Universidad Nacional de Colombia - sede Bogotá
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Curso: Ingeniería de Software I

Estudiantes: Felipe Rojas Marin, Santiago Alejandro Rojas Feo, Juan Diego Rozo Álvarez, Miguel Angel Citarella Camargo

Tecnologías escogidas para SpyNet:

Lenguaje de programación: Java
Framework: JavaFX
Escogimos **Java** como lenguaje principal porque ya lo conocemos bien y tenemos experiencia usándolo, lo cual nos permite avanzar más rápido en el desarrollo sin tener que aprender todo desde cero. Además, es un lenguaje bastante accesible: no es tan complicado como otros ni tiene una curva de aprendizaje muy grande. Otra ventaja es que tiene una comunidad muy activa, lo que significa muchísima documentación, foros y recursos disponibles. Java también nos ofrece todo lo que necesitamos para cumplir con los requisitos técnicos del proyecto, como manejo de interfaces gráficas, conexión con bases de datos y ejecución de comandos del sistema. Por lo tanto es esperable un desarrollo sin mayores complicaciones y obstáculos.

Para la parte gráfica de la aplicación decidimos usar **JavaFX**, que es un framework para crear interfaces gráficas en Java. Es Open Source y ha sido desarrollado tanto por empresas como por la comunidad desde hace años. JavaFX nos permite diseñar una interfaz moderna, eficiente y con buen rendimiento, ideal para una aplicación de escritorio como la que planeamos. Fue lanzado en 2008 y actualmente es parte oficial del OpenJDK, lo que significa que es una herramienta estable, madura y con bastante trayectoria. Gracias a esto, encontramos muchos ejemplos, guías y documentación que nos ayudan a desarrollar la aplicación de forma más ágil y con menos obstáculos.

Tecnologías	Ventajas	Desventajas
Python + PyQt / PySide ❌	Fácil de programar y entender. Buena documentación. Interfaz moderna con Qt Designer. Python tiene librerías para ejecutar comandos (como nmap) fácilmente. SQLite o PostgreSQL se integran sin problema.	El ejecutable puede ser pesado si se empaqueta con PyInstaller. No es tan "nativo" en apariencia en algunos sistemas (Windows/macOS/Linux).

C# (.NET) + Windows Forms o WPF ❌	Excelente integración con Windows. Windows Forms es fácil y rápido; WPF permite interfaces más modernas. Soporte nativo para bases de datos (SQL Server, SQLite). Ejecutar comandos (como nmap) es directo con Process.	Solo funciona nativamente en Windows. WPF puede tener una curva de aprendizaje más alta. .NET Core es multiplataforma, pero GUI sigue limitada a Windows.
Java + JavaFX ✔️	Buen soporte de GUI con JavaFX (más moderno que Swing). Java tiene bibliotecas para ejecutar procesos externos (ProcessBuilder). Fuerte tipado y OOP ayudan en aplicaciones medianas/grandes. Fácil acceso a JDBC para manejo de base de datos.	JavaFX puede ser verboso. Necesita JVM instalada (aunque se puede empaquetar). Apariencia menos nativa.
Electron + Node.js + HTML/CSS/JS ❌	Interfaz moderna y estilizada. Conocimiento web reutilizable. Puedes ejecutar nmap usando Node (child_process). Multiplataforma (Windows/Linux/macOS).	Alto consumo de recursos. Paquetes más grandes (incluye Chromium). La seguridad debe manejarse cuidadosamente.
Rust + egui / Tauri (con HTML frontend) ❌	Alto rendimiento y bajo consumo. Tauri es muy liviano comparado con Electron. Rust puede ejecutar procesos externos y manejar concurrencia eficientemente.	Curva de aprendizaje alta. Herramientas gráficas aún jóvenes (especialmente egui). Menor comunidad que otros lenguajes.

Base de Datos relacional: MySQL 🐬

Escogimos MySQL principalmente por nuestra experiencia previa en este motor de base de datos lo que nos permite un diseño más rápido al no tener que aprender una nueva tecnología desde cero a su misma vez supera ampliamente nuestros requerimientos para la base de datos, es simple de instalar en los dispositivos de los clientes y posee un rendimiento eficiente

Bibliotecas y herramientas complementarias usadas:

nmap: Lo usamos parte de uno de nuestros requerimientos/funcionalidades, esta herramienta que tendrá que ser instalada para el funcionamiento completo de nuestra aplicación, esta herramienta ofrece un poderoso escáner de redes informáticas, lo podemos usar para detectar puertos abiertos, servicios usados, sistemas operativos, etc. En general lo utilizamos para hacer un sondeo básico de las redes de nuestros usuarios, esto para detectar posibles vulnerabilidades en las redes.

API de OpenAI: Incorporamos la API de OpenAI como un complemento inteligente para el análisis de los resultados que obtenemos con Nmap. Esta API nos permite procesar automáticamente la información generada por el escaneo de red y generar una opinión sobre el nivel de vulnerabilidad de los puertos detectados. En otras palabras, el sistema no solo muestra los datos, sino que también los interpreta y ofrece una valoración preliminar sobre la seguridad de la red. Decidimos añadir esta funcionalidad para aprovechar las capacidades actuales de la inteligencia artificial y alinearnos con las tendencias tecnológicas modernas. De esta forma, no solo facilitamos el análisis para el usuario final, sino que también mejoramos la utilidad práctica de la auditoría.

SceneBuilder: Para la parte visual de la aplicación usamos SceneBuilder, una herramienta que nos permite diseñar interfaces gráficas (GUI) para JavaFX de forma visual e intuitiva. En lugar de escribir manualmente todo el código de la interfaz, SceneBuilder nos permite arrastrar y soltar componentes como botones, campos de texto, etiquetas, tablas, etc., lo cual acelera el proceso de desarrollo. Gracias a esto, podemos centrarnos más en la lógica de la aplicación sin dejar de lado una interfaz moderna y funcional. Esta herramienta también ayuda a mantener el código visual separado del código lógico, lo que mejora la organización del proyecto y su mantenimiento.

maven: Para gestionar nuestro proyecto usamos Maven, una de las herramientas más populares para construcción y administración de proyectos en Java. Maven nos permite automatizar tareas importantes como la compilación del código, la ejecución del programa y, sobre todo, la gestión de dependencias externas (por ejemplo, bibliotecas necesarias como JavaFX, la API de OpenAI o controladores JDBC para bases de datos). Esto hace que el desarrollo sea mucho más ordenado y reproducible, especialmente cuando el proyecto crece o cuando se trabaja en equipo. Al usar Maven, también nos aseguramos de tener un entorno más estandarizado, lo cual facilita tanto el desarrollo como la distribución futura de la aplicación.