ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"GENERACIÓN DE SCRIPTS CON MICROSOFT POWERSHELL PARA LA ADMINISTRACIÓN DE SERVIDOR MICROSOFT DE LA ACADEMIA MICROSOFT ESPOCH"

TESIS DE GRADO

Previa obtención del título de:

INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:

BYRON PAÚL HUERA PALTÁN

RIOBAMBA – ECUADOR

2014

AGRADECIMIENTO

Este trabajo agradezco primero a mi DIOS por darme la vida, a mis Padres por estar a mi lado apoyándome, a todos y cada uno de mis amigos que me motivaron y ayudaron a seguir adelante y luchar por alcanzar estos anhelados sueños de ser un profesional, al Ing. Washington Luna excelente profesor, tutor, amigo y al Ing Patricio Moreno por su calidad de profesor y ejemplo a seguir.

Byron Paúl Huera Paltán

DEDICATORIA

A mi madre que la tengo llena de vida y lucha siempre por sus hijos el mejor ejemplo de abnegación y entrega. A mi padre que me apoya con su ejemplo para ser una persona de bien responsable y trabajador.

Byron Paúl Huera Paltán

FIRMAS RESPONSABILIDAD Y NOTA

NOMBRE	FIRMA	FECHA
Ing. Iván Menes Camejo **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRONICA**	……..	……
Ing. Jorge Huilca **DIRECTOR DE LA ESCUELA DE INGENIERIA EN SISTEMAS**	……..	……
Ing. Washington Luna **DIRECTOR DE TESIS**	……..	……
Ing. Patricio Moreno **MIEMBRO DEL TRIBUNAL**	……..	……
Tlgo. Carlos Rodríguez **DIRECTOR DEL CENTRO DE DOCUMENTACIÓN**	……..	……

NOTA DE LA TESIS ……..

RESPONSABILIDAD DEL AUTOR

"Yo Byron Paúl Huera Paltán, soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis de Grado, y el patrimonio intelectual de la misma pertenece a la Escuela Superior Politécnica de Chimborazo."

_____

Byron Paúl Huera Paltán

INDICE DE ABREVIATURAS

A	
AD	Active Directory
ASLR	Address Space Load Randomization
B	
BD	BasedeDatos
D	
DHCP	Dynamic Host Configuration Protocol
DNS	Servicio de Nombres de Domio
G	
GUI	GraphicalUserInterface
H	
HDD	Hard Drive
HTTP	HypertextTransferProtocol–ProtocolodetransferenciadeHipertexto
HW	Hardware
I	
IIS	InternetInformationServer–ServidordeinformacióndeInternet
N	
NAP	Network Access Protection
NAT	NetworkAddressTranslation
NTFS	Sistema de Achivos de Microsft.
O	
OU	Unidad Organizativa
P	
PC	Personal Computer
PKI	Public Key
PS	PowerShell
R	
ROAD	Read Only Domain Controles

S

SSOO	Sistema Operativo
SMB	Server Message Block
T	
TI	Tecnologia de la Informacion
TS	Terminal Server
V	
VGA	Video Graphic Array
VPN	Virtual Private Network

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

La administración de TI es uno de los grandes problemas que enfrenta hoy en día un profesional informático, sin herramientas adecuadas podrían generar grandes problemas de gestión que llevarían incluso al fracaso y quiebra de las empresas e instituciones.

La administración comprende la creación de varios objetos como unidades organizativas, usuarios, grupos de dominio, impresoras, redes, procesos, servicios, respaldos de datos, entres otros.

El desarrollo de nuevos metodos de administración, conlleva cada día a optar técnicas diferentes, para evitar que los costos y tiempos sobrepasen lo planificado, por tal motivo, en este trabajo, se desarrolla un conjunto de herramientas, basada en PowerShell, para entregar a los profesionales que administran centro de computos herramientas adecuadas para una mejor gestión.

Este trabajo de investigación divide el documento en cinco capítulos:

- En el **capítulo I**, se presenta los antecedentes, justificación, objetivos a cumplir y la hipótesis de la investigación.
- En el **capítulo II**, se describe un marco teórico sobre Microsoft Windows Server 2008 y sus características
- En el **capítulo III, s**e analiza el marco teorico de Microsoft PowerShell, y la adaptación entre las metodologías de scripts y su respetivo lenguaje.
- En el **capítulo IV,** se enfoca al análisis y adaptación entre las metodologías de administracion gráfica y script.
- En el **capítulo V,** se escribe la elaboración de script para administrar un servidor.

Finalmente, se emite las conclusiones a las que se llegó luego de realizar la investigación, así como las recomendaciones para la utilización de la nueva metodología de administración.

CAPÍTULO I

1. MARCO REFERENCIAL.

En este capítulo se explica la problemática del trabajo que consta de antecedentes, justificación, objetivos e hipótesis.

1.1. Antecedentes.

La ESPOCH y la Facultad de Informática y Electrónica al haber obtenido varios reconocimientos por parte de Microsoft, necesitan continuar capacitando a sus estudiantes de una manera pertinente, por lo que sus laboratorios deben cumplir ciertos requisitos de administración, acorde a las necesidades que demandan los sistemas operativos modernos

Para poder dictar clases acorde a las exigencias de certificación Microsoft se debe implementar escenarios cliente-servidor reales, para cada curso, el cual implica realizar nueva tareas.

La administración de TI, trata de solucionar problemas que cada día generan los usuarios, atiende las demandas de servicios y recursos, lo cual conlleva tiempo dado a lo complicado de los procedimientos.

La Academia Microsoft consta de 20 computadoras y un servidor el cual esta administrado utilizando la interfaz grafica.

Para ofrecer servicio de administración adecuado y atender de manera rápida y oportuna a los estudiantes es conveniente contar con un método de administración eficente.

DELIMITACIÓN

La investigación se la aplicará en el laboratorio de la Academia Microsoft de la Escuela de Ingeniería en Sistemas de la ESPOCH cuyo problema es la administración que conlleva demasiados pasos para implantar escenarios diversos de acuerdo al tipo de curso, ya que normalmente se lo realiza con interfaz gráfica.

El tema propuesto contendrá el estudio, configuración e implementación de scripts para el control del servidor en el laboratorio de la Academia Microsoft de la ESPOCH utilizando la tecnología de scripts generados con PowerShell, con los cuales se tratara de dar la mejor solución en la administración.

1.2. Justificación del Proyecto de Tesis

1.2.1. Justificación Teórica

Las nuevas tecnologías con toda su sofisticación abren una brecha que facilita los procesos inherentes a los que, los sistemas operativos sean administrados de manera rápida y sencilla utilizando comandos generados en un scripts, con una alternativa potencial para el desarrollo y administración de TI.

El objetivo de la Escuela Superior Politécnica de Chimborazo es el de "Formar profesionales competitivos, emprendedores, conscientes de su identidad nacional, justicia social, democracia y preservación del ambiente sano, a través de la generación,

transmisión, adaptación y aplicación del conocimiento científico y tecnológico para contribuir al desarrollo sustentable de nuestro país", por lo que la Facultad de Informática y Electrónica se ve en la necesidad de implementar laboratorios que sirvan para el mejor proceso enseñanza aprendizaje de los estudiantes.

1.2.2. Justificación Práctica

En vista a las demandas de muchos administradores por lograr una optima y adecuada administración se debe buscar alternativas y estrategias que permitan ir mas allá de la administración tradicional haciendo uso de la ciencia y tecnología para obtener nuevos métodos y técnicas durante el proceso administrativo.

La ESPOCH y particularmente la Academina Microsoft, se beneficiara de los resultados, pudiendo así administrar su servidor y demás computadoras de mejor manera.

La creación de scripts mejora la administración repetitiva causada por el manejo de una interfaz gráfica.

Aportara con herramientas útiles de admininstración de Servidores Microsoft, que servirá con material didáctico con lo cual permitirá implantar escenarios de manera rápida para el dictado de cursos.

1.3. Objetivos

1.3.1. Objetivo General:

Generar Scripts Mediante el uso de Windows PowerShell para automatizar la administración de servidores en la Academia Microsoft ESPOCH

1.3.2. Objetivos Específicos.

- Definir los métodos de administración utilizados en los Sistemas Operativos Microsoft Server 2008, los subsistemas que lo componen, los elementos de su entorno y las relaciones que establece la misma.
- Estudiar el desarrollo de Scripts con la Herramienta Microsoft PowerShell.
- Analizar la administración del servidor Microsoft de la forma gráfica y mediante scripts.
- Utilizar los estándares de desarrollo de scripts mediante PowerShell para la administración del servidor Microsoft de la Academia Microsoft.
- Administrar un dominio en el servidor Microsoft en la Academia Microsoft.

1.4. Hipótesis.

La generación de Scripts mediante Microsoft PowerShell automatizará la administración de los servidores Microsoft en la academia Microsoft de la ESPOCH.

1.5. Métodos y Técnicas.

1.5.1. Métodos.

Los métodos utilizados en la presente investigación son:

Método Científico.

Bajo el cual estarán basadas todas las fases que han sido definidas anteriormente, debido a que mediante este método se constituye el marco general de referencia que guiará toda la investigación.

Método Deductivo

Se utilizará para la elaboración de la investigación, debido a que permite generar un criterio para el manejo de la investigación, en base a la recopilación, análisis y clasificación de toda la información general relacionada con las diferentes metodologías de desarrollo, métodos y herramientas utilizadas dentro del desarrollo de una aplicación de sistemas de administración de sistemas operativos.

1.5.2. Técnicas

Para la recopilación de la información necesaria que sustente este trabajo de investigación, se ha establecido como técnicas las siguientes:

- Investigación Bibliográfica Revisión de Documentos
- Observación
- Lluvia de ideas
- Técnica de Comprobación de ventajas y desventajas mediante cuadros comparativos

CAPÍTULO II

2. SERVIDORES MICROSOFT 2008

El presente capítulo menciona las definiciones conceptuales relacionadas con las metodologías de administración, permitiendo conocer las características, los subsistemas que contiene los elementos, así como también conceptos involucrados con la administración de TI.

2.1. Introducción al Sistemas Operativos Microsoft Windows Server 2008.

2.1.1. Introducción.

Microsoft Windows Server 2008 está implementado para brindar a las organizaciones un sistema que ofrece recursos para cargas de trabajo, creación de aplicaciones eficaces y protección en la red.

Ofrece una plataforma segura, de fácil gestión, para el alojamienro y desarrollo de aplicaciones y servicios en la web.

Windows Server 2008 incluye nuevas funciones de gran valor, eficacia y mejoras impactantes en los sistemas operativos base.

Existe cuatro ediciones de Windows Server 2008: Web, Estándar Edition, Enterprise y Datacenter.

A ellas se añade otra optimizada para plataformas Itanuim y tres más que corresponderían con las tres primeras pero sin la tecnología de virtualización o core.

2.1.2. Características.

Existe algunas diferencias con respecto a estructura y arquitectura de Windows Server 2008, que pueden modificar drásticamente la manera en el que se usa el Sistema Operativo. Estos cambios afectan el sistema operativo hasta el punto de que se puede llegar a controlar el hardware de forma más efectiva, se puede controlar y cambiar de forma radical la seguridad.

Entre las mejoras que se incluyen, esta:

- PowerShell: inclusión de una consola mejorada consola con soporte GUI para administración y creación de script.
- Cierre optimo de servicios en el sistema operativo.
- Creación de sesiones de usuario en paralelo:
- Sistema de Archivos SMB2: de 30 a 40 veces más rápido el acceso a los servidores web y multimedia.
- Windows Hardware Error Architecture (WHEA): protocolo mejorado y estandarizado de reporte de errores en Hardware.
- Address Space Load Randomization (ASLR): protection contra spyware y malware en carga de control de memoria.
- Virtualización de Windows Server: mejora en el rendimiento de virtualización con hipervie.
- Server Core: el núcleo del sistema operativo se ha renovado con muchas y nuevas caracteristicas.

- ❖ Reparación NTFS: proceso se realiza en segundo plano que repara archivos dañados del sistema, de hardware y de los usuarios.

2.1.3. Características y versiones de Windows Server 2008

Windows Server 2008 y sus versiones.

- ❖ Standar Edition.

  Ofrecer servicios y recursos a otros sistemas: servidor de archivos, DNS,DHCP, Web, impresión, controlador de dominio, Terminal Server, etc.

  Admite 4 CPU's y hasta 4 GB (32 GB en sistemas de 64 bits).

- ❖ Enterprise Edition.

  Admite 8 CPU's y hasta 32 GB en 32 bits (o 2 TB en 64 bits) Servicio de clúster, pudiendo gestionar hasta 8 nodos dentro de un mismo CPU.

- ❖ Datacenter Edition.

  Mínimo 8 CPU's (máximo 64 CPU's) y hasta 64 GB en 32 bits(o 2 TB en 64bits) Dispone de características de clúster mejoradas solo se lo puede adquirir con hardware icoporado.

- ❖ Web Edition.

  Creado para desarrollar, ejecutar y alojar un sitio Web. No soporta clúster, ni servicios de administración de redes como Active Directory, DNS o DHCP

  Pero sí puede actuar de servidor miembro de un dominio.

- ❖ Foundation edition.

  Con un límite máximo de 15 usuarios, y no se puede realizar virtualización, diseñado para una empresa que necesita un único servidor, 1 CPU 1 GB y hasta 8 GB.

2.1.4. Requisitos Hardware

Requisitos para la instalación de los sistemas operativos server 2008:

Tabla IIII.I. Requisitos del Sistema Operativo

	MINIMOS	RECOMENDADOS
Procesador	1 GHz (x86) o 1.4 GHz (x64)	2 GHz o superior
Memoria	512 MB DE RAM	2GB o mas -Máximo (sistemas operativos de 32 bits) 4GB en versión Standar y 64 GB en versiones Enterprise y Datacenter. -Máximo (sistemas operativos de 64 bits)32GB en versión Standar y 2TB GB en versiones Enterprise y Datacenter y para los sistemas basados en Itanium.
Disco Duro HDD	10 GB espacio libre	40 GB o mas Los equipos que disponga de más de 16GB de RAM requerirán más espacio libre de disco duro para el archivo de paginación y volcado de memoria.
Tarjeta Grafica	Súper VGA (800 X 600)	Súper VGA (800 X 600) o superior
Unidad Óptica	DVD – ROM	DVD ROM o BLURAY
Otros dispositivos	Monito VGA resolución de (800 x 600) , teclado, mouse	Monito VGA resolución de (1024 x 768) , teclado, mouse

Elaborado. Por el Autor

2.1.5. Beneficios.

Windows Server 2008 proporciona a los administradores de las TI, mayor control sobre sus servidores e infraestructura, las cueles les permite centrarse en las necesidades importantes de la empresa.

Caracteristicas mejoradas en secuencia de comandos con automatización de tareas, con Windows PowerShell que ayudan a los profesionales de la TI a automatizar tareas comunes de un administrador.

La administración e instalación basadas en funciones del Servidor facilita la tarea de administrar y proteger varias funciones de servidor en una empresa.

La nueva consola del Administrador, proporciona un único origen para administrar la configuración del servidor y la información del sistema. El profesional de la TI puede configurar solo las funciones y características que sean necesarias, ya que hay asistentes que automatizan muchas de las tareas de implementación que tardan más tiempo.

Se ha implementado herramientas mejoradas de administración del sistema operativo, tales como el monitor de rendimiento, por lo que ofrecen información sobre sistemas y alertas al profesional de TI sobre problemas potenciales ante de que estos ocurran.

2.1.6. Protección.

Dentro de Windows Server 2008 se proporciona una serie de tecnologías de seguridad nuevas y mejoradas, que aumentan la resguardo del sistema operativo, con una base sólida para dirigir y construir una empresa sólida. Incluye innovaciones de seguridad, como PatchGuard, que reduce la exposición a ataque en el núcleo del sistema operativo, lo que produce un servidor administrador sea más estable y seguro.

2.1.7. Flexibilidad

El sistema operativo Windows Server 2008 está diseñado para permitir que los administradores modifiquen su infraestructura para adaptarla a sus necesidades y a las necesidades de las empresa. Se mejora la flexibilidad en los trabajos móviles mediante

tecnologías que permiten al usuario ejecutar programas desde cualquier parte o ubicación remota, tales como Remoto App y Terminal Servicie, simplificando la administración en los servidores ubicado en las sucursales remotas. Windows Server 2008 acelera la implementación y el mantenimiento de los sistemas de TI con los Servicios de Implementación de Windows (WDS). Para empresas u organizaciones que necesitan controladores de dominio en varias sucursales, Windows Server 2008 ofrece una nueva opción de configuración llamado Controlador de Dominio de solo Lectura (RODC), que evita exponer las cuentas si el controlador de dominio estuviese bajo amenaza o peligro de agentes internos y externos.

2.1.8. Mejoras en la capacidad de administración del Active Directory.

En Windows Server 2008 se incluyeron mejoras en los Servicios de dominio del Active Directory que simplifica la administración de servicios de dominio y ofreciendo a los administradores un mayor grado de flexibilidad para abordar las necesidades de las sucursales. Algunas mejoras en la administración incluyen:

- Un asistente de instalación de dominio del Active Directory (AD DS).
- Cambios en Microsoft Console para administrar AD DS.
- Nuevas opciones de instalación para controladores de dominio.
- Opciones de administración e interfaz mejorada para AD DS.
- Herramientas mejoradas para encontrar controladores de dominio en la empresa.

Con el nuevo asistente de instalación, se agrupa ahora toda la funcionalidad relacionada, con el cual se simplifica el proceso y se ahorra tiempo durante la implantación.

La instalación desatendida de Windows Server 2008 no requiere una respuesta a ninguna solicitud de interface de usuario, lo que se simplifica aún más las instalaciones

remotas. Estas mejoras de interfaz de AD DS que ofrece en Windows Server 2008 reducirá el tiempo de administración de TI y reduce la administración en los sitios remotos.

2.1.9. Acceso centralizado de aplicaciones.

"Windows server 2008 ofrece mejoras e innovaciones a Servicios de Terminal Server que van más allá de una simple habilitación del acceso a aplicaciones, sino que mejora la experiencia de los usuarios al permitir que ejecuten en su propio escritorio aplicaciones remotas en paralelo con aplicaciones locales. También ofrece nuevas opciones para tener acceso de forma centralizada a las aplicaciones a través de Acceso web de servicios de Terminal Server.

Los nuevos componentes de servicios de Terminal Server incluyen una serie de herramientas capaces de optimizar el trabajo tales como:

- RemoteAPP de Servicios de Terminal Server: que permite a los usuarios ejecutar programas Windows de acceso remoto en paralelo con sus aplicaciones locales en el escritorio, mediante el nuevo cliente Conexión remoto.
- Terminal Service Gateway: extiende el alcance de Servicios de Terminal Server más allá del firewall corporativo, al proporcionar protección acceso a Terminal Server y escritorio compartidos sin necesidad de la infraestructura.
- Acceso web de servicios de Terminal Server (TS Web Access) ofrece una solución de aplicación remota que simplifique al administrador el proceso de publicación de aplicaciones remotas, a su vez que simplifica también para el usuario el proceso de búsqueda y ejecución." [1]

---

[1] Pulgar G. análisis comparativo de las tecnologías de virtualización ,2010

2.2. Introducción a la administración

La administración en Microsoft Windows server 2008 de base en dos tipos, mediante interfaz gráfica y la nueva metodología a base de scripts programados mediante la herramienta Microsoft PowerShell, para lo cual se detalla los elementos que posee para ser administrado.

2.2.1. Dominio

Un dominio o nombre de dominio es el nombre que identifica un sitio Web. Cada dominio tiene que ser único en Internet y ser capaz de identificar a una empresa o grupo de empresas. Por ejemplo "www.espoch.edu.ec".

2.2.2. Controlador de Dominio

En un servidor que se encarga de la gestión de un dominio, es decir, administra todo la información correspondiente a los usuarios, grupos y recursos de su dominio. Todo servidor en una empresa necesita al menos un controlador de dominio.

Un controlador de Dominio dentro de la red de una empresa, es primordia para almacenar datos del directorio activo y gestionar la comunicación entre los usuarios y dominios, incluyendo los procesos de inicio de sesión de usuarios, autentificación y búsqueda.

2.2.3. Directorio Activo.

Es un servicio de directorio de red, que identifica todos los recursos existentes en el servidor y los vuelve accesibles a todos los usuarios y a sus aplicaciones. El elemento principal de AD es el directorio, que almacena información del servidor. Los recursos almacenados en el directorio, tales como datos de usuarios, servidores, computadoras

impresoras, páginas web, base de datos, grupos, y políticas del sistema, se denominan objetos del sistema.

El Directorio Activo los organiza en orden jerárquicoa los dominios. Cada dominio tiene uno o más controladores de dominio. Cada vez que se realiza algún cambio en algún controlador de dominio, el resto de dominios se actualiza de forma automática.

Un objeto de sistema es un conjunto de atributos particulares, bajo un nombre especifico, que representa un recurso individual de la red. Los atributos se refieren a las características del objeto, así los atributos de una cuenta de usuario pueden ser el nombre, departamento y dirección de mail, o los de una impresora si es laser o es cartucho.

2.2.4. Unidades Organizacionales.

Las Unidades Organizacionales (UO) son contenedores que se usan para reunir objetos de un dominio, en grupos administrativos lógicos. Cada UO puede contener distintos objetos y cada dominio puede tener su propia lógica de agrupación en la UO.

La Unidad Central de la estructura lógica de AD es un dominio. Agrupando los objetos en uno o más dominios, es posible representar la propia organización de la empresa. Todos los objetos de la red existen en un dominio, pudiendo albergar hasta 10 millones de objetos de sistema operativo.

2.2.5. Sistema de Nombres de Dominio DNS.

El sistema de nombres de dominio es un sistema para asignar nombres a equipos y servicios de red que se organiza en una jerarquía de dominios. La asignación de nombres DNS se utiliza en las redes con protocolos TCP /IP, como internet o intranet,

para localizar equipos y servicios con nombres sencillos. Si el usuario escribe un nombre DNS en una aplicación, los servicios podrán traducir el nombre a otra información asociada con el mismo, como una dirección IP. Los equipos es comunican a través de una red mediante direcciones numéricas, es por eso que un nombre es más sencillo de aprender y recordar.

2.2.6. Grupos locales y globales de dominio.

Los grupos locales de dominio se utilizan para asignar permisos a los recursos del servidor. Un grupo local de dominio tiene la siguiente característica:

Pertenencia abierta: Se pueden añadir miembros desde cualquier dominio.

- Acceso a recursos en un dominio: Se puede utilizar un grupo local de dominio para asignar permisos para acceder solamente a los recursos que se ubican en el mismo dominio donde se ha creado el grupo local de dominio.

Los grupos globales frecuentemente son utilizados para organizar los usuarios que comparten requisitos de acceso similares a la red. Un grupo global tiene la siguiente característica:

Pertenencia limitada: Se pueden añadir miembros solamente desde el dominio en el cual se ha creado el grupo global.

- Acceso a recursos en cualquier dominio: Se puede utilizar un grupo global para asignar permisos para acceder a los recursos que están ubicados en cualquier dominio.

2.2.7. Usuarios locales y de dominio.

Los usuarios locales son cuentas las cuales pueden acceder al sistema de forma local ya que posean atributos y propiedades necesarias para administrar o utilizar determinada computadora, los recursos de ella y los accesos a la red.

Existe dos tipos de usuarios locales , los administradores y los usuarios limitados donde los administradores se encargan de proceso de administración, en tanto los usuarios limitados solo poseen acceso a los recursos y no a la administración.

CAPÍTULO III

3. POWERSHELL

El presente capítulo menciona las definiciones conceptuales relacionadas con Microsoft PowerShell, permitiendo conocer las características, los elementos, así como la sintaxis de programación, sus diferentes tipos de datos, atributos, variables y comandos. Se ilustra cada concepto con su respectivo ejemplo.

3.1. Introducción.

Se ha tomado como referencia la administración utilizando la interfaz PowerShell.

Esta herramienta es capaz de reducir el tiempo de administración de TI con Windows, evitando lentos y complicados pasos

3.2. Definición de PowerShell.

"Windows PowerShell es una interfaz de consola (CLI) con posibilidad de escritura y conjunción de comandos por medio de guiones (scripts en inglés). Es mucho más rica e interactiva que sus predecesores, desde DOS hasta Windows 7. Esta interfaz de consola está diseñada para su uso por parte de administradores de sistemas, con el propósito de automatizar tareas o realizarlas de forma más controlada. Originalmente denominada

como MONAD en 2003, su nombre oficial cambió al actual cuando fue lanzada al público el 25 de Abril del 2006.” [2]

Con más de 130 herramientas estándar de línea de comandos, este lenguaje de scripting centrado en la administración ayuda a conseguir control y productividad. Además de las herramientas estándar de la línea de comandos que se incluyen con Windows PowerShell, AppFabric proporciona otros comandos para administrar el entorno de caché distribuido.

3.3. Estándar de PowerShell

El estándar utilizado por PowerShell son los cmdlets de Microsoft Office 365 para Windows PowerShell para realizar diversas tareas administrativas desde la línea de comandos. Se enumeran y describen todos los cmdlets de Office 365 por tarea administrativa común, como el control de usuarios y la administración de dominios.

Para obtener información requerida, consulte el archivo de ayuda que incluye cada cmdlet, al que podrá acceder escribiendo lo siguiente en la línea de comandos: **get-help <cmdlet-name> -detailed**. La vista detallada del archivo de ayuda del cmdlet incluye una descripción de este cmdlet, la sintaxis de comando, descripciones de los parámetros y uno o varios ejemplos que ilustran el uso del cmdlet.

---

[2] http://es.wikipedia.org/wiki/Windows_PowerShell

3.4. Características de PowerShell.

La característica distintiva de PowerShell, es que es un intérprete de comandos orientado a objetos. La información de entrada y de salida en cada etapa del proceso (cmdlet, "común modulo") es un conjunto de instancias de objeto, a diferencia de lo que ocurre con los intérpretes de comandos tradicionales, que sólo devuelven y reciben texto.

En PowerShell se pueden crear alias al estilo de Unix, es decir, renombrar o nombrar de diferentes maneras a los comandos. Por ejemplo para mostrar directorios se usa dir, ls, gci. El programador puede apodar el comando como quiera. Esto facilita trabajar con el entorno de programación. Utilizando una única sentencia, mediante PowerShell se pueden realizar fácilmente las siguientes acciones:

- Monitorizar contadores.
- Apagar o detener servicios.
- Modificar registros del sistema.
- Crear usuarios locales.
- Administrar Directorio activo.
- Crear unidades organizativas.
- Crear y modificar usuarios del AD.
- Crear grupos de trabajo.
- Administrar impresoras.
- Dar mantenimiento al escritorio.
- Crear sistema de respaldos.
- Administrar servicios de Terminal Services.

- Configurar los servicios de red.
- Configurar DHCP
- Modificar servicio de DNS

3.5. Cmdlets entrada / salida y ejecución de scripts.

3.5.1. Cmdlets: entrada/salida básica.

PS2 tiene una especie de comandos llamados cmdlet que son objetos y hacen de todo. También se utiliza mucho la tubería o redirección "|" para pasar datos de un cmdlet a otro (como *nix). Por ejemplo, Se puede saber la fecha/hora actual, con: [3]

```
PS>get-date

Jueves, 07 de febrero de 2013 12:15:11
```

Para ver todas las propiedades en una lista, con fl (Format-List):

```
PS>get-date | fl
```

Si se quiere ver la salida en formato tabla, la salida a ft (Format-Table)

```
PS>get-date | ft
```

Para obtener funcionalidades de un cmdlet, se puede asignar su salida a una variable

```
PS>$fecha = get-date crea la variable $fecha que tiene

getdate.
```

3 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

De hecho, si se escribe ahora $fecha

```
PS>$fecha
```

Y luego se pulsa ENTER, también aparece la fecha.

Puede ver cualquier propiedad de $fecha indicándola después de un punto, por ejemplo, la propiedad año se ve así:

```
PS>$fecha.year          2013
```

Otra forma de ver todos los miembros, métodos y propiedades de $fecha redirigiendo su salida al cmdlet get-member con una tubería (|):

```
PS>$fecha | get-member
```

Que es lo mismo que hace el cmdlet

```
PS>get-date | get-member
```

Con el cmdlet Write-Host, se puede indicar por pantalla mensajes:

```
PS>write-host hola, ¿Cómo estas?          hola, ¿Cómo estas?

PS>write-host $fecha

Jueves, 07 de febrero de 2013 12:15:11
```

El cmdlet Read-Host pregunta por teclado e introduce la respuesta en una variable designada por el usuario:

```
PS>$nombre = read-host "Introduce tu nombre porfavor"

Introduce tu nombre: Byron

PS>$nombre

Byron
```

En una misma línea de código se ejecuta varias expresiones separadas por punto y coma, tales como:

```
PS>$edad = read-host "Tu edad" ; write-host $edad; "¡Qué Viejo"
```

Para ver una lista de todos los cmdlets disponibles de debe utilizar el comando de busqueda

```
PS>get-command
```

Para acceder a la ayuda predeterminada de PowerShell con el cmlet

```
PS>get-help
```

PowerShell incorpora muchos cmdlets por defecto, además también hay disponibles muchos más que se pueden instalar para realizar operaciones en otras plataformas como Directorio Activo, VMWare, SQL, etc.

3.5.2. Ejecución de scripts

PS2 ejecuta scripts con extensión .ps1, de forma que se puede escribir todo el código PS2 en un archivo de texto con la extensión .ps1.

Para editar scripts sólo se necesita un editor de texto o la herramienta PowerGUI para más funcionalidad como depuración y ver a tiempo real el valor de las variables.

Para ejecutar un script, se debe indicar su ubicación de forma absoluta. Existen dos formas de hacerlo:

Desde la línea de comandos convencional: [4]

```
Powershell .\espoch.ps1
```

En este caso, el archivo script espoch.ps1 debe estar ubicado en la carpeta actual. Se tiene que escribir toda la ruta si el script está en otra carpeta que no es la actual:

```
Powershell c:\scripts\powershell\pruebas\espoch.ps1
```

O desde la línea de comandos (cuando se está en la misma carpeta que el script.

```
PS>.\espoch.ps1
```

Indicando la ruta completa desde la línea de comandos de Powershell:

```
PS>c:\scripts\powershell\pruebas\espoch.ps1
```

4 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

3.6. Tipos de datos: variables, tablas, listas y hashes

Se puede definir y trabajar con los tipos de datos más utilizados, desde la variable común que representa un sólo dato u objeto, hasta las distintas colecciones de datos (tablas, listas y hash). [5]

3.6.1. Variables

Las variables comienzan con ($) y se puede colocar en ella casi cualquier cosa, por ejemplo:

```
PS>$fecha = get-date

PS>$numero = 5

PS>$numero_decimal = .36

PS>$letra = 'y '

PS>$cadena = 'Esto es una cadena'
```

También existe el tipo concreto de variable indicándola entre corchetes. Estos son algunos ejemplos posibles:

5 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

```
[int]$numero_32bit    =        999999999

[long]$numero_64bit =          1111111111111111

[string]$cadena       =        "cadena en unicode"

[char]$caracter       =        "a"

[byte]$numero_8bit    =        254

[bool]$verdad=         $true

[decimal]$numero_decimal =          78.45
```

Para mostrar el valor de una variable, se escribe y pulsa ENTER:

```
PS>numero

1

PS>$cadena

Es una cadena
```

En el caso de la suma dos o más números, se lo realiza de la siguiente manera:

```
PS>5+ 1.5

6.5
```

Sumar dos o más variables que contienen números determinados por el usuario:

```
PS>$numero + $numero_decimal

6.5
```

Incrementar directamente $numero en una unidad el cual fue determinado anteriormente:

```
PS>$numero++

PS>$numero

7
```

Concatenar dos o más cadenas se lo realiza de la siguiente manera:

```
PS>'y ' + ' es una cadena'

y  es una cadena
```

Se debe observar que los espacios en blanco se toman en cuenta.

Para unir dos o más variables que contienen cadenas predetermindas por el usuario :

```
PS>$letra + $cadena

y es una cadena
```

Dentro de Powershell se tiene un tipo de variable especial llamada $

3.6.2. Tablas.

Tabla es una variable que puede contener cualquier número de elementos de cualquier tipo dentro de la misma. El primer elemento de la tabla tiene la posición 0. Los elementos de la tabla se indican separados por comas: [6]

```
PS>$tabla = $fecha,1,'esto es una cadena',$numero_decimal,$letra

PS>$tabla

domingo, 30 de enero de 2013 23:48:28

1

esto es una cadena

1,5

y
```

Los elementos se enumeran desde cero, y se puede acceder a cualquier elemento indicando su posición entre corchetes [ ] como por ejemplo:

```
PS>$tabla[2]

esto es una cadena
```

6 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

También se puede ver un rango de elementos indicando la posición de inicio y de fin. Por ejemplo, para ver los elementos de $tabla desde la posición determinda.

```
PS>$tabla[2..4]

esto es una cadena 1,5 y
```

La posición del último elemento de una tabla es [-1]:

```
PS>$tabla[-1]

y
```

Se utiliza [-1] con este mismo propósito en cualquier otra variable que al igual que las tablas, tengan colecciones de elementos.

Para poder ver el número de elementos de una tabla se utiliza el método *.count*

```
PS>$tabla.count
```

Si desea modificar el valor de un elemento indicando su posición e igualando al nuevo valor

```
PS>$tabla[-1] = 'esto es otra cadena'

PS>$tabla[-1]

esto es otra cadena
```

3.6.3. Listas.

La lista es un array dinámico en el que se pueden añadir, modificar y eliminar elementos fácilmente. Así se declara la lista vacía $lista

```
PS>$lista = New-Object System.Collections.ArrayList
```

Como se determina, la lista es un objeto ArrayList de .NET. [7]

Se debe añadir elementos de cualquier tipo con el método .add, tales como:

```
PS>$lista.add("cadena de carcteres")
PS>$lista.add(50)
PS>$lista.add($fecha)
PS>$lista.add("otra cadena ingresada")
PS>$lista.add(830)
```

Para eliminar un elemento con .remove indicando su valor:

```
PS>$lista.remove (50)
```

Y ver el resultado... el 50 fue eliminado de la lista

```
PS>$lista
Cadena de caracteres
domingo, 30 de enero de 2013 23:48:28
otra cadena ingresada
830
```

7 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

Se quita un rango de elementos con .removerange, indicando la posición donde quiere empezar a quitar y el número de posiciones que quiere quitar en total (el primer elemento tiene la posición 0).

Por ejemplo, quitar desde la posición 2, dos elementos hacia adelante ("otra cadena ingresada " y 830).

```
PS>$lista.removerange(2,2)
```

Se verá el resultado en la lista de la siguiente manera

```
PS>$lista
Cadena de cacarteres

domingo, 30 de enero de 2013 23:48:28
```

También se puede eliminar un elemento indicando su posición con .removeat

```
PS>$lista.removeat(0)
```

Se ha quitado el primer elemento "cadena" en la posición 0, y lo comprueba:

```
PS>$lista
```

Para vaciar la lista se utiliza el método clear(): predefinido por el sistema.

```
PS>$lista.clear()
```

3.6.4. Hashes.

Hash es una lista donde cada elemento se compone de una clave única y su valor correspondiente (clave – valor). Para crear un hash vacío se lo realiza :[8]

```
PS>$hash = @{}
```

Para añadir clave - valor/es de cualquier tipo de dato :

```
PS>$hash["primero"]        =       1

PS>$hash["fecha"]   = $fecha

PS>$hash["tabla"]   =       1,"dos",3,4.5

PS>$hash[2]  =       "dos"
```

Para ver el contenido del hash en el monitor se teclea la variable $hash:

```
PS>$hash

Name            Value

fecha           30/01/2013 15:00:00

tabla           {1, dos, 3, 4,5}

primero         1

2               dos
```

8 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

Para acceder al valor de un elementode la hash (clave), indicar el nombre del hash seguido de un punto y el nombre de la clave de dato.

```
PS>$hash.primero   1

PS>$hash.fecha  Miercoles, 30 de enero de 2013 15:00:01

PS>$hash.tabla   1   dos   3   4,5

PS>$hash.2    error
```

La sentencia ($hash.2) produce un error porque la clave es un número. Esto se soluciona indicando el número entre corchetes [ ]. Las claves de tipo cadena dentro de comillas " " también pueden ir dentro de los corchetes: [9]

```
PS>$hash[2]  dos

PS>$hash["fecha"]

Miercoles, 30 de enero de 2013 15:00:28

PS>$hash["primero"]  1

PS>$hash["tabla"]

1 dos 3 4,5
```

9 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

Es posible acceder sólo a las claves del hash con el método keys:

```
PS>$hash.keys

fecha

tabla

primero

2
```

Y también sólo a los valores con el método determindo values:

```
PS>$hash.values

30/01/2013 15:01:30

1

dos

3

4,5

1

dos
```

Uno de los elementos de $hash es la tabla $tabla. Se pueden acceder a cualquiera de los valores de $tabla indicando su posición [ ]:

```
PS>$hash.tabla[0]

1

PS>$hash.tabla[1]

dos

PS>$hash.tabla[-1]

4,5
```

Para eliminar un elemento , por ejemplo la fecha de $hash:

```
PS>$hash.remove("fecha") $hash

Name            Value

----            -----

tabla           {1, dos, 3, 4,5}

primero         1

2               dos
```

Para vaciar la $hash se utiliza el método clear.

```
PS>$hash.clear()
```

3.7. Control de flujo: condiciones y bucles

3.7.1. Condiciones

3.7.2. If

Como en la mayoría de los lenguajes de programación, *if* se encarga de evaluar y comparar. PowerShell utiliza la estructura básica: if (condición) {acción}. También admite else y elseif. Por ejemplo: [10]

```
$lenguaje = "español"

if ($lenguaje -eq "ingles") {"El lenguaje es ingles."}
```

Si no se cumple la condición y se quiere evaluar otra condición, se debe utilizar elseif. Este ejemplo en varias líneas para verlo de manera optima:

```
$lenguaje = "español"

if ($lenguaje -eq "ingles") {

"El lenguaje es ingles." }

elseif ($lenguaje -eq "chino") {

"El lenguaje es chino." }
```

10 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

Si no se cumple ninguna condición, se puede ejecutar la acción que indica else

```
$lenguaje = "español"

if ($lenguaje -eq "ingles") {

"El lenguaje es ingles." }

elseif ($lenguaje -eq "chino") {

"El lenguaje es chino." }

else {

"El lenguaje no es ingles ni chino." }
```

Esta estructura de evaluación de condiciones siempre comienza con if, el orden de elseif y else no importa, dependiendo del camino que se quiera seguir, como $lenguaje no es ni "ingles" ni "chino", toma la acción de else

```
El lenguaje no es ingles ni chino
```

Los operadores de comparación más importantes:

Tabla IVI.I Operadores de comparación

-eq	igual que
-gt	mayor que
-ge	mayor o igual que
-It	menor que
-le	menor o igual que
-ne	distinto que
-like	parecido a (admite * como comodín)
-notlike	no parecido a (admite * como comodín)

Elaborado. Por el autor.

