

Monitorización en tiempo real de parámetros en hosts remotos

Javier Bravo Fernández

Grado en Ingeniería Informática

Administración de redes y sistemas operativos

Consultor: Manuel Jesús Mendoza Flores

Profesor responsable de la asignatura: Javier Panadero Martínez

Junio 2019



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Monitorización en tiempo real de parámetros en hosts remotos</i>
Nombre del autor:	<i>Javier Bravo Fernández</i>
Nombre del consultor/a:	<i>Manuel Jesús Mendoza Flores</i>
Nombre del PRA:	<i>Javier Panadero Martínez</i>
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	<i>Grado en ingeniería informática</i>
Área del Trabajo Final:	<i>Administración de redes y sistemas operativos.</i>
Idioma del trabajo:	<i>Español</i>
Palabras clave	<i>Monitorización, administración de sistemas</i>

Resumen del Trabajo

El objeto de este proyecto ha consistido en el desarrollo de una aplicación para la monitorización de parámetros en hosts remotos en tiempo real.

En el mercado existen sistemas para recopilar esta información, pero suelen requerir de la instalación de agentes, la parametrización previa de qué datos queremos recopilar y la información que muestran no se actualiza en tiempo real. Para cubrir esta necesidad los administradores recurren a la elaboración de scripts específico, dedicando mucho tiempo. Además se obtienen ficheros de salida no unificados que dificultan su análisis y usabilidad.

La aplicación se orienta a técnicos y administradores de sistemas, persiguiendo evitarles la elaboración y mantenimiento de scripts específicos. Es capaz de consultar múltiple información del sistema, así como obtener o modificar cualquier valor del registro de Windows, sobre cualquier listado de host indicado.

La solución se ha desarrollado en lenguaje C# y tecnología .NET, interactuando con los hosts remotos mediante WMI y Registro Remoto que tiene cualquier sistema Windows, La aplicación se ha estructurado en módulos para dotarla de versatilidad y escalabilidad.

El resultado final ha sido una aplicación portable, que no requiere instalación y que funciona sobre cualquier sistema Windows de 32 o 64 bits sin aplicar configuraciones previas.

Abstract

This Project consisted in the development of an application for remote host parameters monitoring in real time.

In the market there are systems available for collect this information, but usually requires the installation of agents and the previous configuration of data we want to retrieve and the shown information is not updated in real time. To cover this gap system administrators usually develop specific scripts, dedicating many time. Additionally they get not unified output files that difficult it analysis and usability.

The application is oriented for technicians and systems administrators, traying to avoid them the development and maintenance of specific scripts. It is capable to query multiple system information, and also get or modify any Windows registry value, all over any given host list in a file.

The solution has been developed in C# language and .NET technology. The remote host interactions are made using WMI and Remote Registry, booth available in any Windows system. The application is structured in modules to provide it scalability.

The final result is a portable application, that doesn't require any kind of installation, and runs over any Windows 32 or 64 bits without applying previous configuration.

Índice

1. Introducción.....	1
1.1 Contexto y justificación del Trabajo.....	1
1.2 Objetivos del Trabajo.....	2
1.3 Enfoque y método seguido.....	3
1.4 Planificación del Trabajo.....	5
1.4 Análisis de riesgos.....	11
1.5 Sumario de productos obtenidos.....	11
1.6 Descripción de los otros capítulos de la memoria.....	11
2. Análisis.....	12
2.1 Delimitación del entorno.....	12
Selección de tecnologías y análisis de costes.....	13
2.2 Requerimientos y configuraciones previas.....	16
3. Diseño.....	19
3.1 Módulo principal.....	19
3.2 Módulos específicos.....	24
4. Desarrollo y depuración.....	28
4.1 Construcción de entorno de test.....	28
3.2 definición de conjunto de pruebas.....	29
Resultado ejecución tests de validación.....	29
3.3 Valoración y mejoras.....	30
5. Conclusiones.....	36
5.1 Autoevaluación del trabajo realizado y resultados.....	36
5.2 Líneas de trabajo futuras.....	37
6. Glosario.....	38
7. Bibliografía.....	39
8. Anexos.....	41
8.1 Anexo 1: Preparación de paquete ejecutable.....	41
8.2 Anexo 2: User guide.....	43
8.3 Anexo 3: Command syntax.....	45

Lista de figuras

Ilustración 1. Planificación	10
Ilustración 2. Análisis de riesgos	11
Ilustración 3. Scorecard	15
Ilustración 4. Resumen de requerimientos	18
Ilustración 5. Diagrama de flujo del módulo principal	22
Ilustración 6. Interfaz gráfica del módulo principal	23
Ilustración 7. Entorno de pruebas	28
Ilustración 8. Pruebas y resultados	29
Ilustración 9. Compilación	30
Ilustración 10. Estructura de directorios	32
Ilustración 11. Inicialización de directorios	32
Ilustración 12. Interfaz gráfica mejorada	33
Ilustración 13. Inicialización de parámetros	34
Ilustración 14. Fichero de output sin filtrar	35
Ilustración 15. Fichero de output filtrado	35

1. Introducción

1.1 Contexto y justificación del Trabajo

Mediante la ejecución de este proyecto se pretende obtener una herramienta que permita consultar en tiempo real las configuraciones aplicadas en uno o varios hosts: PCs y Servidores. Dado que el abanico de configuraciones puede ser infinito, nos centraremos en asegurar las consultas de los parámetros más habituales y que más valor nos aportan, como son el registro de Windows, las configuraciones del sistema y las propiedades de cualquier fichero.

En el mercado existen herramientas que hacen trabajos similares, pero presentan varios inconvenientes:

1. Requieren de infraestructura: instalación de servidores que centralicen la información y despliegue de agentes a los hosts que capturen y reporten la información al sistema central.
2. Lapso desde que se produce un cambio hasta que la información es reportada por el agente, por tanto, visible en el servidor central.
3. Suelen monitorizarse los parámetros que en cada caso se consideren más necesarios, pero ante una nueva necesidad, se requiere reconfigurar el sistema y esperar a que de nuevo los agentes capturen y reporten la nueva información al sistema central

Para solventar estos puntos, los administradores de TI suelen recurrir a la creación de scripts hechos exprofeso para obtener la información que desean en un momento puntual. Aunque la creación de scripts resuelve la necesidad, implica la dedicación de tiempo para escribirlos y testarlos.

La experiencia profesional en el área de IT me ha hecho observar que la solución de los scripts, a pesar de cubrir la necesidad, implica una pérdida de tiempo, y sobre todo que gran parte de los scripts acaba redundándose. Además, suelen generarse outputs muy básicos, tipo CSV, que después hay que importar en herramientas ofimáticas para poder realizar un análisis de la información obtenida.

La oportunidad de desarrollar un proyecto en esta área hace orientar los esfuerzos a conseguir una herramienta que integre todas las partes comunes que habitualmente se desarrollan en los scripts, para que el administrador de TI sólo tenga que seleccionar las consultas que quiere realizar, indicándolas como parámetros, olvidándose así de hacer scripts a medida cada vez que tiene una necesidad.

Una herramienta de este tipo aportará agilidad en el día a día en un departamento de TI, pudiéndose utilizar para la resolución de incidencias,

comprobación de aplicación correcta de cambios o actualizaciones, o análisis previos a despliegues o proyectos.

1.2 Objetivos del Trabajo

La herramienta que obtengamos debe cumplir los siguientes objetivos principales:

- **Paradigma:** “standalone”, se persigue prescindir de dependencia de otros sistemas como servidores o despliegue de agentes
- **Instalación sencilla:** requerimientos técnicos y de configuración mínimos.
- **Minimización de costes:** el coste debe minimizarse, para aportar atractivo al producto frente a la opción totalmente gratuita de crear scripts, o a otras opciones de pago que permiten crear scripts de forma automática
- **Versatilidad y usabilidad:**
 - Las consultas deben ser parametrizables, para incluir más o menos parámetros
 - Posibilidad de guardar configuraciones, de forma que el usuario pueda volver a ejecutar una consulta que ya configuró en el pasado simplemente seleccionándola en la aplicación.
 - Output: los outputs deben ser de calidad, que permitan fácilmente el filtrado y análisis de la información obtenida.
 - Control de errores: Deben capturarse los errores que se produzcan durante la ejecución e informar al usuario de estos. Algunos ejemplos pueden ser la falta de comunicación con un equipo remoto, o la falta de privilegios para realizar las acciones deseadas.
- **Actualizaciones:** la herramienta no está pensada para que el usuario haga cambios en ella, pero sí debe estar estructurada adecuadamente, para que un desarrollador pueda introducir mejoras o nuevas funcionalidades cuando sea necesario.

Los objetivos parciales para lograrlo son los siguientes:

- **Delimitación de entorno** al que va orientada: Definición del sistema operativo necesario en el doble sentido: por un lado, los requerimientos para ejecutar la aplicación, y por otro, contra qué tipo de hosts y sistemas operativos podrán efectuarse las consultas.
- **Selección de tecnologías a emplear** y obtención del software de desarrollo.

- **Requerimientos de seguridad:** análisis de los requerimientos y permisos necesarios para ejecutar la aplicación y para realizar las conexiones remotas.
- **Preparación de un entorno de test:** disponer de un entorno real o simulado (por ejemplo, con máquinas virtuales), contra el que realizar las pruebas de la aplicación.
- **Módulo principal de la aplicación:** permitirá introducir una lista de hosts a los que escanear. Implementará las capacidades de conexión a equipos remotos y el control de errores: falta de permisos, detección de equipos apagados o inaccesibles
- **Interfaz gráfica:** permitirá la parametrización y mostrará el status en que se encuentran las consultas, así como la visualización de errores capturadas por el núcleo de la aplicación.
- **Módulos adicionales:** desarrollo de los módulos adicionales para cubrir los requerimientos establecidos en la sección “funciones” de los objetivos de este documento.

La herramienta por construir debe ser capaz de rastrear cualquier valor del registro de Windows y las principales configuraciones del hardware y del sistema operativo. En la versión entregable, el producto será capaz al menos de obtener información sobre:

- Registro: cualquier valor del registro de Windows
- Sistema: Sistema operativo, versión, Service Pack, último reinicio, fabricante, modelo, memoria, último usuario
- Unidades de disco: particiones, tamaño, espacio utilizado
- Ficheros: Tamaño, versión, fecha de creación de cualquier fichero, ya sea de usuario o del sistema (como dll's)

1.3 Enfoque y método seguido

El proyecto se enfoca en el desarrollo desde cero de una aplicación que cubra los objetivos citados previamente. Para garantizar el éxito del producto se ha seguido una metodología basada en fases de análisis, diseño, construcción y validación.

La razón de seguir esta estrategia es establecer un proceso lineal, en el que ir resolviendo las tareas específicas a cada ámbito. Así cada fase cerrada es robusta y acota el alcance de las siguientes fases reduciendo el riesgo de errores o de rehacer tareas por una mala definición o diseño.

Fase de análisis

Delimitación de entorno: se ha realizado un estudio de los sistemas operativos más utilizados en las empresas. De este modo orientamos la aplicación a un mayor mercado, pero también establecemos las fronteras de compatibilidad y alcance de la misma.

Selección de tecnología: se evalúan las diferentes tecnologías disponibles que permiten alcanzar los objetivos fijados para el entorno definido en el punto anterior. Además, se ponderan otros factores como el conocimiento previo o el coste.

Requerimientos: se analizan los requerimientos, recursos y medios técnicos necesarios para garantizar que podremos desarrollar el proyecto.

Fase de diseño

Antes de comenzar a desarrollar, se establece una fase de diseño. De este modo prevenimos las dificultades evitando rehacer o redefinir su arquitectura una vez se haya iniciado la fase de programación. En esta fase se ha hecho una marcada distinción entre el módulo principal (núcleo) y los módulos específicos a cada tipo de consulta. El diseño asegura la robustez del módulo principal y a su vez, su versatilidad para integrar módulos específicos.

Fase de desarrollo

En esta fase se construye todo lo definido a nivel teórico en los apartados previos. Es una mera ejecución en el que los requerimientos, recursos y diseños ya están pensados.

Fase de depuración

Construida la aplicación, se prepara un entorno real donde validar que la aplicación cumple todos los requerimientos establecidos. Se ejecutan y corrigen todas las deficiencias hasta conseguir completar satisfactoriamente todos los escenarios de prueba.

Estas fases fueron planteadas desde el inicio y acomodadas en la planificación de las PEC entregadas a lo largo del curso.

1.4 Planificación del Trabajo

1.4.1 Recursos necesarios

Recursos empleados para el desarrollo de la aplicación

A continuación, se describen las necesidades que identificamos para poder desarrollar y probar la herramienta:

- **Visual Studio Community**
Esta versión Community de Visual Studio, ya que es gratuita y la licencia permite su uso para fines educativos. La herramienta puede obtenerse en el siguiente [enlace de descarga](#) [11]
- **PC de desarrollo**
El PC debe cubrir los requisitos de Visual Studio Community. Actualmente contamos con un PC Windows 7 x32, con procesador i5-4300 a 1,9 GH y 4 GB de memoria RAM instalada. Este PC cubre perfectamente los [requisitos para Visual Studio Community](#) [11], por lo que no vemos necesaria la adquisición de hardware adicional
- **.NET Framwerok**
Se instala con el propio Visual Studio. La versión más reciente es accesible desde internet. En la actualidad podemos descargar la última versión desde [este enlace](#).
- **Cuenta “administrador” (*)**
Dado que el acceso a los equipos remotos, tanto para el registro como para WMI requiere elevación de permisos, debemos asegurarnos de que contamos con una cuenta de estas características para ejecutar y testear la aplicación
- **Entorno de pruebas (*)**
Debemos contar con un entorno de PCs y Servidores que cumplan los requisitos ya mencionados para desarrollar y testear la aplicación.

(*) Las primeras fases de desarrollo se harán utilizando dos PCs personales, para después probarlo en un entorno real de empresa, donde podemos probar con diferentes modelos y versiones de PCs y sistemas operativos.

1.4.2 Tareas a realizar

Tarea 1: Delimitación del entorno

Descripción

Los entornos empresariales pueden ser muy heterogéneos en cuanto a la variedad de tipos de dispositivos que se utilizan: dispositivos de oficina, industriales, servidores. También podemos encontrar gran variedad de sistemas operativos: Windows, IOs, Android, Unix/Linux.

Objetivos

Tomar un entorno real donde probar la aplicación y delimitar el alcance. Debemos establecer:

- Sistemas operativos sobre los que ejecutar la aplicación
- Sistemas operativos sobre los que podremos leer

Tarea 2: Selección de tecnología

Descripción

Evaluar las tecnologías disponibles que permitan el desarrollo de una herramienta destinada a la obtención de los parámetros configurados en PCs y servidores. Evaluaremos tecnologías de scripting así como diferentes plataformas y lenguajes de programación.

