UNIVERSIDAD DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Herramienta para la automatización, seguimiento y graficación de un test de intrusión

Autor:
Eloy Pérez González

Directores:
Purificación Cariñena Amigo
Andrés Tarascó Acuña

Grado en Ingeniería Informática

Febrero 2017

Trabajo de Fin de Grado presentado en la Escola Técnica Superior de Enxeñaría de la Universidad de Santiago de Compostela para la obtención del Grado en Ingeniería Informática

Dña. Purificación Cariñena Amigo, profesora del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela, y **D. Andrés Tarascó Acuña**, Director en Tarlogic Security S.L.

INFORMAN:

Que la presente memoria, titulada *Herramienta para la automatización, seguimiento y graficación de un test de intrusión*, presentada por **D. Eloy Pérez González** para superar los créditos correspondientes al Trabajo de Fin de Grado de la titulación de Grado en Ingeniería Informática, se realizó bajo nuestra dirección en la empresa Tarlogic Security S.L.

Y para que así conste a los efectos oportunos, expiden el presente informe en Santiago de Compostela, a 7 de Febrero de 2017:

La directora,	El codirector,	El alumno,
Purificación Cariñena Amigo	Andrés Tarascó Acuña	Eloy Pérez González

Agradecimientos

A mis tutores, Purificación Carñena Amigo y Andrés Tarascó Acuña, por proponer y permitirme realizar un trabajo tan duro como estimulante, y guiarme y resolver mis dudas en todo momento.

A los profesores del Grado de Ingeniería Informática y a los investigadores del CITIUS, que tanto me han enseñado durante estos años.

A mi padre, por ser siempre mi ejemplo a seguir de trabajo duro y sacrificio, así como por enseñarme a pensar antes de actuar y animarme siempre a seguir aprendiendo.

A mi madre, por contagiarme su buen humor y haberme enseñado que la vida se mide en personas y no en monedas.

A mi hermano, por ser mi compañero en esos días que nos pasábamos jugando a la maquinita, en los cuales comencé a interesarme por la informática.

A mi hermana, para que encuentre su camino y nunca pierda su alegría.

A mi querido perrito, Trueno, que falleció durante la realización de este trabajo, y al cual le dedico una de las herramientas desarrolladas. Gracias por todo amigo y descansa en paz.

Al resto de mi familia, abuelos, tíos y primos, que siempre me han apoyado y ayudado.

A mis amigos del grado, por amenizarme las clases y alegrarme las incontables tardes de trabajo durante estos años. Y cómo no, por las risas en las noches compostelanas.

A mis amigos de la residencia, por todas esas comidas y cenas discutiendo sobre los temas más divertidos y surrealistas que uno pueda imaginar. Aquí os dejo el mejor TFG del mundo.

Al equipo de Tarlogic, tanto a los que están como a los que se han marchado, por ser más que compañeros de trabajo, por ser amigos, y un pozo sin fondo de conocimientos y buen humor.

Y por supuesto, a ella, por ser la estrella que ilumina mi vida.

A todos, muchísimas gracias.

¡Pasadlo bien!

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo inicial se muestra cual es el contexto que envuelve al proyecto, así como la motivación que ha llevado a la necesidad de la creación del producto y los objetivos que se esperan de este.

También se exponen en este capítulo un glosario de términos que se utilizarán a lo largo de la presentación, unas aclaraciones que permitan entender ciertos aspectos de la documentación y, finalmente, un listado de los capítulos en los que se divide el presente documento.

Secciones del capítulo:

1. **Contextualización del proyecto**
2. **Motivación del proyecto**
3. **Objetivos del proyecto**
4. **Glosario de términos**
5. **Aclaraciones generales**
6. **Organización del documento**

1.1. Contextualización del proyecto

Hoy en día es común para las empresas contar con multitud de sistemas que necesitan ser administrados en conjunto de una forma sencilla. Por ello lo habitual es establecer un servicio de directorio, en la infraestructura de la compañía, que almacene la información de los usuarios y equipos y permita administrar los privilegios de estos de forma centralizada.

Actualmente, el software de servicio de directorio más ampliamente utilizado es Active Directory de Microsoft, que permite gestionar los ordenadores y usuarios de una red de máquinas Windows.

Active Directory permite definir entidades funcionales denominadas dominios cuya información es almacenada y gestionada desde un servidor o servidores denominados Domain Controllers. La implantación de un servicio de Active Directory permite gestionar los privilegios que sobre cada recurso tengan los usuarios de dominio. Para ello, el enfoque más comúnmente utilizado es asignar los privilegios a los grupos del dominio, de forma que una vez se añade un usuario a un determinado grupo, este cuente con todos los permisos de los que dispone el grupo. Por otro lado, se debe tener en cuenta que también es posible para un grupo ser miembro de otro grupo, de manera que los usuarios heredan los privilegios de todos los grupos de los que son miembros, directa o transitivamente.