Pueden evaluarse dos y/o más condiciones con los operadores lógicos -and y -or

```
if ( ($lenguaje -ne "ingles") -and ($lenguaje -ne "chino") )
```

En este ejemplo se tienen que cumplir las dos condiciones alrededor de -and, y todo ello debe ir dentro de paréntesis.

3.7.3. Switch.

Si hay que evaluar muchas condiciones a la vez, es más cómodo utilizar switch

```
$lenguaje = "español"

Switch

($lenguaje)

{

        "español"       {"El    lenguaje    es  español."}
        "francés"       {"El    lenguaje    es  francés."}
        "alemán"        {"El    lenguaje    es  alemán."}
        "inglés"        {"El    lenguaje    es  inglés."}
        "chino"{"El             lenguaje    es  chino."}
        "danés"         {"El    lenguaje    es  danés."}
        default         {"El    lenguaje    no  está registrado."}

}
```

3.7.4. Where.

Se puede redirigir la salida de un cmdlet, objeto o colección de cosas por una tubería y acceder a sus propiedades. Muchas veces se necesita encontrar alguna propiedad concreta con algún valor concreto, aquí es donde se utiliza Where. Por ejemplo, ¿es 2013 el año actual?. [11]

```
PS>get-date | where {$_.year -eq 2013}
```

11 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

Aparece la variable especial $_ de la se hace referencia en las variables. Where toma la salida del cmdlet y evalúa lo que aparece a continuación entre llaves. $_ realmente es la salida de get-date, y $_.year es la propiedad año de get-date. Después, el operador -eq indica si $_.year es igual a 2013. Si se cumple la condición, devuelve la salida en sí como lo haría get-date, y en caso contrario no devuelve ningún valor. [12]

3.7.5. Bucles

PowerShell posee bucles: foreach, for, while, do-while y do-until. Utiliza tablas, listas, hashes y cualquier cosa que contenga una colección de elementos y tipos de datos.

3.7.6. Foreach

Foreach es el más fácil de usar, hace un recorrido directo por todos los elementos que determine el usuario, por ejemplo de una tabla:

```
PS>$tabla = 1, 'dos', 3, 'cuatro'
PS>foreach ($elementos in $tabla) {$elementos}
1
dos
3
cuatro
```

12 https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

Foreach crea un $elemento por cada elemento de $tabla y después, entre llaves, se puede hacer cualquier cosa con dicho elemento, en este caso lo muestra {$elemento}.

3.7.7. For

For crea e inicializa una variable índice, la cual evalúa una condición con esa variable y luego incrementa si se cumple la condición. Las 3 cosas anteriores deben ir separadas por (;) , todo ello entre paréntesis. Si la condición se cumple, ejecuta lo que hay a continuación entre llaves:

```
PS>for ($i=1; $i -le 3; $i++)
{$i}
1
2
3
```

Si $i es menor que 3, $i se incrementa en uno ($i++) y ejecuta lo que hay a continuación entre llaves (muestra el valor de $i). En la última vuelta, $i vale 3, lo muestra y se incrementa a 4. Ahora $i vale 4. En este momento $i no cumple la condición y el bucle For termina.

3.7.8. While

Es idéntico a un For y también evalúa la condición al principio del bucle, pero no tiene en cuenta ni la creación de la variable índice ni su incremento:

```
PS>$i = 1
PS>while ($i -le 3) {$i ; $i++}
1
2
3
```

La variable índice se crea antes de while y su incremento se produce dentro de la acción entre paréntesis (en una misma línea pueden ir dos o más acciones entre punto y coma como ya vimos). En este caso, si se omite $i++, $i siempre vale 1 y el bucle sería infinito. La condición no tiene porque ser la evaluación de un número, también puede ser la evaluación de una cadena: [13]

```
PS>$cadena = ""

PS>while ($cadena -ne "correcto") {$cadena = read-host "Contraseña"}
```

En este caso, mientras $cadena sea cualquier cosa menos "correcto" pide por teclado un valor nuevo para $cadena. Cuando se teclea "correcto", sale del bucle, y en caso contrario, vuelve a pedir un valor nuevo para $cadena.

3.7.9. Do-While

Igual que While pero evalúa al final de cada bucle. Se toma en cuenta este script de ejemplo:

```
$cadena = "correcto"

do

{ $cadena = read-host "Contraseña" }

while ($cadena -ne "correcto")
```

13. https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

La única diferencia al evaluar al final de una vuelta es que la primera vuelta siempre se produce. En este caso, aunque $cadena es igual a "correcto", en la primera vuelta siempre nos pide un nuevo valor para $cadena.

3.8. Do-Until

Exactamente igual que Do-While pero cambia la lógica. Do-While da vueltas mientras se produzca la condición y Do-Until da vueltas hasta que se produzca la condición:

```
$cadena = "correcto"

do

{  $cadena = read-host "Contraseña" }

until ($cadena -eq "correcto")
```

En este ejemplo, la salida del bucle se produce cuando $cadena sea igual a "correcto".

3.8.1. Bucles especiales

Se puede obtener una enumeración rápida del 1 al 10 así:

```
PS>1..10
```

Si se redirige esa salida por una tubería, cada elemento de la enumeración sale de la tubería como el variable $_ y podemos hacer algo con ello:

```
PS>1..10 | %{$_ * 5}
```

Y se tiene la tabla de multiplicar del 5. Cada $_ (cada elemento de la enumeración que sale de la tubería) es multiplicado por 5 y se visualiza el resultado.

3.9. Funciones

Existen varias formas de definir funciones en PowerShell.

```
PS>function suma($x,$y) { $resultado = $x + $y; return $resultado }
```

Lo anterior define la función suma que toma como parámetros las variables $x e $y (entre paréntesis). A continuación, entre las llaves hace algo con ellas (las suma, coloca el resultado en $resultado y devuelve $resultado). [14]

La llamada a la función es simple: se escribe el nombre de la función y a continuación se indican los parámetros:

```
PS>suma 1 2

3
```

La definición de la función anterior puede abreviarse más así:

```
PS>function suma($x,$y) { return $x + $y }

PS>suma 1 2

3
```

14. https://e740a7b4-a-62cb3a1a-s-sites.googlegroups.com/site/ramiroencinas/doc/PowerShell2.0_v1.pdf

E incluso no tiene porqué devolver resultados con return, también los devuelve sin él:

```
PS>function suma($x,$y) { $x + $y }

PS>suma 1 2

3
```

Por último, crear una variable con el resultado de la llamada a la función

```
PS>function suma($x,$y) { $x + $y }

PS>$resultado = suma 1 2

PS>$resultado 3
```

3.10. Archivos

Todos los datos se guardan en archivos. Veamos algunas de las operaciones con archivos que puede realizar PS2.

3.10.1. Comprobación de archivos

¿Existe el archivo datos.txt en la carpeta actual?

```
PS>test-path .\datos.txt
```

Si devuelve True, existe. En caso contrario devuelve False

CAPITULO IV

4. ANÁLISIS DEL MÉTODO GRÁFICO Y SCRIPT.

En este capítulo se compara la administración entre modo gráfico y modo script con PowerShell, basado en el funcionamiento de los sistemas operativos de la Academia Microsoft de la ESPOCH. Entre los parámetros a comparar se encuentra la portabilidad, con indicadores como transportación, ejecución y la compartición, posteriormente se encuentra el parámetro de usabilidad, con precios, desempeño y tiempo de utilización, finalmente el parámetro de rendimiento con indicadores de seguridad, flexibilidad, consola vs gráfico, recursos hardware, tiempo de respuesta, tiempo de codificación, tiempo de administración y tiempo de ejecución. Parametros que sirven de base para realizar un análisis y obtener como resultado que los scrips desarrollados con Microsoft PowerShell automatizan la administración.

4.1. Administración mediante interface gráfica vs administración scripts

En este capítulo se describe los resultados obtenidos y su análisis correspondiente entre el modo de administración gráfica y el modo de administración con la interfaz Microsoft PowerShell mediante la creación de un pequeño script en el cual se analizara los tiempos tomados para la administración.

Los parámetros que se considerado para este estudio comparativo son:

a) **Portabilidad**: Propiedad de un software para ser utilizadas en distintas computadoras de una empresa o institutos.

   _ Transportación.

   _ Ejecución.

   _ Compartición.

b) **Usabilidad:** Medida en la que un sistema se puede usar por determinados usuarios para conseguir unos objetivos específicos.

   _ Costo

   _ Desempeño

   _Tiempo utilización

c) **Rendimiento:** Permite determinar resultados en tiempos de ejecución de un sistema operativo.

   _ Seguridad

   _ Flexibilidad

   _Consola vs grafica

   _Recursos Hardware

   _Tiempo de respuesta

_Tiempo de codificación

_Tiempo de administración

_Tiempo de ejecución.

La escala asignada para la evaluación esta referenciada bajo los distintos niveles que se muestran a continuación.

Tabla IV. I. Niveles de muestras por medio de Likert

RANGO	PORCENTAJE	NIVEL
1	0% - 20%	Deficiente
2	21% - 40%	Regular
3	40.1% - 60%	Moderado
4	60.1% - 80%	Bueno
5	80.1% - 100%	Máximo

Elaborado. Por el autor.

Los parámetros a evaluar están divididos de la siguiente manera: se ha considerado que la plataforma que cumpla con todas las variables o parámetros definidos anteriormente se les asignara los valores de 5 el cual indicará un nivel máximo de funcionalidad, de igual forma un rango de 1 mostrará que la plataforma posee un nivel deficiente de funcionalidad y finalmente los rango de 2-3-4 evidenciará que la plataforma cumple a mediana regularidad con los requerimientos básicos necesarios.

4.2. Análisis comparativo.

Los parámetros para definir la portabilidad de las plataformas son:

4.2.1. Transportación

Tabla IV. XX. Parámetros de trasportación para el Análisis.

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	0	0%
SCRIPT	5	100%

Elaborado. Por el autor.

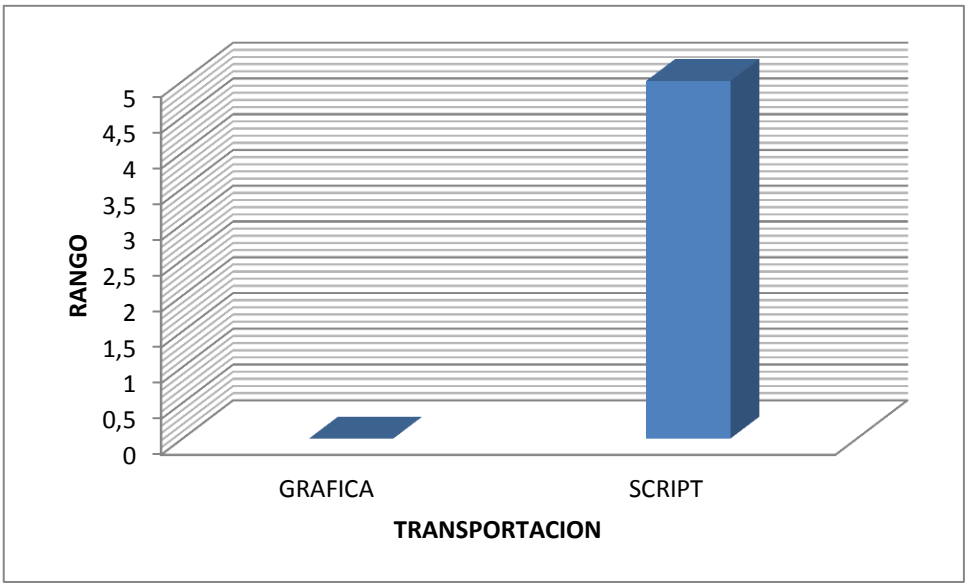


Figura IV. 1. Gráfico de barras de la transportación
Fuente: El autor.

La configuración gráfica no puede ser transportada de un servidor a otro, por lo que obtiene un valor deficiente, mientras que la configuración por scripts es un 100% transportable y obtiene un valor de 5 (excelente), concluyendo que la administración por medio de script es muy optima al momento de transportar entre servidores que actúan en varios a escenarios.

4.2.2. Ejecución

Tabla IV. XXI Parámetro de ejecución para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	2	40%
SCRIPT	4.5	90%

Elaborado. Por el autor.

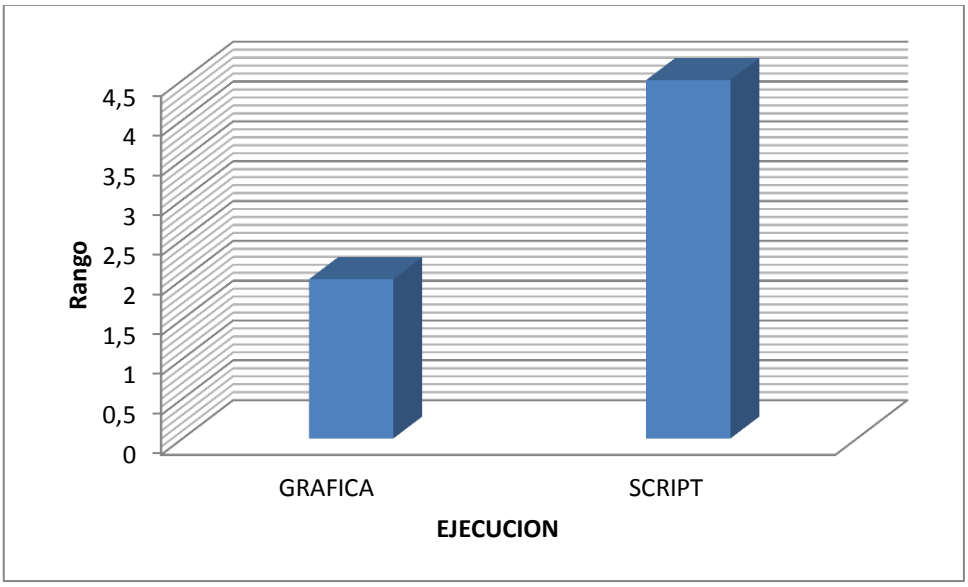


Figura IV. 2. Gráfico de barras de la Ejecución

Fuente: El autor.

La configuración gráfica ejecuta las acciones en un servidor de manera lenta tomándose más de 10s en ejecutarse, por ello se ha colocado un valor de 2, mientras que la ejecución de un script lo realiza en menos de 4s lo que significa un ahorro significativo de tiempo por lo que obtiene un valor de 4.5, se concluye que el script ejecuta con rapidez comparado con la administración gráfica.

4.2.3. Compartición

Tabla IV.IV. Parámetro de Compartición para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	1	20%
SCRIPT	4.5	90%

Elaborado. Por el autor.

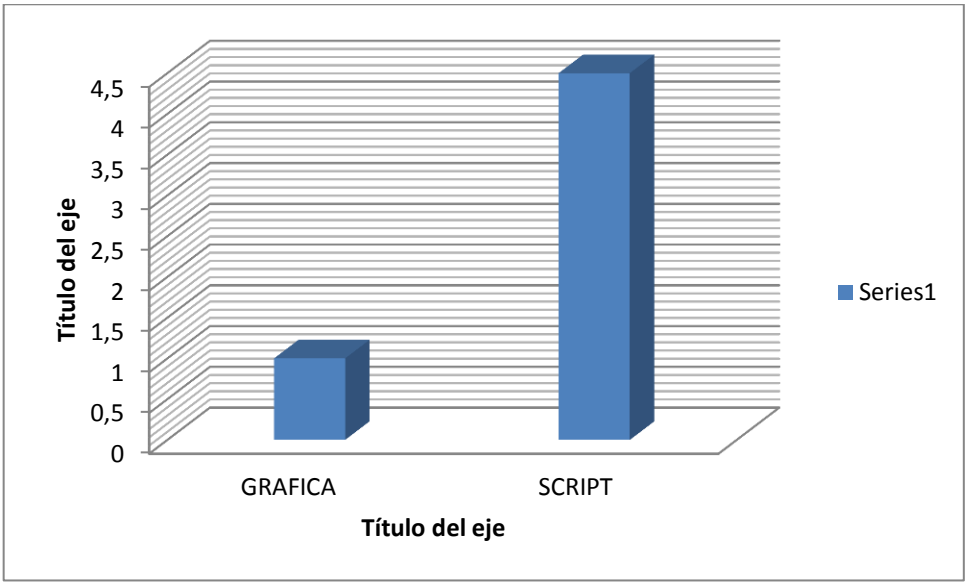


Figura IV. 3. Gráfico de barras de la Comparación

Fuente: el autor.

En este punto se compara la compartición, ya que la compartición de la forma de configuración es muy limitada en la forma gráfica obtiene una puntuación de 1 ya que se deseamos compartir las configuración se debe sacar un backup general del servidor, en cambio la forma script se puede compartir en un 90% por lo que tiene una puntuación de 4,5, ya que si se quiere modificar algo solo se comparte el script deseado en cualquier dispositivo ya sea disco duro, flash memory o por medio de la red

Los parámetros para definir la Usabilidad de las plataformas son:

4.2.4. Costo

Tabla IV. XXII. Parámetro de Costo para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	4,5	90%
SCRIPT	1	10%

Elaborado. Por el autor.

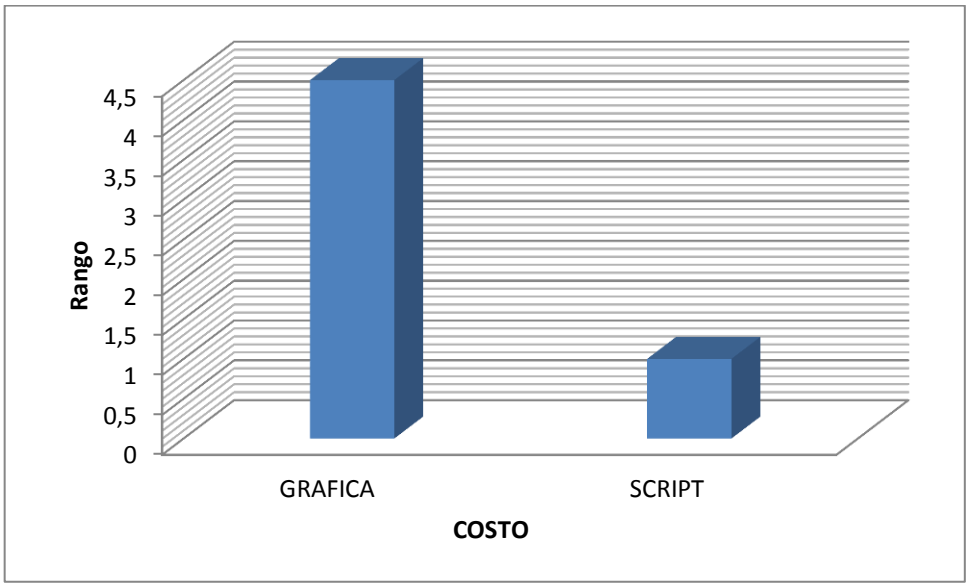


Figura IV. 4. Gráfico de barras deL COSTO.

Fuente: el autor.

Relacionado con el costo en administración en la forma gráfica se debe el tiempo-costo utilizado para gestionar el sistema operativo, los cuales son precios muy elevados, en cambio el software para realizar el script utiliza menos tiempo-costo y pude funcionar en cualquier versión anterior de los sistemas operativos Microsoft.

4.2.5. Desempeño

Tabla IV. XXIIII. Parámetro de Desempeño para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	5	100%
SCRIPT	5	100%

Elaborado. Por el autor.

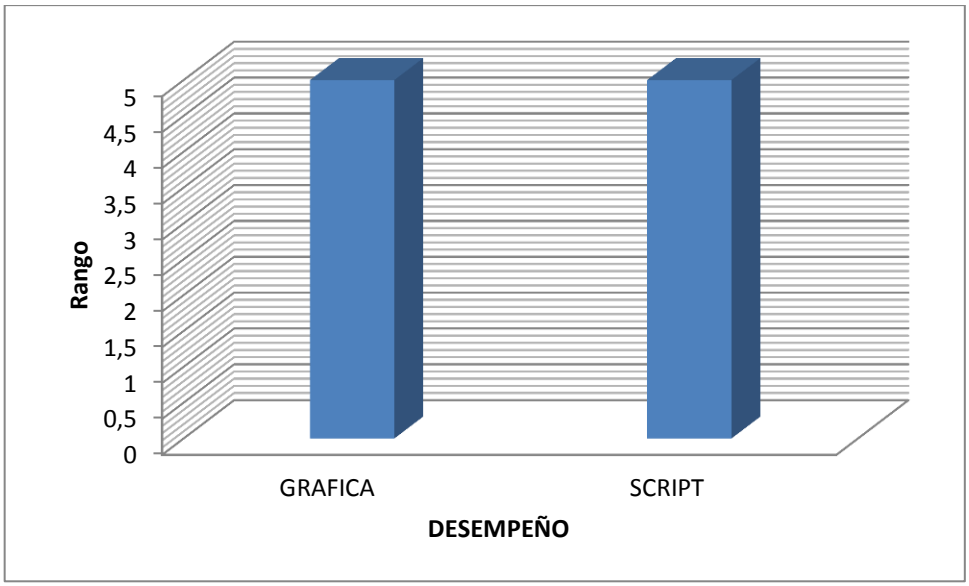


Figura IV. 5. Gráfico de barras de la desempeño
Fuente: El autor.

Ya que tanto la administración gráfica como la realizada por script realizan la misma acción, el desempeño es igual obteniendo el puntaje de 5.

4.2.6. Tiempo utilización en la administración

Tabla IV. XXIVI. Parámetro de Tiempo de utilización para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	5	100%
SCRIPT	3	60%

Elaborado. Por el autor.

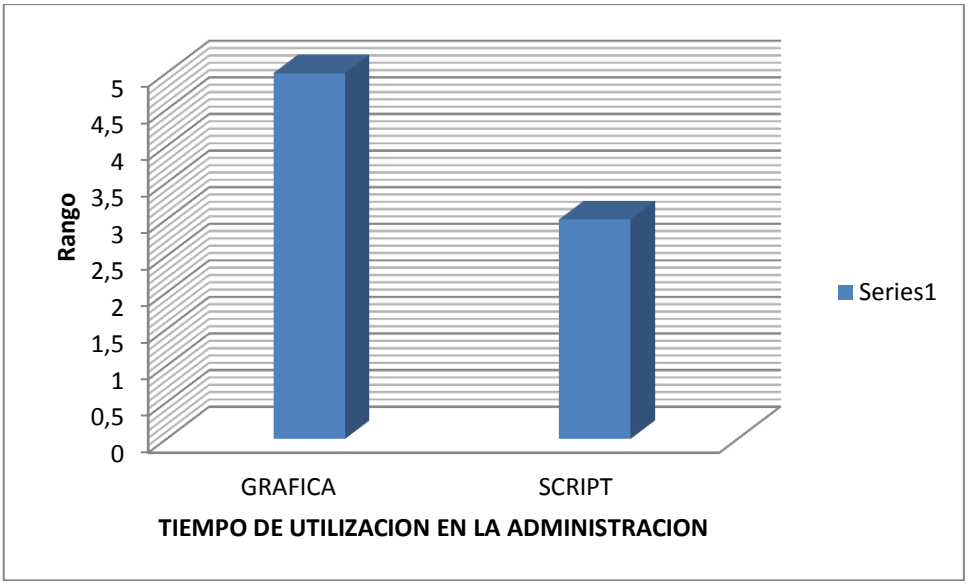


Figura IV. 6. Gráfico de barras del tiempo de utilización de la administración
Fuente: El Autor.

Se analiza el tiempo que se lleva realizar una administración, ya que la administración gráfica conlleva más tiempo se ha colocado un valor de 5, mientras que la forma utilizando script utiliza menos tiempo obtenie un valor de 3.

Los parámetros para definir la Rendimiento de las plataformas son:

4.2.7. Seguridad

Tabla IV. XXVI. Parámetro de seguridad para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	2	40%
SCRIPT	4	80%

Elaborado. Por el autor.

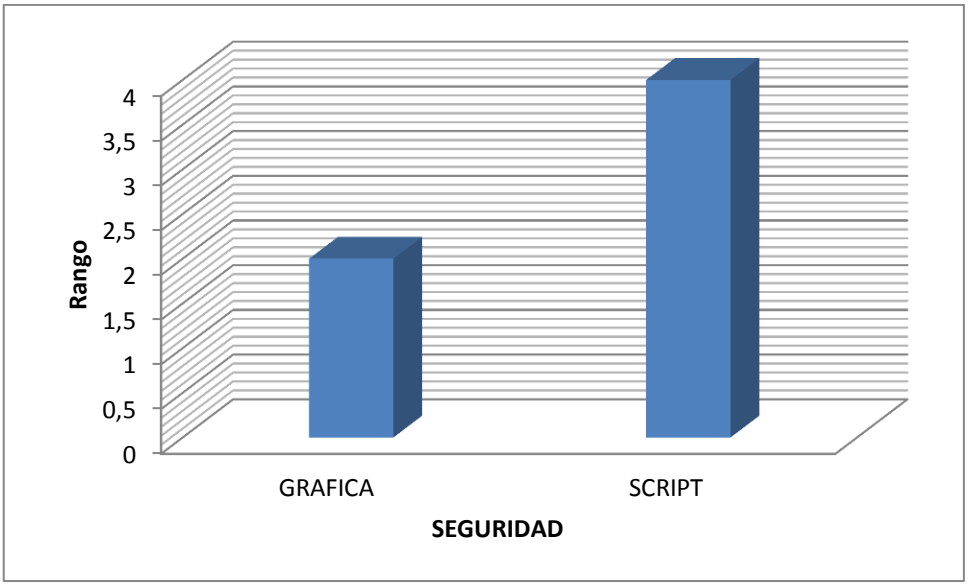


Figura IV. 7. Gráfico de barras de la Seguridad

Fuente: el Autor.

La seguridad es un factor importante dentro de la administración, en este punto se ha analizado que la administración gráfica puede ser cambiada con facilidad si no se tiene las debidas precauciones, mientras que en la administración script, si está firmado digitalmente y se desactiva la ejecución de la misma en el servidor no se podrá realizar cambios en ningún otro sitio.

Además la configuración gráfica es susceptible a cambios ocurridos por virus, en cambio los script son texto plano que no se pueden modificar por la desconfiguración ocurrida por ejecución de virus.

4.2.8. Flexibilidad

Tabla IV.IX. Parámetro de flexibilidad para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	5	100%
SCRIPT	5	100%

Elaborado. Por el autor.

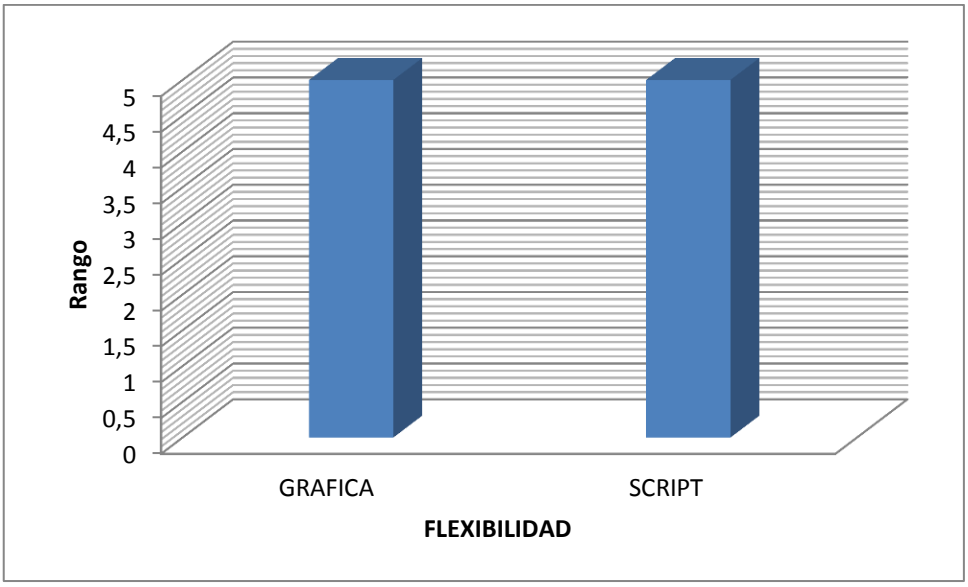


Figura IV. 8. Gráfico de barras de la Flexibilidad
Fuente: el autor.

Se analiza la flexibilidad de los cambios en la administración concluyendo que ambas formas de gestión, se puede realizar cambios de manera fácil, concreta para lo cual obtienen un calificación de 5, con esto se puede decir que son flexibles a la hora de cambios solicitados por los administradores.

4.2.9. Utilización de consolas de texto

Tabla IV. XXVI. Parámetro de ubicación de consolas para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	1	20%
SCRIPT	5	100%

Elaborado. Por el autor.

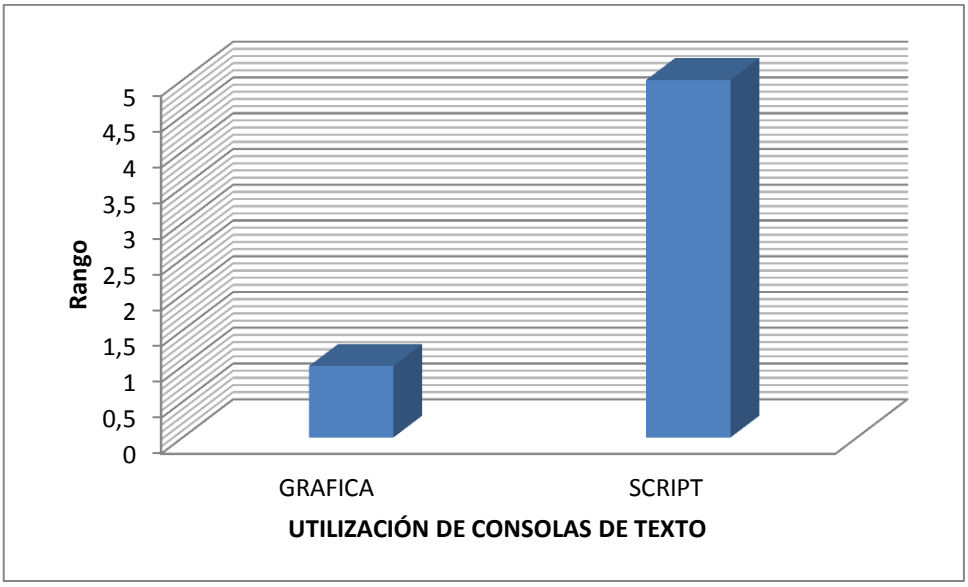


Figura IV. 9. Gráfico de barras de la utilización de consolas de texto
Fuente: El autor.

La utilización de consolas de texto para la administración, mientras que la forma gráfica se utiliza muy poco, por ejemplo la herramienta CMD, en la administración por medio de script se lo realiza 100% en consola de texto.

4.2.10. Recursos Hardware

Tabla IV.XI. Parámetro de recursos Hardware para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	5	100%
SCRIPT	4	80%

Elaborado. Por el autor.

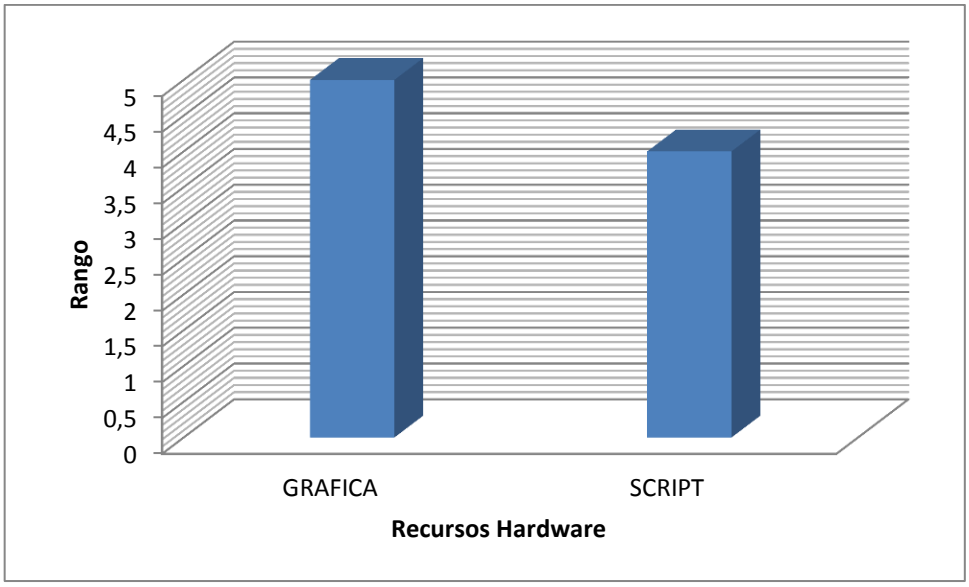


Figura IV. 10. Gráfico de barras de los recursos Hardware
Fuente: el autor.

Ambas metodologías utilizan recursos hardware,pero la gestión por medio de script mucho mas efectiva al momento de utiliza pocos recursos es por eso que se ha asignado un valor de 4, en cambio la administración por modo grafico utiliza mas recursos como por ejemplo tarjeta grafica mas memoria ram , mejor tranferecia en disco duro para realizar una acción.

4.2.11. Tiempo de respuesta a un resultado solicitado

Tabla IV. XXVIII. Parámetro de tiempo de respuesta para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	3	60%
SCRIPT	4.5	90%

Elaborado. Por el autor.

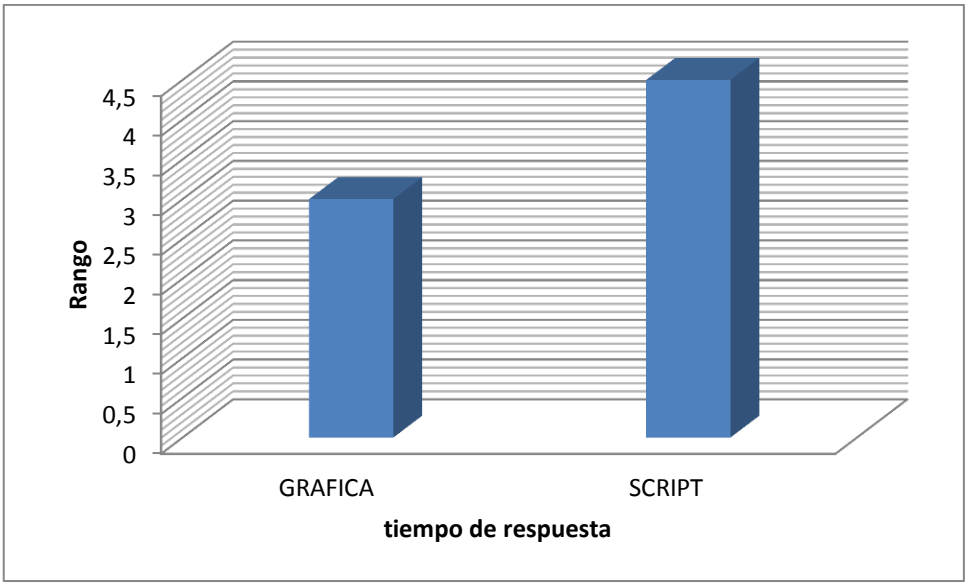


Figura IV. 11. Gráfico de barras de la Tiempo de respuesta
Fuente: El Autor.

En esta parte se analiza el tiempo de respuesta a un resultado solicitado, como por ejemplo, un reporte de los servicios en ejecución, siendo la forma de administración script optima por lo obtiene un valor del 90% de efectividad.

4.2.12. Tiempo de codificación

Tabla IV. XXVIIII. Parámetro de tiempo de codificación para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	0	0%
SCRIPT	4.5	90%

Elaborado. Por el autor.

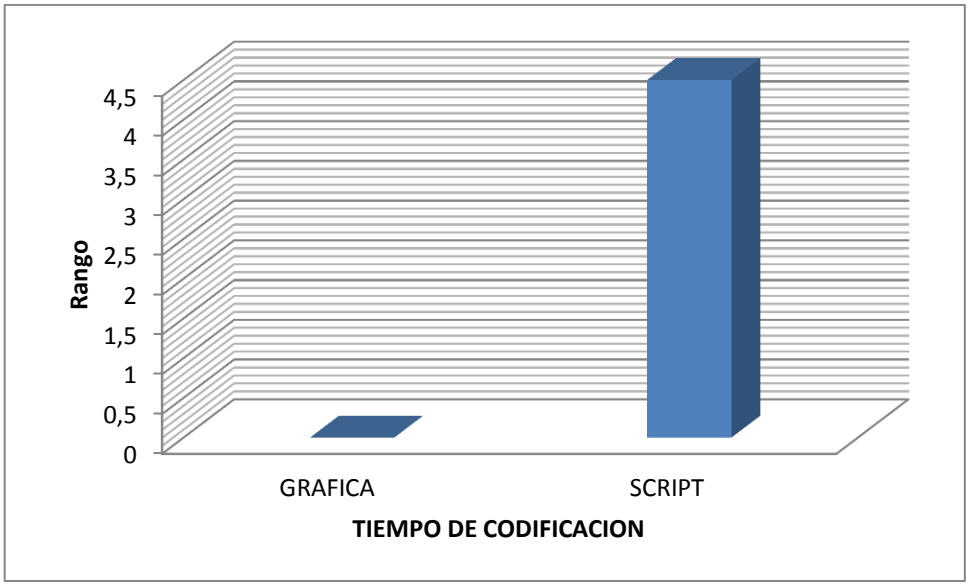


Figura IV.12. Gráfico de barras del tiempo de codificación
Fuente: EL Autor.