Objetivos

Partir de un análisis robusto que nos asegure la elección adecuada de las tecnologías a emplear, antes de entrar en las fases de desarrollo. Así conseguimos:

- Contrastar las posibilidades de la tecnología seleccionada con los objetivos definidos en el proyecto para advertir si pueden existir gaps
- Minimizar el riesgo de seleccionar una tecnología inadecuada que nos obligue a cambiar de paradigma a mitad de desarrollo, con la ingente cantidad de tiempo que supondría reescribir el trabajo que ya tengamos desarrollado.
- Obtener una valoración tipo scorecard, para contrastar objetivamente las posibilidades de cada tecnología contra los objetivos descritos en la sección "Objetivos" de este documento,

Tarea 3: Requerimientos tecnológicos y configuraciones previas

Descripción

Evaluar los requerimientos que deben cumplir los dispositivos de red que queremos escanear

Objetivos

Identificar los requerimientos para que sea posible realizar las lecturas de los valores deseados:

- Privilegios necesarios del usuario que ejecuta la aplicación
- Puertos y protocolos que deben estar permitidos en la red
- Evaluación de otros elementos de seguridad que puedan impedir el correcto funcionamiento de la aplicación como por ejemplo antivirus, firewalls, políticas de dominio que deshabiliten funciones de administración remota.
- Contrastar las posibilidades de la tecnología seleccionada con los objetivos definidos en el proyecto para advertir si pueden existir gaps

Tarea 4: Diseño de la aplicación.

Descripción

Definir la estructura modular de la aplicación para organizar las diferentes funciones de la misma: desde las funciones generales y comunes a todas las consultas, hasta las particulares de cada una de ellas. La tarea debe ayudarnos a escribir una aplicación robusta y con una estructura clara que minimice la cantidad de código a escribir cuando sea necesario añadir nuevas funcionalidades.

Objetivos

Tarea 4.1: Diseño del módulo principal:

- **Inputs:**
 - Parámetro que se desea consultar
 - Host(s) sobre los que deseamos hacer la consulta
- **Outputs:** Para ahorrar trabajo, queremos evitar que cada salida escriba su output, por lo que debemos definir un módulo que sea versátil para generar un fichero de output, independientemente de la consulta que se haya realizado
- **Conexión** a hosts remotos: debe ser común a todas las consultas
- **Log:** Mostrar información útil o de errores que se produzcan en cualquier momento de la ejecución.

Tarea 4.2: Diseño de los módulos específicos:

- Un módulo por cada tipo de consulta que queramos ejecutar: leer valor de registro, información del sistema operativo, disco duro y ficheros.
- Todos los módulos deben tener una parte común para poder comunicarse con los módulos generales: recibir de ellos los parámetros de consulta, la lista de hosts sobre la que consultar, y retornar la respuesta al módulo de output.

Tarea 5: Desarrollo del módulo Principal

Descripción

Desarrollo del módulo principal previamente definido en la Tarea 4.

Objetivos

- Obtención una versión inicial plenamente operativa que lleve a la práctica las definiciones realizadas en la Tarea 4.
- Construcción de una primera versión de la interfaz gráfica que permita la usabilidad del módulo principal.

Tarea 6: Desarrollo del módulo Registro

Descripción

Una vez desarrollado el módulo principal, desarrollaremos el primer módulo operativo para la obtención de valores de registro. Éste será el primer módulo que se acoplará al módulo principal en el que probaremos la versatilidad del mismo para acoplarle módulos adicionales y asegurar un buen comportamiento en las interacciones entre ambos módulos: recepción de parámetros y recuperación del output de la consulta.

Objetivos

Obtención de una versión utilizable en la que, desde el módulo principal, podamos personalizar nuestra **consulta de clave de registro** e indicar una lista de hosts reales sobre los que hacer la consulta. Debe obtenerse como output un listado donde veamos la configuración real del parámetro consultado en los hosts del listado introducido.

Tarea 7: Desarrollo del módulo Sistema

Descripción

Construido el módulo Principal y ya enlazado con el módulo de Registro, continuamos el desarrollo de nuevas funcionalidades replicando la estrategia utilizada en el módulo registro.

Objetivos

- Obtención de una versión utilizable en la que, desde el módulo principal, podamos personalizar nuestra consulta indicando **qué valores del sistema queremos conocer** y una lista de hosts sobre los que realizar la consulta. Debe obtenerse como output un listado donde veamos la configuración real del parámetro consultado en los hosts del listado introducido

Tarea 8: Desarrollo del módulo Discos

Descripción

Continuamos la misma estrategia añadir la funcionalidad de Unidades de Disco al módulo principal

Objetivos

- Obtención de una versión utilizable en la que, desde el módulo principal, podamos personalizar nuestra consulta indicando **qué valores de las unidades de disco queremos conocer** y una lista de hosts sobre los que realizar la consulta. Debe obtenerse como output un listado donde veamos la configuración real del parámetro consultado en los hosts del listado introducido

Tarea 9: Desarrollo del módulo Ficheros

Descripción

Continuamos la misma estrategia añadir la funcionalidad de Ficheros al módulo principal

Objetivos de la tarea

- Obtención de una versión utilizable en la que, desde el módulo principal, podamos personalizar nuestra consulta indicando **qué valores de un fichero concreto queremos conocer** y una lista de hosts sobre los que realizar la consulta. Debe obtenerse como output un listado donde veamos la configuración real del parámetro consultado en los hosts del listado introducido

Tarea 10: Depuración y versión final

Descripción

Desarrollados ya el módulo principal y los módulos funcionales, destinamos un tiempo a la depuración de errores menores, mejoras y depuración de la interfaz gráfica para generar una versión entregable del producto.

Objetivos

- Depuración de errores o ajustes necesarios detectados en las tareas anteriores
- Depuración de interfaz gráfica
- Creación del ejecutable entregable

Tarea 11: Ejecución en entorno real

Descripción

Identificación de necesidades en un entorno real y demostración de cómo la aplicación cubre dichas necesidades.

Objetivos

- Identificación y descripción de 2 necesidades

- Ejecución y evidencias
- Valoración del resultado

Tarea 12: Elaboración de entregables

Descripción

Preparación de los entregables para la presentación del TFG.

Objetivos de la tarea

- Memoria TFG
- Presentación TFG
- Ejecutables
- Manuales de uso

1.4.3 Cronograma

Detallado ya el listado de tareas, se proyectan sobre el calendario en un diagrama Gantt para ver gráficamente las tareas agrupadas por hitos y entregables, que a su vez coinciden con las fechas previstas para entregar las PEC.

Tareas		2019																		
		febrero		marzo				abril					mayo				junio			
		21-feb	26-feb	05-mar	12-mar	19-mar	26-mar	02-abr	09-abr	16-abr	23-abr	30-abr	07-may	14-may	21-may	28-may	04-jun	11-jun	18-jun	25-jun
Tarea	PEC1																			
T0	Elección de la temática del TFG																			
T0.1	Preparación PEC 1																			
Tarea	PEC2																			
T1	Delimitación del entorno																			
T2	Selección de tecnología																			
T3	Req. y Conf. previas																			
T4.1	Diseño Módulo principal																			
T4.2	Diseño Módulo específicos																			
T5	Desarrollo Módulo Principal																			
Tarea	PEC3																			
T6	Desarrollo Módulo Registro																			
T7	Desarrollo Módulo Sistema																			
T8	Desarrollo Módulo Discos																			
T9	Desarrollo Módulo Ficheros																			
Tarea	TFG																			
T10	Depuración y versión final																			
T11	Ejecución en entorno real																			
T12	Elaboración de entregables TFG																			

Ilustración 1. Planificación

1.4 Análisis de riesgos

#	Riesgo	Impacto	Medidas correctivas
1	Tecnológico: que la tecnología seleccionada no cubra los requerimientos planteados	Alto	Definir con exactitud el alcance y objetivos del proyecto y validar que la tecnología seleccionada es adecuada y suficiente
2	Funcional: La no funciona en algunas plataformas de usuario	Medio	Recopilar con detalle todos los requerimientos y fijarlos en el manual de usuario.
3	Planificación: Incurrir en retrasos debido a dificultades en la fase de desarrollo	Alto	Acotar con precisión el alcance Monitorizar el plan definido para reconducirlo o redefinirlo si es necesario

Ilustración 2. Análisis de riesgos

1.5 Sumario de productos obtenidos

Los productos obtenidos a la conclusión de este trabajo han sido:

- Aplicación capaz de rastrear los equipos de la red a partir de un listado, y obtener de ellos toda aquella información que el usuario indica a través de un fichero de comandos
- Manual de usuario que incluye la configuración inicial del producto, así como la sintaxis detallada de cada uno de los comandos permitidos
- Video demostrativo de la aplicación para usos didácticos y promocionales

1.6 Descripción de los otros capítulos de la memoria

Siguiendo con la metodología descrita, en los siguientes capítulos pasamos a detallar el trabajo realizado durante las fases de análisis, diseño, desarrollo y depuración.

Así pues, veremos el proceso mediante el cual se ha determinado el alcance y se han seleccionado las tecnologías empleadas. Igualmente se detallará el diseño de la aplicación, tanto del módulo principal como de los módulos específicos. Finalmente se describe el desarrollo, así como las correcciones y mejoras implementadas durante la fase de depuración.

Finalmente se aporta un enlace para ver el producto final en un ejemplo en entorno real.

2. Análisis

2.1 Delimitación del entorno

Sistemas operativos

Las estaciones de trabajo y servidores pueden correr en una gran diversidad de sistemas operativos y versiones diferentes, por lo que, antes de desarrollar una herramienta para monitorizar los sistemas debemos acotar los entornos en los que queramos que sea compatible.

Una forma de dirigir nuestros esfuerzos hacia las plataformas más frecuentes es analizar el marketshare de sistemas operativos de escritorio. De esta forma contrastamos cuáles son los más frecuentes y enfocamos nuestra herramienta a los entornos con mayor potencial de usabilidad.

A continuación, se muestran estadísticas de uso de sistemas operativos de escritorio y servidor:

Desktop			
OS	Version	%	
Windows	7	40,17%	
Windows	10	37,35%	
Windows	8.1	4,87%	
Windows	XP	3,91%	
Windows	all	86,30%	
Mac OS	all	9,65%	
Linux	all	2,14%	
Others	all	1,91%	
Non-Windows	all	13,70%	

Servers			
Source	OS	%	Porpuse
Datanyze	Windows Server	50,4%	<i>Datanyze websites</i>
W3techs	Windows Server	30,1%	<i>Website hosting</i>
Statista	Windows Server	71,9%	<i>Global</i>

Fuentes: [Datanyze](#), [W3techs](#), [Statista](#)

Fuente: Netmarkshare 2018-03 to 2019-02

Ilustración 3. Uso de sistemas operativos de escritorio

En cuanto a **sistemas operativos de escritorio**, vemos que las estadísticas de [Netmarkshare](#) y nos indican que un 86,3% de los sistemas operativos de escritorio corren en Windows. Otras fuentes como [Statcounter](#) y [w3shools](#) otorgan a Windows una cuota de entorno al 75%.

Para a **sistemas operativos para servidores**, sólo Statista nos arroja un porcentaje de uso de servidores Windows sobre el total y no es contrastable con los datos de Datanyze ni W3techs, dado que sus estadísticas sólo contemplan usos web.

En conjunto, vemos que el sistema operativo predominante es Windows, por lo que orientaremos nuestra herramienta a este sistema operativo ya que nos asegura cubrir un amplio espectro del mercado.

Aprovechamos para remarcar que dentro del mundo Windows existen sistemas operativos de 32 y de 64 bits, para los cuales no hemos conseguido estadísticas de uso. En este punto del proyecto no podemos saber si este cambio de arquitectura puede suponer incompatibilidades con la herramienta, o si puede duplicar el esfuerzo de desarrollo hacer que esta sea compatible con ambas arquitecturas. Dada la incertidumbre que esto ocasiona al comienzo del proyecto, inicialmente acotaremos nuestra solución para sistemas de 32 bits, aunque no renunciamos a que si a nivel de desarrollo no añade demasiada complejidad, esta sea compatible también con sistemas de 64 bits.

A modo de conclusión, nuestra herramienta estará orientada **sistemas operativos Windows de 32 bits**

Selección de tecnologías y análisis de costes

En esta sección vamos a determinar con qué tecnología y lenguajes de programación vamos a desarrollar nuestra herramienta. Para ello fijaremos los siguientes criterios de selección:

- **Compatibilidad**
La tecnología escogida debe ser plenamente compatible con entornos Windows de forma nativa, sin necesidad de hacer ajustes o instalar componentes adicionales. Es un requisito indispensable. (peso ponderado 5)
- **Soporte a objetivos específicos**
Debe ser capaz de proporcionarnos los objetivos que definimos en la PEC 1: lecturas de registro de Windows, información del sistema, discos y ficheros. (peso ponderado 4)
- **Potencial y versatilidad del lenguaje**
Tal como recogimos en la PEC1, queremos que la permita ejecutar cualquiera de las consultas predefinidas, modificando los parámetros deseados, pero sin necesidad de modificar el código de la aplicación. Además, debe ser construida de forma organizada, de forma que permita evoluciones futuras añadiendo módulos nuevos. (peso ponderado 3)
- **Bibliografías disponibles**
La existencia de bibliografía es muy importante para contar con fuentes de documentación y ejemplos tanto a la hora de implementar nuestra solución como en la resolución de problemas. (peso ponderado 4)

- **Experiencia en el lenguaje**

Al igual que con la bibliografía, los lenguajes en los que contemos con mayor experiencia son mejor valorados, ya que ayuda a minimizar la curva de aprendizaje (peso ponderado: 5)

- **Costes**

Se trata de un trabajo académico sin fines comerciales, por lo que debemos minimizar los costes. Entornos de desarrollo freeware o con licencia estudiantil son preferidos sobre aquellos que tengan licencias de pago. (peso ponderado: 4).

Recurriendo a la propia experiencia profesional y consultando en internet la forma de obtener mediante programación los parámetros que queremos consultar desde nuestra aplicación, todas las búsquedas conducen a dos tipos de soluciones predominantes: Scripting (vbscript o PowerShell) o programación: tecnologías .NET predominantemente en C#, o programación tradicional en C++.

A continuación, se evalúa cada una de las opciones siguiendo los criterios de acuerdo a los criterios de selección descritos anteriormente-

Scripting (vbScript y PowerShell)

Es el método más recurrido, como ya habíamos mencionado en la PEC1, tiene gran agilidad en el desarrollo ya que no se necesita ningún entorno de desarrollo. En cuanto a compatibilidad, vbScript es totalmente compatible. PowerShell es más moderno y más potente como lenguaje, sin embargo requiere de instalación de componentes específicos en sistemas anteriores a Windows 7 y Windows Server 2008 ([ver detalles](#)). Ambos son capaces de proporcionarnos toda la información que necesitamos. También existe una amplia bibliografía que nos ayudaría en el desarrollo.

La experiencia previa en vbScript es bastante alta, pero no tanto así con PowerShell en el que apenas he trabajado con anterioridad.

Sin embargo, a la hora de crear una aplicación que integre diferentes funciones, no son una solución adecuada, ya que acabaríamos generando una cantidad exagerada de scripts y líneas de código, sin disponer de un IDE que nos permita organizar la aplicación en objetos, clases... etc. para asegurar una correcta escalabilidad de la misma.

Programación C++

Existe una amplia bibliografía y la compatibilidad es nativa en todas las plataformas Windows. Existen IDEs de desarrollo gratuitas como [Eclipse](#) o [Code::Blocks](#). Es un lenguaje potente, orientado a objetos, que también permite trabajar a bajo nivel, lo cual otorga valiosas capacidades para el desarrollo de aplicaciones escalables. En contra, cabe destacar que la experiencia personal

con este lenguaje de programación es escasa y la curva de aprendizaje puede suponer graves retrasos en la planificación del proyecto.

