (JLabel), tablas (JTable), diálogos (JOptionPane, JFileChooser), barras de herramientas (JToolBar), comboboxes (JComboBox) y campos de texto (JTextField). Es la base para construir y manejar toda la GUI.

java.awt
Uso AWT para layouts y utilidades visuales: BorderLayout, GridBagLayout, FlowLayout, Dimension, Color, Font y Cursor. Permite posicionar y estilizar componentes, complementando Swing.

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

java.io / java.nio.charset
Lectura y escritura de archivos (TXT, CSV, LOG) mediante File, FileOutputStream, BufferedWriter, OutputStreamWriter, Writer. Uso StandardCharsets.UTF_8 para asegurar codificación correcta al guardar archivos y mantener compatibilidad con caracteres especiales.

java.util
Uso de colecciones (ArrayList) para almacenar resultados, ordenamientos (Comparator) para filtrar/ordenar datos, utilidades de fecha (Date), y otras utilidades estándar para procesar resultados y construir la lógica del escaneo.

java.text.SimpleDateFormat
Formateo de fechas y horas para encabezados de informes y logs, por ejemplo `yyyy-MM-dd HH:mm:ss` y `HH:mm:ss` para la hora de cada escaneo.

java.net.InetAddress / ProcessBuilder
InetAddress para operaciones básicas de red: ping (isReachable) y resolución de host (getHostName). ProcessBuilder para ejecutar procesos externos como `nslookup` y obtener información adicional de host server y DNS.

java.util.concurrent
Si se activa paralelismo, uso de ExecutorService y Executors para realizar pings concurrentes y evitar bloquear la interfaz de usuario, acelerando el escaneo.

Clases internas del proyecto

- **scanner.modelo.ResultadoEscaneo** — modelo que representa cada resultado: IP, estado de conexión, host, tiempo de respuesta, TTL, host server, servidor DNS, mensaje de error y fecha/hora del escaneo.

- **scanner.modelo.Metodos** — utilidades: validación de IPs, conversión `ipToLong`, parseo de tiempos, generación de filas fijas para archivos, métodos de red (hacer ping, ejecutar nslookup), escape de CSV y repetición de caracteres.

Otras librerías de utilidad

- **javax.swing.filechooser.FileNameExtensionFilter** para filtrar tipos de archivo en diálogos de guardar (TXT, CSV, LOG).

- **javax.swing.border** para bordes estéticos y TitledBorder en paneles.

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.

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.

Internacionalización y adaptabilidad
Hoy el programa está en español, pero podría ser adaptado para varios idiomas, lo que permitiría su uso en entornos más amplios. También se podrían implementar configuraciones regionales de formato de fechas y red.

Documentación y mantenimiento
Aunque el código está organizado, agregar comentarios más detallados facilitaría futuras modificaciones, la incorporación de nuevos desarrolladores al proyecto y la resolución de errores.