Se analiza el tiempo que lleva codificar, por lo cual la administración script utiliza un 100% de tiempo en codificación mientras que en el modo gráfico no se codifica por lo que no utiliza tiempo para codificar.

4.2.13. Tiempo de administración en la modificación.

Tabla IV.XIV. Parámetro de tiempo de Administración para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	4.5	90%
SCRIPT	2	40%

Elaborado. Por el autor.

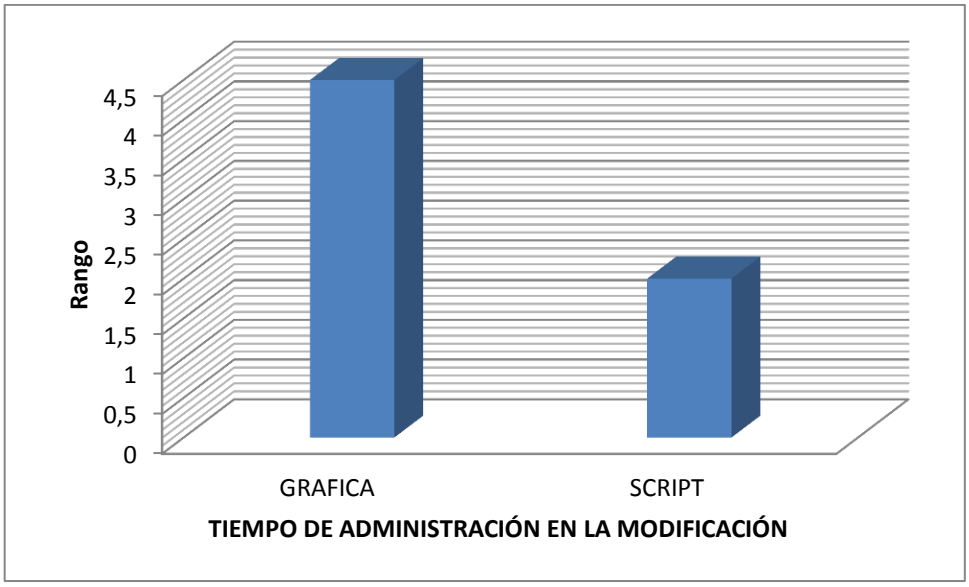


Figura IV. 13. Gráfico de barras del tiempo de administración en la modificación
Fuente: El autor.

En la forma gráfica el tiempo de administración utilizado al momento de varias modificaciones es mayor que la de script por eso obtienen un valor de 4.5, se puede decir que al realizar una modificación en el script se lo hace más rápido.

4.2.14. Tiempo de demora en ejecución de un cambio

Tabla IV. XXIX. Parámetro de Tiempo de demora para el análisis

TEGNOLOGIA	RANGO	PORCENTAJE
GRAFICA	4.5	90%
SCRIPT	2	40%

Elaborado. Por el autor.

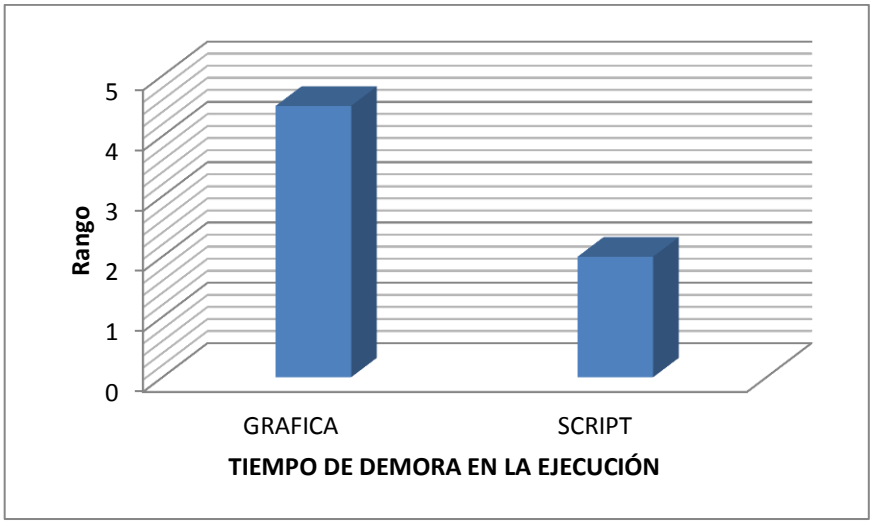


Figura IV. 14. Gráfico de barras del tiempo de demora en la ejecución de un cambio

Fuente: El autor.

Por el análisis realizado, se verifica que el tiempo que se demora al ejecutar un cambio luego de la administración, el script demora menos en realizarlo, mientras que en la administración gráfica utiliza más tiempo ya que se tiene que seguir una serie de pasos antes de completar el cambio.

4.3. Interpretación de los datos.

a) Interpretación de la Portabilidad

Tabla IV. XXXI. Interpretación de Portabilidad

PORTABILIDAD				
Parámetros	**GRAFICO**		**SCRIPT**	
	Rango	**%**	**Rango**	**%**
Trasportación	0	0%	5	100%
Ejecución	2	40%	4.5	90%
Compartición	1	20%	4.5	90%
Total	3	60%	14	280%
Promedio	**1**	**20%**	**4.66**	**90%**

Elaborado por: El autor.

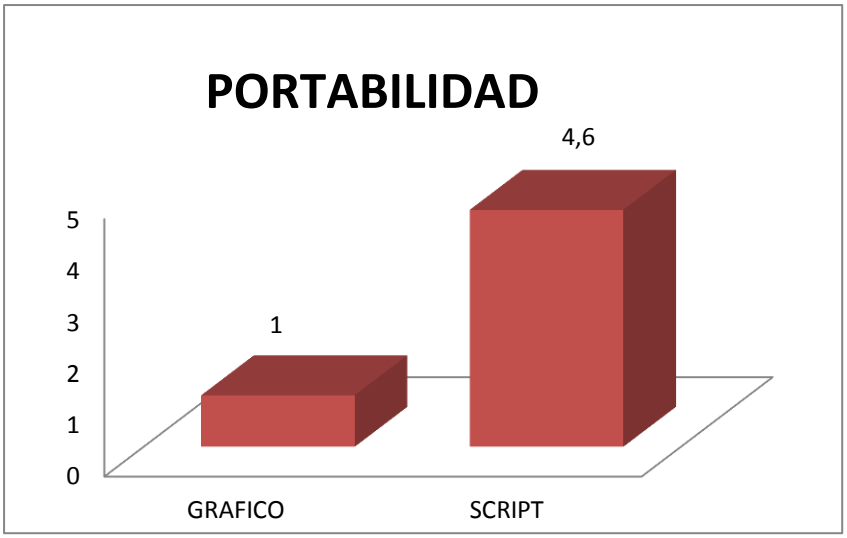


Figura IV. 15. Gráfico de barras de la Portabilidad
Fuente: El autor.

Como resultado de la evaluación en portabilidad que se puede apreciar en la Figura IV.15, la administración gráfica con su valor de 1 que representa al 20%, dentro del rango establecido anteriormente, por lo cual se concluye que es deficiente, mientras que la administración script con un valor de 4,6 que representa el 90%, hay una diferencia al momento de transportar, ejecutar y compartir la configuración de un escenario a otro.

b) Interpretación de la Usabilidad

Tabla IV. XXXI. Interpretación de la Usabilidad

USABILIDAD				
Parámetros	**GRAFICO**		**SCRIPT**	
	Rango	**%**	**Rango**	**%**
Costo	4.5	90%	1	10%
Desempeño	5	100%	5	100%
Tiempo de Utilización de la administración	5	100%	3.5	60%
Total	14.5	290%	9.5	170%
Promedio	**4.83**	**96.66%**	**3.16**	**56.66%**

Elaborado por: el autor.

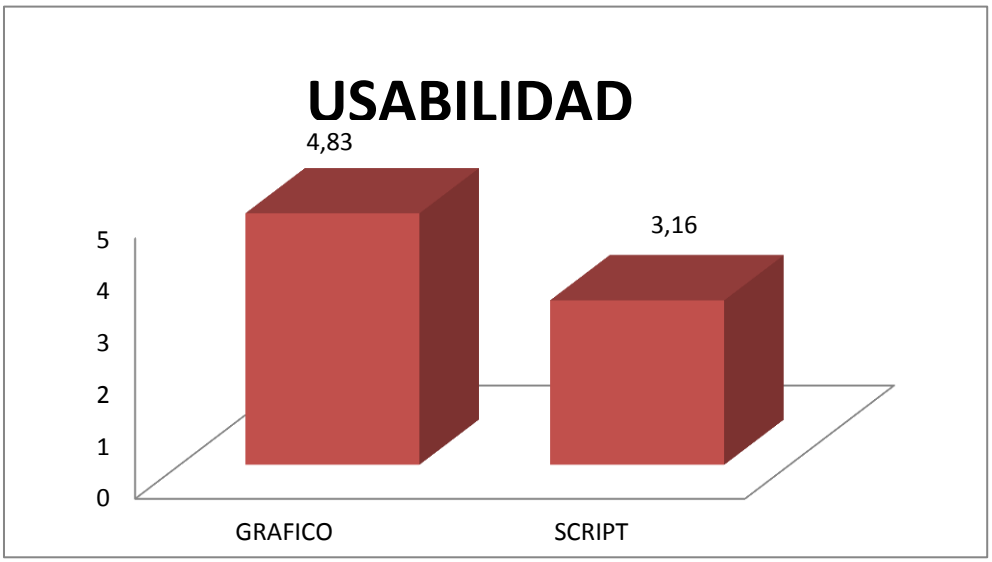


Figura IV. 16. Gráfico de barras de la Usabilidad
Fuente: Los Autores.

En la Figura IV.16 se muestra que el parámetro usabilidad con un valor de 4,83 correspondiente al 96.66% en la administración gráfica, se concluye que utiliza mayor tiempo en administración y sus costos son elevados, mientras que la administración mediante scripts con un valor de 3.16 que equivale al 56,66%, se puede decir que se utiliza menor cantidad de tiempo, costos al momento de realizar una gestión dentro de las TI.

c) Interpretación del Rendimiento.

Tabla IV. XXXIII. Interpretación del Rendimiento

RENDIMIENTO				
Parámetros	**GRAFICO**		**SCRIPT**	
	Rango	**%**	**Rango**	**%**
Seguridad	2	40%	4	80%
Flexibilidad	5	100%	5	100%
Utilización de consola de	1	10%	5	100%

texto				
Recursos Hardware	5	100%	4	80%
Tiempo de respuesta a un resultado solicitado	3.5	60%	4.5	90%
Tiempo de codificación	0	0%	4,5	90%
Tiempo de administración en la modificación	4.5	90%	2	40%
Tiempo de demora ejecución	4.5	90%	2	40%
Total	25.5	490%	32	640%
Promedio	**3.18**	**61%**	**4**	**80%**

Elaborado por: el autor.

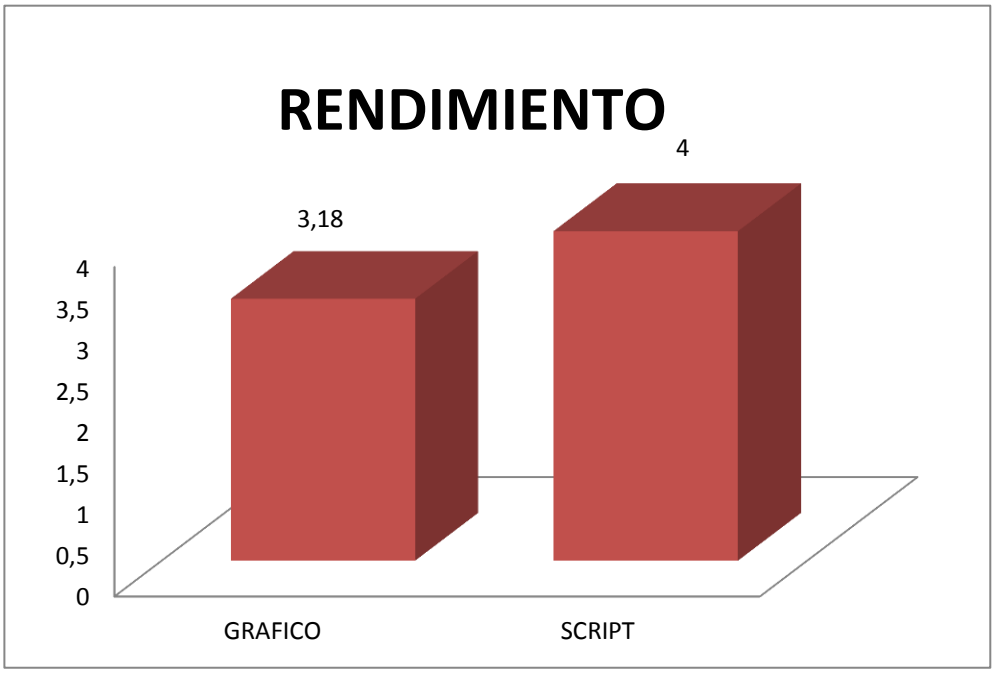


Figura IV. 17. Gráfico de barras del rendimiento
Fuente: Los Autores.

En la Figura IV.17, la administración de forma gráfica con 3.18 que corresponde al 61% obtiene un menor desempeño al momento de mejorar los tiempos de respuesta a cualquier petición del administrador, mientras que la administración por medio de Scripts con un valor de 4 que equivales al 80%, los scripts optimizan tiempo de recursos, tiempo de consultas y modificación.

4.4. Descripción de resultados

Tabla IV. XXXIIIX. Descripción general de los resultados

RESULTADO GENERAL				
Parámetros	**GRAFICO**		**SCRIPT**	
	Rango	**%**	**Rango**	**%**
Portabilidad	1	20%	4.66	60%
Usabilidad	4.83	96.66%	3.16	56.66%
Rendimiento	3.18	61%	4	80%
Total	9,01	178%	11,82	197%
Promedio	**3.003**	**59%**	**3.94**	**66%**

Elaborado por: el autor.

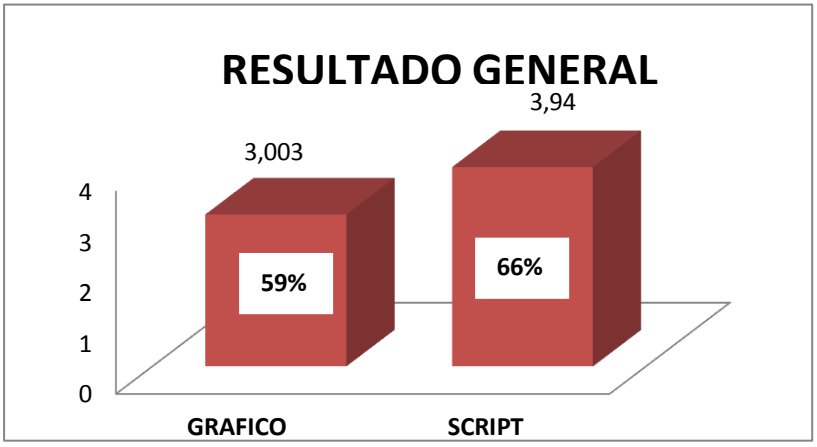


Figura IV. 16. Gráfico de barras del análisis del resultado general
Fuente: Los Autores.

Se concluye que la administración gráfica tiene algunos limitantes como la portabilidad ya que no se puede transportar ni facilita la creación de objetos de manera rápida y sencilla con respecto a los scripts.

También se observa que una de las ventajas de la administración con scripts es la optimización de tiempo y dinero, lo que permite resultados más rápidos, ser manejable para los administradores y optimizar su tiempo la administración del servidor, mientras que la administración grafica conlleva mayor recurso y estabilidad en cuanto a los tiempos de respuesta.

Se puede decir que la administración por medio de scripts automatiza de manera significativa a administración de los servidores Microsoft ya que ayuda a los administradores a ahorrar tiempo en la configuración, creación, modificación y eliminación de procesos administrativos los cuales son muy demorados en la administración gráfica.

CAPITULO V

5. CONFIGURACIÓN E IMPLEMENTACIÓN DEL SISTEMA DE ADMINISTRACIÓN MEDIANTE EL USO E SCRIPTS GENERADOS POR POWERSHELL.

Para la administración de TI en la academia Microsoft, se va a proceder a rediseñar el método de gestión, siguiendo los objetivos y alcances planteados en la justificación práctica.

La aplicación contempla la implantación de scripts, para la gestión de usuarios, unidades organizativas, grupos y otros recursos que sufren constantes cambios durante el periodo de un curso.

Los scripts generados se pondrán al alcance de los estudiantes mediante un carpeta compartida, que se podrá revisar instantáneamente.

5.1.Ingeniería de la Información.

El objetivo fundamental de Windows PowerShell es proporcionar un control administrativo mejor y más sencillo de los sistemas operativos, de forma interactiva o mediante scripts.

La Metodología comprende las fases de: Análisis, Planificación, Contenido, Programación y ejecución.

5.2.Definición del Ámbito

Crear una solución informática que provea la automatización, reutilización, planificación, organización de los procesos que realiza el administrador de la Academia MICROSOFT - ESPOCH.

Además se pretende generar la creciente interacción entre el usuario y el servidor, para que pueda ser más práctico con la finalidad de llevar un mejor control de los accesos a los sistemas servidores.

La solución que se pretende realizar se ve enfocada en el mejoramiento de los procesos que realiza en el sistemas servidor de la Academia MICROSOFT - ESPOCH, para mejor su calidad de servicio y adaptarse a la tecnología actual.

El modelo de administración de TI para el sistema de la academia MICROFOST, se lo realiza con el desarrollo de pequeños scripts programados en Microsoft PowerShell, se hará énfasis en los usuarios y sus permisos.

Dentro de ámbito de administración existen dos tipos de usuarios que posee el servidor de la academia Microsoft:

- Usuarios del dominio o usuarios locales.
- Administradores.

Los usuarios del dominio son sencillamente usuarios de academia.espoch.edu.ec que entran a utilizar los servicios del servidor. Dependiendo de cómo el administrador ha configurado el servidor, los usuarios del dominio podrán utilizar libremente por todo el contenido o tener restringido el acceso a cierto tipo de contenidos, reservados para administradores.

Los administradores están registrados en su sitio con un nombre de usuario y contraseña. Este nombre de usuario y contraseña les permite acceder al área restringida del servidor, recibiendo privilegios especiales no disponibles para los usuarios del dominio.

5.3.Requerimientos.

5.3.1. Requerimientos Funcionales.

Para solucionar los inconvenientes citados en el literal anterior se puede desarrollar una administración del centro de cómputo, el mismo que permitirá.

1. Crear un dominio.
2. Configurar DNS.
3. Ingresar, Consultar y Actualizar las unidades organizativas.
4. Ingresar, Consultar y Actualizar los usuarios del dominio.
5. Ingresar, Consultar y Actualizar los grupos de dominio.
6. Ingresar, Consultar y Actualizar los servicios utilizados.
7. Consultar los procesos en ejecución.
8. Consultar servicios de DHCP.
9. Consultar impresoras en la red.
10. Consultar y modificar archivos de usuario.

5.3.2. Requerimientos No Funcionales

A continuación se muestran los requerimientos no funcionales más relevantes del sistema operativo con sus respectivas características:

- **Rendimiento.**

- Tiempos de respuesta al, abrir un login de usuario para la autenticación será de 1s.

- Tasa esperada de velocidad de respuesta dada la tasa de clientes conectados será mucho más rápida.

- **Fiabilidad.**

- Capacidad para tolerar errores en un 98%.

- Capacidad para tolerar sobrecargas en el volumen de información, de usuarios o de procesos en un 70%.

- **Disponibilidad**

- El sistema estará funcionando 24*7 ósea 24 h por 7 días.

- Empleo de sistemas de respaldo.

- **Seguridad**

- El sistema tendrá una autenticación óptima.

- Uso de sesiones de usuario.

- **Portabilidad**

- Utilización de estándares para la presentación de información en reportes.

- Capacidad de manejo y transportación de un sitio a otro.

- **Mantenibilidad**

- Documentación del diseño de los scripts y de la codificación de la solución.

- **Escalabilidad**

- Diseño de un modelo se administración cambiable.

- Empleo de scripts para hacer al sistema compatible con otros sistemas.

- **Reusabilidad**

- Uso de estándares en los formatos para los datos sea reutilizable en otras ocasiones.

5.4. Estudio de Factibilidad

5.4.1. Factibilidad Técnica

Recurso Humano

Existe el recurso humano capacitado para realizar la administración del sistema, obteniendo una operación garantizada y uso garantizado.

Ing. Washington Luna E. (Administrador)

Recurso Hardware

El recurso hardware existente en la academia Microsoft es el siguiente**:**

- Intel Core 2 Duo 2.1Ghz
- Memoria ram de 4gb
- Disco duro de 1000gb
- DVD-RW
- Teclado, Mouse, Monitor

Recurso Software

El software disponible para los administradores y los estudiantes se detalla a continuación:

- Sistema Operativo: Microsoft Windows Sever 2008 R2
- Tecnología de Desarrollo: PowerShell 2.0

- Microsoft Office 2010.

5.4.2. Factibilidad Operativa

- Ing. Washington Luna y Sr. Byron Huera.
- Usuarios de la Academia Microsoft.

5.4.3. Factibilidad Legal

Existe la autorización de las autoridades respectivas por lo que no existe ningún tipo de impedimento legal para el desarrollo de la administración.

5.5.Planificación Temporal

Ver Anexo 1

5.6.Análisis del Sistema

El uso de herramientas externas constituye un objetivo de diseño a largo plazo de Windows PowerShell. A medida que crezca el sistema, los usuarios seguirán encontrándose con situaciones en las que las herramientas disponibles no hacen todo lo que necesitan que hagan. Este sistema se basa en una administración centralizada el cual posee controlador de dominio y sus diferentes unidades organizativas, para lo cual se hace énfasis en dicha configuración.

5.7. Diseño de la red.

En el diseño de la red se basa generalmente en los puntos de acceso al internet y a la intranet ya que posee un total de 24 puertos conectados directamente a un switch el cual trasmite información entre las PC's clientes y el servidor.

5.7.1. Diagrama de administración son controlador de dominio.

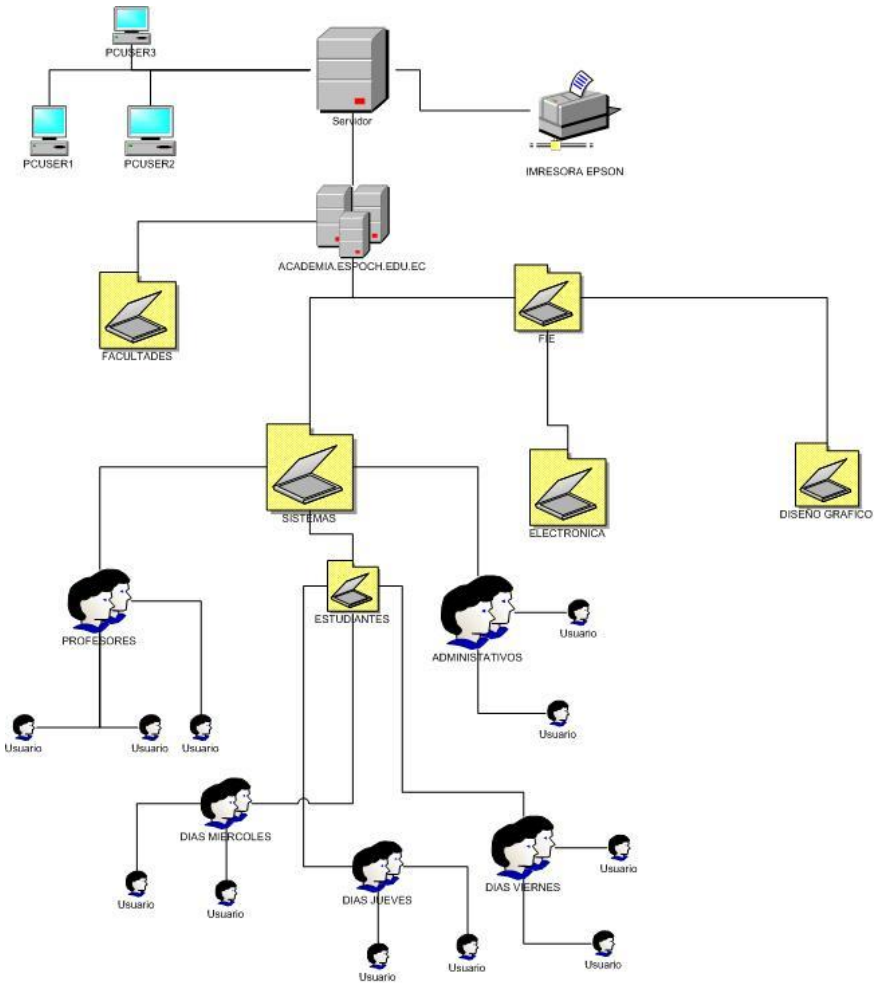


Figura V. 2. Diagrama de Controlador de dominio de la academia Microsoft.
Fuente: EL Autor

5.8. Implementación y Pruebas.

5.8.1. Implementación y Pruebas del servidor

Configuración y administración del servidor.

La implementación de cmdlets, Windows PowerShell integra soluciones de todos los escenarios alternativos posibles.

1) El primer escenario de administración es la automatización de un computador local en donde se realiza los siguientes puntos:

a) Crear y administrar usuario local.

b) Crear y administrar grupos locales.

c) Administrar procesos locales.

d) Administrar servicios locales.

e) Recopilar información acerca de los equipos locales.

f) Trabajar con las instalaciones de software.

g) Trabajar con el estado de un equipo.

h) Trabajar con impresoras.

i) Realizar tarea de red.

j) Trabajar respaldos del sistema.

.

Escenario de administración local.

a) Crear y administrar usuario local.

- **Creación de usuarios locales.**

Código

```
param($computer="localhost",$user="academiaespochfie",
$password="P@ssw0rd",$help)
function funHelp()
{
$helpText= @ "
DESCRIPCIÓN:
NOMBRE: creaciondeusuarioslocales.ps1
Crea un usuario local ya sea en un equipo local o remoto.
PARÁMETROS:
-equipo Especifica el nombre del equipo en el que ejecutar
 la secuencia de comandos  usuario Nombre de usuario para crear
-help Muestra el archivo de ayuda
SINTAXIS:
creaciondeusuarioslocales.ps1
Genera un error. Debe proporcionar un nombre de usuario
creaciondeusuarioslocales.ps1 ordenador tesisfie usuario academiaespochfie
-password Passw0rd
Crea un usuario local denominado academiaespochfie
en un equipo denominado tesisfie con la contraseña Passw0rd
creaciondeusuarioslocales.ps1  usuario academiaespochfie -password Passw0rd
creaciondeusuarioslocales.ps1 -ayuda?
Muestra el tema de ayuda para el script
"@$helpText
exit
}
if($help){ "Obtaining help ..." ; funhelp }
if(!$user -or !$password)
```

```
{
$(Throw 'A value for $user and $password is required.
Try this: creaciondeusuarioslocales.ps1 -help ?')
}
$objOu = [ADSI]"WinNT://$computer"
$objUser = $objOU.Create("User", $user)
$objUser.setpassword($password)
$objUser.SetInfo()
$objUser.description = "Test user"
$objUser.SetInfo()
```

Resultado optenido despues de la ejecución.

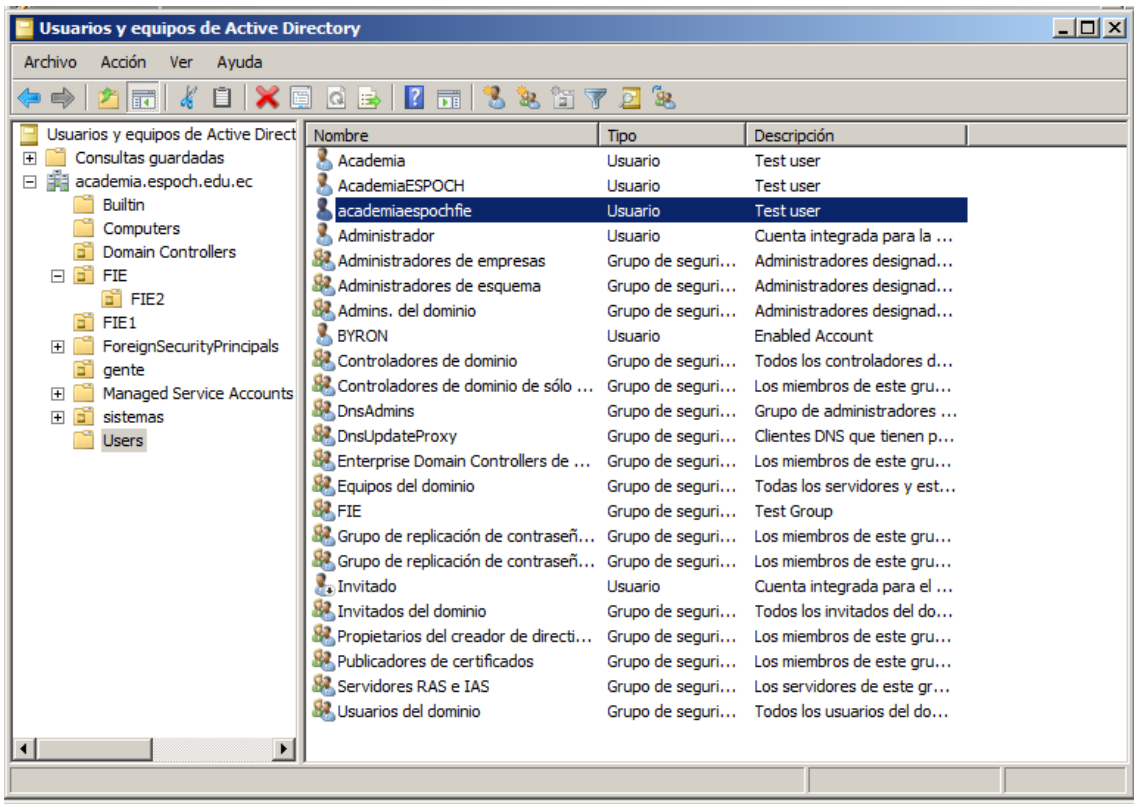


Figura V. 2. Creación de usarios locales.
Fuente: EL Autor

- **Habiltar o deshabiltar cuentas de usuarios locales**

Código

```
param($computer="localhost", $a="e", $user="byron", $password, $help)
function funHelp()
{
$helpText= @ "
DESCRIPCIÓN:
NOMBRE: habilitarydeshabilitarusuario.ps1
Activa o desactiva un usuario local ya sea en un equipo local o remoto.
PARÁMETROS:
-equipo Especifica el nombre del equipo en el que ejecutar la secuencia de comandos
-a (cción) Acción para realizar <e (nable) d (Isable)>
usuario Nombre de usuario para modificar
-help Muestra el archivo de ayuda
SINTAXIS:
habilitarydeshabilitarusuario.ps1
Genera un error. Debe proporcionar un nombre de usuario
habilitarydeshabilitarusuario.ps1 ordenador tesisfie usuario byron
-password Passw0rd
Permite a un usuario local llamado byron en un equipo denominado tesisfie
con la contraseña Passw0rd –a
habilitarydeshabilitarusuario.ps1 -ayuda?
Muestra el tema de ayuda para el script
"@$helpText
exit
}
$EnableUser = 512
$DisableUser = 2
if($help){ "Obtaining help ..." ; funhelp }
if(!$user)
{
```

```
$(Throw 'A value for $user is required.
Try this: EnableDisableUser.ps1 -help ?')
}
$ObjUser = [ADSI]"WinNT://$computer/$user"
switch($a)
{
"e" {
if(!$password)
{
$(Throw 'a value for $password is required.
Try this: habilitarydeshabilitarusuario.ps1 -help ?')
}
$objUser.setpassword($password)
$objUser.description = "Enabled Account"
$objUser.userflags = $EnableUser
$objUser.setinfo()
}
"d" {
$objUser.description = "Disabled Account"
$objUser.userflags = $DisableUser
$objUser.setinfo()
}
DEFAULT
{
"You must supply a value for the action.
Try this: EnableDisableUser.ps1 -help ?"
}
}
```

Resultado optenido despues de la ejecución.

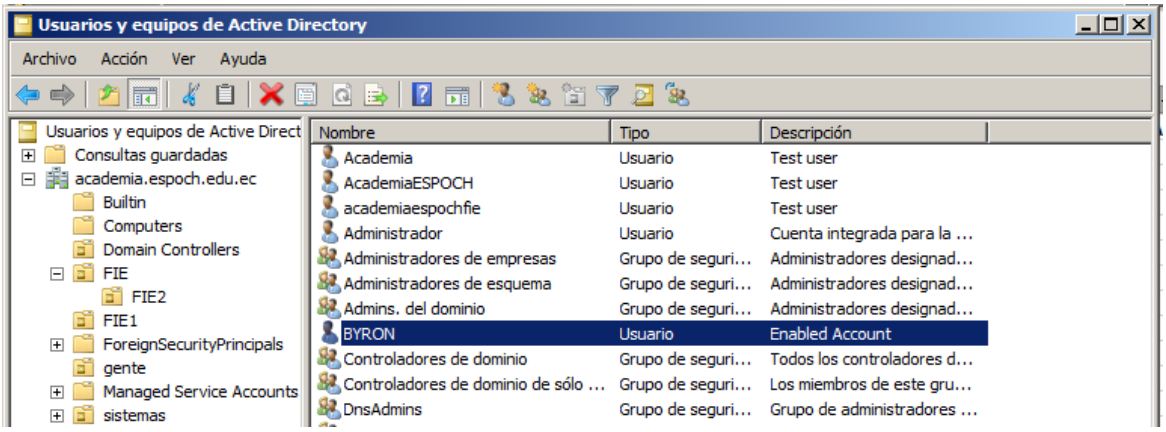


Figura V. 3. Habilitar o deshabilitar cuentas locales.
Fuente: EL Autor

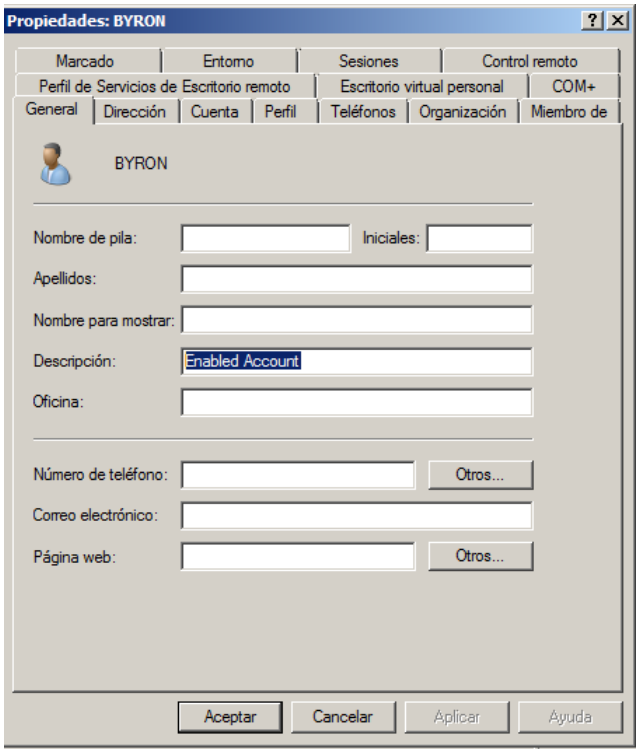


Figura V. 4. Habilitación de cuentas locales.
Fuente: EL Autor

- **Mostrar usuarios y propietarios locales**

La información general sobre los usuarios locales (número de usuarios con licencia, número actual de usuarios y nombre del propietario) está disponible mediante una selección de propiedades de Win32_OperatingSystem. Puede seleccionar explícitamente las propiedades.

Código

```
Get-WmiObject -Class Win32_OperatingSystem -ComputerName . | Select-
Object -Property NumberOfLicensedUsers,NumberOfUsers,RegisteredUser
```

Resultado optenido despues de la ejecución.

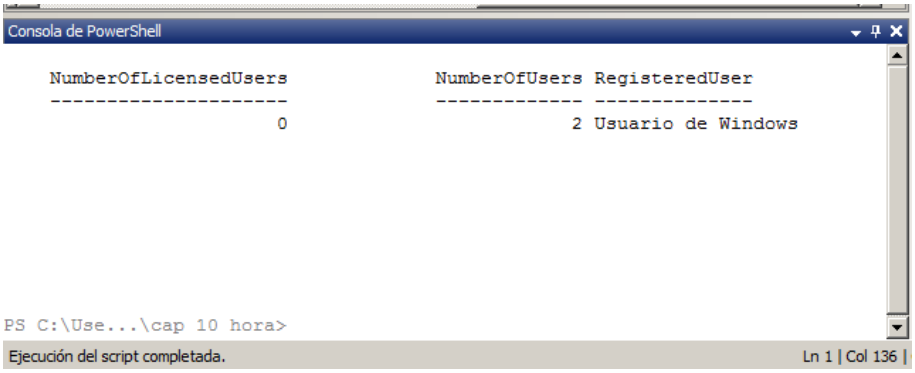


Figura V. 5. Muestra usuario y propiedades locales.
Fuente: EL Autor

b) Creación y administración de grupo locales.

- **Creación de grupo**

Código

```
param($computer="localhost", $group="Facultad", $help)
function funHelp()
{
$helpText=@ "
```

DESCRIPCIÓN:

NOMBRE: creaciondegruposlocales.ps1

Crea un grupo local ya sea en un equipo local o remoto.