C#

Asegura completa compatibilidad para trabajar con las consultas que queremos realizar: interacción con el registro y con información del sistema a través de WMI. Es un lenguaje de programación basado en .NET Framework de Microsoft que permite la creación de clases y objetos y por tanto, el desarrollo avanzado de aplicaciones y su escalabilidad. También se cuenta con experiencia previa en este lenguaje y tecnologías .NET por proyectos anteriores.

En cuanto a costes, existen versiones freeware de IDEs para desarrollar en .NET como [MonoDevelop](#). Sin embargo, la herramienta específica de Microsoft para desarrollar en .NET es [Visual Studio](#), de la cuál además hay amplia bibliografía y experiencia en el mundo de desarrolladores de C#, lo que nos puede ser de mucha ayuda a la hora de utilizar el IDE y resolver problemas. Esta plataforma tiene un coste de licencia mensual de \$45 o \$499 en licencia perpetua para la edición Profesional. Afortunadamente, la versión **Visual Studio Community** está disponible de forma gratuita para estudiantes y desarrolladores particulares con fines no lucrativos. En este [enlace](#) podemos ver en detalle las condiciones de esta licencia.

A continuación, introducimos estas valoraciones en una tabla de puntuación para obtener una valoración objetiva de las diferentes alternativas:

Criterio	peso	Puntuación				Puntuación ponderada			
		vbscript	powershell	C#	C++	vbscript	powershell	C#	C++
Compatibilidad	5	5	3	5	5	25	15	25	25
Soporte a objetivos	4	3	3	5	5	12	12	20	20
Potencial en desarrollo	3	2	2	5	4	6	6	15	12
Bibliografía disponible	4	4	4	5	3	16	16	20	12
Experiencia previa	5	4	3	5	2	20	15	25	10
Coste	4	5	5	3	3	20	20	12	12
						99	84	117	91

Ilustración 4. Scorecard

Como vemos, atendiendo a los criterios y objetivos fijados, la plataforma que obtiene mayor puntuación es C#, por lo que será la elegida para el desarrollo de este proyecto.

2.2 Requerimientos y configuraciones previas

La aplicación está pensada para funcionar en modo standalone en un pc del administrador de sistemas, y desde ahí poder conectarse al resto de PCs y servidores para extraer la información que en cada caso se requiera.

Como ya hemos explicado anteriormente, hay dos grandes grupos de información que deseamos poder obtener: Información del registro de Windows, e información sobre configuración de Windows.

Vemos detalladamente qué necesitamos para asegurar que cada una de estas categorías puede ser consultada remotamente.

Registro de Windows

Para trabajar con el registro de Windows utilizaremos el servicio Registro Remoto de Windows. Para permitir la conexión al registro remoto [se requiere](#) [12]:

- **Firewall de Windows:** La administración remota debe estar abierta
- **Usuario y contraseña configurados** en el equipo remoto. Si tiene un usuario con password en blanco el acceso remoto al registro fallará
- **Servicio “Registro remoto”** debe estar arrancado tanto en el PC donde ejecutemos la aplicación como en los sistemas a los que vayamos a conectarnos

Dado que la aplicación la desarrollaremos en .NET, deberemos importar el [namespace “Microsoft.Win32”](#) [13] a nuestro proyecto dado que es el namespace que contiene las clases necesarias para el acceso a registros remotos.

Información del sistema WMI

El resto de información que queremos conseguir: Sistema, Discos y Ficheros puede obtenerse a través de la plataforma de administración “Windows Management Instrumentation”, que ya viene integrada en los sistemas operativos Windows, y que está pensada para permitir la gestión y administración remota a través de scripting o programación.

Dado que nos apoyaremos en WMI, pasamos a enumerar los requerimientos (*) que deben cumplir los sistemas a los cuales nos conectemos para extraer información.

- **Firewall de Windows**
Debe estar abierto el tráfico WMI

- **Control cuentas de usuarios**
WMI está protegido por el control de cuentas de usuario (UAC) y requiere elevación de permisos mediante “run as..” para poder interactuar con él, desde una cuenta del grupo de administradores locales. En las conexiones remotas debemos utilizar cuentas de dominio que previamente hayan sido incluidas en el grupo de administradores locales de las máquinas a las que nos vamos a conectar.
- **Configuración DCOM y CIMOM**
Si vamos a trabajar con cuentas que no sean del dominio o sean de dominios con los cuales no se hayan configurado relaciones de confianza, debemos ajustar estos parámetros para permitir este tipo de conexiones. En nuestro caso nos orientaremos sólo al trabajo desde cuentas de dominio incluidas en el grupo de administradores locales de los equipos
- **Versiones de Windows**
WMI está [disponible](#) [14] en todas las versiones de Windows, aunque es posible que algunos componentes no estén en determinadas versiones. Puede consultarse una lista detallada en el siguiente enlace
- **PC donde se ejecuta la aplicación**
Como ya decidimos anteriormente, nuestra aplicación estará desarrollada en C# dentro de la plataforma .NET. Para trabajar en dicha plataforma debemos tener instalado .NET Framework e importar la clase “Microsoft.Management.Infrastructure” tal como recoge [este artículo](#) [15]

(*) En el siguiente enlace pueden consultarse los [requisitos](#) [16] de forma detallada. También podemos consultar una lista de errores frecuentes y [troubleshooting](#) [17]

Requerimientos tecnológicos para el desarrollo de la aplicación

A continuación, se describen las necesidades que identificamos para poder desarrollar y probar la herramienta:

- **Visual Studio Community**
Como ya mencionamos anteriormente, trabajaremos con la versión Community de Visual Studio, ya que es gratuita y la licencia permite su uso para fines educativos. La herramienta puede obtenerse en el siguiente [enlace de descarga](#).
- **PC de desarrollo**
El PC debe cubrir los requisitos de Visual Studio Community. Actualmente contamos con un PC Windows 7 x32, con procesador i5-4300 a 1,9 GH y 4 GB de memoria RAM instalada. Este PC cubre perfectamente los [requisitos para Visual Studio Community](#) [11], por lo que no vemos necesaria la adquisición de hardware adicional

- **.NET Framwerok**
Se instala con el propio Visual Studio. La versión más reciente es accesible desde internet. En la actualidad podemos descargar la última versión desde [este enlace](#) [18].
 - **Cuenta “administrador” (*)**
Dado que el acceso a los equipos remotos, tanto para el registro como para WMI requiere elevación de permisos, debemos asegurarnos de que contamos con una cuenta de estas características para ejecutar y testear la aplicación
 - **Entorno de pruebas (*)**
Debemos contar con un entorno de PCs y Servidores que cumplan los requisitos ya mencionados para desarrollar y testear la aplicación.
- (*) Las primeras fases de desarrollo se harán utilizando dos PCs personales, para después probarlo en un entorno real de empresa, donde podemos probar con diferentes modelos y versiones de PCs y sistemas operativos.

Resumen de requerimientos

La siguiente tabla muestra un resumen de todos los requerimientos identificados hasta ahora, así como la comprobación de que cumplimos cada uno de ellos para poder iniciar las siguientes fases de diseño y desarrollo.

Función	Requisito	Status para empezar
General	- Microsoft Windows x32 (all versions)	OK
Acceso registro remoto	- Firewall Windows: permitir administración remota	OK
	- Usuario y password configurados (no en blanco)	OK
	- Servicio "Registro remoto" running	OK
WMI	- Firewall Windows: permitir tráfico WMI	OK
	- UAC: cuenta de dominio incluida en administradores locales de los equipos	OK
	- DCOM y CIMOM: no requiere ajustes si trabajamos con cuentas de dominio	OK
Desarrollo	- PC con Windows x32	OK
	- Visual Studio Community	OK
	- .NET Framwork (versión más reciente)	OK
	- Cuenta "administrador" de los equipos remotos	OK
	- Entorno de pruebas con PCs y Servidores	OK

Ilustración 5. Resumen de requerimientos

3. Diseño

3.1 Módulo principal

Recuperamos los objetivos descritos para este módulo que habíamos definido en el capítulo 1:

- **Inputs:**
 - Parámetro que se desea consultar
 - Host(s) sobre los que deseamos hacer la consulta
- **Outputs:** Para ahorrar trabajo, queremos evitar que cada salida escriba su output, por lo que debemos definir un módulo que sea versátil para generar un fichero de output, independientemente de la consulta que se haya realizado
- **Conexión** a hosts remotos: debe ser común a todas las consultas
- **Log:** Mostrar información útil o de errores que se produzcan en cualquier momento de la ejecución.

Inputs

- **Host(s)**

Este dato será provisto a partir de un **fichero de texto** plano, con un nombre de equipo remoto por cada línea. La aplicación debe recorrer este fichero.
- **Comandos**

Deseamos que nuestra aplicación sea lo más versátil posible, por lo que dejaremos al usuario un alto grado de libertad para parametrizar su consulta. En este sentido optamos por que el usuario introduzca las consultas **mediante un fichero de texto**, de manera que pueda guardar varios ficheros con consultas diferentes que le puedan ser interesantes, y en cada momento ejecutar la más oportuna.

Esta decisión da mayor versatilidad, pero también introduce algunas dificultades, ya que debe definirse una forma unificada de introducir las consultas de forma que la aplicación sea capaz de comprobar que la consulta introducida es correcta antes de tratar de ejecutarla.

A la hora de definir el formato de entrada de las consultas, debemos tener en cuenta los 5 tipos de consultas que nos hemos fijado como objetivo y cómo estas se realizarán a partir del lenguaje C# que vamos a utilizar, de esta forma podemos establecer con mayor seguridad el formato de entrada de las consultas para que luego sean procesadas por la aplicación.

Se decide implementar una clase que haga de brocker para los diferentes comandos. Su cometido será recibir un comando y verifica que existe, está correctamente escrito e invocar al método concreto que corresponda para la obtención de la información

Output

Uno de los objetivos fijados en Capítulo 1 era que el output sea manejable, por lo que la mejor opción es que sea procesable en Excel. Sin embargo, si la cantidad de datos es muy pequeña, a veces es suficiente un simple fichero CSV, que se abre desde cualquier procesador de textos sin esperar a que se abra toda la aplicación de Excel.

En este sentido decidimos que la aplicación trabajará nativamente con un CSV y una vez finalizada la consulta, contaremos con dos botones para abrir el output: o bien en CSV o bien en un Excel formateado que creará la aplicación de forma automática a partir del CSV.

Después del análisis anterior observamos que el output puede resultar complejo de gestionar si pretendemos hacerlo de una forma unificada desde el módulo principal, sin importar de qué consulta provenga, ya que el número de filas y columnas que este pueda tener es variable e indeterminado.

Para abordar este problema la aplicación debe ser capaz de analizar previamente la consulta, ver qué parámetros se han consultado para generar la cabecera del fichero. Si esperamos que haya varias líneas de respuesta por cada equipo consultado (como por ejemplo para los discos o impresoras), debemos asegurar que el volcado no acaba en la primera línea.

Para manejar todas estas dificultades se trabajará con arrays de dos dimensiones, así conseguimos tener en un único objeto todo el output, sin importar el número de columnas y filas que nos devuelva la consulta. De esta forma también podemos volcar su contenido a un CSV con sólo escribir cada elemento de cada vector separados por comas, e introducir un salto de línea al pasar de un vector a otro en el array bidimensional. Una vez tenemos el fichero CSV, hacer que la aplicación lo importe a un Excel de forma automática resultará más sencillo.

Conexión

La propuesta al inicio del proyecto fue que la conexión a los hosts para realizar las consultas se haga desde el módulo principal. Sin embargo, al profundizar en el diseño, vemos que cada consulta se hará apoyándonos en diferentes clases de C# y que estas ya manejan directamente la conexión al equipo remoto.

Para conseguir un enfoque lo más unificado posible, haremos que el módulo principal sea el encargado de recorrer el fichero de hosts y verificar la conexión

mediante un ping a cada uno de ellos. Si la conexión es correcta pasará el control a las clases específicas a la consulta introducida. Si la conexión es incorrecta lo reflejará en el log y en el fichero de output como “no ping” para ese host.

Log

La aplicación dispondrá de una ventana a modo de consola o log, donde visualizaremos en tiempo real el comportamiento de la aplicación y el progreso de las consultas. Esta ventana mostrará:

Progreso de la consulta indicando el número de hosts escaneados y pendientes de escanear

Errores o excepciones que puedan ocurrir durante la ejecución y sean de utilidad para la resolución de errores.

Al poco de empezar a programar la aplicación nos damos cuenta de que para conseguir esto, la aplicación debe ser **multitarea**, ya que no es capaz de actualizar el formulario principal al mismo tiempo que ejecuta las consultas.

Tras todo este análisis a continuación se expone el diagrama de componentes propuesto como diseño del módulo principal. Tal como puede comprobarse en los diagramas que se muestran a continuación, finalmente se ha considerado oportuno agregar algunos módulos auxiliares para dar soporte a las complejidades de los outputs múltiples descritas anteriormente. De esta manera estas complejidades quedan aisladas en módulos y es mucho más escalable a la hora de capturar problemas o introducir modificaciones.

Directorios de trabajo

La aplicación debe contar con directorios predefinidos en los que albergar los ficheros de input (hosts a analizar, comandos a ejecutar), y fichero de output (resultado de la consulta)

Diagrama de flujo del módulo principal

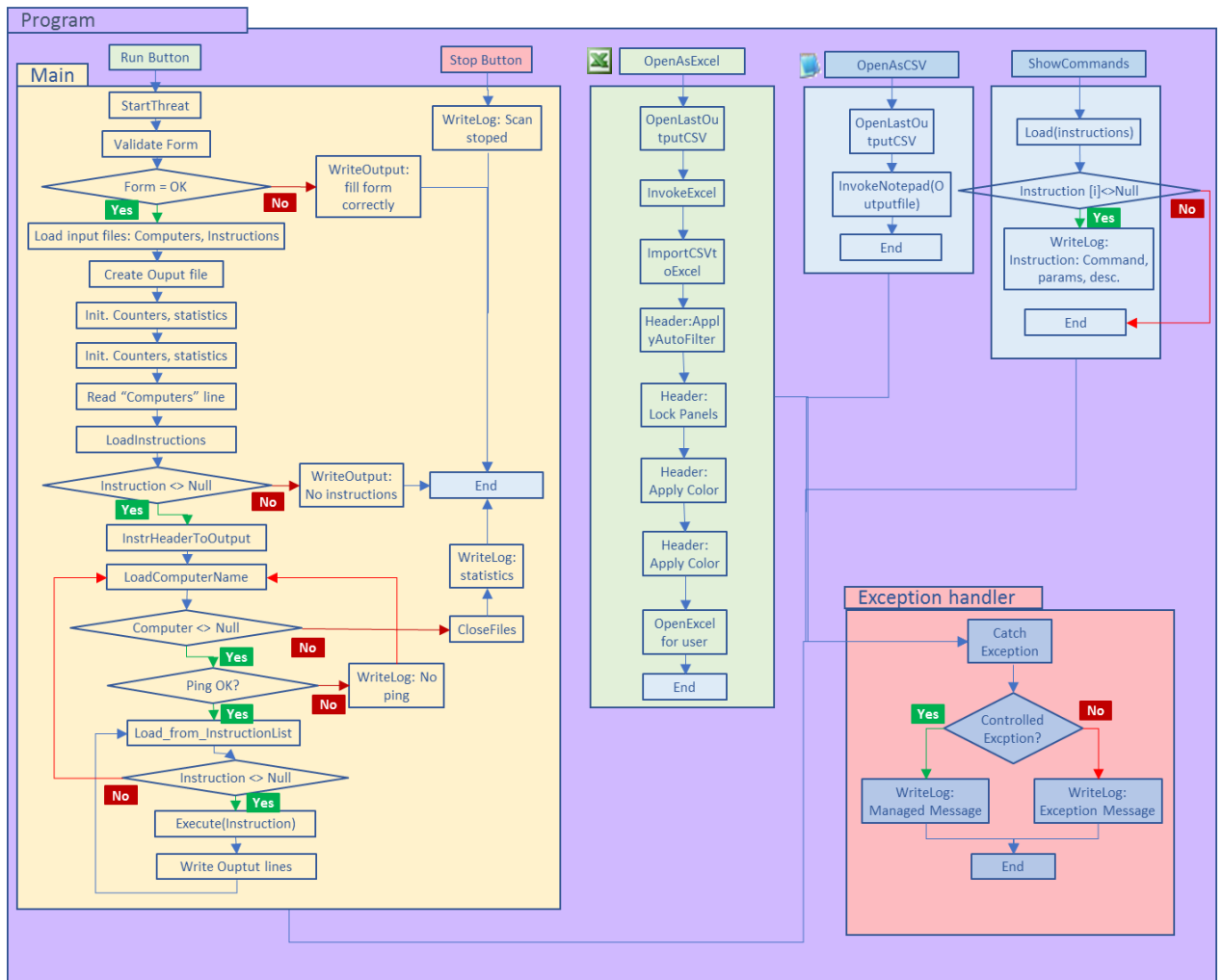


Ilustración 6. Diagrama de flujo del módulo principal

** **ExceptionHandler:** en todos los módulos se implementa la captura de excepciones. Para aquellas que son controladas se reporta al log de actividades un mensaje específico que ayude al usuario a corregir el error. Para las excepciones no controladas se volcará al log de actividades el mensaje propio de la excepción

** **WriteLog:** es una función que se ha implementado, invocable desde cualquier punto de la aplicación, para escribir de forma sencilla pero formateada y tabulada al log de actividades

Como puede observarse, al mismo tiempo que se está ejecutando el scan, hay que escribir en el log y los listener de botones deben estar disponibles (por ejemplo, para parar el scan en curso). Esto hace que la aplicación sea implementada mediante diferentes threads (hilos)

A continuación, se muestra una la interfaz gráfica del módulo principal y una descripción de sus principales botones:

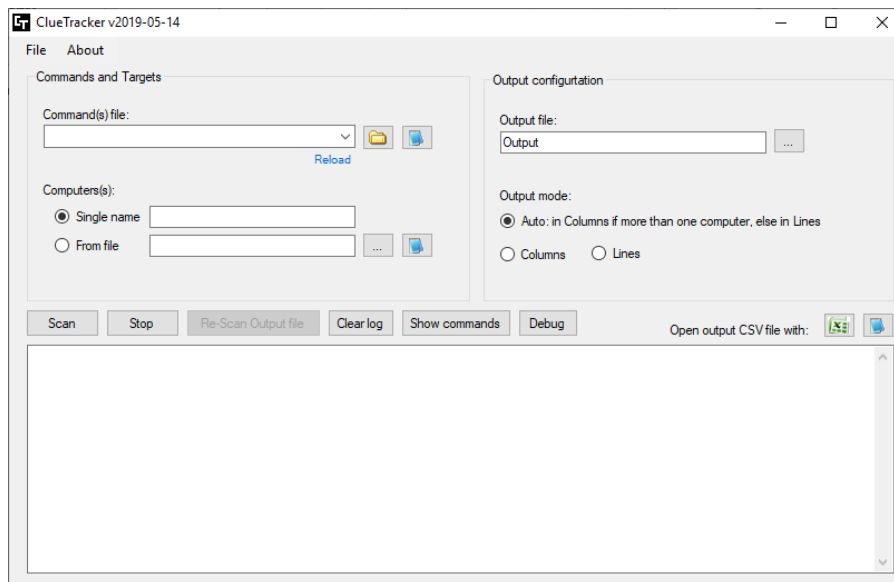


Ilustración 7. Interfaz gráfica del módulo principal

- **Commands:** Permite la selección del fichero de texto con el/los comandos que queremos ejecutar en la consulta. Se acompaña de un botón para explorar la carpeta donde están todos los ficheros que vayamos creando, y un segundo botón para abrir directamente la edición del fichero y poder modificarlo sin tener que navegar hasta el fichero por el explorador de Windows.

Computers: En este apartado señalamos el host o hosts sobre los que queremos lanzar la consulta. En la fase de pruebas se observó que muchas veces, sólo queremos lanzar la consulta sobre un equipo y resultaba incomodo tener que crear/editar un fichero para un solo equipo, por lo que se modificó el formulario con la opción de teclear directamente un nombre para simplificar el proceso. . Al igual que commands, se acompaña de un botón para abrir el directorio donde se encuentran los ficheros con listados de PCs y un segundo botón para editar directamente dichos ficheros.

Output: Indicamos un fichero csv sobre el que volcar el resultado. Nos permite configurar si queremos el fichero organizado en filas secuenciales (una fila por cada parámetro recibido de cada host consultado), o en columnas (una fila por cada host consultado y una columna por cada parámetro recibido del host).

- **Run:** es el botón principal y el que lanza el scan sobre los hosts indicados en el fichero con las instrucciones indicadas en el fichero de instrucciones.
- **Stop:** Permite cancelar un scan en ejecución.

- **OpenAsExcel:** abre el último CSV generado en un fichero Excel formateado y con filtros.
- **OpenAsCSV:** abre el último CSV directamente en un Notepad.
- **ShowCommands:** Aprovechando la arquitectura de la aplicación y el módulo “InstructionBrocker” que nos da visibilidad de los comandos disponibles, se implementa un botón para que el usuario pueda ver los comandos disponibles, así como la sintaxis que debe emplear para escribirlos

3.2 Módulos específicos

En el módulo principal hemos definido la arquitectura de la aplicación para que pueda recibir información proveniente de los diferentes módulos que implementaremos después.

Tal como explicábamos anteriormente, esto lo conseguimos mediante la clase “**Instruction**”, que estandariza la manera en que se escriben las instrucciones, y la clase “**OperationsBrocker**” que pasa las instrucciones a los módulos específicos y recibe de estos la información en objetos “**Data_moduleX**”, que representan el modelo de datos de lo que estamos consultando. También deben crearse objetos “**Access_moduleX**”, que nos darán la capa de acceso para cada consulta: ya sea que vamos a consultar el registro de Windows, impresoras... etc.

Definidos los componentes y las clases que hacen de interfaz entre unos y otros, podemos ya pasar a diseñar cada uno de los módulos para que se acoplen en esta arquitectura.

El módulo **Registro** es un caso particular, ya que además de lecturas va a permitir hacer actualizaciones de valores. El módulo **System** lo tomamos de ejemplo para explicar la modelización de datos múltiples a un objeto. El **resto de módulos** siguen exactamente el mismo patrón que System, por lo que los explicamos más brevemente centrándonos sólo en las diferencias.

.

Módulo “Registro de Windows”

Para trabajar con el registro de Windows utilizamos la clase [Microsoft.Win32.RegistryKey](#) [12.1], que nos permite consultar y actualizar el registro de hosts remotos.

Acceso operaciones: Access Registry

Esta clase nos proporcionará todos los métodos necesarios que queramos implementar para acciones sobre el registro.

Antes de comenzar la ejecución de cada método propio, ejecuta el método Validate() de la instrucción que ha recibido para asegurar que la sintaxis del comando introducido por el usuario es correcta.

Inicialmente pensábamos únicamente en consultar valores en claves concretas. Sin embargo, con esta arquitectura hemos visto la oportunidad de mejorar el enfoque inicial. Así pues, pasamos a implementar:

- **BuscarClave:** dada una rama de registro y un nombre de clave a buscar, se indica también una profundidad de búsqueda, hasta la cual el programa recorrerá las diferentes subclaves para localizar el valor. Lógicamente si ya sabemos la rama exacta, le daremos una profundidad de 1. Sin embargo, en ocasiones sabemos el nombre de la clave, pero dependiendo de la versión del producto, los nodos anteriores pueden variar en nombre.

```
private static string RegValReadSearch(string computerName, string[] args)
```

- **ActualizarClave:** aprovechamos la opción de búsquedas anterior para además de leer una clave, poder también actualizarla.

```
private static string RegValUpdateSearch(string computerName, string[] args)
```

El output nos devolverá un string indicando si se ha encontrado la clave, la rama concreta del registro en que se encontró. Si además estamos haciendo una actualización, nos indicará el valor anterior y el valor después de la actualización (de gran utilidad por si hacemos un cambio masivo que por cualquier motivo tengamos que revertir)

Este enfoque **mejora sustancialmente el objetivo inicial**.

Módulo “Sistema”

Para obtener información del sistema operativo nos basamos en las clases [Win32 OperatingSystem](#) [19] y [Win32 ComputerSystem](#) [20]. Entre las dos clases sumamos más de 130 parámetros que podríamos consultar. Aprovechando que la arquitectura definida nos permite representar en un solo objeto los parámetros que deseamos de ambas clases, seleccionamos los más frecuentes.

Acceso a operaciones: Access_SysInfo

Al igual que con el registro, esta clase nos provee los métodos relativos a las opciones del comando introducido por el usuario, para el cual también comprueba la sintaxis. En este caso es mucho más sencillo ya que la única operación es la consulta de la información del sistema.

```
public static string SysGetInfo(string computerName, string requestedAttributes)
```

A pesar de tener sólo una operación disponible para la lectura de valores, se decide mantener la misma arquitectura que utilizamos para el registro. Esto lo respetaremos en todos los módulos para dar versatilidad a la aplicación y poder añadir operaciones de forma sencilla si fuera necesario.

Modelo de datos: Data_SysInfo

A diferencia del registro donde el ouptu siempre era un valor único, para el resto de módulos implementamos el modelo de Datos que explicábamos en los capítulos anteriores. Así definimos el objeto “Data_SysInfo”, que modeliza en una clase los datos. Esto nos permitirá volcar en él el resultado obtenido de consultar esta información sobre un host. Si en lugar de un solo host fueran varios, se crea un array de estos objetos para tener perfectamente accesible la información relativa a cada host.

El constructor de esta clase recibe el nombre del host al que vamos a consultar, y los atributos que deseamos como un string separados por comas. Antes de llegar a este punto, la clase OperationsBrocker se habrá en cargado de validar la sintaxis de este string:

```
public Data_SysInfo(string computerName, string strRequestedAttributes)
```

Este constructor implementa las consultas necesarias a las clases Win32_ComputerSystem [19] y Win32_OperatingSystem [20] y vuelca los datos a los atributos de la clase. Implementa también un control de errores y excepciones, como por ejemplo, falta de permisos para hacer las consultas. Cabe destacar que en este punto ya no se comprueba que el host esté encendido, ya que la comprobación del ping se hace en la parte principal del programa, antes de pasar el control a esta clase:

```
ManagementObjectSearcher searcher = new ManagementObjectSearcher(scope, new ObjectQuery("SELECT * FROM Win32_BIOS"));  
searcher = new ManagementObjectSearcher(scope, new ObjectQuery("SELECT * FROM Win32_OperatingSystem"));
```

Módulo “Discos”

Se basará en la clase [Win32_LogicalDisk](#) [21]. Siguiendo la misma arquitectura explicada anteriormente, contaremos con la clase “Access_LogicalDisk” que nos proporcionará las operaciones (en esta versión sólo lecturas), y la clase “Data_LogicalDisk”, que nos mapeará los valores obtenidos a un objeto fácilmente accesible. Como ya explicamos, si la consulta es a múltiples hosts, obtendremos un array de objetos.

Output: será multilínea, ya que un host puede tener varios discos enlazados. Por eso este parámetro se recoge en un array:

```
private ArrayList logicalDiskList = new ArrayList();
```


Módulo “Impresoras”

Se basará en la clase [Win32 Printer](#) [22]. Es muy similar a Discos, ya que como **input** tenemos diversos parámetros seleccionables para ser consultados, y el **output** dependerá de la cantidad de impresoras que haya instaladas en el sistema

Módulo “Ficheros”

Se basará en la clase [System.IO.FileInfo](#) [23] de C#. En el **input** indicaremos el directorio del fichero y las propiedades del fichero que queremos consultar. En el **output** obtendremos el valor para las propiedades consultadas

4. Desarrollo y depuración

En este capítulo describimos el proceso de desarrollo y depuración que se ha seguido. También se explica la preparación del entorno de test que asegura las pruebas en las diferentes arquitecturas.

4.1 Construcción de entorno de test

Para depurar correctamente la aplicación necesitamos contar con un entorno en que haya diferentes sistemas operativos en arquitecturas x86 y x64 y permisos adecuados para ejecutar las consultas que requieren privilegios de acceso a máquinas remotas.

En este sentido hemos dispuesto lo siguiente:

- Windows Active Directory 2008
- PCs y Servidores en arquitecturas X86 y x64 pertenecientes al dominio (ver inventario). Nos aseguramos de contar con equipos en la LAN y también en la WAN para probar el rendimiento sobre hosts remotos.
- Cuenta de dominio con privilegios administrativos sobre los equipos del inventario

Hostname	Description	Location	Type	OS	Arch.
computer1	Javier Prod	LAN	Laptop	Windows 10 Pro	x64
computer2	Raúl Test	LAN	Laptop	Windows 10 Pro	x64
computer3	Wharehouse	WAN	Desktop	Windows 10 Pro	x64
computer4	IT PC	WAN	Desktop	Windows 10 Pro	x64
computer5	IT MM	LAN	Desktop	Windows 10 Pro	x64
computer6	IT MA	LAN	Desktop	Windows 10 Pro	x64
computer7	Javier Test	LAN	Desktop	Windows 7	X32
computer8	Distribution Center	WAN	Desktop	Windows 7	x86
computer9	Manager Laptop	WAN	Laptop	Windows 7	x86
computer10	Materials	WAN	Laptop	Windows 7	x86
computer11	IT EG	WAN	Laptop	Windows 7	x86

server1	FileServer1	LAN	Server	W2008	x64
server2	FileServer2	WAN	Server	W2003	x86
server3	AppServer1	WAN	Server	W2008 R2	x64
server4	AppServer2	WAN	Server	W2008 R2	x64
server5	AppServer3	WAN	Server	W2008 R2	x64
server6	AppServer4	WAN	Server	W2008 R2	x64

Ilustración 8. Entorno de pruebas

** Por motivos de confidencialidad se han ocultado los nombres reales de los equipos. Para poder trabajar con los nombres mostrados en esta memoria, se han añadido como alias en el fichero hosts del PC en que se ejecuta la aplicación.

3.2 definición de conjunto de pruebas

El conjunto de pruebas a realizar debe orientarse a garantizar el cumplimiento de los requerimientos establecidos al inicio del proyecto y también a asegurar la robustez de la aplicación. Para ello, retomamos los requerimientos establecidos en el Capítulo 1, a partir de los cuales establecemos las pruebas a realizar y el resultado esperado, para evaluar si el resultado es satisfactorio.