PARÁMETROS:

-equipo Especifica el nombre del equipo en el que ejecutar la secuencia de comandos

-grupo Nombre del grupo para crear

-help Muestra el archivo de ayuda

SINTAXIS:

creaciondegruposlocales.ps1

Genera un error. Debe proporcionar un nombre de grupo

creaciondegruposlocales.ps1 ordenador tesisfie grupo facultad

Crea un grupo local llamado tesisfie en un equipo denominado tesisfie

creaciondegruposlocales.ps1 grupo tesisfie

creaciondegruposlocales.ps1 -ayuda?

Muestra el tema de ayuda para el script

"@$helpText

exit

}

if($help){ "Obtaining help ..." ; funhelp }

if(!$group)

{

$(Throw 'A value for $group is required.

Try this: creaciondegruposlocales.ps1 -help ?')

}

$objOu = [ADSI]"WinNT://$computer"

$objUser = $objOU.Create("Group", $group)

$objUser.SetInfo()

$objUser.description = "Test Group"

$objUser.SetInfo()

Resultado optenido despues de la ejecución.

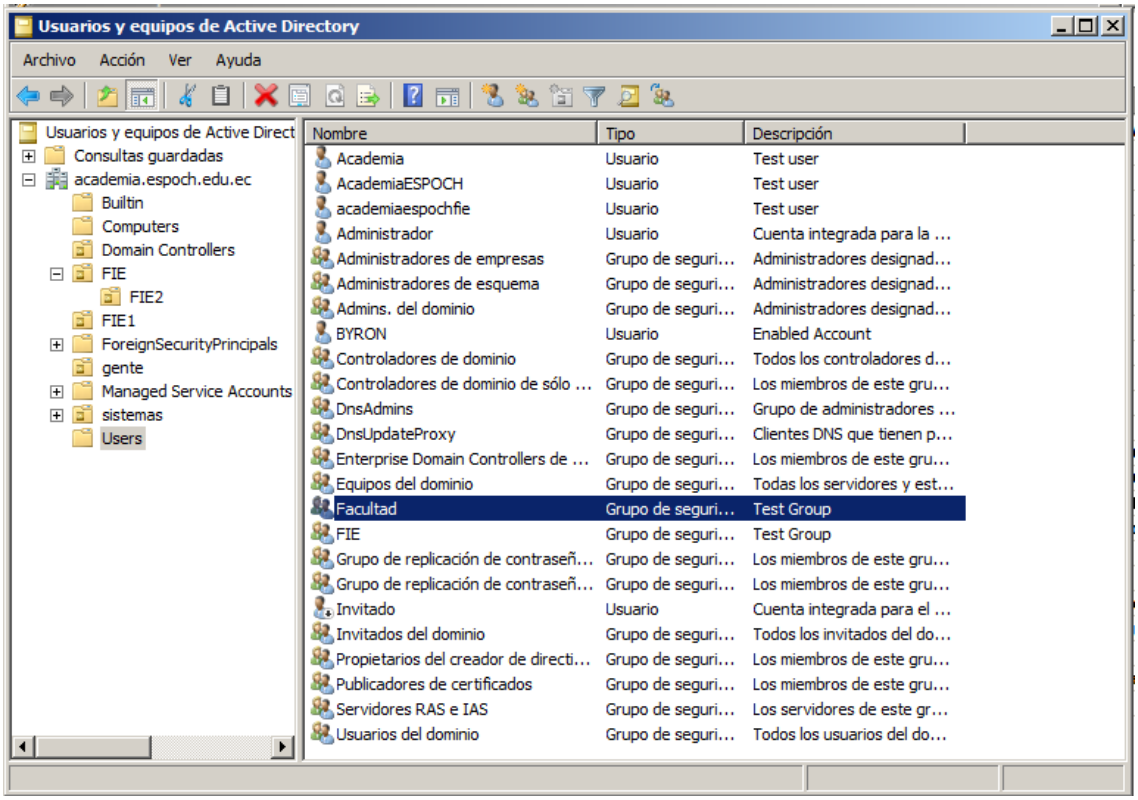


Figura V. 6. Creación de grupos locales.
Fuente: EL Autor

c) Administrar procesos locales

Hay únicamente dos cmdlets Process principales: Get-Process y Stop-Process. Puesto que es posible inspeccionar y filtrar procesos utilizando parámetros o los cmdlets Object, puede realizar algunas tareas complejas con sólo estos dos cmdlets.

- **Mostrar la lista de procesos**

Código

Get-Process

Resultado optenido despues de la ejecución.

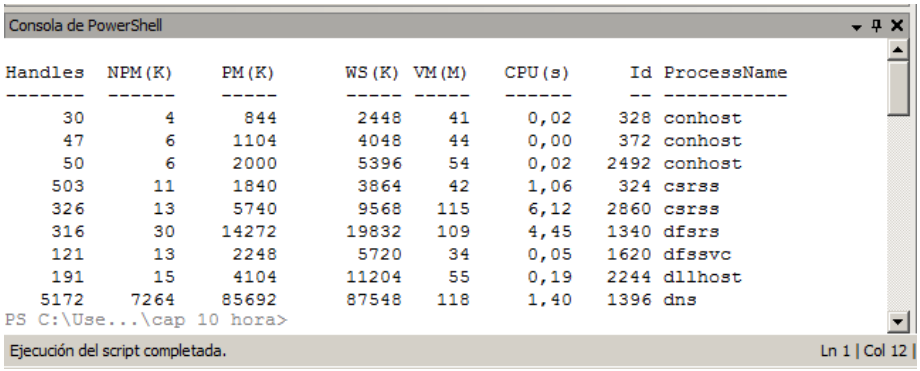


Figura V. 7. Muestra lista de procesos
Fuente: EL Autor

- **Mostrar la lista de procesos con una parámetro específico**

Código

Get-Process -Name dns

Resultado optenido despues de la ejecución.

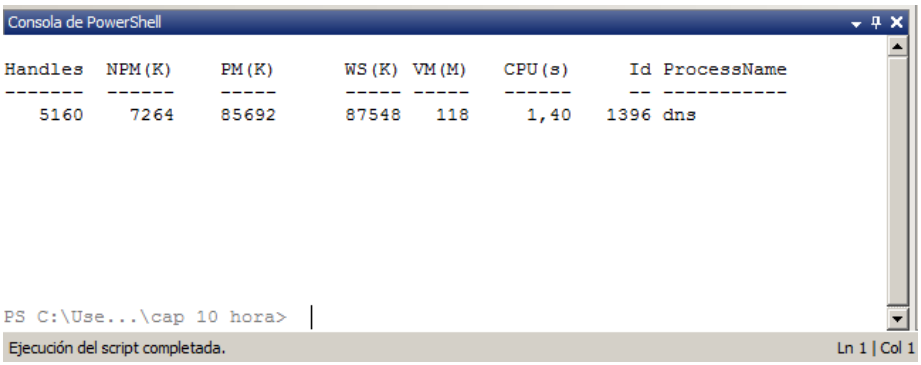


Figura V. 8. Litado de procesos por un parámetro especifico.
Fuente: EL Autor

- **Lista los procesos con un comodín**

Código

*Get-Process -Name ex**

Resultado optenido despues de la ejecución.

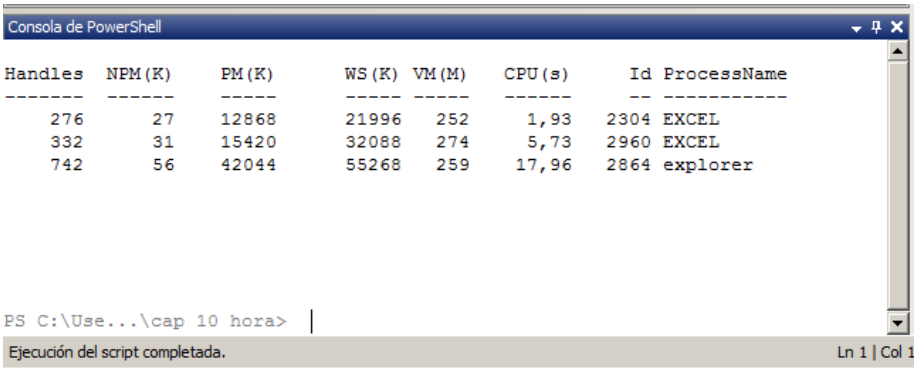


Figura V. 9. Listado de procesos con comodín.
Fuente: EL Autor

- **Detener procesos.**

Código

Stop-Process -Name excel –Confirm

Resultado optenido despues de la ejecución.

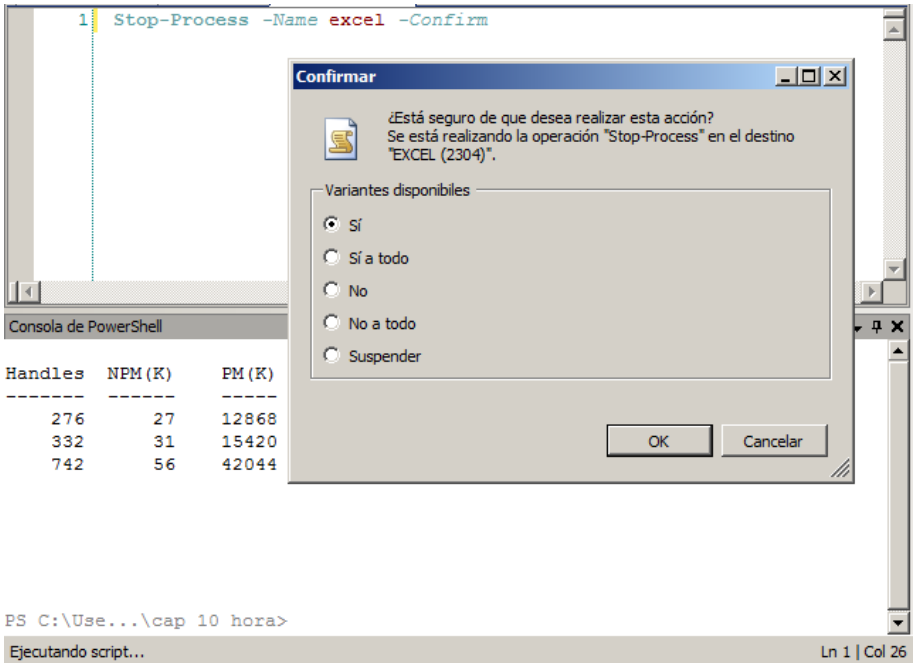


Figura V. 10. Detener Procesos
Fuente: EL Autor

d) Administrar servicios locales.

Hay ocho cmdlets Service principales, diseñados para una amplia variedad de tareas relacionadas con servicios. Se explicara únicamente cómo se enumeran y cambian los estados de ejecución de servicios, pero puede obtener una lista de los cmdlets Service con Get-Help *-Service y también puede obtener información acerca de cada cmdlet Service con Get-Help<nombreCmdlet> (p. ej., Get-Help New-Service).

- **Mostrar la lista de los servicios**

Código

Get-Service

Resultado optenido despues de la ejecución.

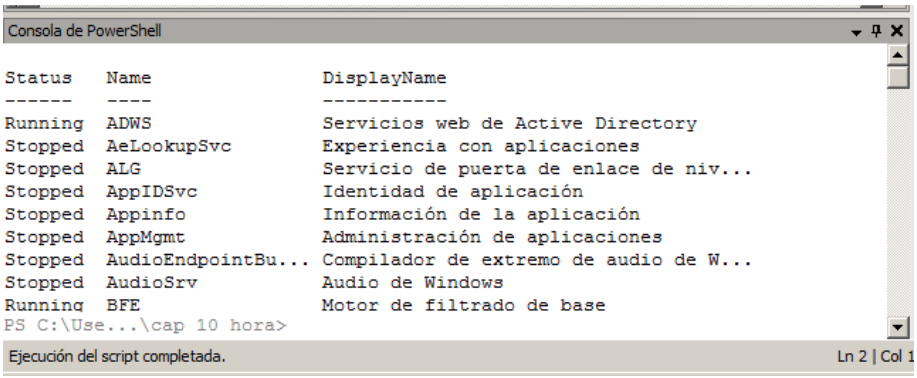


Figura V. 11. Listado de servicios
Fuente: EL Autor

- **Muestra la cantidad de servicios ejecuntados en el servidor**

Código

Write-Host "Cantidad de sevicios en la Servidor"

(Get-Service).length

Resultado optenido despues de la ejecución.

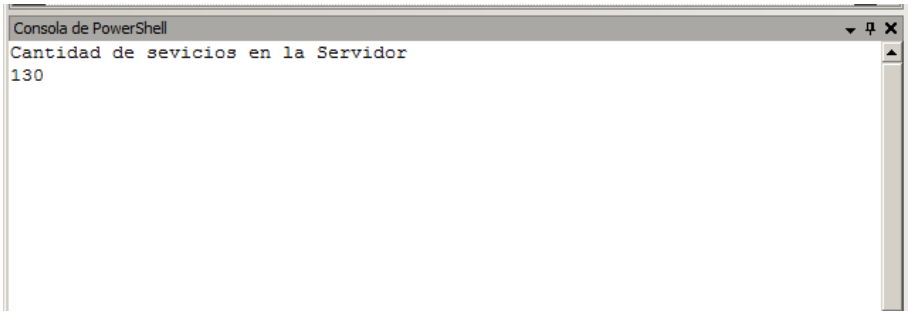


Figura V. 12. Cantidad de servicios ejecutados.
Fuente: EL Autor

- **Cantidad de servicios en ya sea en ejecución, supendido o detenido**

Código

```
Write-Host "Cantidad de sevicios ejecutandose"

(Get-Service | where-object { $_.status -eq "running" }).length
```

Resultado optenido despues de la ejecución.

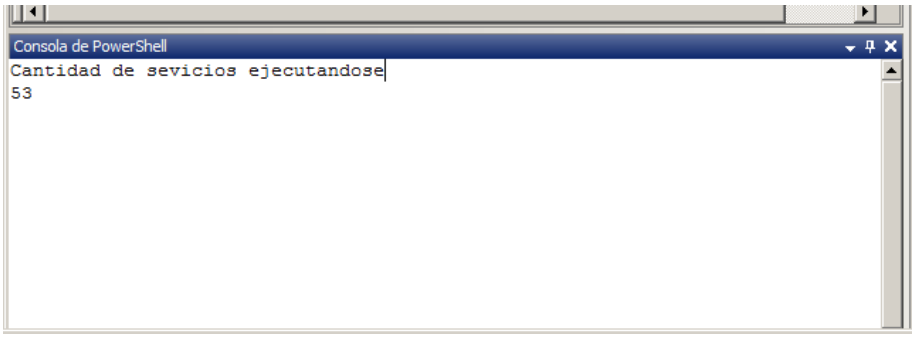


Figura V. 13. Cantidad de servicio con un parámetro especifico.
Fuente: EL Autor

- **Busca la característica de un servicio específico.**

Código

```
$strService = "bits"

Get-Service -Name $strService |

Format-list *
```

Resultado

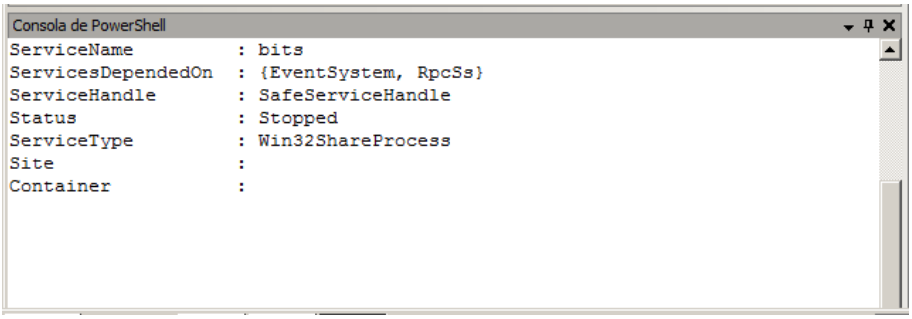


Figura V. 14. Busca característica de un servicio especifico.
Fuente: EL Autor

- **Chequea si un servicio esta levantado o no**

Código

```
$strService = "bits"
Get-Service -name $strService |
Foreach-object { if ($_.status -ne "running")
{
Write-Host "starting $strService ..."
Start-Service -Name $strService
}
ELSE
{
Write-Host "$strService is already started"
}
}
```

Resultado optenido despues de la ejecución.

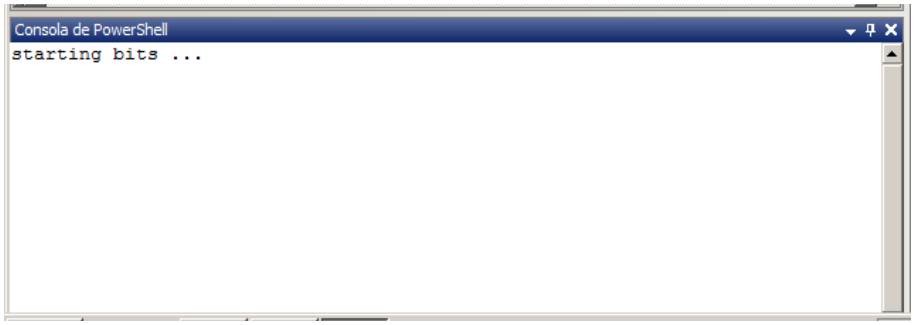


Figura V. 15. Gráfico de una servicio levantado
Fuente: EL Autor

- **Levantamiento de uno o varios servicios.**

Código

```
$aryServices = "bits", "wuauserv"
foreach ($strService in $aryServices)
{
Write-Host "Starting $strService ..."
Start-Service -Name $strService
}
```

Resultado optenido despues de la ejecución.

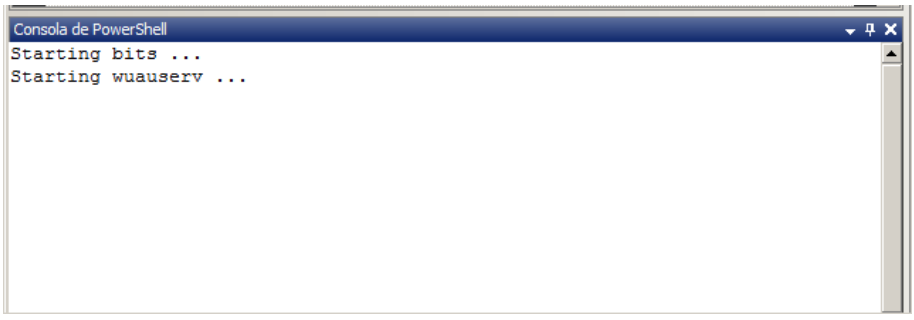


Figura V. 16. Gráfico de varios servicios levantados
Fuente: EL Autor

- **Detencion de uno o varios servicios**

Código

```
$aryServices = "bits", "wuauserv"
foreach ($strService in $aryServices)
```

{

Write-Host "Stopping $strService ..."

Stop-Service -Name $strService

}

Resultado optenido despues de la ejecución.

Figura V. 17. Gráfico de detención de uno o varios servicios
Fuente: EL Autor

- **Importa el estado de los servicios a un archivo plano o txt**

Código

Get-Service > c:\servicios.txt

Resultado optenido despues de la ejecución.

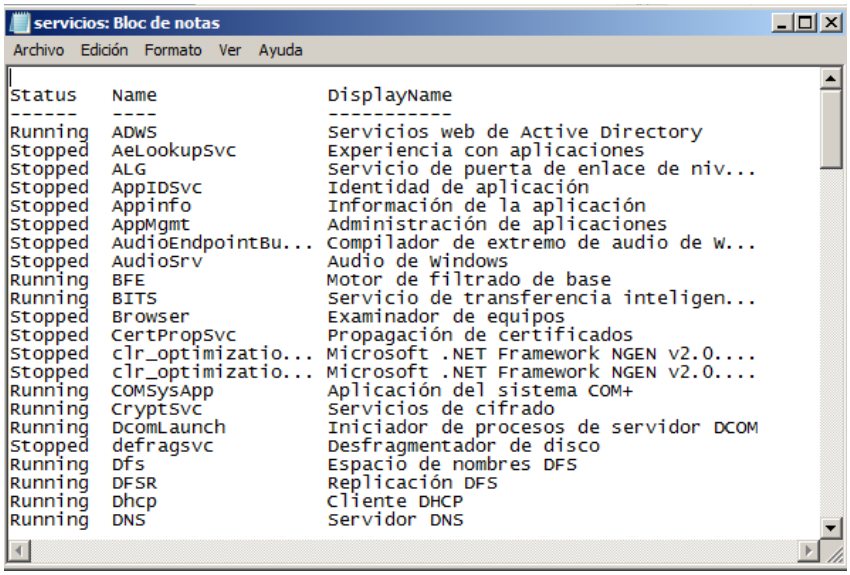


Figura V. 18. Gráfico del estado de los servicios en txt
Fuente: EL Autor

e) Recopilar información acerca de los equipos locales

- **Mostrar la lista de configuraciones del escritorio**

Código

Get-WmiObject -Class Win32_Desktop

Resultado optenido despues de la ejecución.

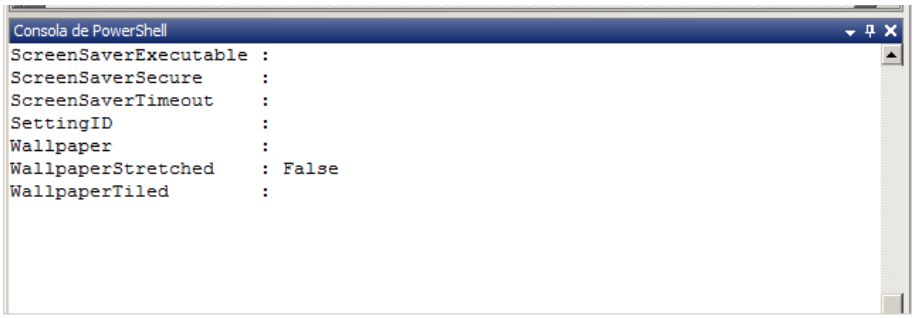


Figura V. 19. Gráfico de la configuración del escritorio
Fuente: EL Autor

- **Muestra la versión del bios**

Código

Get-WmiObject -Class Win32_BIOS

Resultado optenido despues de la ejecución.

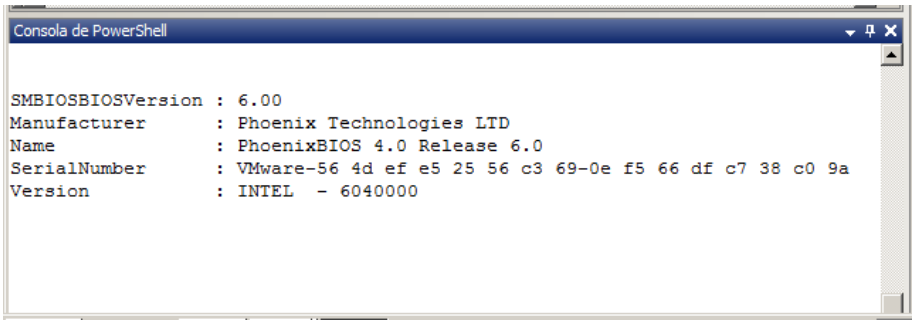


Figura V. 20. Gráfico de la versión del bios
Fuente: EL Autor

- **Muestra si el procesador es basado en tecnología de 32 o 64 bits**

Código

Get-WmiObject -Class Win32_ComputerSystem -ComputerName .| Select-Object -Property SystemType

Resultado optenido despues de la ejecución.

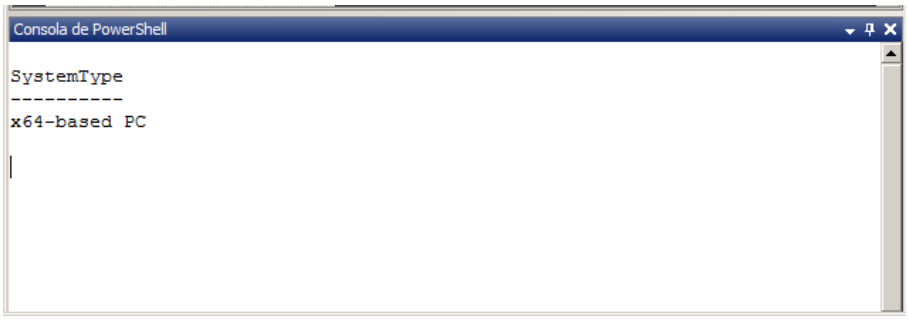


Figura V. 21. Gráfico de la tecnología del procesador
Fuente: EL Autor

- **Muestra el fabricante y modelo del equipo**

Código

Get-WmiObject -Class Win32_ComputerSystem

Resultado optenido despues de la ejecución.

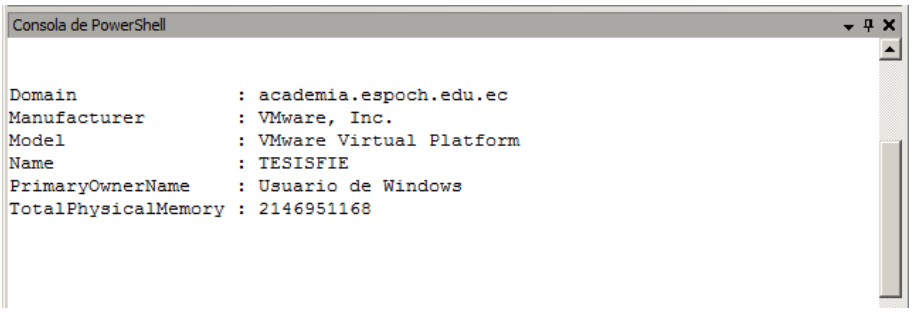


Figura V. 21. Gráfico del fabricante ymodelo del equipo
Fuente: EL Autor

- **Cambiar nombre al computador**

Codigo

```
param(
$computer="WIN-2JULL4B2TE8",
$newName="TESISFIE",
$user = "administrador",
$password="P@ssw0rd",
$help
)
function funHelp()
{
$helpText=@ "
DESCRIPCIÓN:
NOMBRE: renombrarcomputadora.ps1
Cambia el nombre de un equipo local o remoto.
PARÁMETROS:
-equipo Especifica el nombre del equipo en el que ejecutar la secuencia de
comandos
-newname nuevo nombre del equipo
fácil de credenciales de usuario
-password contraseña del usuario
-help Muestra el archivo de ayuda
SINTAXIS:
renombrarcomputadora.ps1 ordenador WIN-2JULL4B2TE8 -newname
TESISFIE
Cambia el nombre de un equipo denominado WIN-2JULL4B2TE8  a TESISFIE
renombrarcomputadora.ps1 ordenador WIN-2JULL4B2TE8 -newname
TESISFIE
fácil de munich \ admin-password P@ss0rd
Genera un error. Debe suministrar nuevo nombre para el ordenador
RenameComputer.ps1-ayuda?
Muestra el tema de ayuda para el script
```

```
"@$helpText
exit
}
if($help) { "Obtaining help ..." ; funhelp }
if($computer -ne "WIN-2JULL4B2TE8")
{
$objWMI = Get-WmiObject -Class Win32_Computersystem `
-computername $computer -credential $user
$objWMI.rename($newName)
}
ELSE
{
$objWMI = Get-WmiObject -Class Win32_Computersystem `
-computername $computer
$objWMI.rename($newName)
}
```

Resultado optenido despues de la ejecución.

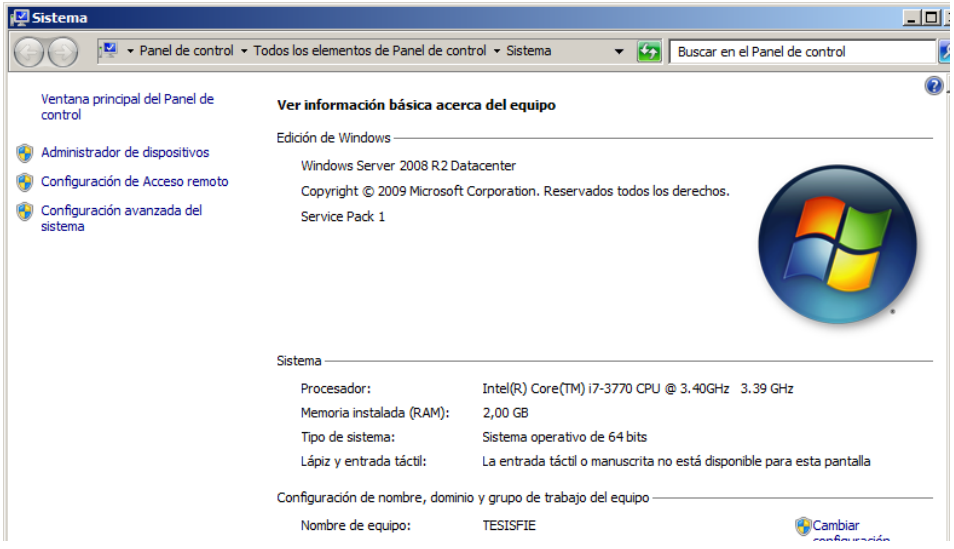


Figura V. 22. Gráfico del cambio de nombre de un computador
Fuente: EL Autor

- **Setear la hora de forma remota**

Código

```
param($computer="localhost", $a, $help)
function funline ($strIN)
{
$num = $strIN.length
for($i=1 ; $i -le $num ; $i++)
{ $funline += "=" }
Write-Host -ForegroundColor yellow `n$strIN
Write-Host -ForegroundColor darkYellow $funline
}
function funHelp()
{
$helpText= @ "
DESCRIPCIÓN:
NOMBRE: setearhoraremotamente.ps1
Muestra o ajusta la hora actual en un equipo local o remoto.
PARÁMETROS:
-computerName Especifica el nombre del equipo en el que ejecutar la secuencia de comandos
-a (acción) determina si establece o se pone la hora actual
-help Muestra el archivo de ayuda
SINTAXIS:
setearhoraremotamente.ps1 -ordenador localhost
Muestra la hora actual en un equipo denominado locahost
setearhoraremotamente.ps1 -a q
Muestra la hora actual en el equipo local
setearhoraremotamente.ps1 –ayuda ?
Muestra el tema de ayuda para el script
"@$helpText
exit
}
```

```
if($help){funline("Obtaining help ...") ; funhelp }
$date = [Management.ManagementDatetimeConverter]::`
ToDmtfDateTime($(get-date))
$objWMI = Get-WmiObject -ComputerName $computer `
-Class win32_operatingsystem
$localUTC=$objwmi.localDateTime
switch($a)
{
"q" {
funline("The time on $($objWMI.csname) is")
[Management.ManagementDatetimeConverter]::`
ToDateTime($localUTC)
}
"s" {
funline("Setting current time on $computer ...")
$objWMI.SetDateTime($date)
}
DEFAULT {
funline("The time on $($objWMI.csname) is")
[Management.ManagementDatetimeConverter]::`
ToDateTime($localUTC)
}
}
```

Resultado optenido despues de la ejecución.

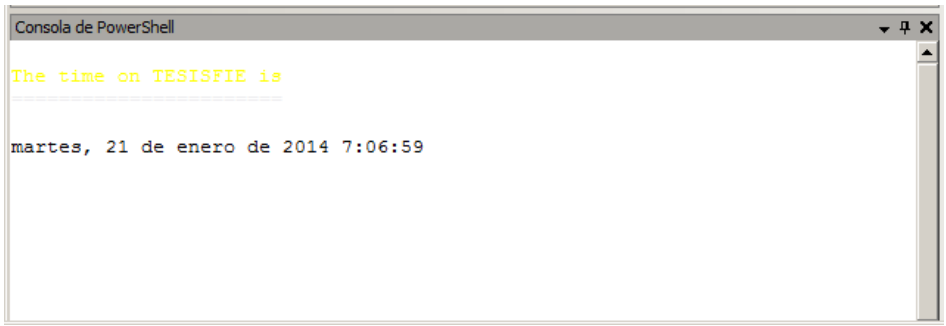


Figura V. 23. Gráfico de setear la fecha y hora remota
Fuente: EL Autor

f) Trabajar con las instalaciones de software.

Se puede tener acceso a aplicaciones diseñadas correctamente para utilizar Windows Installer mediante la clase Win32_Product de WMI, pero actualmente no todas las aplicaciones usan Windows Installer. Dado que Windows Installer proporciona la mayor variedad de técnicas estándar para trabajar con aplicaciones que se pueden instalar. Las aplicaciones que utilizan otras rutinas de instalación no se suelen administrar con Windows Installer. Las técnicas concretas para trabajar con estas aplicaciones dependen del software de instalación y de las decisiones que adopte el programador de la aplicación.

- **Muestra las aplicaciones instaladas con Windows instaler**

Código

Get-WmiObject -Class Win32_Product | ft > c:\listado.xlsx

Resultado optenido despues de la ejecución.

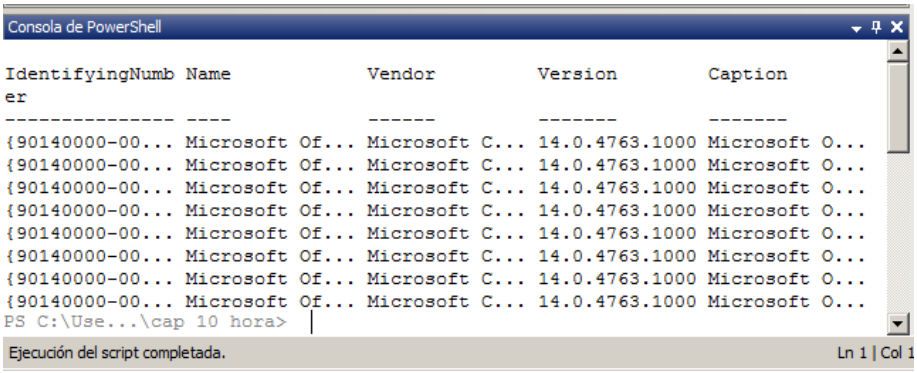


Figura V. 24. Gráfico de las aplicaciones intaladas con Windows instaler
Fuente: EL Autor

- **Muestra el listado detalla de los programas instalados**

Código

Get-WmiObject -Class Win32_Product -ComputerName . | Format-List Name,InstallDate,InstallLocation,PackageCache,Vendor,Version,IdentifyingNumber

Resultado optenido despues de la ejecución.

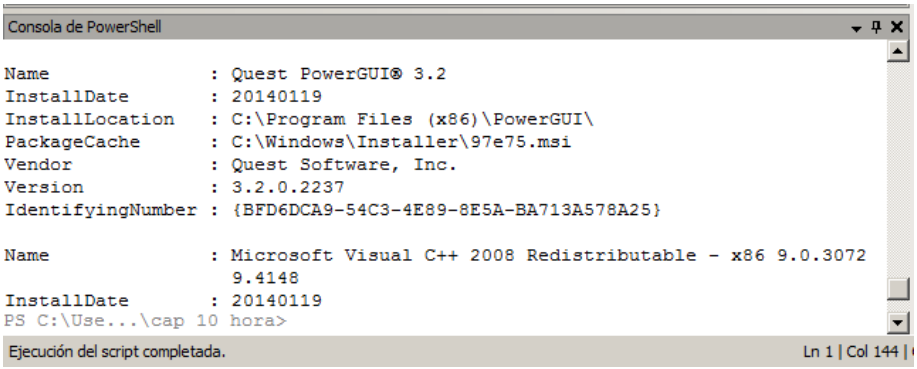


Figura V. 25. Gráfico de del listado detallado de un programa
Fuente: EL Autor

g) Trabajar con el estado de un equipo.

Puede restablecer un equipo de varias maneras distintas desde Windows PowerShell pero en la primera versión debe usar una herramienta estándar de línea de comandos o WMI. Aunque está utilizando Windows PowerShell sólo para invocar una determinada herramienta, recorrer el proceso para cambiar el estado de energía de un equipo ilustra algunas partes importantes del uso de herramientas externas.

- **Bloquear un equipo**

Código

rundll32.exe user32.dll,LockWorkStation

Resultado optenido despues de la ejecución.

Figura V. 26. Gráfico de bloqueo de equipo
Fuente: EL Autor

- **Cerrar sesion**

Código

(Get-WmiObject -Class Win32_OperatingSystem -ComputerName .).InvokeMethod("Win32Shutdown",0)

Resultado optenido despues de la ejecución.

Figura V. 27. Gráfico de cierre de sesión
Fuente: EL Autor

- **Apagar o reinicia una computadora remota**

Código

```
param(
$computer="localhost",
$user = "administrador",
$password = "P@ssw0rd",
$a="r",
$help
)
function funHelp()
{
$helpText= @ "apaga o reinicia un equipo remoto
"@$helpText
exit
}
if($help){ "Obtaining help ..." ; funhelp }
switch($a)
{
"s" {
if($computer -ne "localhost")
{
$objWMI = Get-WmiObject -Class Win32_operatingsystem `
-computername $computer -credential $user
$objWMI.psbase.Scope.Options.EnablePrivileges = $true
$objWMI.shutdown()
}
ELSE
{
$objWMI = Get-WmiObject -Class Win32_operatingsystem `
-computername $computer
$objWMI.psbase.Scope.Options.EnablePrivileges = $true
$objWMI.shutdown()
```

```
}
}
"r" {
if($computer -ne "localhost")
{
$objWMI = Get-WmiObject -Class Win32_operatingsystem `
-computername $computer -credential $user
$objWMI.psbase.Scope.Options.EnablePrivileges = $true
$objWMI.reboot()
}
ELSE
{
$objWMI = Get-WmiObject -Class Win32_operatingsystem `
-computername $computer
$objWMI.psbase.Scope.Options.EnablePrivileges = $true
$objWMI.reboot()
}
}
DEFAULT { "You must supply an action. Try this"
"apagaoreyniciaunequiporemotamente.ps1 -help ?" }
}
```

h) Trabajar con impresoras.

Las tareas de administración de impresoras pueden realizarse en Windows PowerShell con WMI y con el objeto COM WScript.Network de WSH. Se debe utilizar una combinación de ambas herramientas para demostrar la forma de realizar tareas específicas.