Para todas las pruebas utilizamos equipamientos que cumplen los requerimientos establecidos en la PEC2 relativos a permisos de acceso, firewall, registro remoto e instalación de .NET Framework... etc.

Resultado ejecución tests de validación

Requerimiento	Test ID	Test a realizar	Pass / No Pass
Generales			
Standalone	1	Scan hosts remotos sin realizar configuraciones previas	OK
Instalación sencilla	2	La aplicación se inicializa correctamente sólo con abrir el ejecutable sin hacer configuraciones previas	OK
Minimización costes	3	Usabilidad completa sin necesidad de licencias	OK
Específicas			
Registro: Buscar Clave	4	Retorno del valor del registro consultado. Output muestra: - Offline: si está inaccesible - Error Permisos: si no tenemos privilegios	OK
Registro: Buscar subclaves	5	Retorno del valor del registro consultado.	OK
Registro: Actualizar valor	6	Sobreescribir valor. Output muestra: nuevo valor	OK
Registro: Actualizar valor en subclave	7	Sobreescribir valor. Output muestra: nuevo valor	OK
Sistema	8	Retorno de todos los parámetros consultados.	OK
Unidades de disco	9	Retorno de todos los parámetros consultados.	OK
Ficheros	10	Retorno de todos los parámetros consultados O Fichero no encontrado.	OK
Impresoras	11	Retorno de todos los parámetros consultados.	OK
Usabilidad			
Guardado de consultas	12	Las consultas guardadas son seleccionables y reconfigurables cada vez que abrimos la aplicación	OK
Inicialización del formulario	13	El formulario se abre con la última selección realizada por el usuario en cada uno de sus parámetros seleccionables	OK
Fichero multi-consulta	14	Retorno en un solo fichero de output de todos los valores definidos en el ejemplo.	OK
Output	15	Fichero Excel formateado, con cabeceras y filtros	OK
Errores de sintaxis en comandos	16	El log de la aplicación muestra el error detectado, indicando la línea conflictiva y no continúa con la ejecución	OK
Errores en ejecución	17	Los errores detectados al acceder a host remotos son capturados y reportados como mensaje en el output, para no interrumpir el escaneado del resto de hosts	OK
Errores de permisos	18	Se refleja en el output para cada host	OK

Ilustración 9. Pruebas y resultados

(*) Todos los TestID marcados en negrita corresponden a funcionalidades adicionales que no estaban previstas en los objetivos iniciales del proyecto.

Los resultados mostrados en la tabla son los finales, tras todo el trabajo de depuración mejoras que se han realizado para conseguir poner OK en todos ellos.

Dado que algunos casos de test han sido especialmente relevantes, en la siguiente sección de la memoria pasamos a explicar con más detalle las acciones realizadas en algunos de ellos.

3.3 Valoración y mejoras

En esta sección nos centramos en describir las correcciones y mejoras más relevantes que se han hecho durante la fase de depuración. Algunas de estas mejoras no eran parte de los objetivos iniciales, sin embargo se ha visto la oportunidad de aplicarlas para dar mayor valor a la solución final.

3.3.1 Ejecución de la aplicación en Windows x64

La versión de la PEC2 no fue totalmente validada en la última versión del sistema operativo de Microsoft (Windows 10). Aunque en los objetivos iniciales del proyecto se especificaba que esto no era un requerimiento, dado que nos hemos adelantado en el plan, consideramos oportuno hacer esta validación para dar más valor al resultado final.

Debemos tener en cuenta que la aplicación debe ser compatible en el doble sentido: por una parte poder correr en máquinas Windows x86 o x64, pero también poder conectarse a hosts remotos x86 y x64 para obtener la información que consulta cualquiera de los módulos.

Tal como encontramos en la [documentación](#) [24] de Microsoft, Visual Studio nos permite utilizar la opción “AnyCpu” para compilar nuestros proyectos. Esto hace que el ensamblado corra en modo nativo por defecto en máquinas de 64 bits y recurra a la compilación JIT para la ejecución en otras plataformas. Este tipo de compilación multi-plataforma podría causar algún problema de lentitud en aplicaciones que fueran muy exigentes a nivel de procesador, pero no es nuestro caso, por lo que consideramos que es la opción más cómoda y versátil para compilar nuestro proyecto y asegurar que corre en cualquier plataforma Windows con .NET 4.5.

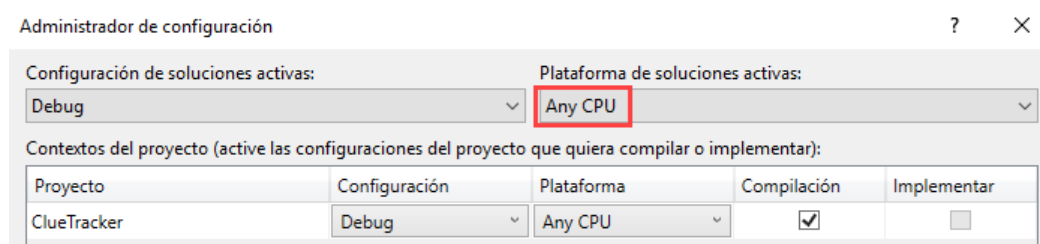


Ilustración 10. Compilación

Aclarada la parte teórica, hemos procedido a la compilación del proyecto utilizando esta opción. El resultado ha sido realmente satisfactorio: la compilación se ha hecho a la primera, sin detectarse ningún error. Igualmente se ha validado el ejecutable en las máquinas de desarrollo descritas en la sección “Entorno de test”. La ejecución en todos los casos ha sido correcta a la primera.

3.3.2 Ejecución de la aplicación contra máquinas Windows x64

En esta ocasión debemos asegurar que todos los módulos que implementa la aplicación son capaces de extraer información de los sistemas remotos, ya sean Windows x86 o x64.

Problema: No es posible leer el registro de máquinas x64

En la versión entregada en la PEC2 se validaron algunos módulos, pero haciendo el análisis en detalle observamos que todos los módulos funcionan correctamente contra equipos x64, salvo el módulo de registro. Esto se debe a que la compatibilidad está garantizada por .NET para todos los módulos que trabajan con WMI. Sin embargo, el módulo “Registro” presenta problemas con la lectura de valores en máquinas de 64 bits.

Solución: utilizar la vista [Registry64](#) [25] al abrir el registro. De acuerdo con la documentación de MS, es posible invocar el método [RegistryKey.OpenRemoteBaseKey](#) [26] con el parámetro Registry64, que por defecto devuelve una vista del registro remoto en 64 bits, pero que si detecta que es de 32, no hace ninguna alteración. Este parámetro es opcional, y en la implementación inicial no se había utilizado. Se ha verificado que al implementarlo los resultados son correctos en todas las combinaciones de ejecución y conexión remota en x86 y x64

A continuación se muestra un extracto de la modificación realizada en el código:

```
ServiceController sc = new ServiceController("REMOTEREGISTRY", computerName);  
  
if ((regHive == "HKCR") || (regHive == "HKEY_CLASSES_ROOT"))  
    baseRegistryKey = RegistryKey.OpenRemoteBaseKey(RegistryHive.ClassesRoot, computerName, RegistryView.Registry64);
```

3.3.3 Portabilidad

Errores de inicialización de variables: Al ejecutar la aplicación en PCs donde no se había ejecutado antes, se producen algunos problemas e impactos en la usabilidad de la aplicación

Problema: Root path de la aplicación no establecido

En el primer uso: causa que un usuario no habituado a la aplicación no sepa cómo proceder para encontrar las carpetas con los scans y listados de PCs a escanear

Solución: En el rediseño de la aplicación se elimina el root path del formulario y se inicializa automáticamente. Para ello, se parte de la ubicación del exe y se sube dos directorios arriba. A partir de ahí se encuentra la carpeta “Scans” que contiene el resto de directorios necesarios para el usuario: QuickScans, Commands y Output en la raíz jerárquica de la carpeta de la aplicación. En el fichero AppConfig se inicializa dicha variable y se establecen las subcarpetas de Command Files, Computer Files y Output Files.

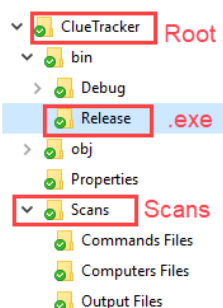


Ilustración 11.
Estructura de directorios

```
private void initializeFoldersPaths(Data_FormMirror frm)
{
    //AppFolder: Initialize with value reported in config file
    appDefaultFolder = ConfigurationManager.AppSettings["txtAppFolder"];
    if (appDefaultFolder == "")
    {
        //Initialize folder where exe main folder is located
        DirectoryInfo df = new DirectoryInfo(Environment.CurrentDirectory);
        df = new DirectoryInfo(df.Parent.FullName);
        df = new DirectoryInfo(df.Parent.FullName);
        appDefaultFolder = df.ToString();
        frm.txtAppFolder = appDefaultFolder;
    }
}
```

Ilustración 12. Inicialización de directorios

Problema: validación de licencia

Las dlls que se han creado para el encriptado y validación de licencias no estaban incluidas en la carpeta del proyecto. Esto produce una excepción en la primera ejecución porque no puede validarse la licencia introducida.

Solución: Se depura la aplicación y se comprueba que la excepción se produce al tratar de usar `qatLicVal.dll` para descryptar la clave de licencia introducida y ver si es válida. Se comprueba que dejando una copia de la dll en el mismo directorio del ejecutable el error desaparece.

3.3.4 Usabilidad

Problema: escanear un único equipo

Resulta incómodo tener que crear un fichero para escribir en el nombre de un equipo, cuando el scan queremos realizarlo sobre una única máquina.

Solución: se crea un radiobutton para seleccionar el escaneado de un solo equipo, junto a un textbox para introducir el nombre del equipo a escanear.

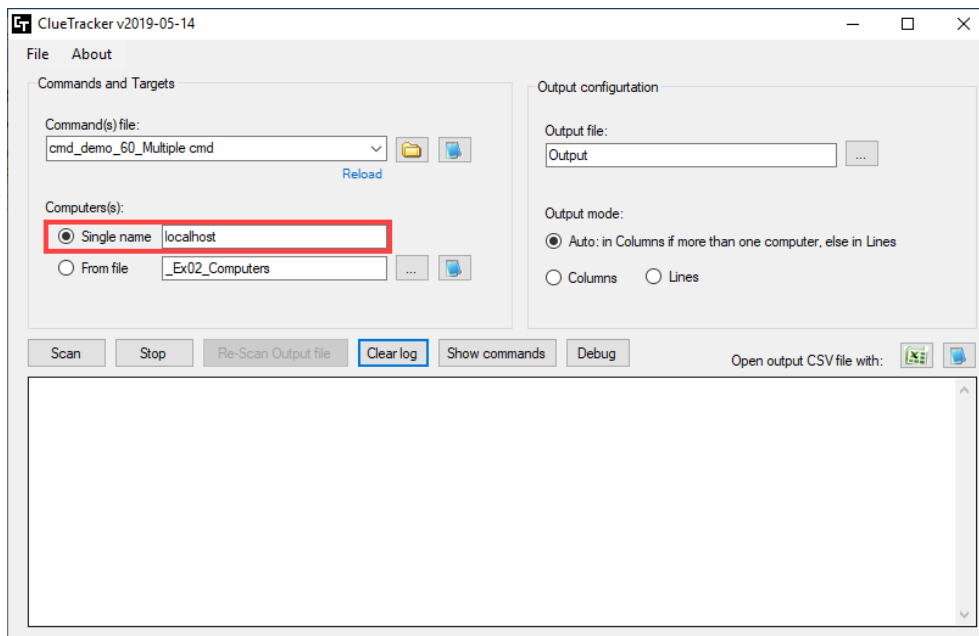


Ilustración 13. Interfaz gráfica mejorada

Problema: Selección de parámetros en cada ejecución

Al realizar las pruebas se ha detectado que resulta muy molesto seleccionar cada vez los parámetros con los que queremos lanzar el scan. En muchas ocasiones queremos repetir un mismo scaneado, o sólo cambiamos el listado o equipo sobre el que queremos realizar la acción.

Solución: Guardar último estado del formulario

Se implementa una clase que hace de espejo entre el formulario, y los valores fijados en el App.config. De este modo cada clic que hace el usuario para modificar algún parámetro del formulario, es reflejado en el fichero App.config. Dado que la aplicación al abrirse carga los parámetros de ese fichero, siempre se abrirá inicializando el formulario tal como el usuario lo dejó la última vez.

Este apartado va relacionado con el problema 1. En la primera ejecución, el App.config no tiene cargado el parámetro de carpeta root de la aplicación. Esto es controlado e inicializado por el programa para que no cause un error.

```

1  <?xml version="1.0"?>
2  <configuration>
3  <startup>
4      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
5  </startup>
6  <appSettings>
7      <add key="AppVersion" value="ClueTracker v2019-05-14"/>
8      <add key="txtAppFolder" value="" />
9      <add key="QuickScansMainFolder" value="Scans" />
10     <add key="QuickScansCommandsFilesSubFolder" value="Commands Files" />
11     <add key="QuickScansCommandsFileDefaultName" value="" />
12     <add key="QuickScansComputerSubFolder" value="Computers Files" />
13     <add key="QuickScansOutputFilesSubFolder" value="Output Files" />
14     <add key="QuickScansOutputFileDefaultName" value="Output.csv" />
15
16     <add key="SavedScansMainFolder" value="Saved Scans" />
17     <add key="QuickScansCommandsFilesSubFolder" value="Commands Files" />
18
19     <add key="txtCommandsFile" value="" />
20     <add key="rbCompsSingleName" value="True" />
21     <add key="txtCompSingleName" value="" />
22     <add key="rbCompsFromFile" value="False" />
23     <add key="txtComputersInputFile" value="" />
24     <add key="txtOutputFile" value="" />
25     <add key="rbOutAuto" value="True" />
26     <add key="rbOutColumns" value="False" />
27     <add key="rbOutLines" value="False" />
28
29 </appSettings>
30
31 </configuration>

```

Ilustración 14. Inicialización de parámetros

Problema: Legibilidad del output en consultas múltiples

Como ya explicamos en la PEC2, la aplicación nos permite combinar en un único fichero de comandos, más de una consulta. Esto nos evita tener que lanzar un escaneado diferente por cada parámetro a consultar. Sin embargo, dificulta la lectura del fichero de output, ya que en un mismo fichero encontramos datos de diferente tipo y dimensiones. Por ejemplo, si consultamos una clave de registro y las impresoras asociadas, por cada PC obtendremos: 1 línea con una columna del valor del registro que estamos consultando, y X líneas y N columnas por cada una de las impresoras que encontremos en ese PC.

Solución: Añadir cabecera con filtros para cada comando

Se opta por añadir una cabecera al inicio el fichero de output, con una línea por cada uno de los comandos utilizados. Para ello se procesa primero el fichero de comandos, se identifican los IDs de cada uno de ellos y se escriben las líneas de cabecera.

Además, al pulsar el botón de abrir en Excel, la aplicación aplicará un formato automático para facilitar aún más su lectura: Inmoviliza las líneas que corresponden a la cabecera, las aplica un color diferente y establece autofiltros.

De esta manera, aunque el fichero contiene diferente número de filas y columnas para cada uno de los comandos consultados, es posible filtrar por el ID del comando y obtener vistas simplificadas para cada uno de ellos.

En la siguiente imagen vemos un ejemplo de un informe ejecutado contra todos los equipos del entorno de test en el que estamos mostrando toda la

información que la aplicación es capaz de proveer (consultas sobre el registro, sistema, discos, ficheros e impresoras).

Como vemos, hay consultas que retornan más de una línea por cada equipo, hay diferente número de columnas utilizadas para cada línea ... etc.:

Computer Name	Date Time	Query	OuputVal_0	OuputVal_1	OuputVal_2	OuputVal_3	OuputVal_4	OuputVal_5	OuputVal_6	OuputVal_7	OuputVal_8	OuputVal_9
computer2	14/05/2019 17:18:23	id_FileInf_excel2016.exe	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists
computer3	14/05/2019 17:18:49	id_SysInfo	ERROR: Access is denied. (Exception from HRESULT: 0x80070005)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
computer3	14/05/2019 17:18:49	id_DiskInfo	0	No Info	C:	FALSE	No Info	Local Fixed Disk	C:	3	NTFS	1,01604E+11
computer3	14/05/2019 17:18:49	id_FileInf_winword2016.exe	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists
computer3	14/05/2019 17:18:49	id_FileInf_excel2010.exe	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists
computer3	14/05/2019 17:18:49	id_FileInf_excel2016.exe	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists	Not exists

Ilustración 15. Fichero de output sin filtrar

Sin embargo, simplemente filtrando el command ID que deseamos, obtenemos una vista perfectamente estructurada del fichero para los datos que deseamos visualizar. En este ejemplo se muestran los datos relativos a la parte de la consulta “Discos”:

Computer Name	Date Time	Query	OuputVal_0	OuputVal_1	OuputVal_2	OuputVal_3	OuputVal_4	OuputVal_5	OuputVal_6	OuputVal_7	OuputVal_8	OuputVal_9
computer3	14/05/2019 17:18:49	id_DiskInfo	ERROR: The RPC server is unavailable. (Exception from HRESULT: 0x80070005)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
computer4	14/05/2019 17:20:01	id_DiskInfo	0	No Info	C:	FALSE	No Info	Local Fixed Disk	C:	3	NTFS	1,01604E+11
computer4	14/05/2019 17:20:01	id_DiskInfo	0	No Info	D:	FALSE	No Info	Local Fixed Disk	D:	3	NTFS	3,21886E+11
computer4	14/05/2019 17:20:01	id_DiskInfo	No Info	No Info	E:	No Info	No Info	CD-ROM Disc	E:	5	No Info	No Info
computer5	14/05/2019 17:20:04	id_DiskInfo	ERROR: Access is denied. (Exception from HRESULT: 0x80070005)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
computer6	14/05/2019 17:20:22	id_DiskInfo	ERROR: Access is denied. (Exception from HRESULT: 0x80070005)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
computer7	14/05/2019 17:20:38	id_DiskInfo	ERROR: Access is denied. (Exception from HRESULT: 0x80070005)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
computer8	14/05/2019 17:20:55	id_DiskInfo	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING
computer9	14/05/2019 17:21:00	id_DiskInfo	ERROR: Access is denied. (Exception from HRESULT: 0x80070005)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
computer10	14/05/2019 17:21:22	id_DiskInfo	0	No Info	C:	FALSE	No Info	Disco fijo local	C:	3	NTFS	72244391936
computer10	14/05/2019 17:21:22	id_DiskInfo	0	No Info	D:	FALSE	No Info	Disco fijo local	D:	3	NTFS	3,18065E+11
computer11	14/05/2019 17:21:37	id_DiskInfo	0	No Info	C:	FALSE	No Info	Disco fijo local	C:	3	NTFS	79266639872
computer11	14/05/2019 17:21:37	id_DiskInfo	0	No Info	D:	FALSE	No Info	Disco fijo local	D:	3	NTFS	1,44114E+11
computer11	14/05/2019 17:21:37	id_DiskInfo	No Info	No Info	E:	No Info	No Info	Disco CD-ROM	E:	5	No Info	No Info

Ilustración 16. Fichero de output filtrado

5. Conclusiones

En este capítulo revisaremos desde un punto de vista crítico el trabajo que hemos realizado, considerando también posibles líneas futuras en las que complementar el trabajo realizado hasta ahora.

5.1 Autoevaluación del trabajo realizado y resultados

Desde un punto de vista crítico, considero que en el análisis y planteamiento inicial se subestimó la complejidad en algunos puntos del desarrollo de la herramienta. Ha sido necesario invertir gran cantidad de tiempo en depurar el código para obtener una solución estable y libre de errores.

Durante el desarrollo del proyecto se ha cumplido con los hitos marcados en el calendario. Las dificultades técnicas encontradas y su resolución han llevado a anticipar algunas tareas para obtener la solución integral. Sin embargo, este adelanto en las tareas técnicas ha afectado al tiempo disponible relativo a la elaboración de las memorias y entregables del proyecto, pudiendo afectar quizá a su calidad final.

Por otro lado, se ha puesto de manifiesto la relevancia que tiene hacer un correcto planteamiento y diseño técnico antes de iniciar la programación. A pesar de haber encontrado muchas dificultades, el tener un diseño bien estructurado nos ha permitido aislar los problemas y resolverlos más fácilmente. El diseño obtenido, además, nos ha permitido incluir nuevas funcionalidades de forma sencilla que no estaban previstas inicialmente.

A nivel general se puede concluir que los objetivos planteados inicialmente se han cumplido y que incluso se han aportado algunas funcionalidades adicionales que no estaban previstas al inicio, como son la compatibilidad con sistemas de 64 bits y funciones como la recolección de información sobre impresoras.

Además, se ha podido elaborar el proyecto sin soportar ningún coste, dado que la licencia de Visual Studio es gratuita para estudiantes, y no ha sido necesaria la compra de ningún otro material.

En la actualidad la herramienta se ha puesto a disposición de profesionales de IT para su evaluación y el resultado está siendo muy positivo.

5.2 Líneas de trabajo futuras

Como ya hemos mencionado, el producto obtenido cumple y mejora sustancialmente los objetivos previstos inicialmente. Además, la arquitectura que se ha diseñado permite incluir nuevas funcionalidades., lo cual nos deja la puerta abierta a implementar nuevos módulos que podamos considerar de utilidad.

A nivel técnico y funcional se hacen las siguientes propuestas de mejora y ampliación para siguientes versiones:

- Copiado de ficheros, que nos permita copiar un determinado fichero desde el PC del administrador a un directorio concreto sobre los PCs definidos en el directorio hosts.
- Ejecución de scripts remotos, esto, unido al punto anterior, nos permitiría copiar y ejecutar scripts en equipos remotos, por ejemplo para la aplicación de soluciones conocidas a incidencias.
- Recopilación de ficheros, que nos permita copiar a un directorio del PC del administrador o varios ficheros de PCs remotos. Esto sería muy útil para recoger logs y analizarlos de forma centralizada, y también podría combinarse con las 2 funciones mencionadas anteriormente.

Además de la parte técnica, una segunda fase del proyecto podría enfocarse en la promoción, difusión y venta de la aplicación a Pymes y pequeñas empresas, haciendo un análisis de mercado, vías posibles de acceso a los clientes así como profundizar en los modelos de licenciamiento de software y contratos de soporte que podrían ofrecerse a los compradores.

6. Glosario

A continuación se detallan algunos de los términos más relevantes en esta memoria, junto a la definición y sentido con el que se ha utilizado

- **Comando:** datos introducidos por el en la aplicación, mediante un fichero, que representan la información que se desea obtener de los hosts de la red.
- **Host:** Cualquier equipo en la red al que la aplicación se conecta para extraer información de él.
- **Módulo principal:** en la aplicación desarrollada, este módulo aporta la interfaz gráfica de usuario y rige toda la lógica de la aplicación, haciendo de intermediario entre los diferentes módulos adicionales de funciones específicas o auxiliares.
- **Módulos específicos** (o adicionales): son los módulos orientados a funciones específicas, como la obtención de información remota (registro, sistema, impresoras, discos, ficheros), o funciones auxiliares como el procesamiento de instrucciones o comunicación con los hosts remotos.
- **Output:** fichero sobre el que se recoge el resultado de los comandos ejecutados sobre los host. Es estructurado y fácilmente manipulable mediante filtros.
- **Registro de Windows:** Base de datos.
- **Standalone:** Software que puede ser ejecutado sin realizar instalaciones previas y que además no depende de un sistema centralizado o de servidor con el que comunicarse.
- **WMI:** Windows Management Instrumentation. Modelo de información común para compartir información de la administración de sistemas en una red empresarial.
- **.NET Framework:** Plataforma de Microsoft instalable en entornos Windows que provee de funcionalidades para programación de aplicaciones en estos entornos.

7. Bibliografía

Este trabajo se ha desarrollado con el apoyo de información disponible en la red.

Cuotas de mercado por sistema operativo (abril 2019)

- [1] <https://www.datanyze.com/market-share/operating-systems/windows-server-market-share>
- [2] https://w3techs.com/technologies/overview/operating_system/all
- [3] <https://www.statista.com/statistics/915085/global-server-share-by-os/>
- [4] <http://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide>
- [5] https://www.w3schools.com/browsers/browsers_os.asp

Foros y webs de programación (marzo-mayo 2019)

- [6] <https://www.codeproject.com>
- [7] <https://stackoverflow.com>
- [8] <https://social.msdn.microsoft.com>

IDEs de programación (marzo-mayo 2019)

- [9] Eclipse: <https://www.eclipse.org/downloads/>
- [10] Mono: <https://www.monodevelop.com/download/>
- [11] Visual Studio <https://visualstudio.microsoft.com/es/downloads/>

Documentación de Microsoft (marzo-mayo 2019)

- [12] Registro remoto de Windows, <https://helpdeskgeek.com/networking/how-to-connect-to-a-remote-registry-windows-7-10/>
- [12.1] Registro Key properties <https://docs.microsoft.com/es-es/dotnet/api/microsoft.win32.registrykey?view=netframework-4.7.2>
- [13] Win32 namespace, <https://docs.microsoft.com/es-es/dotnet/api/microsoft.win32?view=netframework-4.7.2>
- [14] WMI, <https://docs.microsoft.com/en-us/windows/desktop/wmisdk/operating-system-availability-of-wmi-components>
- [15] WMI namespace, <https://docs.microsoft.com/en-us/windows/desktop/wmisdk/using-wmi>
- [16] WMI connection, <https://docs.microsoft.com/en-us/windows/desktop/wmisdk/troubleshooting-a-remote-wmi-connection>
- [17] WMI Troubleshooting, <https://docs.microsoft.com/en-us/windows/desktop/wmisdk/troubleshooting-a-remote-wmi-connection>

.NET Framework (marzo-mayo 2019)

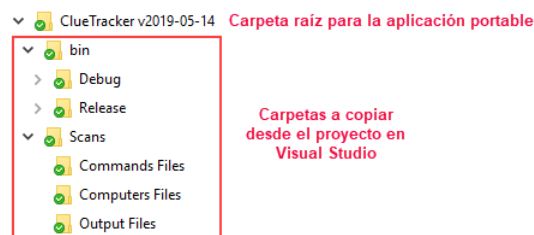
- [18] Enlace de descarga <https://dotnet.microsoft.com/download/dotnet-framework/net472>

- [19] Módulo Sistema: Sistema operativo, <https://docs.microsoft.com/en-us/windows/desktop/cimwin32prov/win32-operatingsystem>
- [20] Módulo Sistema: ComputerSystem, <https://docs.microsoft.com/en-us/windows/desktop/cimwin32prov/win32-computersystem>
- [21] Módulo discos: Win32_LogicalDisk, <https://docs.microsoft.com/en-us/windows/desktop/cimwin32prov/win32-logicaldisk>
- [22] Módulo Impresoras, <https://docs.microsoft.com/en-us/windows/desktop/cimwin32prov/win32-printer>
- [23] Módulo Ficheros, <https://docs.microsoft.com/es-es/dotnet/api/system.io.fileinfo?view=netframework-4.7.2>
- [24] Opciones de compilación en Visual Studio, <https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/compiler-options/platform-compiler-option>
- [25] Registro de 64 bits, <https://docs.microsoft.com/en-us/dotnet/api/microsoft.win32.registryview?view=netframework-4.8#Microsoft.Win32.RegistryView.Registry64>
- [26] Registro remoto, <https://docs.microsoft.com/en-us/dotnet/api/microsoft.win32.registrykey.openremotebasekey?view=netframework-4.8>

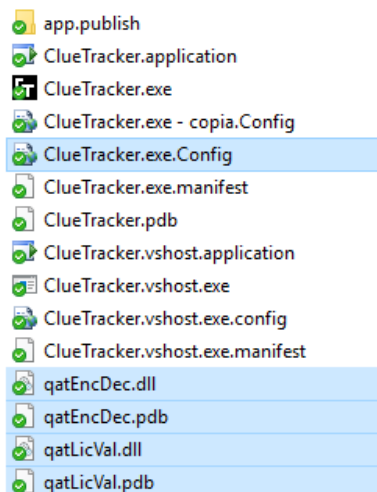
8. Anexos

8.1 Anexo 1: Preparación de paquete ejecutable

1. Rehacer el proyecto desde Visual Studio > Project > Build, seleccionando “Release”
2. Crear la carpeta raíz de la aplicación, en blanco, con el nombre “ClueTracker vYYYY-MM-DD”
3. A esta carpeta, copiar los siguientes directorios “bin” y “Scans” y todo su contenido desde la carpeta del proyecto de Visual Studio a la carpeta raíz creada en el paso anterior.



4. Asegurar que el directorio “\Release” contiene los ficheros señalados en la imagen:

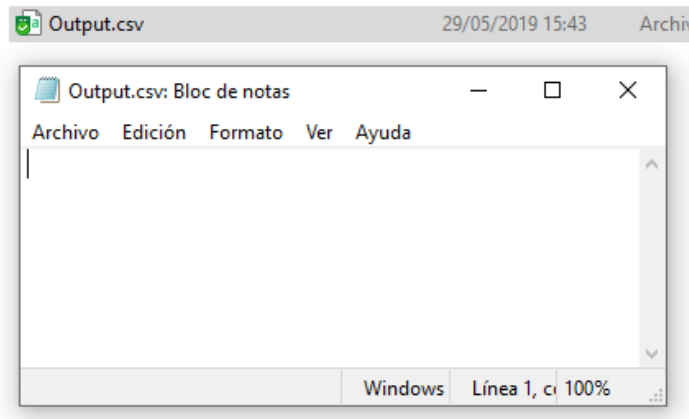


5. Asegurar que el fichero “ClueTracker.exe.Config” tiene los paths inicializados en blanco. Si no, restaurar el fichero a partir de “ClueTracker.exe – Copia.Config” que tiene la inicialización correcta

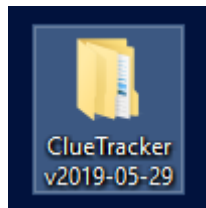
```
ClueTracker.exe.Config: Bloc de notas
Archivo Edición Formato Ver Ayuda
<?xml version="1.0"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
  </startup>
  <appSettings>
    <add key="AppVersion" value="ClueTracker v2019-05-14" />
    <add key="txtAppFolder" value="" />
    <add key="QuickScansMainFolder" value="Scans" />
    <add key="QuickScansCommandsFileDefaultName" value="" />
    <add key="QuickScansComputerSubFolder" value="Computers Files" />
    <add key="QuickScansOutputFilesSubFolder" value="Output Files" />
    <add key="QuickScansOutputFileDefaultName" value="Output.csv" />
    <add key="SavedScansMainFolder" value="Saved Scans" />
    <add key="QuickScansCommandsFilesSubFolder" value="Commands Files" />
    <add key="txtCommandsFile" value="" />
    <add key="rbCompsSingleName" value="True" />
    <add key="txtCompSingleName" value="" />
    <add key="rbCompsFromFile" value="False" />
    <add key="txtComputersInputFile" value="" />
    <add key="txtOutputFile" value="" />
    <add key="rbOutAuto" value="True" />
    <add key="rbOutColumns" value="False" />
    <add key="rbOutLines" value="False" />
  </appSettings>
</configuration>
```

** Si se ha restaurado el fichero, asegurar que el campo AppVersion coincide con la versión del fichero original a la versión que estamos paquetizando.