- **Crea una lista de impresoras instaldas**

Código

Get-WmiObject -Class Win32_Printer -ComputerName .

Resultado optenido despues de la ejecución.

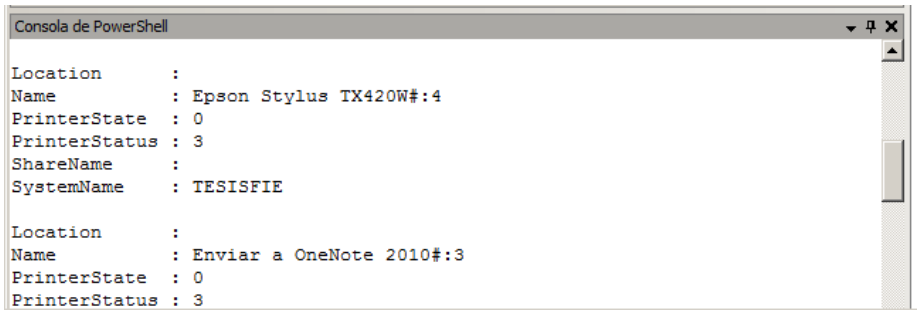


Figura V. 28. Gráfico de lista de impresoras instaladas
Fuente: EL Autor

- **Configurar la impresora como predeterminada**

Código

(Get-WmiObject -ComputerName . -Class Win32_Printer -Filter "Name='Epson Stylus TX420W#:4'").InvokeMethod("SetDefaultPrinter",$null)

Resultado optenido despues de la ejecución.

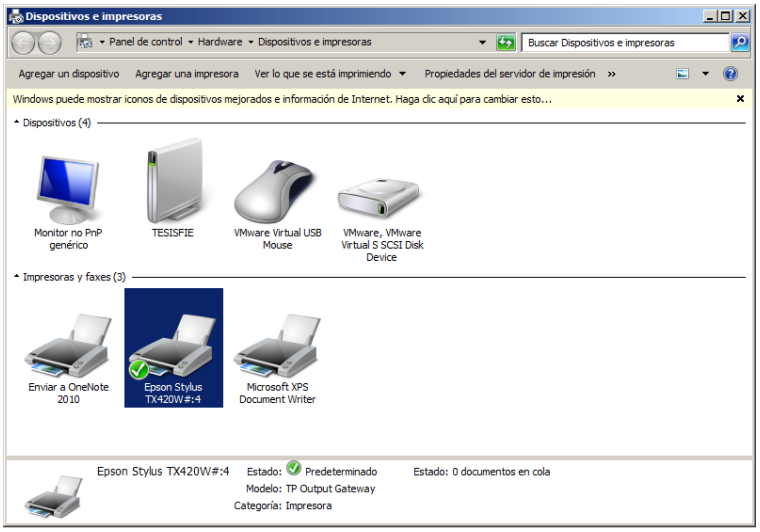


Figura V. 29. Gráfico de una impresora predeterminada
Fuente: EL Autor

i) Realizar tarea de red.

En la mayoría de las tareas básicas de administración de protocolos de red implican el uso de TCP/IP, ya que éste es el protocolo de red más utilizado. Se vera ahora cómo efectuar una selección de estas tareas desde Windows PowerShell con WMI.

- **Crea ua lista de ip utilizada en el equipo.**

Codigo

Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter IPEnabled=TRUE -ComputerName .| Select-Object -Property IPAddress

Resultado optenido despues de la ejecución.

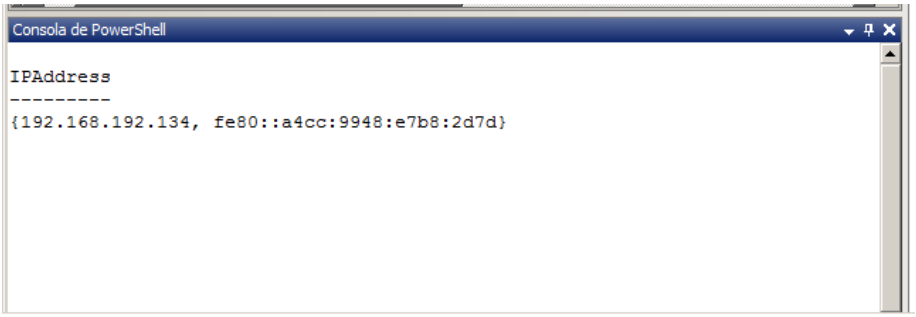


Figura V. 30. Gráfico de la lista de ip utilizada en un equipo
Fuente: EL Autor

- **Muestra los datos de configuración de la ip.**

Código

Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter IPEnabled=TRUE -ComputerName .

Resultado optenido despues de la ejecución.

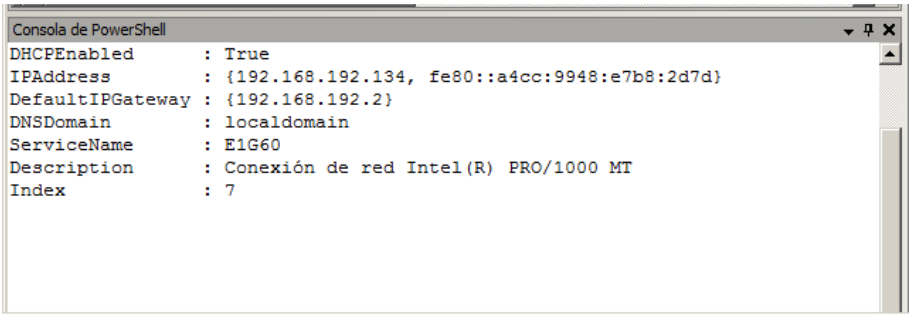


Figura V. 31. Gráfico de la configuración Ip
Fuente: EL Autor

- **Verificar el estado de una red**

Codigo

```
param($computer="localhost",$help)
function funStatus($status)
{
switch($status)
{
0 { " Disconnected" }
1 { " Connecting" }
2 { " Connected" }
3 { " Disconnecting" }
4 { " Hardware not present" }
5 { " Hardware disabled" }
6 { " Hardware malfunction" }
7 { " Media disconnected" }
8 { " Authenticating" }
9 { " Authentication succeeded" }
10 { " Authentication failed" }
}
}
function funHelp()
```

```
{
$helpText= @ "
DESCRIPCIÓN:
NOMBRE: verificaelestadodeunared.ps1
Produce una lista de adaptadores de red y el estado en un equipo local o
remoto.
PARÁMETROS:
-computerName Especifica el nombre del equipo en el que ejecutar la secuencia
de comandos
-help Muestra el archivo de ayuda
SINTAXIS:
verificaelestadodeunared.ps1 -ordenador
Lista todos los adaptadores de red y el estado de un equipo denominado tesisfie
verificaelestadodeunared.ps1
Lista todos los adaptadores de red y el estado en el equipo local,
verificaelestadodeunared.ps1 -ayuda?
Muestra el tema de ayuda para el script
"@$helpText
exit
}
function funline ($strIN)
{
$num = $strIN.length
for($i=1 ; $i -le $num ; $i++)
{ $funline = $funline + "=" }
Write-Host -ForegroundColor blue $strIN
Write-Host -ForegroundColor darkYellow $funline
}
if($help) { "Printing help now..." ; funHelp }
$objWMI=Get-WmiObject -Class win32_networkadapter -computer $computer
funline ("Network adapters and status on $computer")
```

Resultado optenido despues de la ejecución.

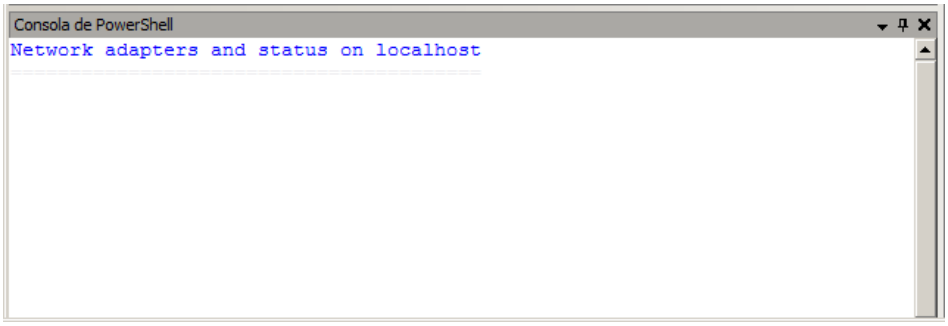


Figura V. 32. Gráfico muestra el estado de una red
Fuente: EL Autor

- **Verificar adptador conectado.**

Código

```
$computer="localhost"
$connected=2
Get-WmiObject -Class win32_networkadapter -computername $computer `
-filter "netconnectionstatus = $connected" |
foreach-object `
{
Get-WmiObject -Class win32_networkadapterconfiguration `
-computername $computer -filter "Index = $($_.deviceID)"
}
```

Resultado optenido despues de la ejecución.

Figura V. 33. Gráfico de la verificación de un adaptador conectado
Fuente: EL Autor

- **Habilitando DHCP**

Codigo

```
param($computer="localhost",$action,$help)
function funHelp()
{
$helpText=@ "
DESCRIPCIÓN:
NOMBRE: trabajandoconDHCP.ps1
Funciona con la configuración de DHCP en un equipo local o remoto.
PARÁMETROS:
-computerName Especifica el nombre del equipo en el que ejecutar la secuencia de comandos
-action <q (query) e (enable) r (elease) rr (release / renew) acción a realizar
-help Muestra el archivo de ayuda
SINTAXIS:
trabajandoconDHCP.ps1 -q "sí" en computadora tesisfie
Ajustes consultas DHCP en un equipo denominado tesisfie
trabajandoconDHCP.ps1 de acción e
Habilita DHCP en el equipo local
trabajandoconDHCP.ps1 acción r
Libera la dirección DHCP en la máquina local
trabajandoconDHCP.ps1 acción rr
Estrenos y luego renueva la dirección DHCP en la máquina local
trabajandoconDHCP.ps1 -ayuda?
Muestra el tema de ayuda para el script
"@$helpText
exit
}
function FunEvalRTN($rtn)
{
Switch ($rtn.returnvalue)
{
```

```
0 { Write-Host -foregroundcolor green "No errors for $strCall" }
82 { Write-Host -foregroundcolor red "$strCall reports" `
" Unable to renew DHCP lease" }
83 { Write-Host -ForegroundColor red "$strCall reports" `
" Unable to release DHCP lease" }
91 { Write-Host -ForegroundColor red "$strCall reports" ` " access denied"}
DEFAULT { Write-Host -ForegroundColor red "$strCall service reports" `
" ERROR $($rtn.returnValue)" }
}
$rtn=$strCall=$null
}
if($help) { funhelp }
$global:RTN = $null
if(!$action) { $action="q" }
$objWMI = Get-WmiObject -Class win32_networkadapterconfiguration `
-computer $computer -filter "ipenabled = 'true'"
Switch($action)
{
"e" {
$rtn = $objWMI.EnableDHCP() ;
$strCall = "Enable DHCP" ;
FunEvalRTN($rtn)
}
"r" {
$rtn = $objWMI.ReleaseDHCPLease() ;
$strCall = "Release DHCP address" ;
FunEvalRTN($rtn)
}
"rr" {
$rtn = $objWMI.RenewDHCPLease() ;
$strCall = "Release and Renew DHCP address" ;
FunEvalRTN($rtn)
```

```
}
"q" {
"DHCP Server: $($objWMI.dhcpserver)"
"Lease obtained: " + [Management.ManagementDatetimeConverter]::`
todatetime($objWMI.DHCPleaseObtained)
"Lease expires: " + [Management.ManagementDatetimeConverter]::`
todatetime($objWMI.DHCPleaseExpires)
}
}
```

Resultado optenido despues de la ejecución.

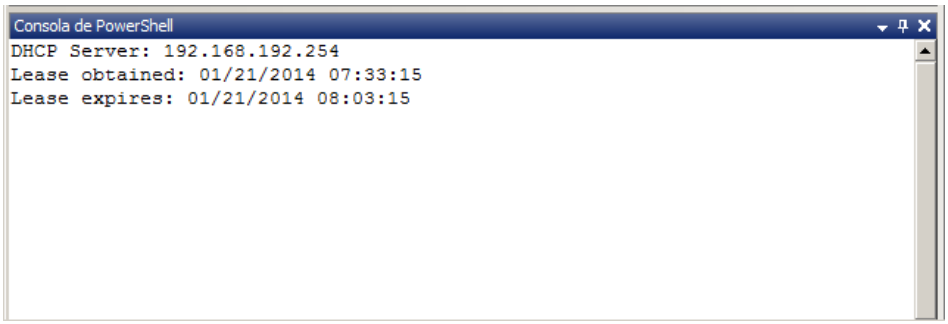


Figura V. 34. Gráfico del levantamiento de dhcp
Fuente: EL Autor

- **Verifica el si esta habilitado DHCP**

Codigo

```
Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter
"DHCPEnabled=true" -ComputerName .| Select-Object -Property DHCP*
```

Resultado optenido despues de la ejecución.

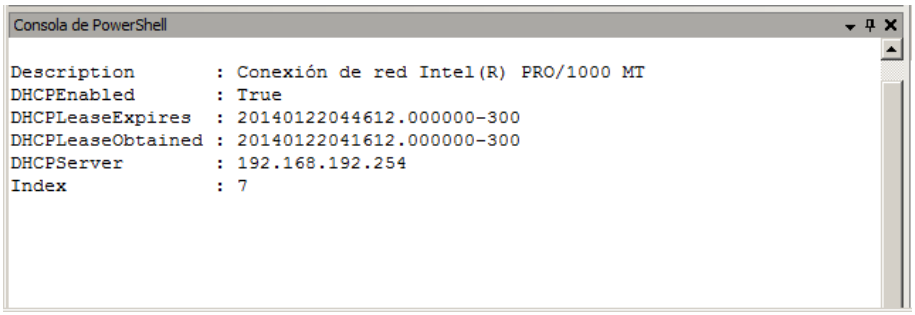


Figura V. 35. Gráfico del dhcp habilitado
Fuente: EL Autor

- **Libera y renova conceciones DHCP en todos los adaptadores**

Codigo

Get-WmiObject -List | Where-Object -FilterScript {$_.Name -eq "Win32_NetworkAdapterConfiguration"}

Resultado optenido despues de la ejecución.

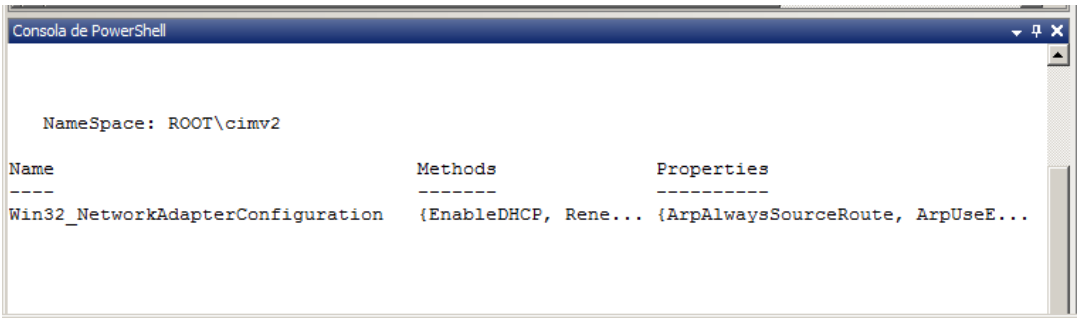


Figura V. 36. Gráfico de liberación concesiones de dhcp
Fuente: EL Autor

- **Verificar la hora en la red**

Codigo

param($computer="localhost",$a,$timeServer,$help)
function funHelp()
{
$helpText= @ "
DESCRIPCIÓN:
NOMBRE: verificalahoradelared.ps1
Graba y establece el origen de la hora actual en un equipo local o remoto.
PARÁMETROS:
-equipo Especifica el nombre del equipo en el que ejecutar la secuencia de comandos
-a (cción) La acción específica a realizar <qt, qs, s>
-timeserver El nombre del servidor de tiempo para usar
-help Muestra el archivo de ayuda
SINTAXIS:
verificalahoradelared.ps1 -ordenador tesisfie

Muestra la hora actual en un equipo denominado tesisfie

verificalahoradelared.ps1

Muestra la hora actual en el equipo local

verificalahoradelared.ps1 -informáticos tesisfie-a QS

Listas de servidor de tiempo actual en un equipo denominado MunichServer

verificalahoradelared.ps1 ordenador tesisfie-as-timeserver 192.168.2.5

Configura el servidor de tiempo actual en un equipo denominado tesisfie

a 192.168.2.5

verificalahoradelared.ps1 -ayuda?

Muestra el tema de ayuda para el script

"@$helpText

exit

}

if($help){("Obtaining help ...") ; funhelp }

switch($a)

{

"qt" { net time \\$computer }

"qs" { net time \\$computer /querySNTP}

"s" { net time \\$computer /setSNTP:$timeServer }

DEFAULT { net time \\$computer }

}

Resultado optenido despues de la ejecución.

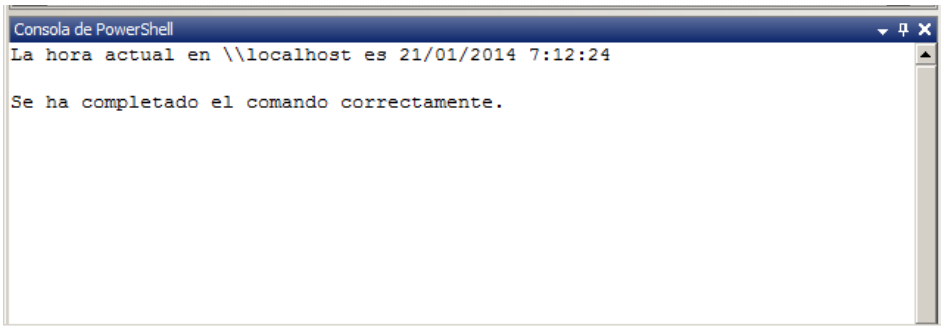


Figura V. 37. Gráfico de la hora actual en la red

Fuente: EL Autor

- **Crear un recurso compartido**

Código

(Get-WmiObject -List -ComputerName . | Where-Object -FilterScript {$_.Name -eq "Win32_Share"}).InvokeMethod("Create",("C:\prueba2","TempShare",0,25,"test share of the temp folder"))

Resultado optenido despues de la ejecución.

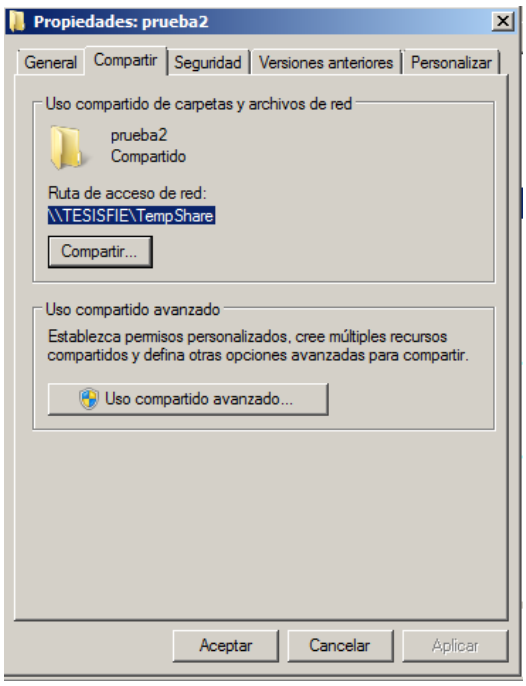


Figura V. 38. Gráfico de un recurso compartido
Fuente: EL Autor

- **Elimina un recurso compartido**

Código

(Get-WmiObject -Class Win32_Share -ComputerName . -Filter "Name='TempShare'").InvokeMethod("Delete",$null)

Resultado optenido despues de la ejecución.

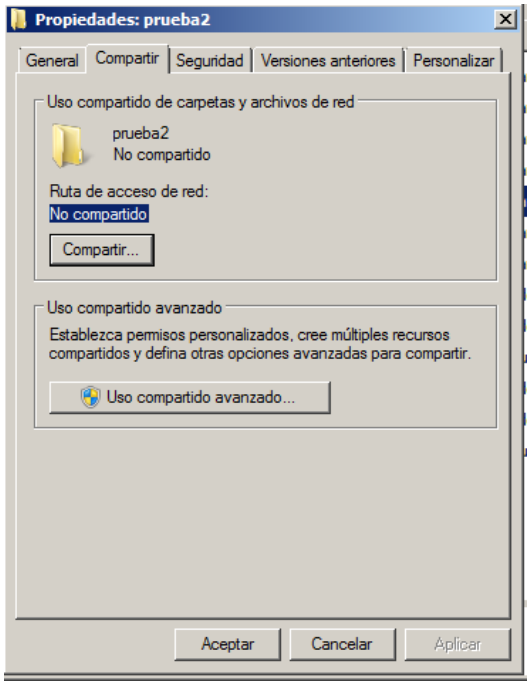


Figura V. 39. Gráfico de la elimininación de un recurso compartido
Fuente: EL Autor

- **Crea una unidad de red compartida**

Código

(New-Object -ComObject WScript.Network).MapNetworkDrive("H:", "\\TESISFIE\tempshare")

Resultado optenido despues de la ejecución.

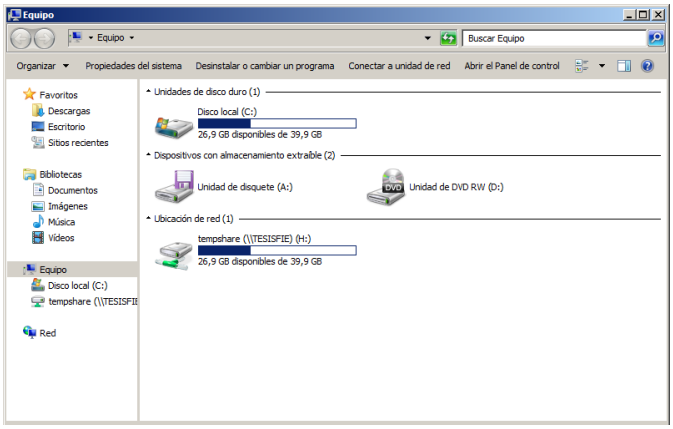


Figura V. 40. Gráfico de una unidad de red compartida
Fuente: EL Autor

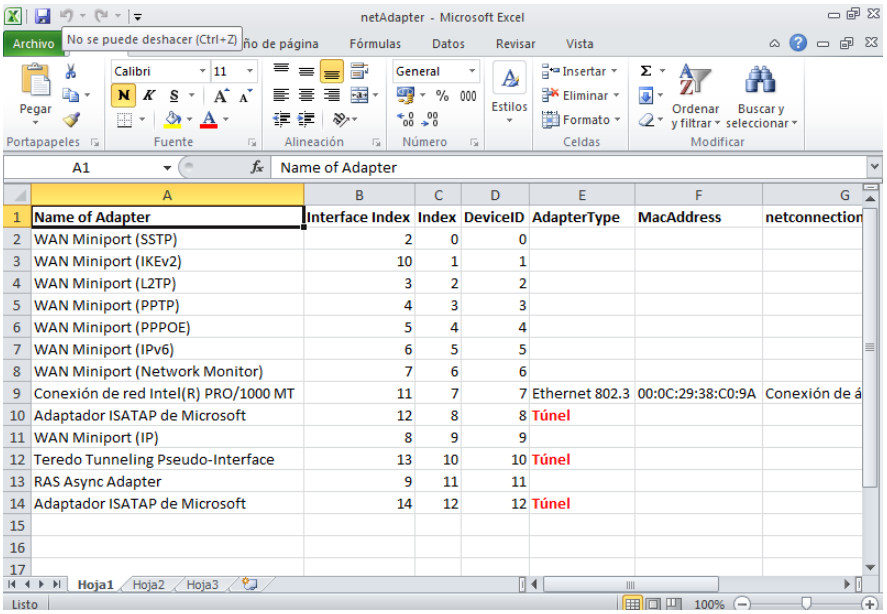
- **Informacion de red en archivo Excel**

Código

```
$strPath="c:\netAdapter.xls"
$objExcel=New-Object -ComObject Excel.Application
$objExcel.Visible=-1
$workbook=$objExcel.workbooks.Add()
$sheet=$workbook.worksheets.item(1)
$x=2
$Computer = $env:computerName
$objWMIService = Get-WmiObject -class win32_NetworkAdapter `
-computer $Computer
for($b=1 ; $b -le 10 ; $b++)
{$sheet.Cells.item(1,$b).font.bold=$true}
$sheet.Cells.item(1,1)=("Name of Adapter")
$sheet.Cells.item(1,2)=("Interface Index")
$sheet.Cells.item(1,3)=("Index")
$sheet.Cells.item(1,4)=("DeviceID")
$sheet.Cells.item(1,5)=("AdapterType")
$sheet.Cells.item(1,6)=("MacAddress")
$sheet.Cells.item(1,7)=("netconnectionid")
$sheet.Cells.item(1,8)=("NetConnectionStatus")
$sheet.Cells.item(1,9)=("NetworkAddresses")
$sheet.Cells.item(1,10)=("PermanentAddress")
ForEach ($objNet in $objWMIService)
{
$sheet.Cells.item($x, 1)=($objNet.Name)
$sheet.Cells.item($x, 2)=($objNet.InterfaceIndex)
$sheet.Cells.item($x, 3)=($objNet.index)
$sheet.Cells.item($x, 4)=($objNet.DeviceID)
$sheet.Cells.item($x, 5)=($objNet.adapterType)
$sheet.Cells.item($x, 6)=($objNet.MacAddress)
$sheet.Cells.item($x,7)=($objNet.netconnectionid)
```

```
$sheet.Cells.item($x,8)=($objNet.NetConnectionStatus)
$sheet.Cells.item($x,9)=($objNet.NetworkAddresses)
$sheet.Cells.item($x,10)=($objNet.PermanentAddress)
If($objNet.AdapterType -notMatch 'ethernet')
{
$sheet.Cells.item($x,5).font.colorIndex=3
$sheet.Cells.item($x,5).font.bold=$true
}
$x++
}
$range = $sheet.usedRange
$range.EntireColumn.AutoFit()
IF(Test-Path $strPath)
{
Remove-Item $strPath
$objExcel.ActiveWorkbook.SaveAs($strPath)
}
ELSE
{
$objExcel.ActiveWorkbook.SaveAs($strPath)
}
```

Resultado optenido despues de la ejecución.



netAdapter - Microsoft Excel

	Name of Adapter	Interface Index	Index	DeviceID	AdapterType	MacAddress	netconnection
2	WAN Miniport (SSTP)	2	0	0			
3	WAN Miniport (IKEv2)	10	1	1			
4	WAN Miniport (L2TP)	3	2	2			
5	WAN Miniport (PPTP)	4	3	3			
6	WAN Miniport (PPPOE)	5	4	4			
7	WAN Miniport (IPv6)	6	5	5			
8	WAN Miniport (Network Monitor)	7	6	6			
9	Conexión de red Intel(R) PRO/1000 MT	11	7	7	Ethernet 802.3	00:0C:29:38:C0:9A	Conexión de á
10	Adaptador ISATAP de Microsoft	12	8	8	Túnel		
11	WAN Miniport (IP)	8	9	9			
12	Teredo Tunneling Pseudo-Interface	13	10	10	Túnel		
13	RAS Async Adapter	9	11	11			
14	Adaptador ISATAP de Microsoft	14	12	12	Túnel		

Figura V. 41. Gráfico de los dispositivos de la red en excel
Fuente: EL Autor

j) Trabajar respaldos del sistema y verificaciones de disco duro.

El repaldos es algo sumamente importante dentro de la administacion es por eso que se planta un ayuda específica para los administradores con herramientas en la cuales se pueda respaldar y ver el tamaño de almacenamiento disponible.

- **Respaldar datos a una unidad deteminada**

Código

```
param($source= "c:\prueba", $destination="c:\prueba2", $help)
function funHelp()
{
$helpText @ "
DESCRIPCIÓN:
NOMBRE: respaldaunacarpetadelservidor.ps1
```

Realiza copias de seguridad de archivos en una carpeta en una unidad asignada. El destino
carpeta no tiene que estar presente
PARÁMETROS:
de código fuente de los archivos y carpetas
destino donde los archivos se copiarán
-help Muestra el archivo de ayuda
SINTAXIS:
respaldaunacarpetadelservidor.ps1 -source c: \ fso-destino h: \ fso
Realiza copias de seguridad de todos los archivos y carpetas en c: \ fso en el equipo local para una unidad asignada llama h. La carpeta FSO \ no necesita existir en el h: \ unidad.
respaldaunacarpetadelservidor.ps1
genera un error. la fuente y el destino de los parámetros debe estar presente
respaldaunacarpetadelservidor.ps1 -ayuda?
Muestra el tema de ayuda para el script
"@$helpText
exit
}
if($help){ "Obtaining help ..." ; funhelp }
if(!$source -or !$destination)
{
$(throw "You must supply both source and destination.
Try this BackupFolderToServer.ps1 -help -?")
}
Copy-Item -Path $source -destination $destination -recurse

Resultado optenido despues de la ejecución.

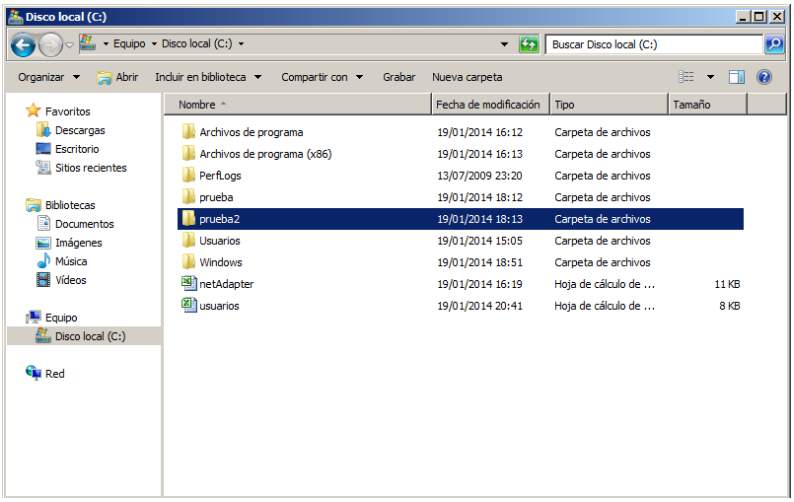


Figura V. 42. Gráfico de respaldo de una carpeta del servidor
Fuente: EL Autor

- **Configuración del sistema de restauración**

Código

Param($computer = "localhost", $help)

function funHelp()

{

$helpText= @ "

DESCRIPCIÓN:

NOMBRE: configuraciondelsistemaderestauracion.ps1

Imprime los archivos sin conexión config en un equipo local o remoto.

PARÁMETROS:

-equipo Especifica el nombre del equipo en el que ejecutar la secuencia de comandos

-help Muestra el archivo de ayuda

SINTAXIS:

configuraciondelsistemaderestauracion.ps1 -ordenador tesisfie

Sistema de listas restaurar config en un equipo denominado tesisfie

configuraciondelsistemaderestauracion.ps1

```
Sistema de listas restaurar config en el equipo local
configuraciondelsistemaderestauracion.ps1 -ayuda?
Muestra el tema de ayuda para el script
"@$helpText
exit
}
if($help){ funline("Obtaining help ...") ; funhelp }
New-Variable -Name SecInDay -option constant -value 86400
$objWMI = Get-WmiObject -Namespace root\default `
-Class SystemRestoreConfig -computername $computer
for($i=0; $i -le 15; $i++)
{
Write-Host -ForegroundColor $i "Retrieving System Restore Settings"
Start-Sleep -Milliseconds 60
cls
}
if($computer -eq "localhost")
{
Write-Host "System Restore Settings on $env:computername"
}
ELSE
{
Write-Host "System Restore Settings on $computer"
}
format-table -InputObject $objWMI -property `
@{
Label="Max disk utilization" ;
expression={ "{0:n0}"-f ($_.DiskPercent ) + " %"}
},
@{
Label="Scheduled Backup" ;
expression={ "{0:n2}"-f ($_.RPGlobalInterval / $SecInDay) + " days"}
```

```
},
@{
Label="Max age of backups" ;
expression={ "{0:n2}"-f ($_.RPLifeInterval / $SecInDay) + " days" }
}
```

Resultado optenido despues de la ejecución.

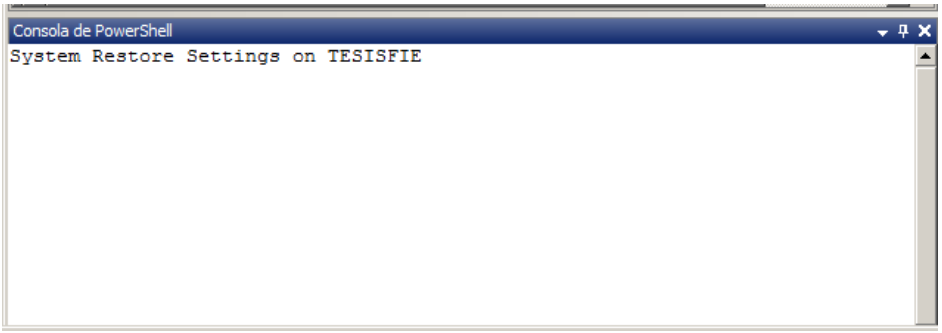


Figura V.43. Gráfico sistema de restauración creado
Fuente: EL Autor

- **Monitor de rendimiento de disco**

Código

```
$numRep = 3
$sleep = 2
$n1=$d1=$n2=$d2=$r1=$r2=$w1=$w2=$null
for ($i=1 ; $i -le $numRep ; $i++)
{
$wmiPerf=Get-WmiObject -class win32_perfrawdata_perfdisk_logicaldisk `
-Filter "name = '_Total'"
[double]$n1 = $wmiperf.percentIdleTime
[double]$r1 = $wmiperf.percentDiskTime
[double]$d1 = $wmiperf.TimeStamp_Sys100NS
Start-Sleep -Seconds $sleep
$wmiPerf=Get-WmiObject -class win32_perfrawdata_perfdisk_logicaldisk `
-Filter "name = '_Total'"
[double]$n2 = $wmiperf.percentIdleTime
```

```
[double]$r2 = $wmiperf.percentDiskTime
[double]$d2 = $wmiperf.TimeStamp_Sys100NS
"rep $i . counting to rep $numrep ..."
$PercentIdleTime = (1 - (($N2 - $N1)/($D2-$D1)))*100
"`tPercent Disk idle time is: " + "{0:N2}" -f $PercentIdleTime
$PercentDiskTime = (1 - (($r2 - $r1)/($D2-$D1)))*100
"`tPercent Disk time is: " + "{0:N2}" -f $PercentDiskTime
}
```

Resultado optenido despues de la ejecución.

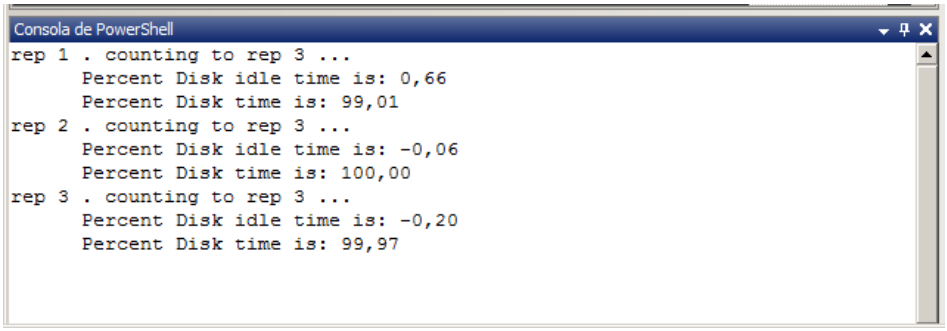


Figura V. 44. Gráfico de monitoreo del disco duro
Fuente: EL Autor

- **Monitoreando espacio en el disco**

Codigo

```
function funline ($strIN)
{
$num = $strIN.length
for($i=1 ; $i -le $num ; $i++)
{ $funline = $funline + "=" }
Write-Host -ForegroundColor yellow $strIN
Write-Host -ForegroundColor darkYellow $funline
}
$arycomputer = "localhost", "loopback"
foreach($computer in $arycomputer)
{
$volumeSet = Get-WmiObject -Class win32_volume -computer $computer `
-filter "drivetype = 3"
```

```
foreach($volume in $volumeSet)
{
$drive=$volume.driveLetter
[int]$free=$volume.freespace/1GB
[int]$capacity=$volume.capacity/1GB
funline("Drives on $computer computer:")
"Analyzing drive $drive $($volume.label) on $($volume.__server)"
"`t`t Percent free space on drive $drive " + "{0:N2}" -f `
(($free/$capacity)*100)
}
}
```

Resultado optenido despues de la ejecución.

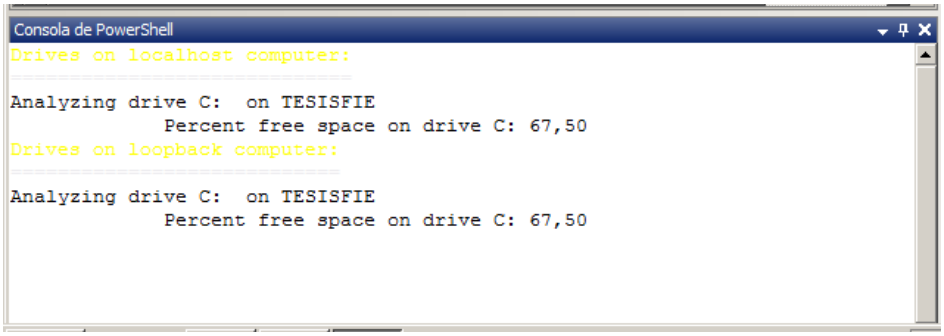


Figura V. 45. Gráfico delas unidades físicas de disco duro
Fuente: EL Autor

2) En el segundo escenario de administración es la automatización de un computador con directorio activo en donde se realiza los siguientes puntos:

a) Crear dominio.

b) Crear OU en el dominio

c) Crear grupos del dominio.

d) Crear y modificar usuarios del dominio.

e) Trabajar con IIS

f) Trabajar con Terminal Server,

g) Trabajar con DNS

Cabe recalcar que algunos procesos tales como Directorio Activo, DNS, IIS deben ser activados de manera grafica antes de ser utilizados ya que solo se puede visualizar configuraciones.

a) Crear dominio.