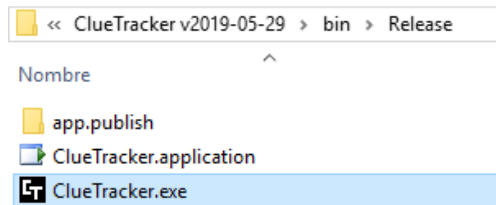
6. Asegurar que el directorio “**Output**” contiene sólo el fichero “Output.csv” en blanco:



7. La carpeta ya está lista para ser distribuida y pueda ser ejecutada en cualquier PC con Windows y .NET Framework 4.5 o superior.



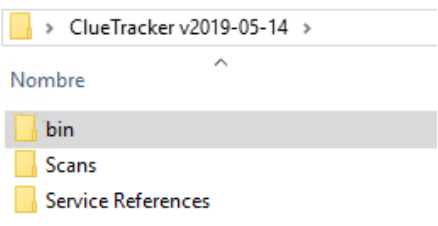
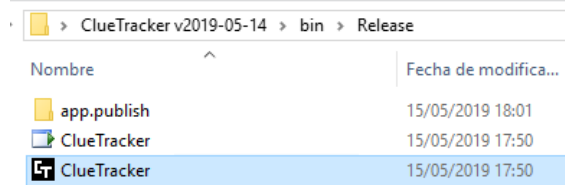
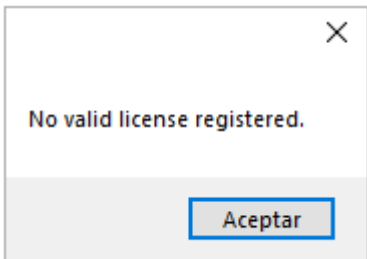
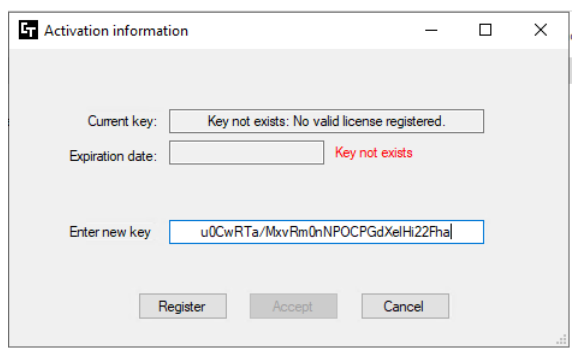
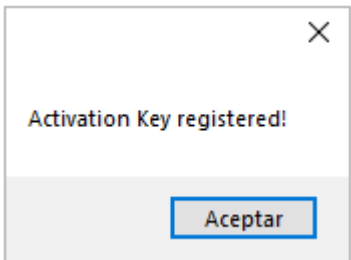
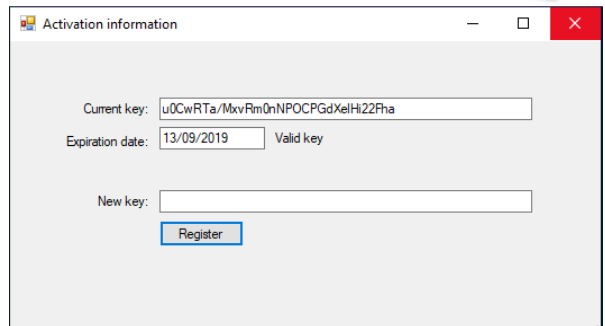
8. Ejecutar el fichero “\ClueTracker v2019-05-29\bin\release\ClueTracker.exe”



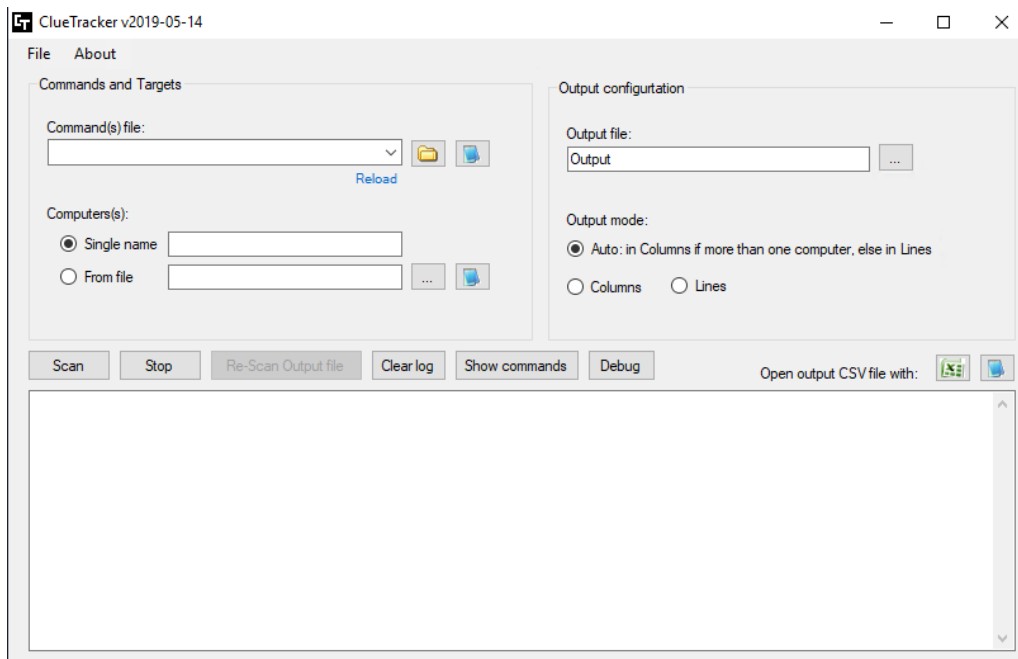
** Ver manual de usuario en la siguiente sección de este documento (Anexo 2)

8.2 Anexo 2: User guide

Un-zip “ClueTracker vYYYY-MM-DD” folder to local directory into a PC where tool will be used

<p>1 . Go to “Release” directory</p> 	<p>2 . Run “Cluetracker.exe” with account with administrative rights over hosts you want to scan</p> 
<p>3 . In first execution the message below is prompted. Click “Accept”.</p> 	<p>4 . Enter activation key and click “Register”</p>  <p style="text-align: center;">Key: u0CwRTa/MxvRm0nNPOCPGdXelHi22Fha</p>
<p>5 . Accept licenes confirmation:</p> 	<p>6 . License expiration date will be displayed. Play “x” to continue:</p> 

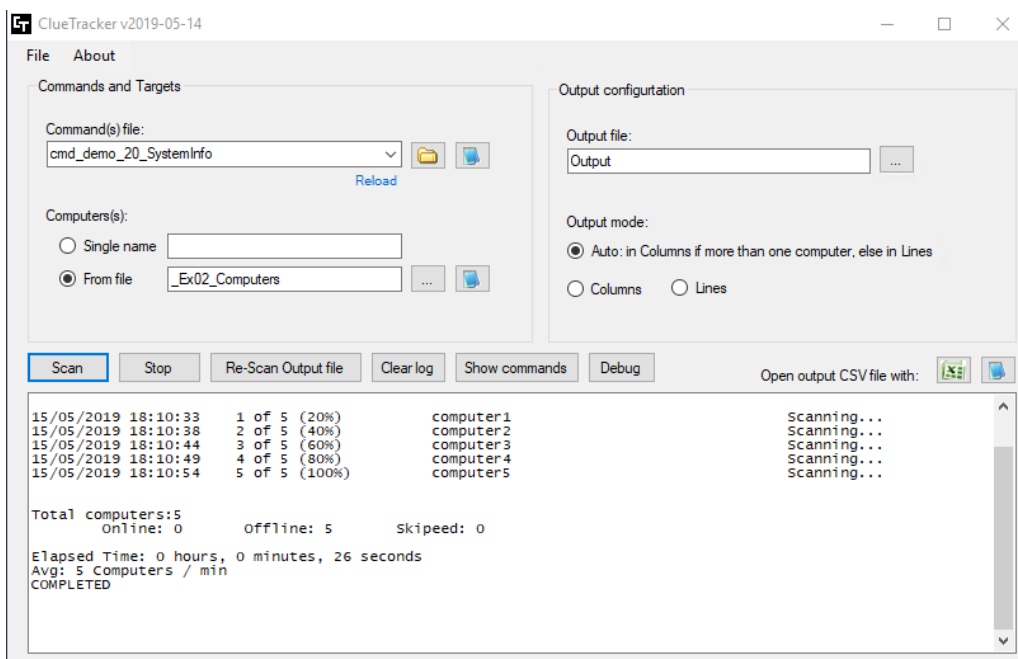
7. Main window is displayed:



8 . Enter parameters to run desired queries:

- **Comand(s) file:** Select command file or create new one.
- **Computer(s):** Enter the name of the host to scan or select file with hosts list.
- **Output file:** Select CSV file to use as output.

9 . Once parameters are initialized, press “Scan” botton to start the scan:



8.3 Anexo 3: Command syntax

A continuación se describe la sintaxis de los comandos disponibles en esta versión de la aplicación.

8.3.1 Command RegValRead

```
RegValRead p1:Registry_Root p2:Registry_Key_Path  
p3:Value_Name
```

Description

Search for registry value name (p3) in given registry hive (p1) and its relative path (p2)

Parameters

- *P1: Registry_Root*
Registry Hive where search for path and value. Long and short formats are allowed:

```
HKEY_LOCAL_MACHINE or HKLM  
HKEY_CLASSES_ROOT or HKCR  
HKEY_USERS or HKU)  
HKEY_CURRENT_USER or HKCU  
**Other possible hives has not been tested
```

Example: P1:HKLM

- *P2: Registry_Key_Path:* Relative path from give registry hive where search for value name.
Example: P2:SOFTWARE\JavaSoft\Java Runtime Environment
- *P3: Value_Name:* Registry value name in given hive (p1) and path (p2) to be readed
Example: P3: Java7FamilyVersion
- *P4: details_yes_no:* Set if value subkey details must be reported or not (1 yes, 2 no)
Example: P4: 1

Controlled errors

- Unreacheable machine
- Remote Registry disabled in local computer
- Remote Registry disabled in target computer
- No rights to read register
- Hive not exists
- Key not exists
- ValueName not exists

Example 1: Get Java version for each online computer. With no subkey details

id_Jversion RegValRead p1:HKLM p2:SOFTWARE\JavaSoft\Java Runtime Environment p3:Java7FamilyVersion p4:0

Computer Name	Date Time	InsOutLineID	Valid	id_Jversion_Output
computer1	22/05/2019 13:36	id_Jversion	Valid	Java7FamilyVersion (String)="1.7.0_045"
computer2	22/05/2019 13:36	id_Jversion	Valid	_NO PING
computer3	22/05/2019 13:36	id_Jversion	Valid	Java7FamilyVersion (String)="1.7.0_045"

Example 2: Get Java version for each online computer. With subkey details

id_Jversion RegValRead p1:HKLM p2:SOFTWARE\JavaSoft\Java Runtime Environment p3:Java7FamilyVersion p4:1

Computer Name	Date Time	InsOutLineID	Valid	id_Jversion_Output
computer1	29/04/2019 13:32	id_Jversion	Valid	Java7FamilyVersion (String) = "1.7.0_45" in subkey 'HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment'
computer2	29/04/2019 13:32	id_Jversion	Valid	_NO PING
computer3	29/04/2019 13:32	id_Jversion	Valid	Java7FamilyVersion (String) = "1.7.0_45" in subkey 'HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment'

8.3.2 Command RegValReadInSubKeys

RegValReadInSubKeys p1:nlevels, p2:Registry_Root p3:Registry_Key_Path p4:Value_Name p5:details

Description

Search for registry value name (p4), starting in given registry hive (p2) and registry path (p3). If value does not exist in this path, search will continue in next subkeys, with max subkey number determined by p1

Parameters

- *P1: nlevels:* Number of subkey levels to continue the search if value does not exist in first given level by p3
- *P2: Registry_Root* (see details in **RegValRead** command)
- *P3: Registry_Key_Path* (see details in **RegValRead** command)
- *P4: Value_Name* (see details in **RegValRead** command)
- *P5: details_yes_no:* (see details in **RegValRead** command)

Output

Execution	Output
Correct execution	
Unreachable machine	NO_PING
Remote Registry disabled in local computer	Normal Output (Remote Registry service is not needed in local computer)
Remote Registry disabled in target computer	Cannot open Registry Hive 'HKLM'. Check that Remote Registry service is running
No rights to read register	ERROR opening key: Requested registry access is not allowed.
No rights to update register	Not apply to this command
Registry hive not exists	Cannot open Registry Hive 'HKLM_ERR'. Check that Remote Registry service is running
Registry key not exists	Cannot open Registry Key 'HKLM\SOFTWARE\JavaSoft\Java Plug-in_ERR', Check that exists and Remote Registry service is enabled in this system and in target computer
ValueName not exists	Value Name 'UseJava2IExplorer_ERR' not found in subkey 'HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Plug-in' searching in 1 levels deep
Invalid type for update parameter	Not apply to this command

Example 1: Get if Java is enabled for browsers. With no subkey details

```
id_JavaForBrowser: RegValReadInSubKeys p1:1 p2:HKLM
p3:SOFTWARE\JavaSoft\Java Plug-in p4:UseJava2IExplorer
p5:0
```

Computer Name	Date Time	InsOutLineID	ValID	id_JavaForBrowser_Output
computer1	22/05/2019 13:42	id_JavaForIE	Valid	UseJava2IExplorer (DWord) = "1"
computer2	22/05/2019 13:42	id_JavaForIE	Valid	_NO PING
computer3	22/05/2019 13:42	id_JavaForIE	Valid	UseJava2IExplorer (DWord) = "1"

Example 2: Get if Java is enabled for browsers. With subkey details

```
id_JavaForBrowser: RegValReadInSubKeys p1:1 p2:HKLM
p3:SOFTWARE\JavaSoft\Java Plug-in p4:UseJava2IExplorer
p5:1
```

Computer Name	Date Time	InsOutLineID	ValID	id_JavaForBrowser_Output
computer1	22/05/2019 13:41	id_JavaForIE	Valid	UseJava2IExplorer (DWord) = "1" in subkey 'HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Plug-in\10.45.2'
computer2	22/05/2019 13:41	id_JavaForIE	Valid	_NO PING
computer3	22/05/2019 13:41	id_JavaForIE	Valid	UseJava2IExplorer (DWord) = "1" in subkey 'HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Plug-in\10.45.2'

**Note that value actually is located 1 level under given subkey. In this output you can see that this value location depends on Java version. Because registry structure is known for Java

settings, we have given 1 level up path to search, no matter the installed Java version and its key name. With details_yes_no parameter, we also get information about installed Java version

8.3.3 Command RegValUpdate

RegValUpdate p1:Registry_Root p2:Registry_Key_Path
p3:Registry_Value_Name p4:New Value p5:details

Description

Updates registry value name (p4) in registry path given by p1, p2 and p3, with value given by p5.