Configuración del Directorio Activo.

Se debe utilizar el asistente para poder levantar la configuración o el comando dcpromo.exe

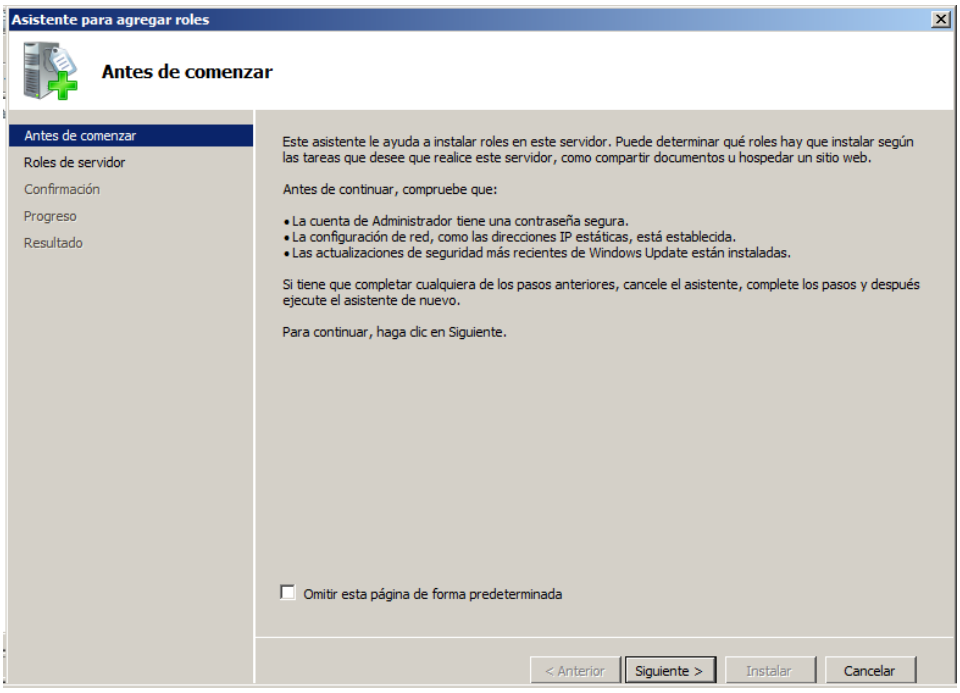


Figura V. 46. Configuración
Fuente: EL Autor

Luego se escoje la ocion servicios de dominio del directorio activo y siguiente

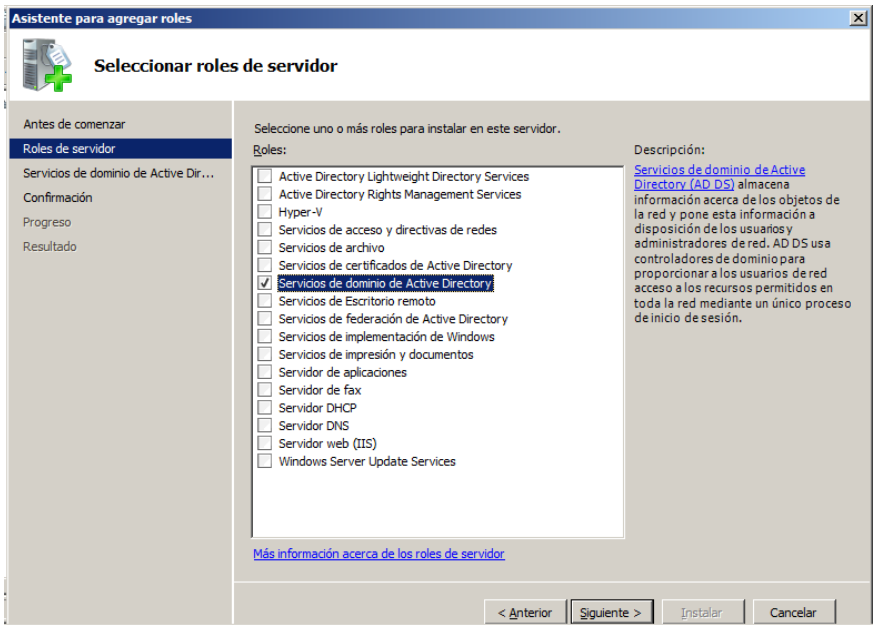


Figura V. 47. Configuración Directorio
Fuente: EL Autor

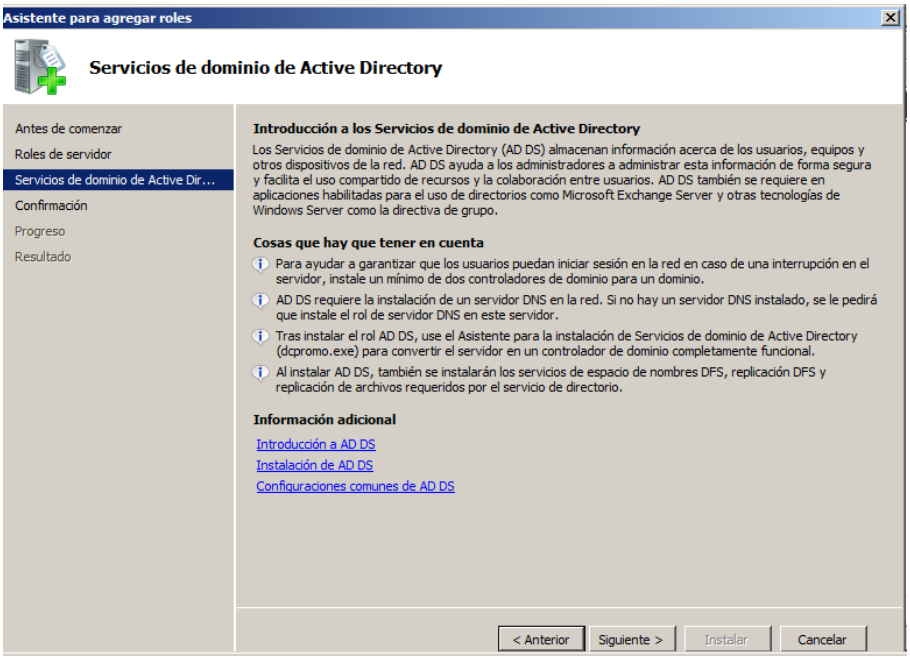


Figura V. 48. Configuración servicios
Fuente: EL Autor

Se confirma las secciones de instalación y se procede a instalar.

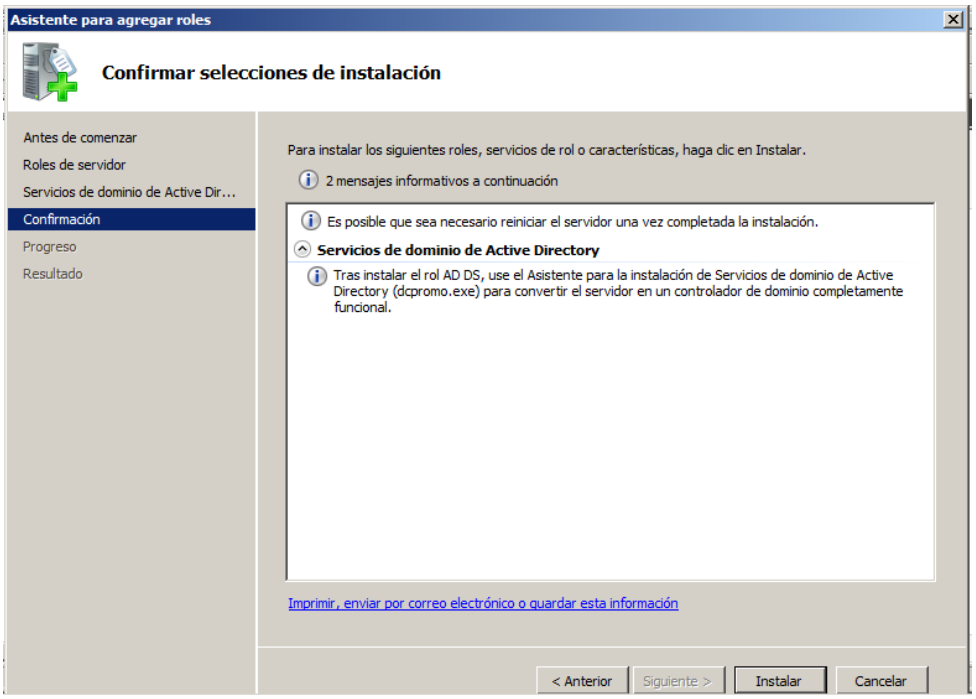


Figura V. 49. Configuración AD
Fuente: EL Autor

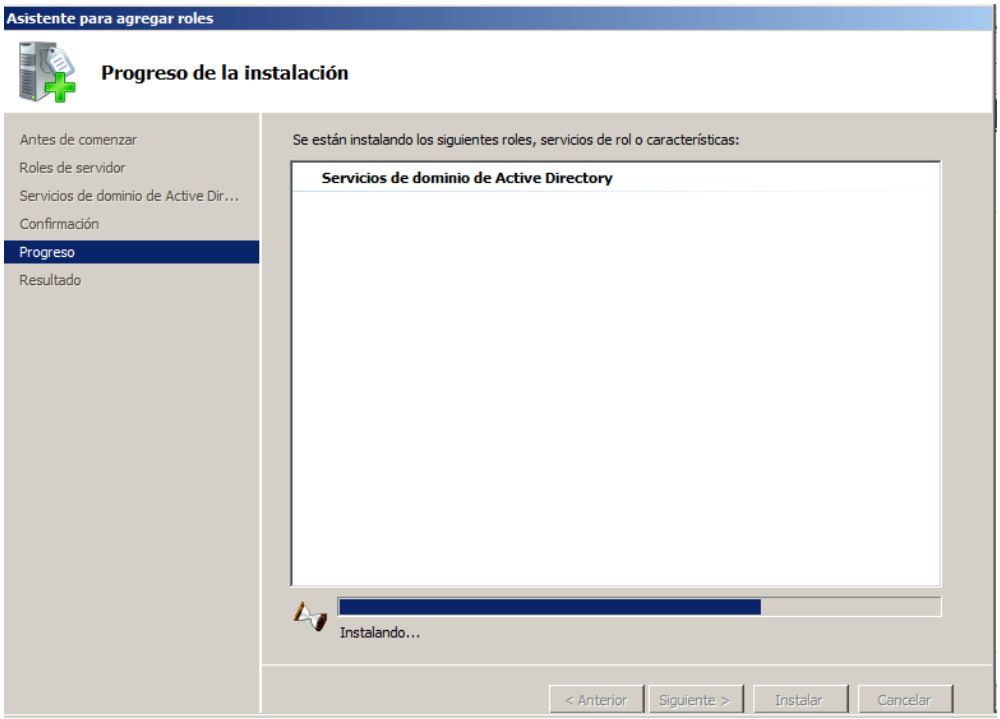


Figura V. 50. Configuración del dominio de AD
Fuente: EL Autor

Configuracion del DNS

Luego de la configuración del directorio activo de debe configural el servicio DNS, para ello se ejecuta el asistente de instalación.

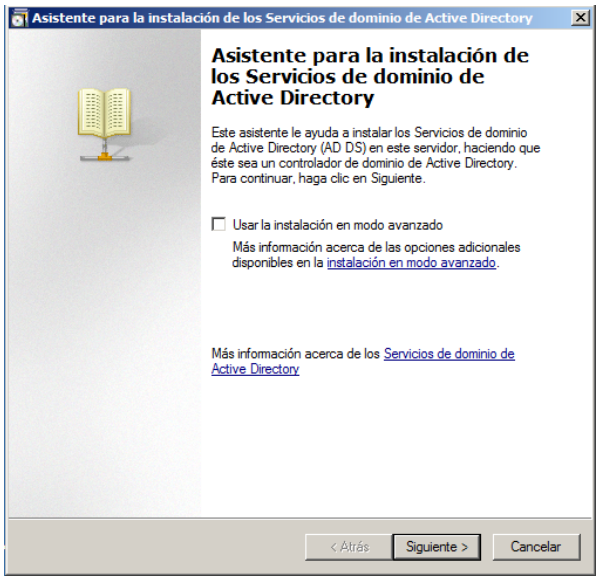


Figura V. 51. Asistente de dns
Fuente: EL Autor

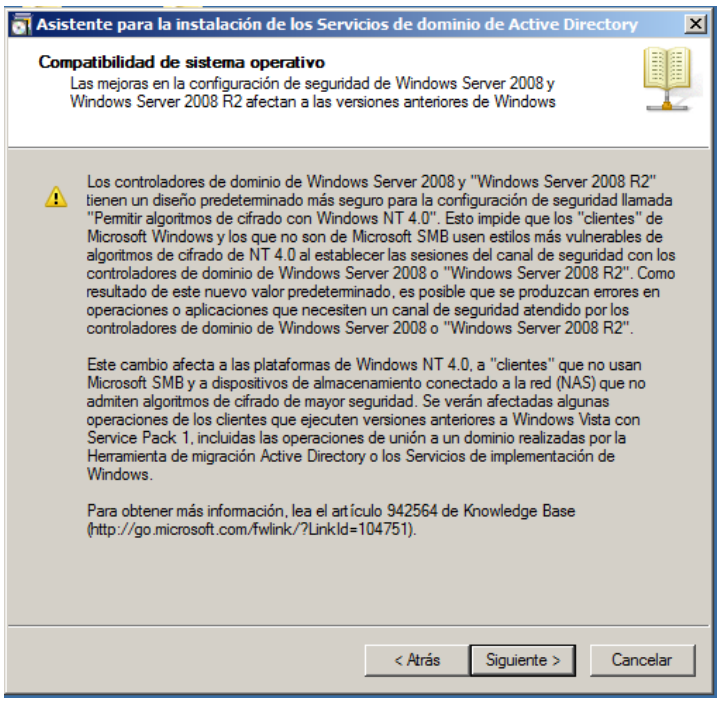


Figura V. 52. Configuración del sistema operativo
Fuente: EL Autor

Se crea un nuevo dominio, ya que no tenenmos otro dominio instado anteriomente para poder escoger la opción de bosque existente.

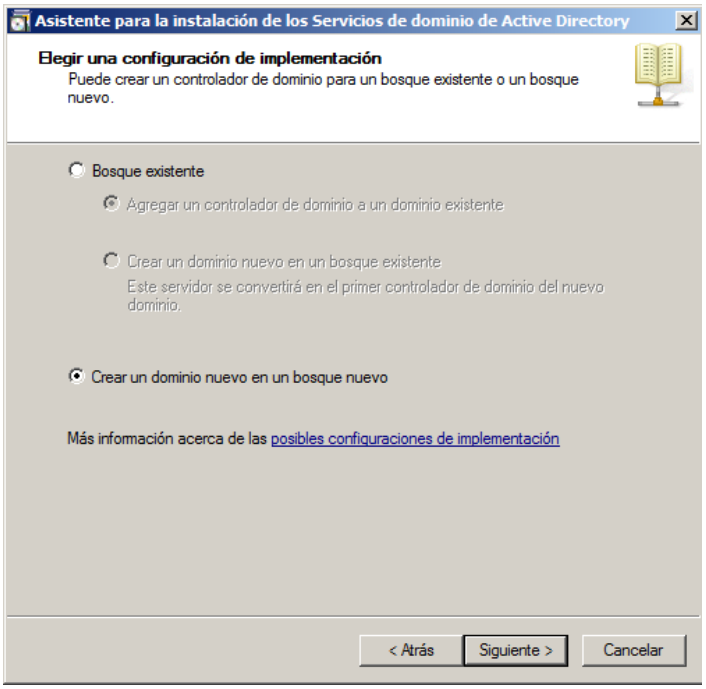


Figura V. 53. Configuración de nuevo dominio
Fuente: EL Autor

Se procede a colocar el nombre del dominio para el cual es "academia.espoch.edu.ec"

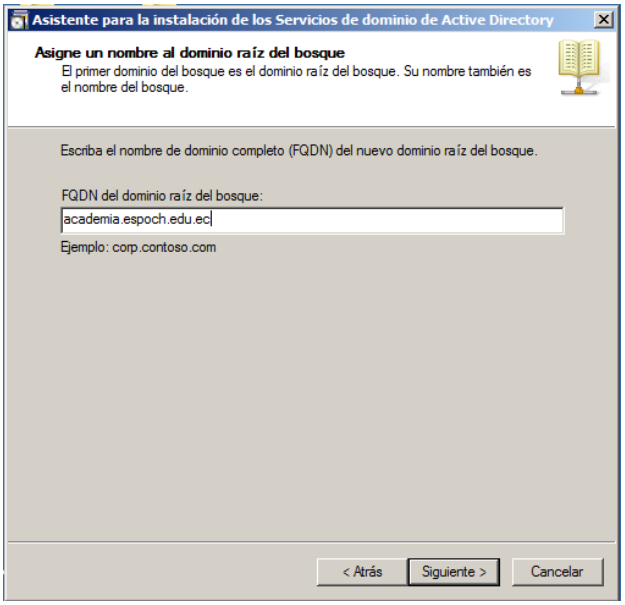


Figura V. 54. Configuración del nombre de dominio
Fuente: EL Autor

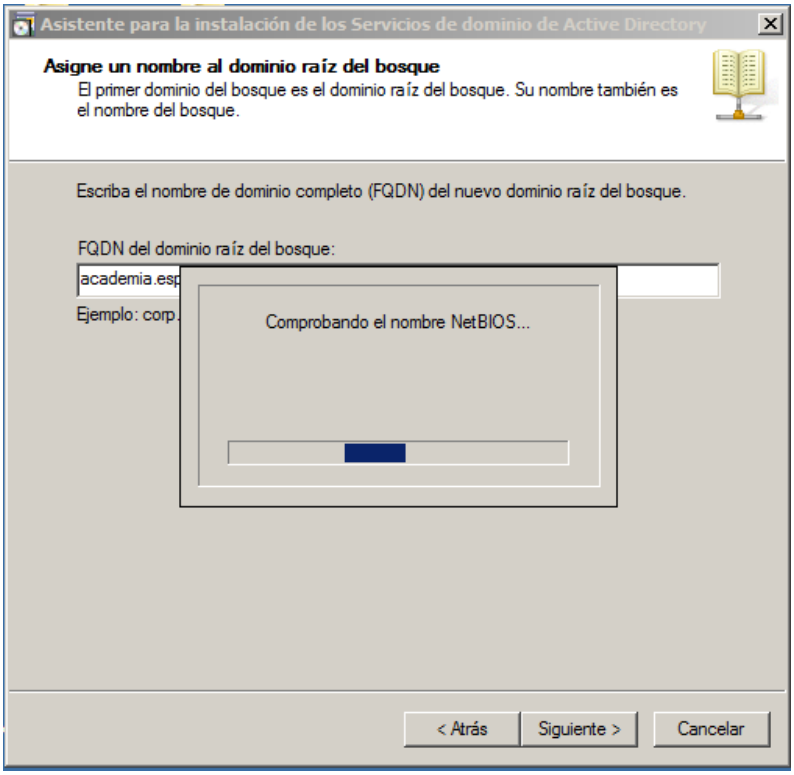


Figura V. 55. Comprobación de netbios
Fuente: EL Autor

Escoger que sea Windows server 2008 R2 ya que se han corregido muchas fallas con respecto al anterior sistema operativo

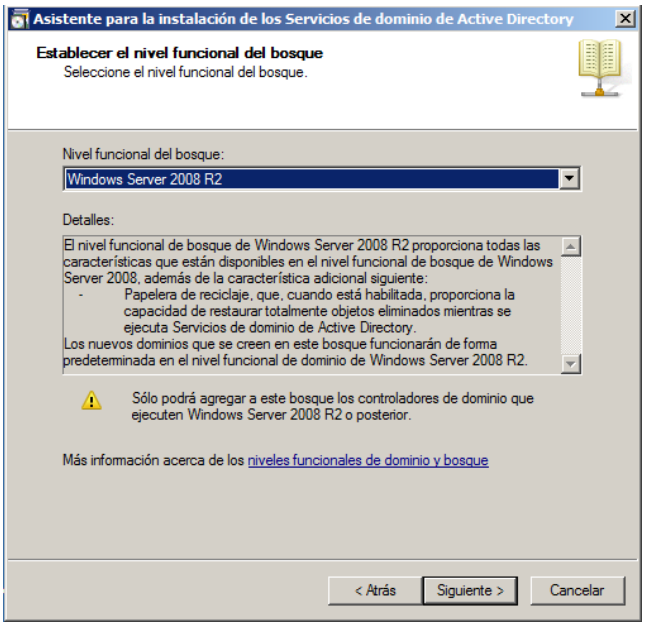


Figura V. 56. Compatibilidad del sistema operativo
Fuente: EL Autor

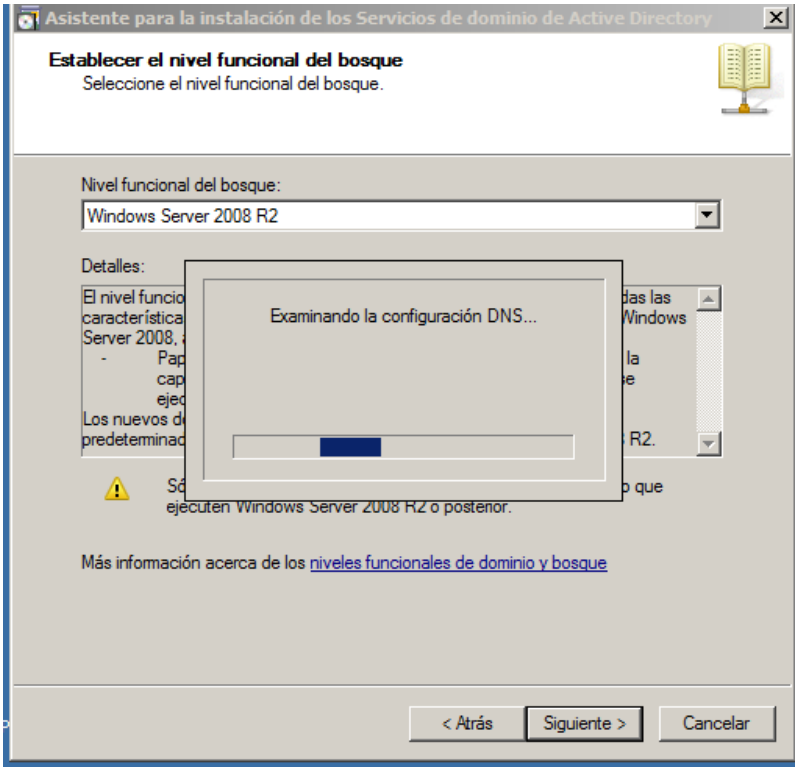


Figura V. 57. Configuración examinianción DNS
Fuente: EL Autor

Se selección a opción adicional de DNS ya que como se explica anteriormente no se lo puede hacer en modo script en Windows server 2008.

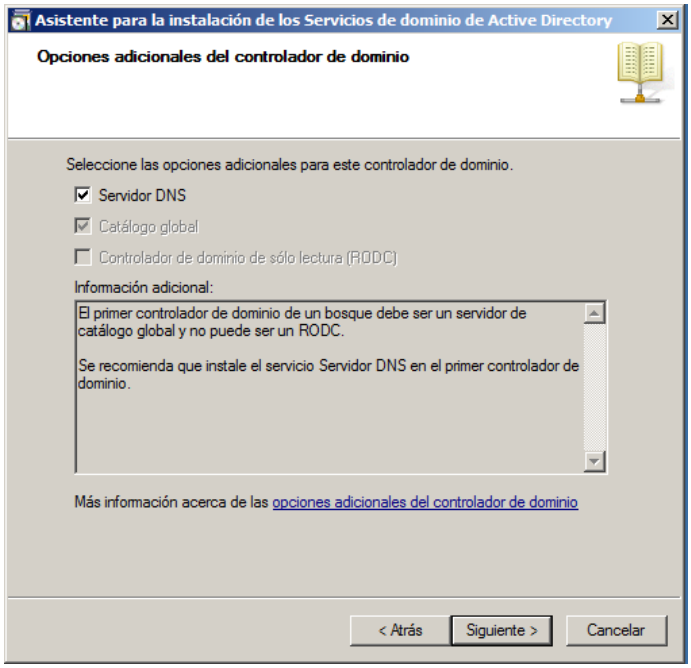


Figura V. 58. Opciones adicionales
Fuente: EL Autor

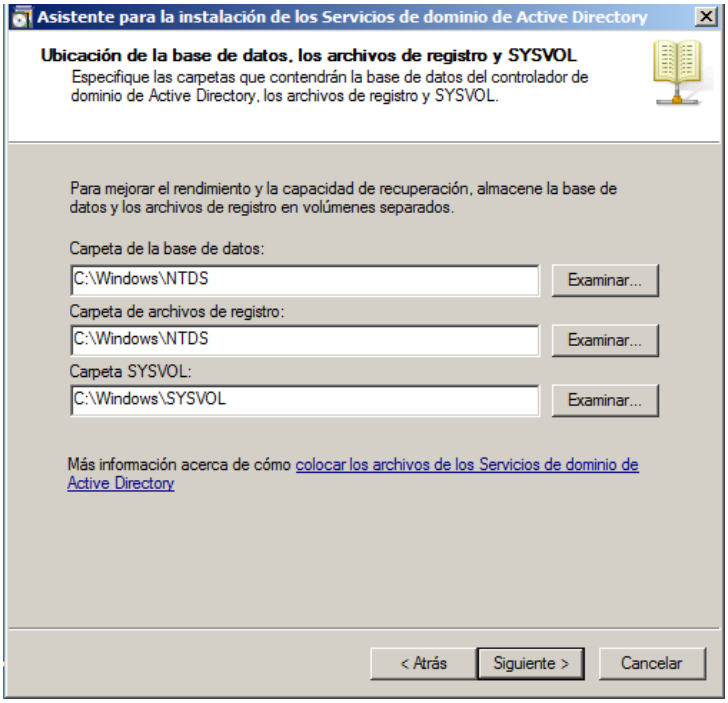


Figura V. 59. Ubicación de la base de datos
Fuente: EL Autor

Colocar la contraseña que sea segura, tal clave debe poseer al menos una letra mayúscula, una miniscula, un numero y un símbolo deacuerdo a los standares de seguridad.

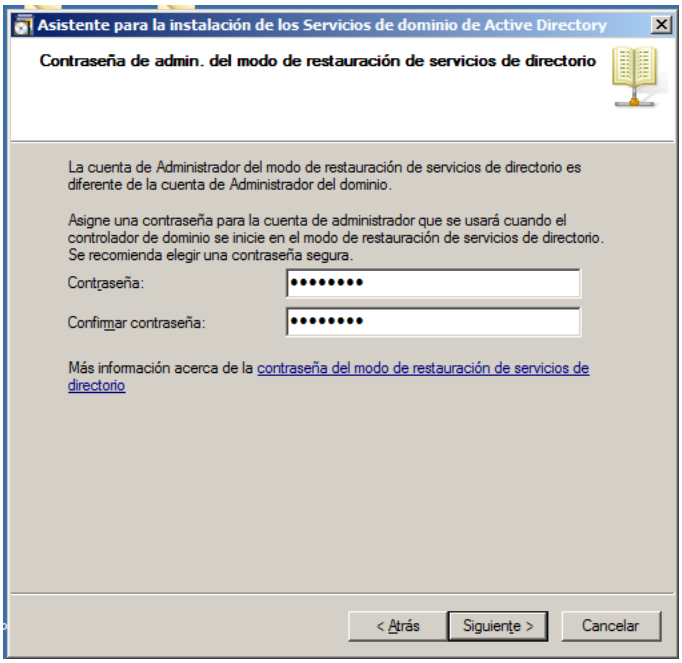


Figura V. 60. Contraseña
Fuente: EL Autor

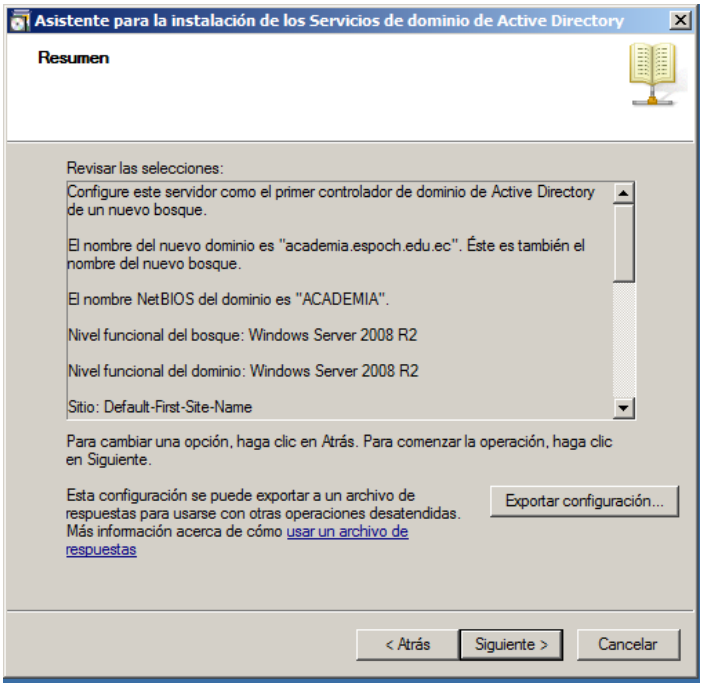


Figura V. 61. Resumen
Fuente: EL Autor

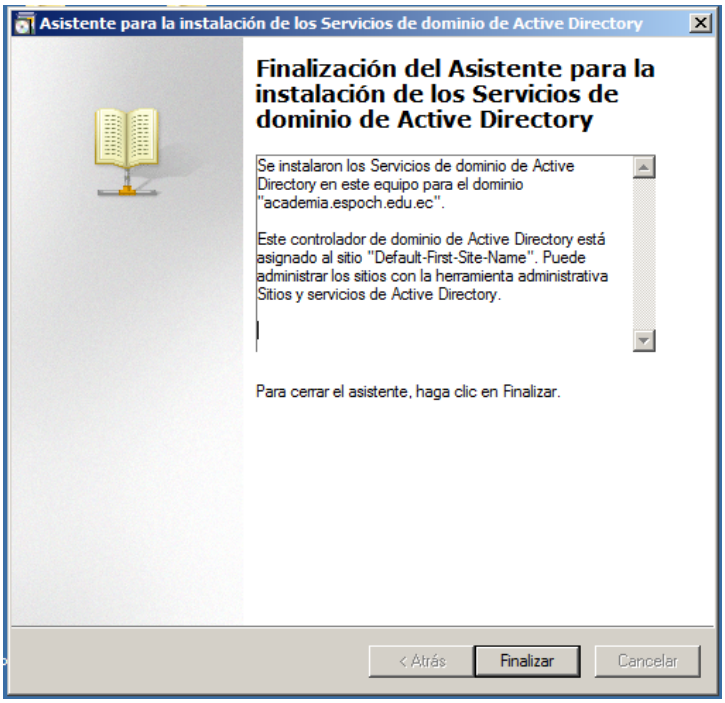


Figura V. 62. Finalización de la configuración
Fuente: EL Autor

Figura V. 63. Login
Fuente: EL Autor

Es necesario recalcar que powershell no pude realizar ciertas tarea de administración por lo que es transfoma en una desventaja al momento de las configuraciones iniciales

b) Creacion de OU en el dominio.

- **Crear OU**

Codigo.

```
param($name="ou=FIE2",$ou,$dc="ou=FIE,dc=academia,dc=espoch,dc=edu
,dc=ec",[switch]$help)
function funHelp()
{
$helpText= @ "
DESCRIPCIÓN:
NOMBRE: CreateOU.ps1
Crea un OrganizationalUnit
PARÁMETROS:
nombre-nombre de la OrganizationalUnit para crear
-ou ou OrganizationalUnit para crear en
```

-dc de dominio para crear OrganizationalUnit en

-help Muestra el archivo de ayuda

SINTAXIS:

CreateOU.ps1-name "OU = MyNewOU" ou "Myou" `

-dc "dc = nwtraders, dc = com"

Crea un OrganizationalUnit llamado MyNewOU en el Myou unidad organizativa en el dominio nwtraders.com

CreateOU.ps1-name "ou = mynewou" dc "dc = nwtraders, dc = com"

Crea un OrganizationalUnit llamado MyNewOU en la raíz del dominio nwtraders.com

CreateOU.ps1-ayuda Muestra el tema de ayuda para el script

"@$helpText

exit

}

if($help){ "Obtaining help ..." ; funhelp }

if(!$name -or !$dc) { "Missing parameter ..." ; funhelp }

if($ou)

{ "Creating OU $name in LDAP://$ou,$dc"

$ADSI = [ADSI]"LDAP://$ou,$dc"

}

ELSE

{ "Creating OU $name in LDAP://$dc"

$ADSI = [ADSI]"LDAP://$dc"

}

$Class = "OrganizationalUnit"

$OrganizationalUnit = $ADSI.create($CLass, $Name)

$OrganizationalUnit.setInfo()

Resultado optenido despues de la ejecución.

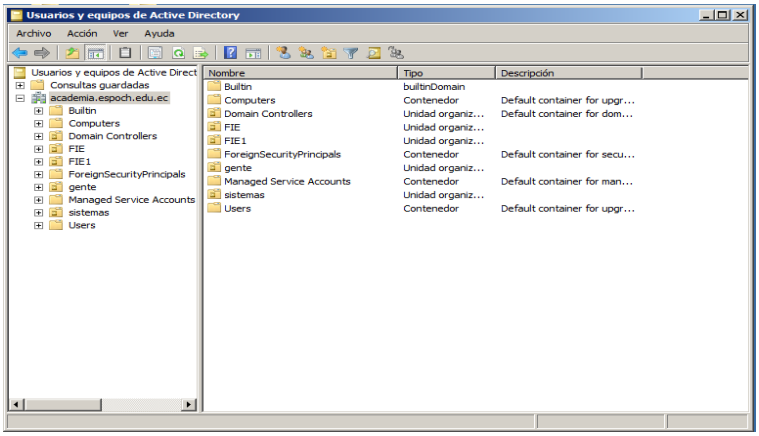


Figura V. 64. Gráfico de la creacion de OU
Fuente: EL Autor

c) Creacion de grupos en el dominio.

- **Crear grupo**

Codigo

```
param($name="cn=grupoprofesores",$ou="ou=fie",$dc="dc=academia,dc=es
poch,dc=edu,dc=ec",[switch]$help)
function funHelp()
{
$helpText=@ "
DESCRIPCIÓN:
NOMBRE: CreateGroup.ps1
Crea un grupo de
PARÁMETROS:
nombre-nombre del grupo para crear
-ou ou para crear grupo en
-dc de dominio para crear el grupo de
-help Muestra el archivo de ayuda
SINTAXIS:
CreateGroup.ps1-name "CN = MyNewGroup" ou "Myou" `
```

```
-dc "dc = nwtraders, dc = com"
Crea un grupo llamado MyNewGroup en el Myou
unidad organizativa en el dominio nwtraders.com
CreateGroup.ps1-name "CN = MyNewGroup" `
-dc "dc = nwtraders, dc = com"
Crea un grupo llamado MyNewGroup en los usuarios
contenedor del dominio nwtraders.com
CreateGroup.ps1-ayuda
Muestra el tema de ayuda para el script
"@$helpText
exit
}
if($help){ "Obtaining help ..." ; funhelp }
if(!$name -or !$dc) { "Missing name parameter ..." ; funhelp }
if($ou)
{ "Creating group $name in LDAP://$ou,$dc"
$ADSI = [ADSI]"LDAP://$ou,$dc"
}
ELSE
{ "Creating group $name in LDAP://cn=users,$dc"
$ADSI = [ADSI]"LDAP://cn=users,$dc"
}
$Class = "Group"
$Group = $ADSI.create($Class, $Name)
$Group.setInfo()
```

Resultado optenido despues de la ejecución.

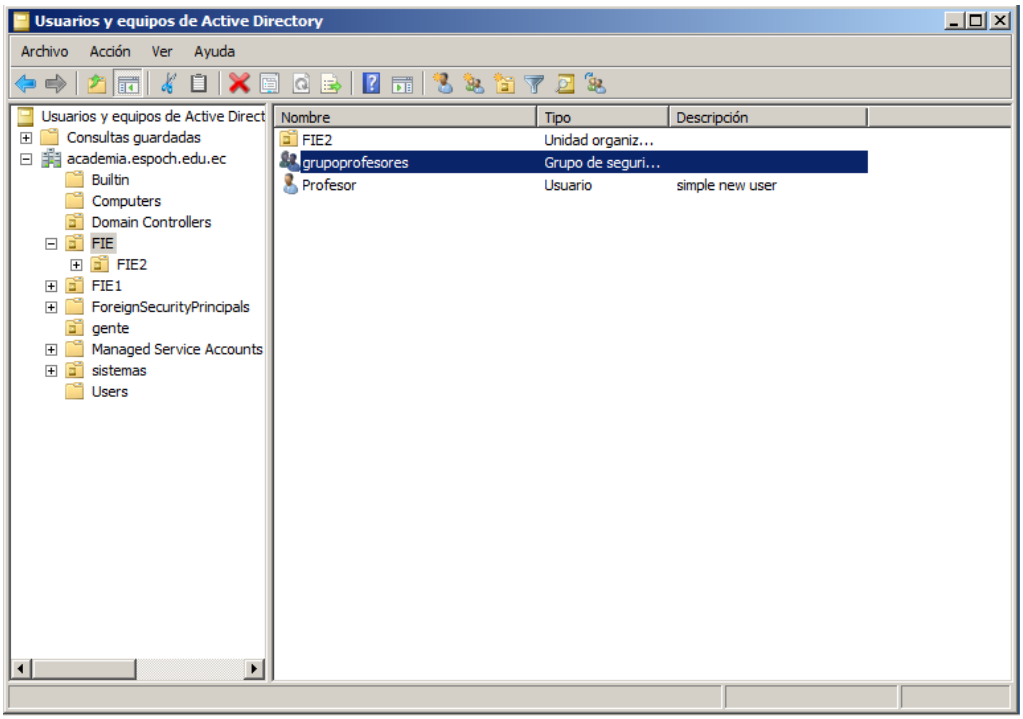


Figura V. 65. Configuración de grupos
Fuente: EL Autor

d) Crear y Modificar usuarios del Dominio.