Parameters

- *P1: Registry_Root (see details in RegValRead command)*
- *P2: Registry_Key_Path (see details in RegValRead command)*
- *P3: Value_Name (see details in RegValRead command)*
- *P4: New_Value: Value to write in registry. Put attention to use valid type according with registry value type to update (DWORD... etc)*
- *P5: details_yes_no: (see details in RegValRead command)*

Output

Execution	Output
Correct execution	
Unreachable machine	NO_PING
Remote Registry disabled in local computer	Normal Output (Remote Registry service is not needed in local computer)
Remote Registry disabled in target computer	Cannot open Registry Hive 'HKLM'. Check that Remote Registry service is running
No rights to read register	ERROR opening key: Requested registry access is not allowed.
No rights to update register	ERROR opening key: Requested registry access is not allowed.
Registry hive not exists	Cannot open Registry Hive 'HKLM_ERR'. Check that Remote Registry service is running
Registry key not exists	Cannot open Registry Key 'HKLM\SOFTWARE\JavaSoft\Java Plug-in_ERR', Check that exists and Remote Registry service is enabled in this system and in target computer
ValueName not exists	Value Name 'UseJava2IExplorer_ERR' not found in subkey 'HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Plug-in'
Invalid type for update parameter	Trying to set value "abc" for UseJava2IExplorer, type: REG_DWORD:

	ERROR: AccessRegistry.RegValRead():The type of the value object did not match the specified RegistryValueKind or the object could not be properly converted.
--	--

Example 1: Enable Java for Browser, with no subkey details

```
id_JavaForBrowser: RegValUpdate p1:HKLM
p2:SOFTWARE\JavaSoft\Java Plug-in\10.45.2
p3:UseJava2IExplorer p4:0 p5:0
```

Computer Name	Date Time	InsOutLineID	Valid	id_JavaForBrowser_Output
computer1	22/05/2019 14:25	id_JavaForBrowser	Valid	UseJava2IExplorer (DWord) = "1" updated to "0"
computer2	22/05/2019 14:25	id_JavaForBrowser	Valid	_NO PING
computer3	22/05/2019 14:25	id_JavaForBrowser	Valid	Cannot open Registry Key 'HKLM\SOFTWARE\JavaSoft\Java Plug-in\10.45.2', Check that exists and Remote Registry service is enabled in this system and in target computer

**Note error for computer3. Its Java version is 10.36.6 so regkey \Java Plug-In\10.45.2 is not found in its registry. In these cases use command "RegValUpdateInSubKeys " explained in next section of this document.

8.3.4 Command RegValUpdateInSubKeys

```
RegValUpdateInSubKeys p1:nlevels p2:Registry_Root
p3:Registry_Key_Path p4:Registry_Value_Name p5:New Value
p6:details
```

Description

Same as RegValUpdate command, but allows to search the value name to be updated in different subkeys levels with deep search detailed in p1.

Parameters

- *P1: nlevels:* Number of subkey levels to continue the search if value does not exists in first given level by p3
- *P2: Registry_Root* (see details in **RegValRead** command)
- *P3: Registry_Key_Path* (see details in **RegValRead** command)
- *P4: Value_Name* (see details in **RegValRead** command)
- *P5: New Value:* Value to write in registry. Put attention to use valid type according with registry value type to update (DWORD... etc)
- *P6: details_yes_no:* (see details in **RegValRead** command)

Output

Execution	Output
Correct execution	
Unreachable machine	NO_PING
Remote Registry disabled in local computer	Normal Output (Remote Registry service is not needed in local computer)
Remote Registry disabled in target computer	Cannot open Registry Hive 'HKLM'. Check that Remote Registry service is running
No rights to read register	ERROR opening key: Requested registry access is not allowed.
No rights to update register	ERROR opening key: Requested registry access is not allowed.
Registry hive not exists	Cannot open Registry Hive 'HKLM_ERR'. Check that Remote Registry service is running
Registry key not exists	Cannot open Registry Key 'HKLM\SOFTWARE\JavaSoft\Java Plug-in_ERR', Check that exists and Remote Registry service is enabled in this system and in target computer
ValueName not exists	Value Name 'UseJava2IExplorer_ERR' not found in subkey 'HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Plug-in' searching in 1 levels deep
Invalid type for update parameter	Trying to set value "abc" for UseJava2IExplorer, type: REG_DWORD: ERROR: AccessRegistry.RegValRead():The type of the value object did not match the specified RegistryValueKind or the object could not be properly converted.

Example 1: Enable Java for Browser, with subkey details

```
id_JavaForBrowser: RegValReadInSubKeys p1:1 p2:HKLM
p3:SOFTWARE\JavaSoft\Java Plug-in p4:UseJava2IExplorer
p5:0
```

Computer Name	Date Time	InsOutLineID	Valid	id_JavaForBrowser_Output
computer_01	01/01/2019 14:25	id_JavaForBrowser	Valid	UseJava2IExplorer (DWord) = "0" updated to "1" in subkey 'HKEY_LOCAL
computer_02	01/01/2019 14:25	id_JavaForBrowser	Valid	_NO PING
computer_03	01/01/2019 14:25	id_JavaForBrowser	Valid	UseJava2IExplorer (DWord) = "1" updated to "1" in subkey 'HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Plug-in\10.36.6'

**Note that in this case computer_03 have different Java version that computer_01. Input command is set to start search from parent key path ...Java Plug-In\ so no matter next subkey name, value name will be found and updated. This solves problem of Example 1 in RegValUpdate command section

8.3.5 Command FileGetInfo

FileGetInfo p1:Folder_Path\File_Name
 p2: |Name|fileVersion|fileProductVersion|Length|CreationTime|IsReadOnly|LastAccessTime|LastWriteTime

Description

Get available file information for given full file path (p1). Requested information is detailed in (p2)

Parameters

- *P1: File full path and name:* Full path for despaired file in target computer
- *P2: Requested attributes:* Enter desired field names separated by "|". See full fields list in command syntax

Output

Execution	Output
Correct execution	Normal ouput
Unreachable machine	NO_PING
No permissions to read file or access through network	"Not exists" for each of the required fields
File path not exists	"Not exists" for each of the required fields
File name not exists	"Not exists" for each of the required fields
Required field is empty or not available	"not exists" for each of required fields that not exists

Example 1: Get Notepad.exe file information. Output in columns

id_Notepad.exe: FileGetInfo p1:C:\windows\notepad.exe
 p2: |Name|fileVersion|fileProductVersion|Length|CreationTime|IsReadOnly|LastAccessTime|LastWriteTime

Computer	Date Time	InsOutLineID	ValID	id_Notepad	fileVersion	ProductVersi	Length	CreationTime	IsReadOn	LastAccessTim	LastWriteTime
computer1	22/05/2019 14:25:26	id_Notepad.exe	Valid	notepad.exe	6.1.7600.16385 (win7_rtm.090713-1255)	6.1.7600.16385	179712	14/07/2018 1:41	FALSO	14/05/2019 9:41	14/07/2018 3:14
computer2	22/05/2019 14:25:26	id_Notepad.exe	Valid	_NO_PING	_NO_PING	_NO_PING	_NO_PING	_NO_PING	_NO_PING	_NO_PING	_NO_PING
computer3	22/05/2019 14:25:26	id_Notepad.exe	Valid	notepad.exe	6.1.7600.16385 (win7_rtm.090713-1255)	6.1.7600.16385	179712	14/07/2018 1:41	FALSO	16/05/2019 2:21:34	14/07/2018 3:14

Example 1: Get winword.exe file information. Valid to know exact version and patches level

id_winword.exe: FileGetInfo p1:C:\Program Files\Microsoft Office\Office14\winword.exe
 p2: |Name|fileVersion|fileProductVersion|Length|CreationTime|IsReadOnly|LastAccessTime|LastWriteTime

Computer Name	Date Time	InsOutLineID	Valid	id_winword.exe	fileVersion	fileProductVersion	Length	CreationTime	IsReadOnly	LastAccessTime	LastWriteTime
computer1	22/05/2019 14:25:26	id_winword.exe	Valid	winword.exe	14.0.7121.5004	14.0.7121.5004	1423008	19/03/2018 15:10	FALSE	04/05/2019 0:59	19/03/2019 15:10
computer2	22/05/2019 14:25:26	id_winword.exe	Valid	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING	_NO PING
computer3	22/05/2019 14:25:26	id_winword.exe	Valid	winword.exe	14.0.7121.5004	14.0.7121.5004	1423008	20/05/2018 9:12	FALSE	09/12/2018 10:25:02	09/12/2019 10:25:02

8.3.6 Command SysInfoGet

SysInfoGet

p1: |SerialNumber|BuildNumber|BIOSCaption|CountryCode|CSName|LastBootUpTime|OSLanguage|ServicePackMajorVersion|ServicePackMinorVersion|CompSystemCaption|DNSHostName|Domain|Manufacturer|Model|PrimaryOwnerName|SystemType|TotalPhysicalMemory|CompSystemUserName

Description

Get detailed system information

Parameters

- *P1: Field list to retrieve. Take all or only once needed separated by “|”*

CompSystemUserName: Retrives current logged on user (local or domain user). Terminal sessions are not reported and appears as “No Session”

Outputs

Execution	Output
Correct execution	Normal output
Unreachable machine	NO_PING
No permissions read information	ERROR: Acceso denegado. (Exception from HRESULT: 0x80070005 (E_ACCESSDENIED))
Required field is empty or not available	Not exists

Example 1: Get the following system information: Build Number, BIOS Caption, Country Code, CSName, Last BootUp Time, OS Language, Service Pack Major Version

Computer Name	Date Time	InsOutLineID	Valid	_SerialNumber	_BuildNumber	_BIOSCaption	_CountryCode	_CSName	_LastBootUpTime	_OSLanguage	_ServicePackMajorVersion
computer1	23/05/2019 14:04	1	Valid	CNU9278LD9	7601	Microsoft Windows 7 Professional	34	computer1	20/05/2019 8:40	3082	1
computer2	05/02/2014 14:04	1	Valid	_NO PING	_NO PING	_NO PING	_NO PING	_NO	_NO PING	_NO PING	_NO PING
computer3	23/05/2019 14:04	1	Valid	CNU9278LD9	7601	Microsoft Windows 7 Professional	34	computer3	20/05/2019 10:40	3082	1

Command PrintersGetInfo

PrintersGetList

p1: |Caption|Default|DeviceID|Name|Local|DriverName|PortName|Published|Shared|ShareName|InstallDate|SpoolEnabled|HorizontalResolution|VerticalResolution|PrintQuality|Duplex|Color

Description

Get printers information

Example

Computer Name	Date Time	Query	Caption	Default	DeviceID	Name	Local	DriverName
computer1	24/05/2019 18:11:42	1	Snagit 13	FALSO	Snagit 13	Snagit 13	VERDADERO	Snagit 13 Printer
computer1	24/05/2019 18:11:42	1	Send To OneNote 2016	FALSO	Send To OneNote 2016	Send To OneNote 2016	VERDADERO	Send to Microsoft OneNote 16 Driver
computer2	24/05/2019 18:11:42	1	Microsoft XPS Document Writer	FALSO	Microsoft XPS Document Writer	Microsoft XPS Document Writer	VERDADERO	Microsoft XPS Document Writer v4
computer3	24/05/2019 18:11:42	1	Microsoft Print to PDF	FALSO	Microsoft Print to PDF	Microsoft Print to PDF	VERDADERO	Microsoft Print To PDF
computer4	24/05/2019 18:11:42	1	HP Deskjet F2400 series	FALSO	HP Deskjet F2400 series	HP Deskjet F2400 series	VERDADERO	HP Deskjet F2400 series
computer5	24/05/2019 18:11:42	1	Fax	FALSO	Fax	Fax	VERDADERO	Microsoft Shared Fax Driver
computer6	24/05/2019 18:11:42	1	\\Print-vitoria.domain.com\Safe	VERDADERO	\\Print-vitoria.domain.com\Safe	\\Print-vitoria.domain.com\Safe	FALSO	RICOH PCL6 UniversalDriver V4.6
computer2	24/05/2019 18:11:42	1	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING
computer3	24/05/2019 18:11:42	1	Snagit 13	FALSO	Snagit 13	Snagit 13	VERDADERO	Snagit 13 Printer
computer3	24/05/2019 18:11:42	1	Send To OneNote 2016	FALSO	Send To OneNote 2016	Send To OneNote 2016	VERDADERO	Send to Microsoft OneNote 16 Driver
computer3	24/05/2019 18:11:42	1	Microsoft XPS Document Writer	FALSO	Microsoft XPS Document Writer	Microsoft XPS Document Writer	VERDADERO	Microsoft XPS Document Writer v4
computer3	24/05/2019 18:11:42	1	Microsoft Print to PDF	FALSO	Microsoft Print to PDF	Microsoft Print to PDF	VERDADERO	Microsoft Print To PDF
computer3	24/05/2019 18:11:42	1	HP Laserjet 3015l	FALSO	Fax	Fax	VERDADERO	HP PCL6
computer3	24/05/2019 18:11:42	1	\\Print-barcelona.domain.com\Safe	VERDADERO	\\Print-vitoria.domain.com\Safe	\\Print-vitoria.domain.com\Safe	FALSO	RICOH PCL6 UniversalDriver V4.6

Command LogicalDiskInfo

LogicalDiskInfo

p1: |Access|Availability|Caption|Compressed|ConfigManagerErrorCode|Description|DeviceID|DriveType|FileSystem|FreeSpace|InstallDate|MediaType|Name|PNPDeviceID|ProviderName|Purpose|Size|Status|StatusInfo|SystemName|VolumeDirty|VolumeName|VolumeSerialNumber

Description

Get logical disk information

Example

Computer Name	Date Time	Query	Access	Availability	Caption	Compressed	ConfigManagerErrorCode	Description	DeviceID	DriveType	FileSystem	FreeSpace	InstallDate
computer1	24/05/2019 18:19:58	1	0	No Info	C:	FALSO	No Info	Disco fijo local	C:	3	NTFS	95459188736	No Info
computer1	24/05/2019 18:19:58	1	0	No Info	D:	FALSO	No Info	Disco fijo local	D:	3	NTFS	2,76354E+11	No Info
computer2	24/05/2019 18:19:58	1	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING	NO_PING
computer3	24/05/2019 18:19:58	1	0	No Info	C:	FALSO	No Info	Disco fijo local	C:	3	NTFS	95459188736	No Info
computer3	24/05/2019 18:19:58	1	0	No Info	D:	FALSO	No Info	Disco fijo local	D:	3	NTFS	7,7635422	No Info
computer3	24/05/2019 18:19:58	1	0	No Info	E:	FALSO	No Info	Disco externo	D:	3	NTFS	1,47635E+12	No Info