- **Creación de usuarios del dominio.**

Codigo

```
param($name="CN=Profesor",$ou="ou=fie",$dc="dc=academia,dc=espoch,d
c=edu,dc=ec",[switch]$help)
function funHelp()
{
$helpText= @ "
DESCRIPCIÓN:
NOMBRE: CreateUser.Ps1
Crea una cuenta de usuario
PARÁMETROS:
```

```
nombre-nombre del usuario para crear
-ou ou para crear el usuario en
-dc de dominio para crear el usuario en
-help Muestra el archivo de ayuda
SINTAXIS:
CreateUser.Ps1-name "CN = MyNewUser" ou "ou = Myou" `
-dc "dc = nwtraders, dc = com"
Crea un usuario llamado MyNewUser en el Myou
unidad organizativa en el dominio nwtraders.com
CreateUser.ps1-nombre "cn = miUsuario" ou "ou = ou2, ou = mytestou" `
-dc "dc = nwtraders, dc = com"
Crea un usuario llamado MyNewUser en la organización ou2
unidad. Una OU niño de la unidad organizativa mytestou
en el dominio nwtraders.com
CreateUser.Ps1-name "CN = MyNewUser" `
-dc "dc = nwtraders, dc = com"
Crea un usuario llamado MyNewUser en los usuarios
contenedor del dominio nwtraders.com
CreateUser.Ps1-ayuda
Muestra el tema de ayuda para el script
"@$helpText
exit
}
if($help){ "Obtaining help ..." ; funhelp }
if(!$name -or !$dc) { "Missing name parameter ..." ; funhelp }
if($ou)
{ "Creating user $name in LDAP://$ou,$dc"
$ADSI = [ADSI]"LDAP://$ou,$dc"
}
ELSE
{ "Creating user $name in LDAP://cn=users,$dc"
$ADSI = [ADSI]"LDAP://cn=users,$dc"
```

```
}
$CLass = "User"
$User = $ADSI.create($CLass, $Name)
$User.setInfo()
```

Resultado optenido despues de la ejecución.

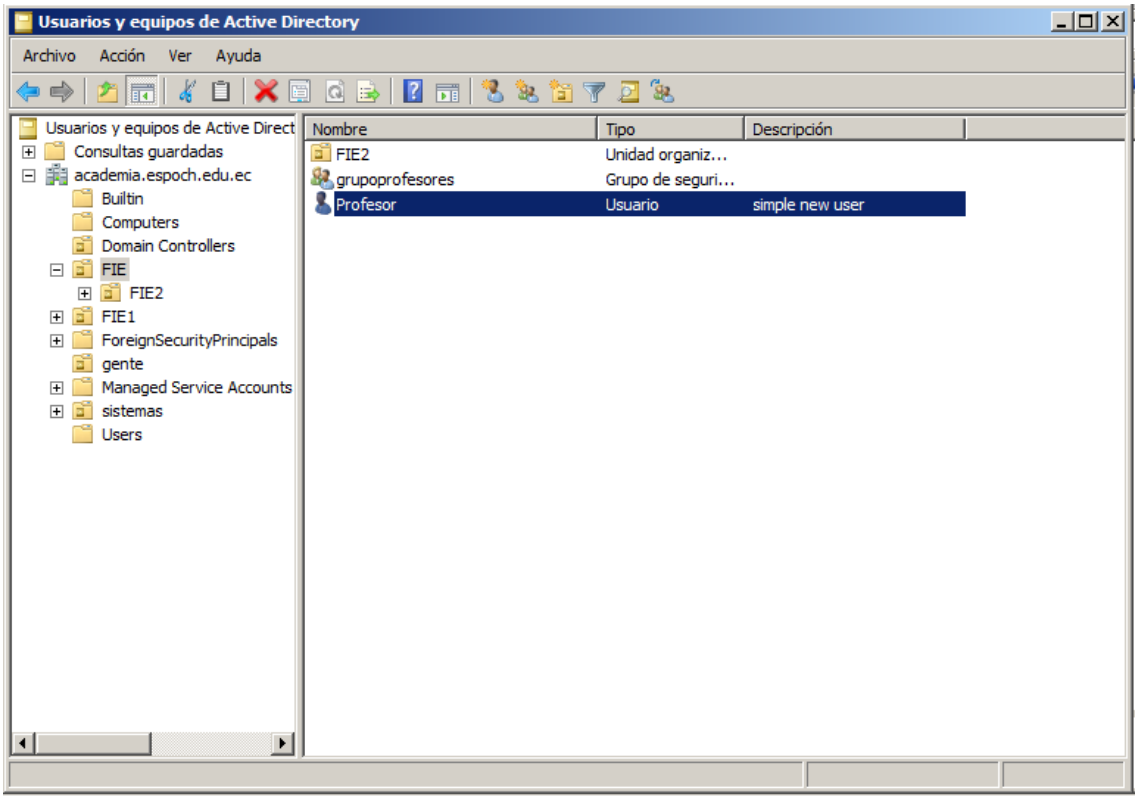


Figura V. 66. Configuración de usuario del dominio
Fuente: EL Autor

- **Adicionar un usuario a un grupo**

Código

```
param($name="cn=profesor",$group="cn=grupoprofesores",$ou="ou=fie",$d
c="dc=academia,dc=espoch,dc=edu,dc=ec",[switch]$help)
function funHelp()
{
$helpText= @ "
```

```
DESCRIPCIÓN:
NOMBRE: AddUserToGroup.ps1
Agrega una cuenta de usuario a un grupo
PARÁMETROS:
nombre-nombre del usuario
-ou ou del grupo
-dc dominio del usuario
-grupo de grupo para modificar
-help Muestra el archivo de ayuda
SINTAXIS:
AddUserToGroup.ps1-nombre "cn = MyNewUser" ou "ou = Myou" `
-dc "dc = nwtraders, dc = com" `
-grupo "cn = MiGrupo"
Agrega un usuario llamado MyNewUser en el Myou
unidad organizativa en el dominio nwtraders.com
al grupo MiGrupo en la misma unidad organizativa.
AddUserToGroup.ps1-ayuda
Muestra el tema de ayuda para el script
"@$helpText
exit
}
if($help){ "Obtaining help ..." ; funhelp }
if(!$name -or !$dc -or !$group -or !$ou)
{ "Missing parameter ..." ; funhelp }
$CLass = "User"
"Modifying $name,$ou,$dc"
$ADSI = [ADSI]"LDAP://$group,$ou,$dc"
$ADSI.add("LDAP://$name,$ou,$dc")
```

Resultado optenido despues de la ejecución.

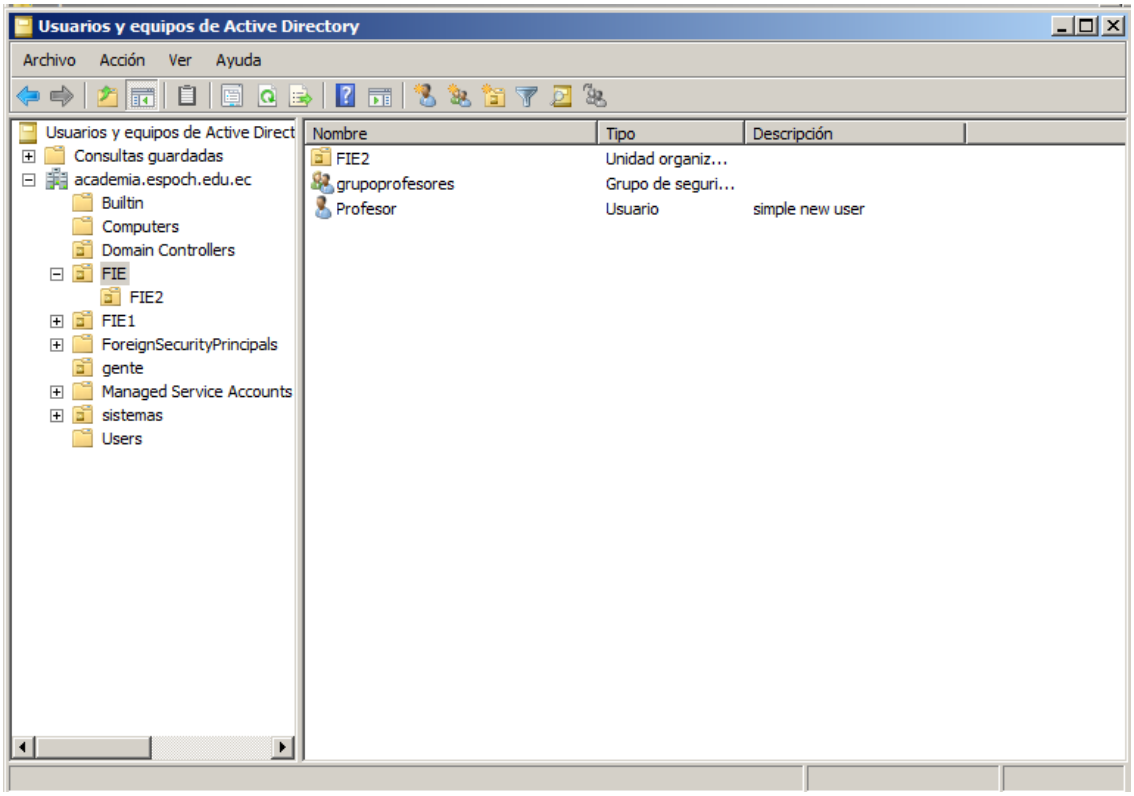


Figura V. 67. Adición de un usuario a grupo
Fuente: EL Autor

- **Modificacion de los datos de un usuario**

Código

```
$objUser =
[ADSI]"LDAP://cn=profesor,ou=Fie,dc=academia,dc=espoch,dc=edu,dc=ec"
$objUser.put("SamaccountName", "myNewUser")
$objUser.put("givenName", "My")
$objUser.Put("initials", "N.")
$objUser.Put("sn", "User")
$objUser.Put("DisplayName", "My New User")
$objUser.Put("description" , "simple new user")
$objUser.Put("physicalDeliveryOfficeName", "RQ2")
$objUser.Put("telephoneNumber", "999-222-1111")
```

$objUser.Put("mail", "mnu@hotmail.com")

$objUser.Put("wwwHomePage", "http://www.mnu.msn.com")

$objUser.setInfo()

Resultado optenido despues de la ejecución.

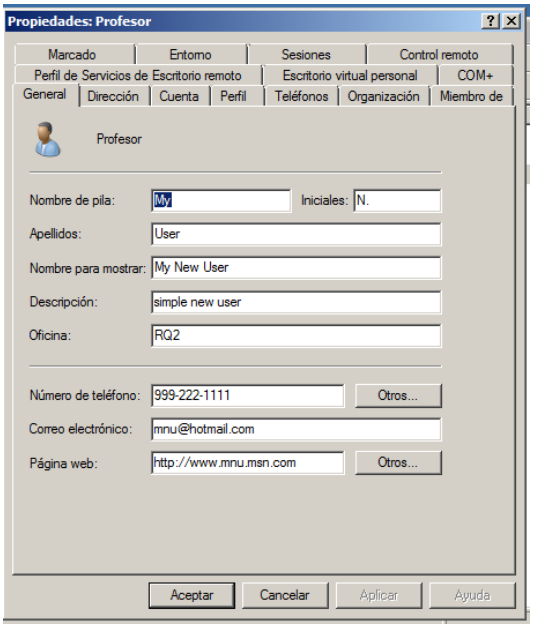


Figura V. 68. Modificación de datos de usuario
Fuente: EL Autor

- **Modificacion de la direccion de un usuario**

Codigo.

$objUser =

[ADSI]"LDAP://cn=profesor,ou=Fie,dc=academia,dc=espoch,dc=edu,dc=ec"

$objUser.put("streetAddress", "123 main st")

$objUser.put("postOfficeBox", "po box 12")

$objUser.put("l", "Bedrock")

$objUser.put("st", "Arkansas")

$objUser.put("postalCode" , "12345")

$objUser.put("c", "US")

$objUser.put("co", "United States")

$objUser.put("countryCode", "840")

$objUser.setInfo()

Resultado optenido despues de la ejecución.

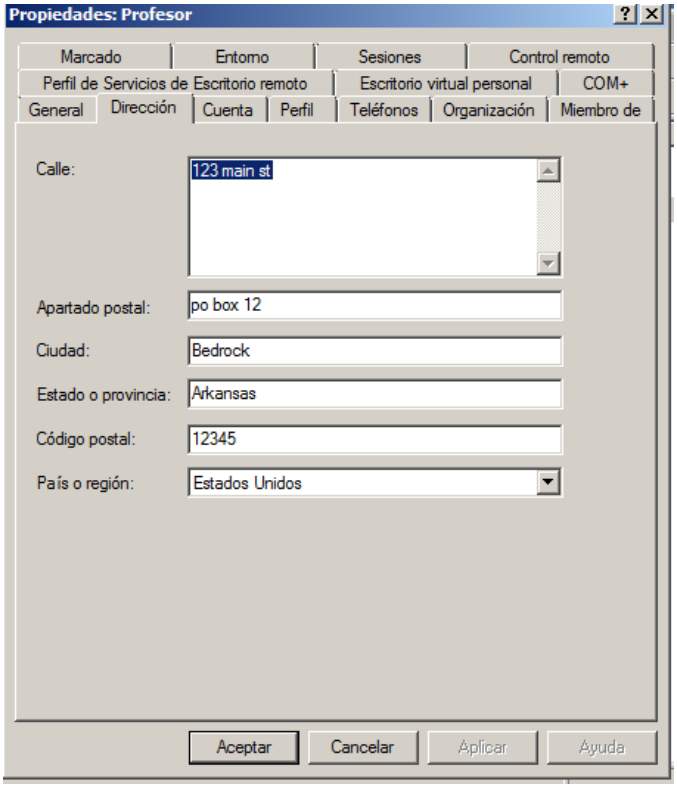


Figura V. 69. Configuración de dirección
Fuente: EL Autor

- **Modificacion del perfil del usuario**

Código.

```
$objUser =
[ADSI]"LDAP://cn=profesor,ou=Fie,dc=academia,dc=espoch,dc=edu,dc=ec"
$objUser.put("profilePath", "\\London\profiles\myNewUser")
$objUser.put("scriptPath", "logon.vbs")
$objUser.put("homeDirectory", "\\london\users\myNewUser")
$objUser.put("homeDrive", "H:")
$objUser.setInfo()
```

Resultado optenido despues de la ejecución.

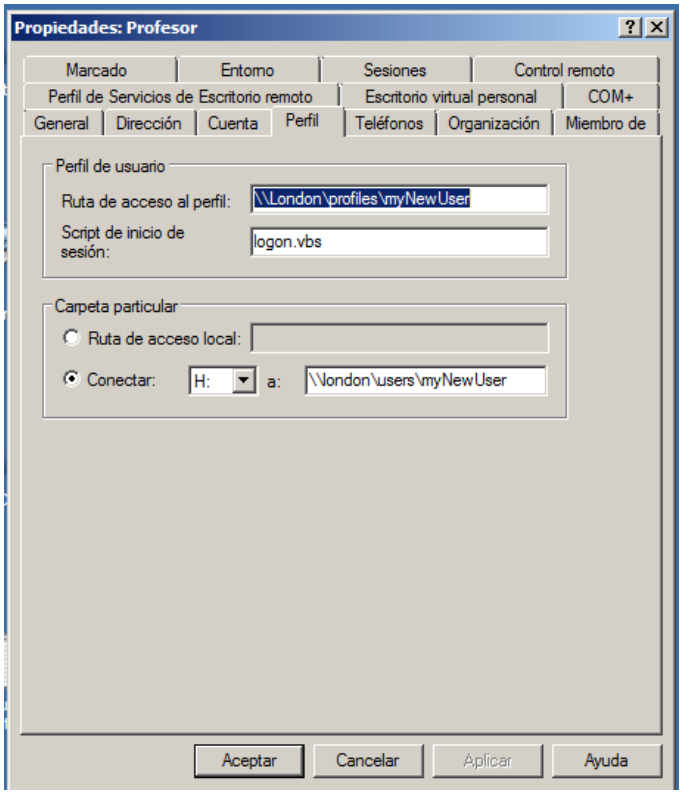


Figura V. 70. Configuración de pefil
Fuente: EL Autor

- **Modificacion de teléfono del usuario**

Código

```
$objUser =
[ADSI]"LDAP://cn=profesor,ou=Fie,dc=academia,dc=espoch,dc=edu,dc=ec"
$objUser.Put("homePhone", "(215)788-4312")
$objUser.Put("pager", "(215)788-0112")
$objUser.Put("mobile", "(715)654-2341")
$objUser.Put("facsimileTelephoneNumber", "(215)788-3456")
$objUser.Put("ipPhone", "192.168.6.112")
$objUser.Put("info", "All contact information is confidential," `
+ "and is for official use only.")
$objUser.setInfo()
```

Resultado optenido despues de la ejecución.

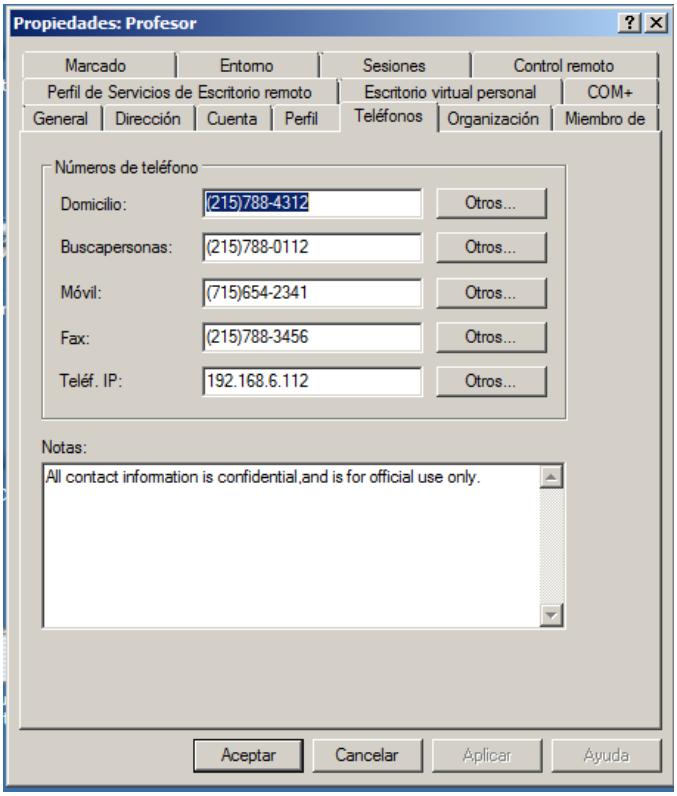


Figura V. 71. Configuración de telefono
Fuente: EL Autor

- **Modificar un único atributo del usuario**

Código

```
param($name="cn=profesor",$property="title",$value="jefe
administrador",$ou="ou=fie",$dc="dc=academia,dc=espoch,dc=edu,dc=ec",[
switch]$help)
function funHelp()
{
$helpText=@ "
DESCRIPCIÓN:
NOMBRE: ModifyUser.ps1
Modifica una cuenta de usuario
PARÁMETROS:
```

nombre-nombre del usuario para modificar

-ou ou del usuario

-dc dominio del usuario

-propiedad de atributo para modificar

valor valor del atributo

-help Muestra el archivo de ayuda

SINTAXIS:

ModifyUser.ps1-name "CN = MyNewUser" ou "ou = Myou" `

-dc "dc = nwtraders, dc = com" `

-propiedad "sAMAccountName" `

-valor "MyNewUser"

Modifica un usuario llamado MyNewUser en el Myou unidad organizativa en el dominio nwtraders.com agrega el attriute samAccountName con un valor de MyNewUser

ModifyUser.ps1-ayuda Muestra el tema de ayuda para el script

"@$helpText

exit

}

if($help){ "Obtaining help ..." ; funhelp }

if(!$name -or !$dc -or !$property -or !$value)

{ "Missing parameter ..." ; funhelp }

$CLass = "User"

"Modifying $name,$ou,$dc"

$ADSI = [ADSI]"LDAP://$name,$ou,$dc"

$ADSI.put($property, $value)

$ADSI.setInfo()

Resultado optenido despues de la ejecución.

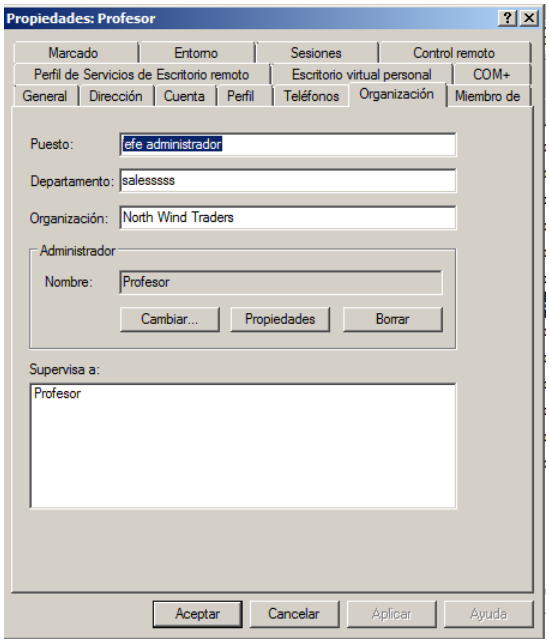


Figura V. 72. Configuración de atributos
Fuente: EL Autor

e) Trabajando con IIS

- **Lista los sitios**

Codigo

```
param($computer="localhost", [switch]$help)
function funHelp()
{
$helpText=@"
DESCRIPTION:
"@
$helpText
exit
}
if($help) { "Printing help now..." ; funHelp }
Get-WmiObject -Namespace root\webadministration `
-computername $computer -class site |
```

format-table -property name

- **Muetra el grupo de aplicaciones en el equipo local**

Codigo

param($computer="localhost", [switch]$help)
function funHelp()
{
$helpText=@"
DESCRIPCIÓN:
NOMBRE: GetAppPool.ps1
Obtiene una lista de los grupos de aplicaciones en un equipo local o remoto.
PARÁMETROS:
-equipo Especifica el nombre del equipo para ejecutar la secuencia de comandos
-help Muestra el archivo de ayuda
SINTAXIS:
GetAppPool.ps1
Obtiene una lista de los grupos de aplicaciones en el equipo local
GetAppPool.ps1-ordenador "webserverII"
Obtiene una lista de los grupos de aplicaciones en un servidor web llamado webserverII.
GetAppPool.ps1-ayuda
Imprime el tema de la ayuda para el script
"@$helpText
exit
}
if($help) { "Printing help now..." ; funHelp }
Get-WmiObject -Namespace root\webadministration `
-computername $computer -Class applicationpool |
format-table -property name, autostart, `
@{

Label = ".Net Version" ;

Expression = { $_.ManagedRuntimeVersion }

}, `

QueueLength –autosize

Resultado

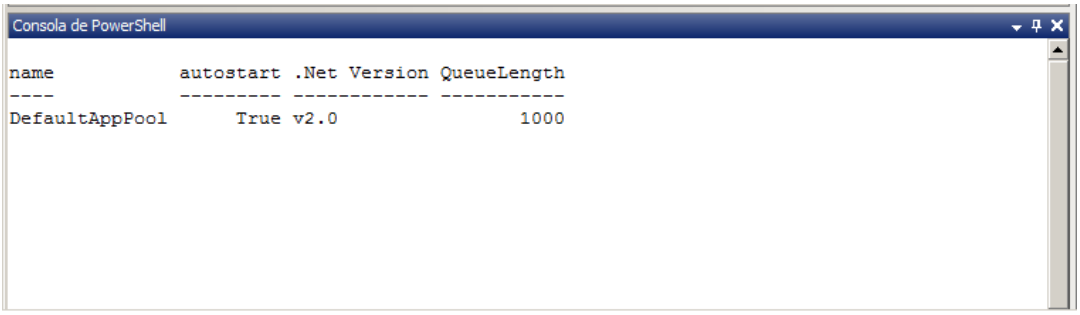


Figura V. 73. Configuración grupo de aplicaciones de equipo local
Fuente: EL Autor

- **Informa los límites del sitio**

Código

param($computer="localhost", [switch]$help)

function funHelp()

{

$helpText=@"

DESCRIPCIÓN:

NOMBRE: GetSiteLimits.ps1

Obtiene una lista de los límites del sitio en un equipo local o remoto.

PARÁMETROS:

-equipo Especifica el nombre del equipo para ejecutar la secuencia de comandos

-help Muestra el archivo de ayuda

SINTAXIS:

GetSiteLimits.ps1

Obtiene una lista de los límites del sitio en el equipo local,

GetSiteLimits.ps1-ordenador "webserverII"

Obtiene una lista de los límites del sitio en el servidor web llamado webserverII.

GetSiteLimits.ps1-ayuda

Imprime el tema de la ayuda para el script

"@

$helpText

exit

}

if($help) { "Printing help now..." ; funHelp }

$server = Get-WmiObject -Namespace root\webadministration `

-computername $computer -class server

$server.SiteDefaults.limits.maxconnections

$server.SiteDefaults.limits.ConnectionTimeout

$server.SiteDefaults.limits.MaxBandwidth

Resultado optenido despues de la ejecución.

Figura V. 74. Configuración limites del sitio
Fuente: EL Autor

f) Trabajndo con terminal server

- **Reporte de servicios de teminal**

Codigo

param(

```
$computer = "localhost",
[switch]$help
)
function funHelp()
{
$helpText=@"
DESCRIPCIÓN:
NOMBRE: ReportTerminalServiceSetting.ps1
Muestra la configuración de Terminal Server en un local o un mando a
distancia
servidor de terminal
PARÁMETROS:
-equipo el equipo para dirigir el guión para
-help Muestra el archivo de ayuda
SINTAXIS:
ReportTerminalServiceSetting.ps1
Muestra la configuración de Terminal Server en el equipo local
TS1 ReportTerminalServiceSetting.ps1-ordenador
Informes configuración de Terminal Server en el servidor de terminal remoto
llamado ts1
ReportTerminalServiceSetting.ps1-ayuda
Imprime el tema de la ayuda para el script
"@
$helpText
exit
}
if($help) { "Printing help now ..." ; funHelp }
$namespace = "root\cimv2\TerminalServices"
$class = "win32_TerminalServiceSetting"
get-wmiobject -namespace $namespace -computername $computer `
-class $class |
format-list [a-z]*
```

Resultado optenido despues de la ejecución.

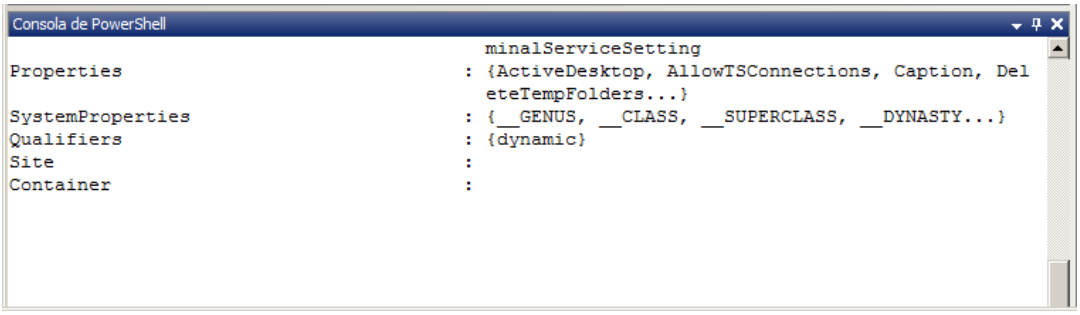


Figura V. 75. Terminal server
Fuente: EL Autor

g) Trabajando con DNS

- **Reporta el estado de DNS**

Codigo

```
param($computer="localhost",$query,[switch]$help)
function funHelp()
{
$helpText=@"
DESCRIPCIÓN :
NOMBRE: GetDNSServerConfig.ps1
Produce una lista de la información de configuración del servidor DNS
en un equipo local o remoto.
PARÁMETROS :
- equipo Especifica el nombre del equipo para ejecutar la secuencia de
comandos
- consultar el tipo de consulta < todo, avanzado, memoria caché, hacia
adelante,
intervalo , registro, recurse >
- help Muestra el archivo de ayuda
"@
$helpText
exit
```

```
}
if($help) { "Printing help now..." ; funHelp }
$class="MicrosoftDNS_Server"
$logProperty = "EventLogLevel","LogFileMaxSize","LogFilePath", `
"LogIPFilterList","LogLevel"
$forwardProperty = "ForwardDelegations", "Forwarders", `
"ForwardingTimeout"
$recurseProperty = "NoRecursion","RecursionRetry","RecursionTimeout"
$cacheProperty = "AutoCacheUpdate","EDnsCacheTimeout","MaxCacheTTL",
`
"MaxNegativeCacheTTL"
$intervalProperty = "DefaultNoRefreshInterval", `
"DefaultRefreshInterval", "DisjointNets", `
"DsPollingInterval", "DsTombstoneInterval", `
"ScavengingInterval"
$advroperty = "roundrobin","SecureResponses","EnableDnsSec", `
"BindSecondaries"
if($query)
{
switch($query)
{
"log" { $query=$logProperty }
"forward" { $query=$forwardProperty }
"recurse" { $query= $recurseProperty }
"cache" { $query=$cacheProperty }
"interval" { $query=$intervalProperty }
"advanced" { $query=$advroperty }
"all" {
Get-WmiObject -class $class -computername $computer `
-namespace root\microsoftDNS| format-list * ;
exit
}
```

```
DEFAULT { "
Using default: all items. For options try this:
GetDNSServerConfig.ps1 -help
"
Get-WmiObject -class $class -computername $computer `
-namespace root\microsoftDNS| format-list * ;
exit
}
}
}
ELSE
{
"
Using default: all items. For options try this:
GetDNSServerConfig.ps1 -help
"
Get-WmiObject -class $class -computername $computer `
-namespace root\microsoftDNS| format-list * ;
exit
}
Get-WmiObject -class $class -computername $computer `
-namespace root\microsoftDNS |
format-list -property $query
```

Resultado obtenido despues de la ejecución.

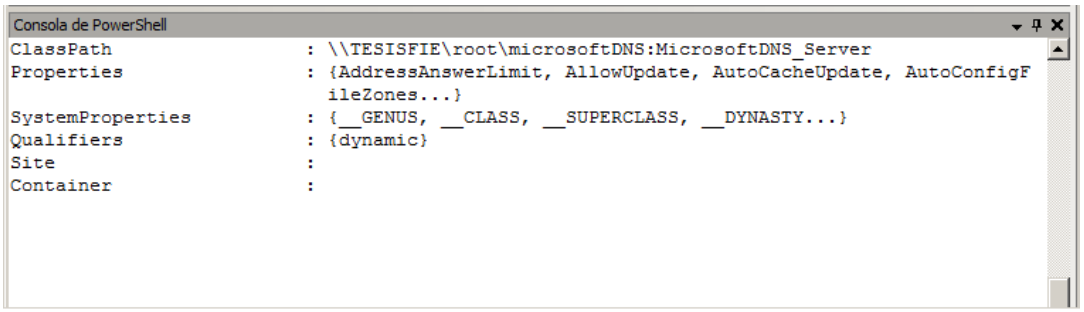


Figura V. 76. Configuración reporte dns
Fuente: EL Autor

5.8.2. Definición de estándares Scripts

Para realizar la codificación del sistema se han definido los siguientes estándares:

- El lenguaje de programación esta en Ingles.
- Se usara objetos de .net para su configuración.
- Los métodos que interactúan con los scripts reciben como parámetros de entrada:
    - Connection: Representa la cadena de conexión hacia la base de datos de Microsoft Windows server.
    - Lista de parámetros en el caso de ser necesaria.

5.8.3. Pruebas Unitarias

Para asegurar el correcto funcionamiento del Sistema se han probado sus métodos de forma independiente, enviando datos de entrada desde el código, para luego obtener los a través de los diferentes métodos para realizar consultas. Se han probado especialmente todas las funciones para validaciones de datos.

5.8.4. Pruebas de Módulos y del Sistema

Las pruebas finales consistieron en verificar que la información ingresada se vea inmediatamente reflejada en las consultas del sistemaservidor, esto sirve para comprobar que la información se está registrando correctamente en el sistema operativo.

Se provocaron errores intencionales para verificar el correcto funcionamiento del sistema, así como de las funciones de validación de datos, como por ejemplo:
- Realizar consultas de usuarios no existentes.
- Ingresar campos incorrectos en los parámetros.

CONCLUSIONES

1. Windows Server 2008, posee métodos de administración dentro de la TI, utilizando la administración tradicional o gráfica y la administración por consola o scripts.

2. La interfaz de gestión en forma de scripts es una metodología nueva de administración de TI, para servidores Microsoft que optimiza la gestión de recursos ahorrando tiempo.

3. El estándar de programación de los scripts es la utilización de cmlets con objetos de .net.

4. La seguridad es muy importante en la interfaz de administración, ya que cuenta con encriptación a nivel de firmas digitales y protección de ejecución.

5. Dentro del análisis comparativo, Microsoft PowerShell posee un 66% de desempeño en la utilización de los recursos de hardware y software, superando con un 7% a la gestión gráfica al realizar una tarea; pero al realizar tareas repetitivas o que afecten a varios objetos el margen puede ser mayor.

6. Microsoft PowerShell utilización mínimos recursos de hardware como CPU y memoria comparado con intergaz gráfica.

7. La trasportabilidad mediante el uso de Powershell posee las mejores formas de trasporte y vinculación con un 70%, ya que la gestión gráfica no se pude trasportar.

8. La creación de usuarios, grupos y unidades organizativas en la configuración de Directorio Activo utilizan recursos LDAP.

9. Los cambios que se realizan con scripts son fáciles de manejar ya que se puede saber que cambio se realizó después de la ejecución de la misma.

10. La automatización facilita a los administradores la optimización de recursos en la implementación de nuevas configuraciones.

11. Finalmente se puede concluir que la administración por interfaz script utilizando la interfaz Microsoft PowerShell automatiza de una mejor manera los procesos de configurarción y administración del servidor Microsoft de la Academia Microsoft de la ESPOCH.

RECOMENDACIONES

1. Se recomienda plantear correctamente con el método de administración que se va a realizar y recoger todos los requisitos que desea para no perder tiempo en la administración del Servidor.

2. Revisar infraestructuras y sistemas informáticos existentes antes de decidir la utilización de una determinada tecnología, ya que esto nos ayudará a tener más claro el panorama.

3. Realizar un análisis previo de características de diferentes metodos para seleccionar el que mejor se preste para desarrollar la administración de sistemas.

4. Investigar los cambios en las metodologías ya que estas facilitarán una mejor administración y optimización del tiempo.

5. Se debería implementar en cada laboratorio un método alternativo de administración el cual facilite al personal que gestiona los servidores y la información ahorro en los tiempos de configuración

6. El manejo de recursos de hardware y software es complejo por lo tanto al momento de elegir un método de administración para el manejo de los sistemas operativos servidores se debe realizar un previo análisis obligatorio de diferentes métodos para elegir cual se adapta a las necesidades de la empresa para obtener un mejor sistema administrado.

7. Se recomienda implantar una materia optativa que cubra la administración TI mediante scripts ya que brinda muchas oportunidades a los nuevos profesionales para una mejor gestión en sus trabajos.

RESUMEN

La investigación sobre scripts generados con Microsft PowerShell y la gestión grafica para administrar Windows Server 2008, permite seleccionar la gestión óptima en la academia Microsoft de la Escuela Superior Politécnica de Chimborazo.

El análisis comparativo se sustenta en método Científico, con técnicas de observación directa, y experimentación, mediante la implemetación de herramientas Software: Powershell 2.0, PowerGUI y monitor del Sistema.

La metodología de administración motivo de investigación, fueron integrados en Powershell y modo grafico, para automatizar sistemas servidores, permitiendo determinar cuál posee mejor rendimiento, usabilidad y portabilidad, dando como resultado cuantitativo, obtenido mediante comparación de Indicadores un 59% administración gráfica vs un 66% administración con PowerShell.

Los resultados denotados anteriormente, permiten concluir que la administración mediante scripts es ideal para una automatización clara y óptima obteniendo un 7% de efectividad con respecto a la gestión gráfica, a fin de desarrollar métodos de gestión cuando se realice la administración en la academia Microsoft de la Escuela Superior Politécnica de Chimborazo.

En conclusión, script de Microsoft PowerShell ofrece mejor rendimiento para administración de servidores Microsoft, superando el desempeño con gestión gráfica.

Se recomienda al administrador y estudiantes de la Academia Microsoft de la Escuela Superior Politécnica de Chimborazo, el uso de metodología script, que facilita la gestión en forma técnica, sencilla y rápida.

SUMMARY

The research on generated with PowerShell Microsft and graphics management to manage Windows Server 2008, it allows to select optimal management in the Microsoft Academy of the Escuela Superior Politecnica de Chimborazo.

The comparative analysis is based on scientific method with techniques of direct observation and experimentation through the implemetation of software tools: Powershell 2.0, and monitor the system.

The methodology of management of this reseach were integrated in Powershell and graphic mode to automate server systems, allowing to determine which one has better performance, usage, and portability, giving quantitative results by comparing indicators 59 % graphic management versus 66 % management with PowerShell.

The results allow to conclude that management through scripts is ideal for clear and optimum automation in 7% of effectiveness with respect to the graphical management in order to develop management methods in the Microsoft Academy of the Escuela Superior Politecnica de Chimborazo.

In conclusion the script PowerShell Microsoft provides better performance to manage Microsoft server, outperforming graphical management.

It is recomended to the manager and students of the Microsoft Academy of the Escuela Superior Politecnica de Chimborazo to use script methodology, to facilitate management in a formal, easy adn quickway.

GLOSARIO

Arquitectura: Representación abstracta de los componentes de un sistema y su comportamiento.

Controlador: Es el elemento más abstracto. Recibe, trata y responde los eventos enviados por el usuario o por la propia aplicación. Interactúa tanto con el modelo como con la vista.

Modelo: Es la representación de la información en el sistema. Trabaja junto a la vista para mostrar la información al usuario y es accedida por el controlador para añadir, eliminar, consultar o actualizar datos.

Objeto: Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

Patrón de diseño: Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

Persistencia: Capacidad de almacenar y recuperar el estado de los objetos, de forma que sobrevivan a los procesos que los manipulan.

Rendimiento: La velocidad de procesar la información y devolver el resultado al usuario. Esto es: qué tan rapido carga el sistema operativo, programas, páginas de internet, gráficos en 3d, crear o modificar datos como sonidos, video, texto, imagen, en el menor tiempo posible.

Sistema Operativo: Un sistema operativo (SO, frecuentemente OS, del inglés Operating System) es un programa o conjunto de programas que en un sistema informático gestiona los recursos de hardware y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes y anteriores próximos y viceversa

Usabilidad: efiere a la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto

Transportación: La finalidad es la obtención de un sistema de transporte eficiente, seguro, de acceso a todas las personas y ambientalmente amigable.

Tiempo de respuesta: Una señal de entrada del tipo escalón permite conocer la respuesta del sistema frente a cambios abruptos en su entrada

BIBLIOGRAFÍA

1. BARRENA, A., Administración avanzada de Windows Server 2008., Valencia - España., Politécnica de Valencia., 2010., Pp17-150.

2. ED, W., Windows PowerShell Scripting Guide., Washington - Estados Unidos de Norteamérica., Microsoft Press., 2008., Pp 1-583.

3. PAYATE,B., Windows PowerShell In Action., Washington- Estados Unidos de Norteamérica., Manning Publications Co., 2 007., Pp 1-519

4. PULGAR,G., Análisis comparativo de las tecnologías de virtualización hyper-v y virtual server 2005 R2 aplicada al centro Pearson Vue Riobamba., Tesis Ing. en Sistemas., Escuela Superior Politécnica de Chimborazo., Facultad de Informática y Electrónica., 2010., Pp 26-34

5. TOBIAS, W., Mastering PowerShell., Washington- Estados Unidos de Norteamérica., Microsoft Corporation., 2009., Pp 6-552.

6. ACTIVE DIRECTORY ADMINISTRATION WITH WINDOWS POWERSHELL

http://technet.microsoft.com/library/dd378937(WS.10).aspx

2011/02/01

7. WINDOWS SERVER 2008 – IMPLEMENTANDO LA INFRAESTRUCTURA – PARTE 1

http://www.aprendeinformaticaconmigo.com/windows-server-2008-implementando-la-infraestructura-parte-1

2012/04/05

8. WINDOWS SERVER 2008 – IMPLEMENTANDO LA INFRAESTRUCTURA – PARTE 2

http://www.aprendeinformaticaconmigo.com/windows-server-2008-implementando-la-infraestructura-parte-2

2012/04/05

9. WINDOWS SEVER2008 REDES GRUPOS LOCALES Y GLOBALES

http://redeselie.blogspot.com/2010/05/grupos-locales-de-dominio-los-grupos.html

2012/05/29

10. PAQUETE DE INSTALACIÓN LOCALIZADO PARA WINDOWS POWERSHELL 1.0 PARA WINDOWS XP

http://www.microsoft.com/es-es/download/details.aspx?id=9591

2012/09/13

11. WINDOWS POWERSHELL

http://technet.microsoft.com/es-es/library/bb978526.aspx

2013/05/21

12. SCRIPTING CON POWERSHELL

http://technet.microsoft.com/es-es/scriptcenter/powershell.aspx

2012/11/29

13. WINDOWS POWERSHELL

http://es.wikipedia.org/wiki/Windows_PowerShell

2013/11/22

14. INTRODUCTION TO MICROSOFT POWERSHELL – WHAT IS IT AND SETUP

http://www.darkoperator.com/blog/2012/3/28/introduction-to-microsoft-powershell-ndash-what-is-it-and-se.html

2012/03/27

15. POWERSHELL SCRIPTS, TIPS, EXPERT ADVICES, FORUMS, AND RESOURCES

http://powershell.com/cs/

2012/09/04

16. GUIA DE POWERSHELL 2.0

https://sites.google.com/site/ramiroencinas/

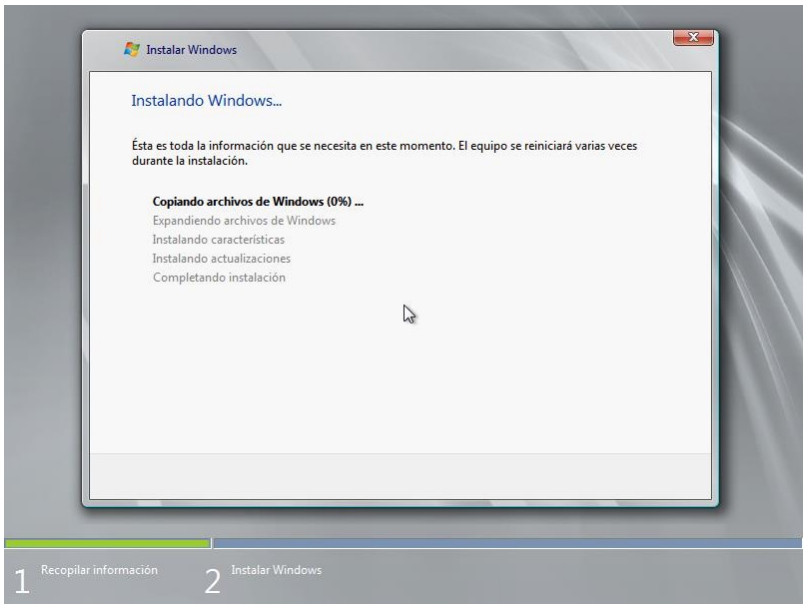
2012/12/12

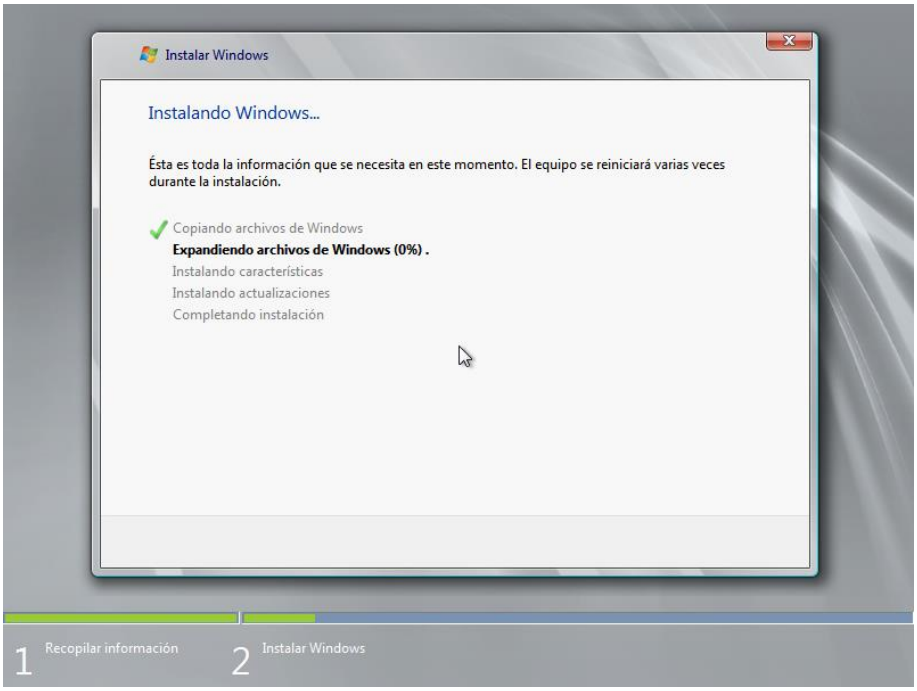
ANEXOS

INSTALACIÓN Y CONFIGURACIÓN DE WINDOWS SEVER 2008

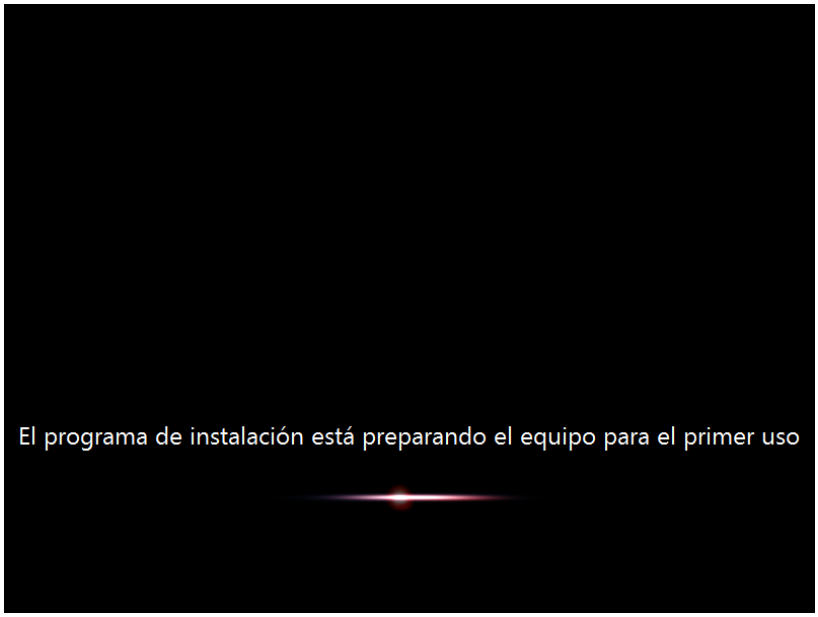
Para la instalación de Windows Server se utiliza en primera instancia un cd que contenga el sistema operativo.

1) Inserta el cd y pulsar enter



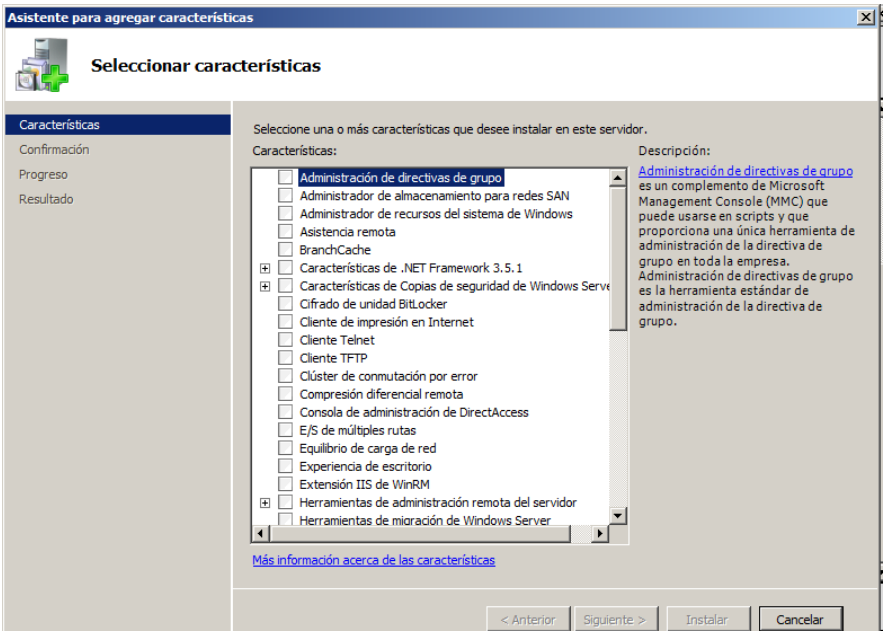


2) Una vez que concluye el proceso de instalación se debe esperar a que cargue todos los componentes del software.

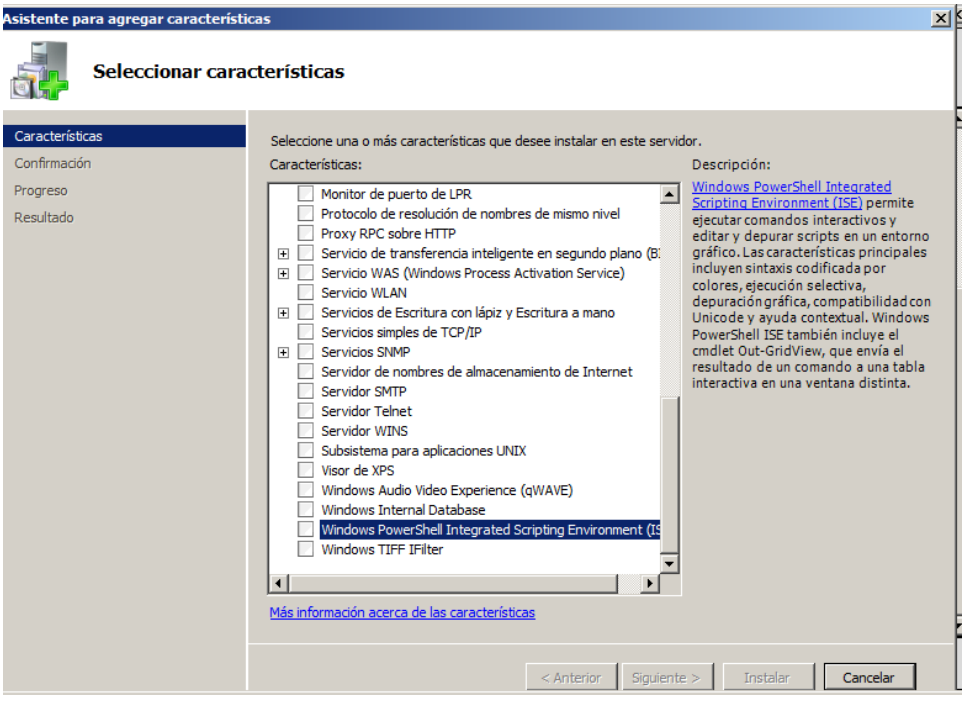


3) Ingresar con el login predeterminado que es el ADMINISTRADOR para realizar las primeras configuraciones.

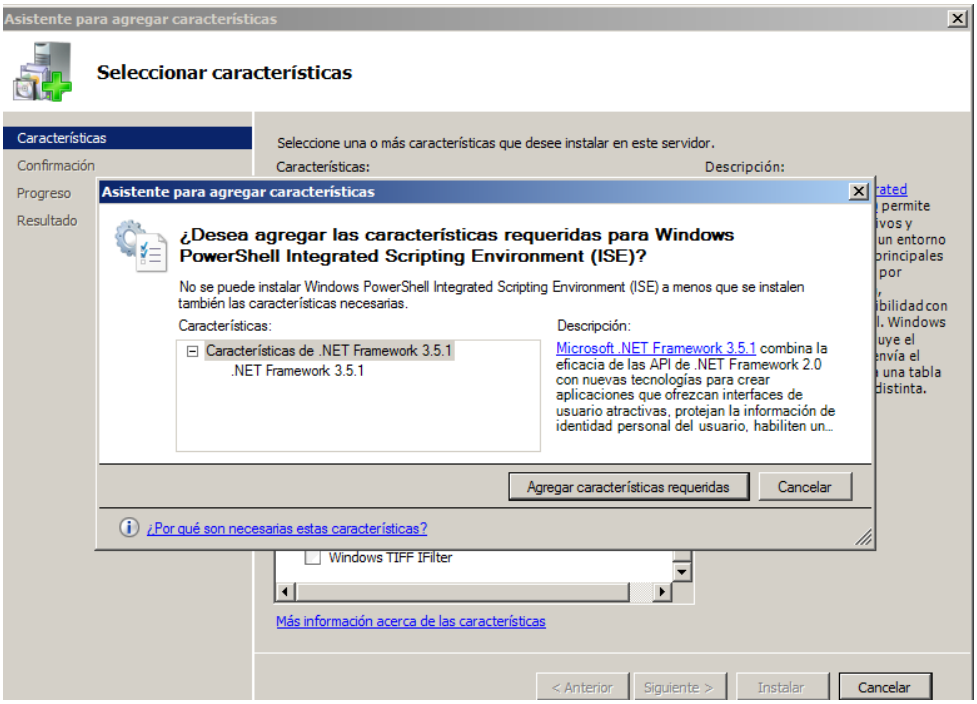
4) Después muestra la pantalla de configuración inicial y se agrega una nueva característica.



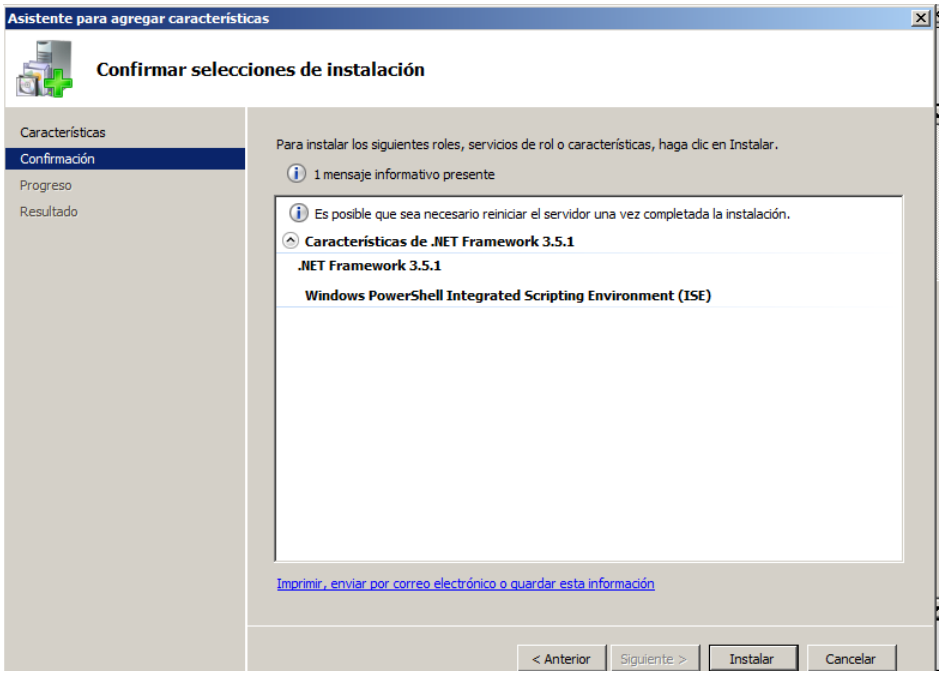
5) Se escoge la característica de PowerShell.

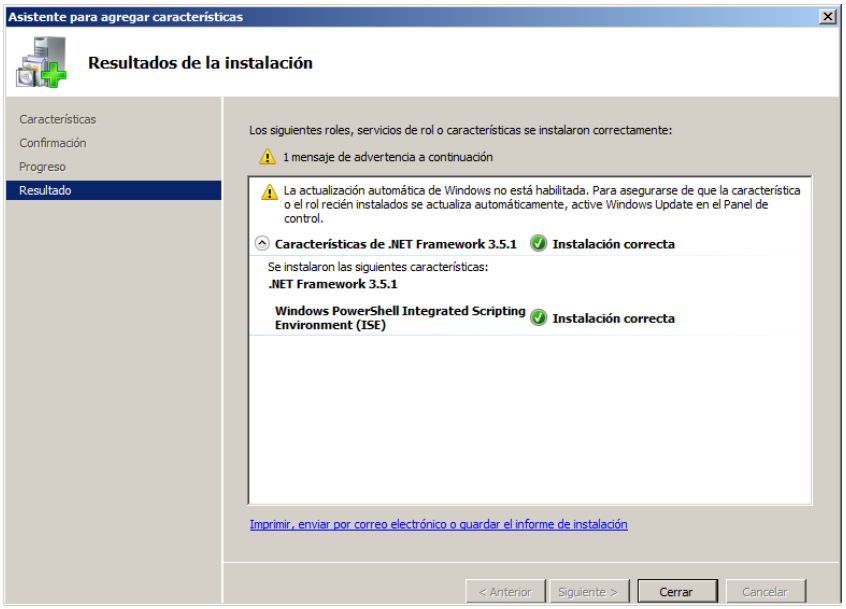


6) Pedirá que se instale .net framework 3.5.1



7) Si está conectado a Internet se procederá a descargar automáticamente.



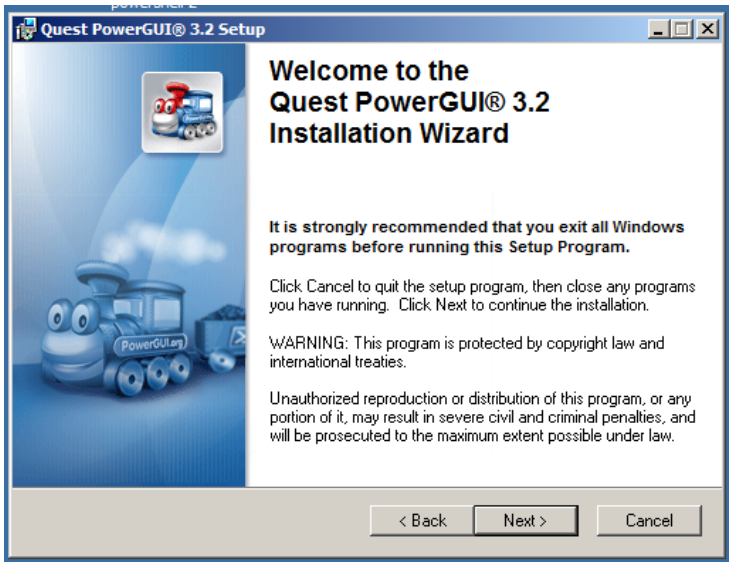


Se puede instalar una herramienta de ayuda para la codificación de los Scripts ya que facilita su mantenimiento y desarrollo.

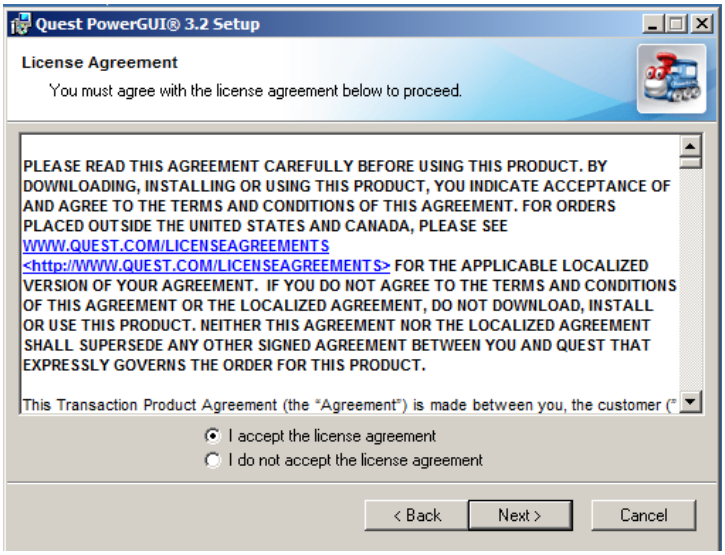
INSTALACIÓN DE PORWERGUI 3.2

Descargamos el programa de http://www.powergui.org/index.jspa cualquier versión actual que esté disponible y procedemos con la instalación.

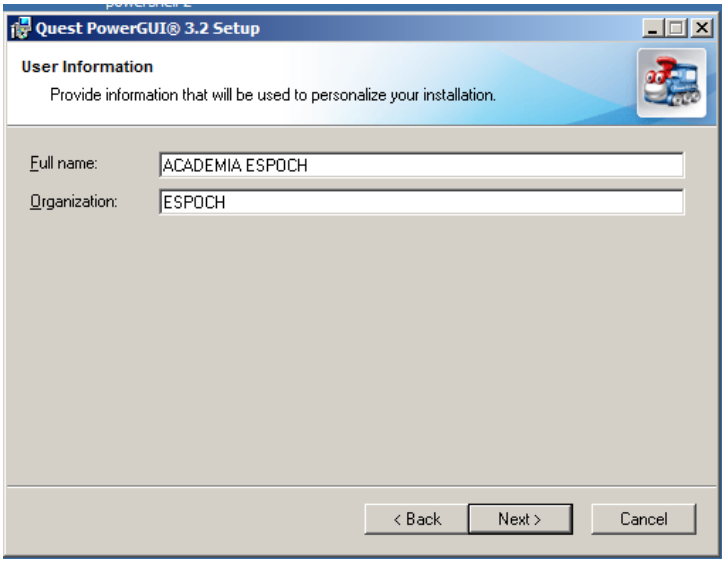
1) Ejecutamos los instaladores de PorwerGui 3.2.



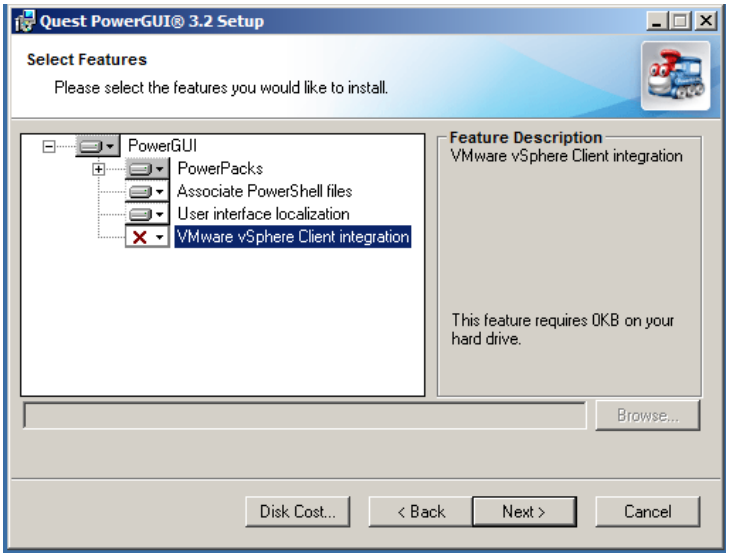
2) Se acepta la licencia



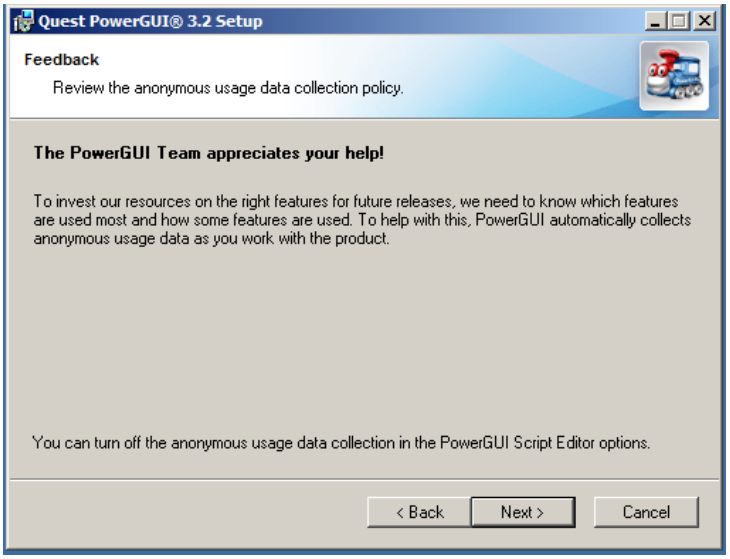
3) Escogemos el nombre y la organización

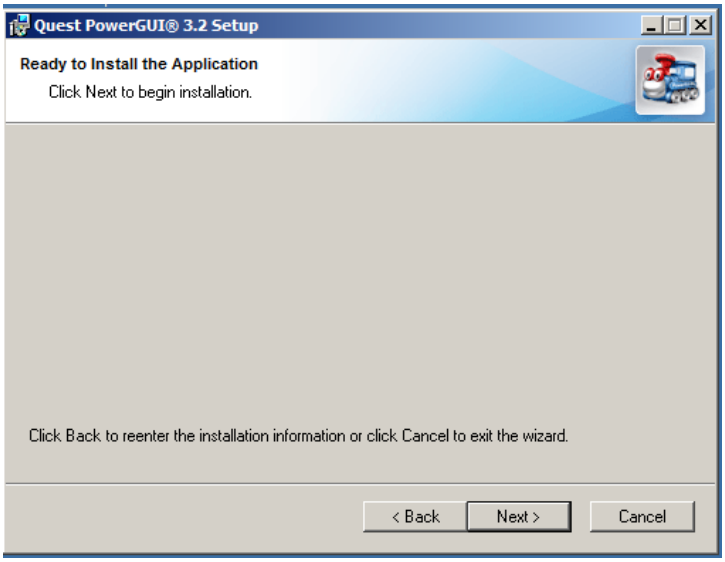


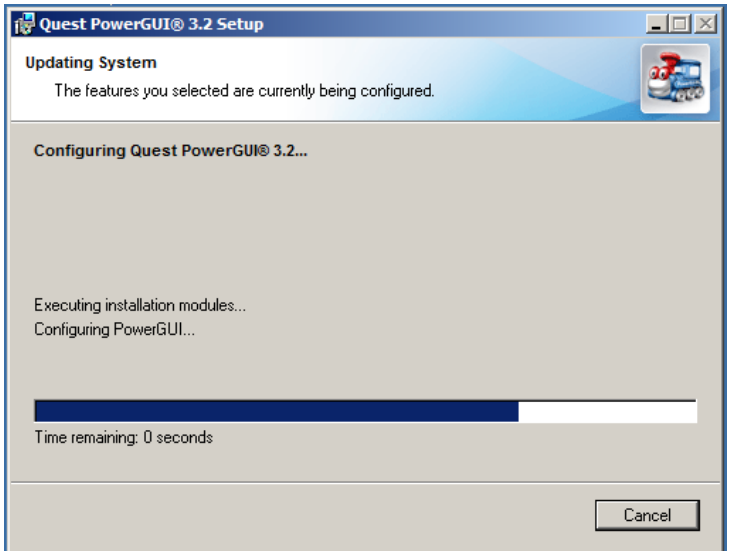
4) Se integra los paquetes restantes.



5) Y se procede con la instalación



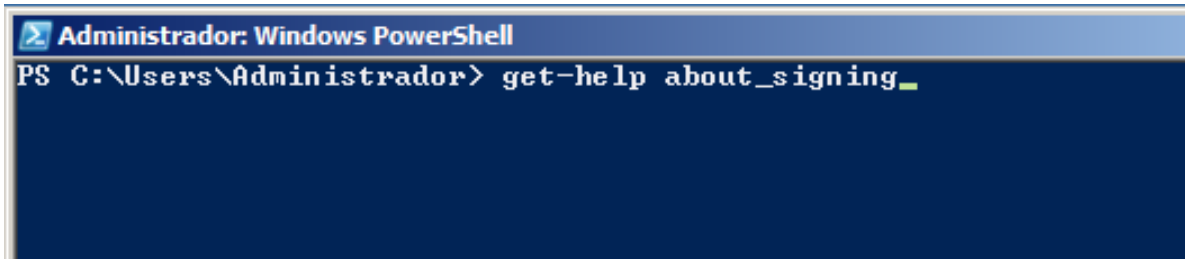


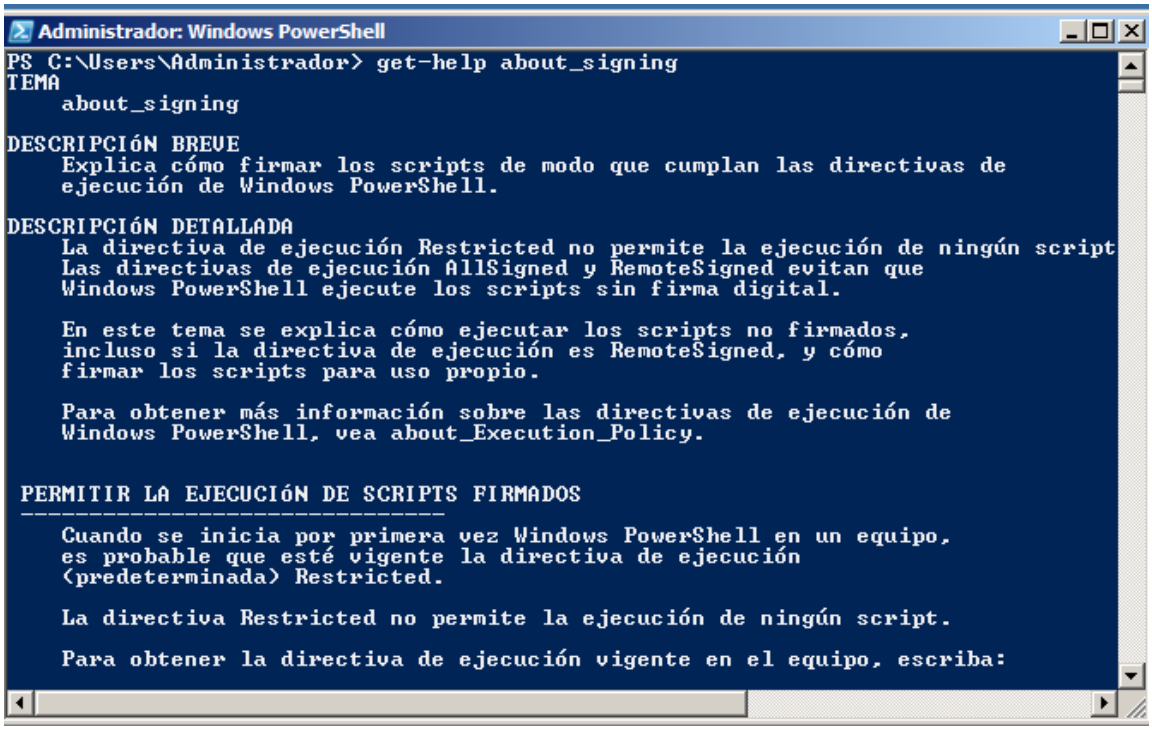


PROCEDIMIENTO PARA LA CONFIGURACION DE EJECUCION DE SCRIPTS

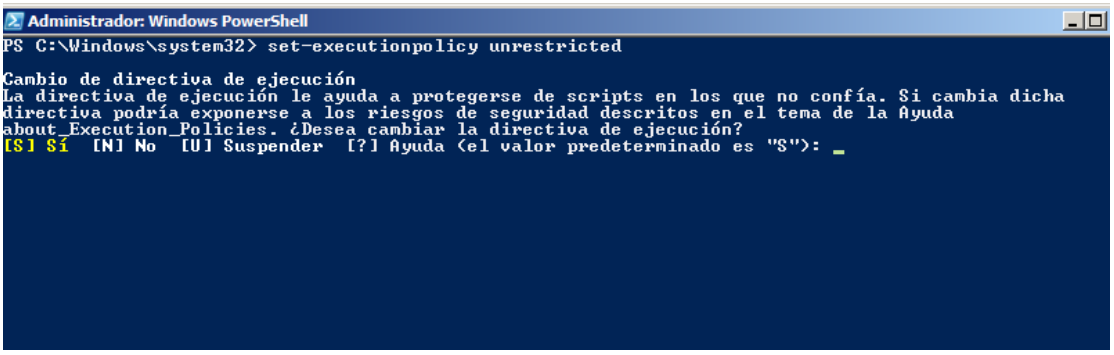
Para la activación de las políticas de ejecución de scripts se debe ejecutar lo siguiente:

1) Colocamos el comando a continuación se detalla para verificar el estado de la ejecución el cual muestra una ayuda para la configuración.

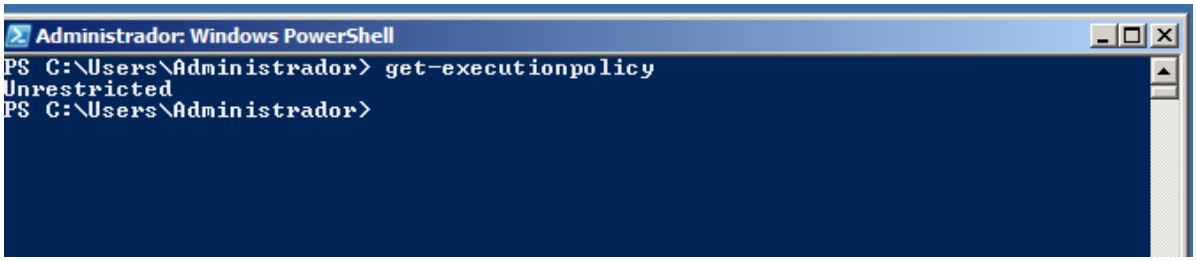




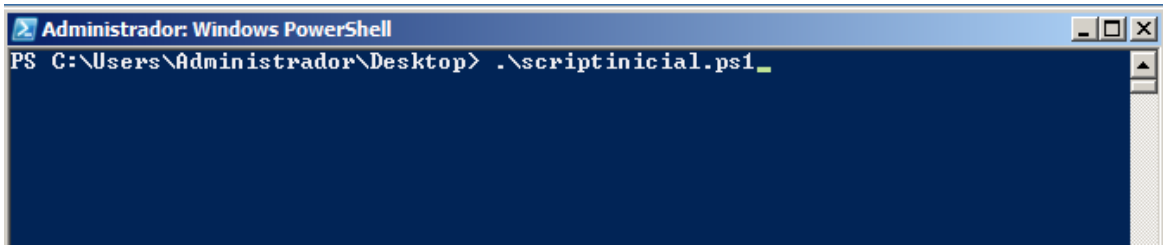
2) Se escoge que política se quiere utilizar y se pulsa si , en este caso se escoge Unrestricted (no restringida) y pulsamos la letra “S”



3) Verificamos si se ha ejecutado la política deseada



4) Para ejecutar un script tenemos que colocarnos en la carpeta donde estén respaldados nuestro script y colocar el nombre del script a ejecutarse anteponiendo .\ como muestra el siguiente gráfico.



O podemos utilizar la herramienta PorwerGui para su ejecución pulsando F5 o en icono de inicio de depuración