La asignación de privilegios permite definir que relaciones de confianza que se dan entre los usuarios y los diferentes recursos que se ofrecen en la infraestructura administrada por Active Directory, como servicios y estaciones de trabajo. Por tanto, los privilegios de un usuario permiten establecer en que máquinas del dominio, incluidos los Domain Controllers, puede este iniciar sesión, de manera interactiva o remota, y en cuales de ellas puede contar con permisos de administrador.

En consecuencia, en una red con un gran número de usuarios existen también un elevado número de relaciones de confianza, muchas de las cuales podrían ser aprovechadas por un atacante para acceder a los sistemas críticos de la red, como los Domain Controllers o servidores de bases de datos.

Por tanto para validar y evolucionar la seguridad de las redes de Active Directory, es común contratar a un auditor de seguridad para que realice un test de intrusión o pentest, en el cual el auditor simula ser un atacante. En estos test al auditor se le es dado por lo general un equipo de trabajo sobre el cual es administrador y se le pide que busque vías de elevación de privilegios dentro de la red y formas de acceder a información corporativa o a los sistemas críticos para la infraestructura.

1.2. Motivación del proyecto

Como se ha comentado en la sección anterior, un test de intrusión, o pentest (penetration test), es una simulación de un ataque sobre un sistema informático, en este caso una red de Active Directory, con objeto de exponer sus fallos de seguridad, para que puedan ser posteriormente solucionados.

A la hora de buscar realizar un pentest sobre una infraestructura de Active Directory, se trabaja con 2 activos principales. Por una lado se tienen los sistemas que se están testeando, entre los cuales el auditor se irá moviendo hasta llegar a un equipo crítico, y por otro lado se encuentran las credenciales de usuario,que permiten acceder a sus cuentas y disponer de sus privilegios, de modo que el auditor pueda aprovecharse de estos para realizar los saltos entre las máquinas.

Teniendo lo anterior en cuenta, en un pentest se realizan las siguientes acciones:

- **Identificación de sistemas**: En un pentest se deben identificar los equipos que componen el dominio o bosque de dominios que se está auditando.
- **Búsqueda de credenciales**: El auditor recurre a varios métodos para obtener credenciales de los usuarios del dominio. Una de las técnicas más empleadas es revisar la memoria de los equipos, ya que los sistemas Windows mantienen cacheadas las credenciales de los usuarios que tienen la sesión iniciada y estas pueden ser leídas si se cuenta con permisos de administrador en la máquina.
- **Comprobación de credenciales**: Una vez se han encontrado credenciales y se han identificado los equipos de la red, se comprueba si es posible acceder a estos utilizando las credenciales descubiertas.

Estas acciones se realizan de manera recursiva, es decir, para realizar la comprobación de credenciales es necesario previamente conseguirlas, así como descubrir equipos en los que probarlas para ver si se consigue obtener acceso. Y en caso de conseguir dicho acceso, se puede volver a realizar la búsqueda de credenciales en el nuevo equipo, para posteriormente volver a comprobar estas contra otras máquinas. Este proceso se repite hasta que se consigue acceder a los sistemas críticos de la infraestructura.

La motivación de este proyecto es la creación de un sistema que permita automatizar las tareas previamente listadas, de modo que se puedan localizar y explotar fallos de seguridad en una red de Active Directory, derivados de una mala

configuración de los privilegios de los usuarios, con objeto de tomar el control de la infraestructura, accediendo, para ello, a los sistemas críticos de la misma, en concreto los Domain Controllers.

Por otro lado se desea que el sistema guarde la información que se recolecte durante su ejecución, incluyendo las máquinas identificadas, las credenciales obtenidas y el método utilizado para realizar los saltos entre equipos, de modo que se pueda determinar los caminos, es decir la secuencia de saltos, que permiten acceder de manera más directa a los sistemas críticos desde cualquier otra máquina de la red.

Por último se requiere que el sistema permita visualizar un grafo que represente principalmente a los equipos y usuarios de los dominios y muestre sus relaciones, de forma que se visualice las relaciones de confianza existentes entre ellos que fueron aprovechadas por el sistema para tomar el control total de la red.

1.3. Objetivos del proyecto

El principal objetivo del proyecto será la creación de un sistema basado en agentes que permitan automatizar el proceso de descubrimiento de credenciales y escalada de privilegios en una red de dominio de Active Directory.

El sistema debe contar con las siguientes características:

- Debe realizar descubrimientos de nuevas máquinas en la infraestructura de Active Directory.
- Debe permitir recolectar credenciales almacenadas en las máquinas controladas.
- Debe comprobar las credenciales obtenidas en las máquinas identificadas y desplegar un agente en todas aquellas máquinas a las que se permita el acceso.
- Debe permitir controlar de forma remota las máquinas del dominio en las cuales se haya desplegado un agente.
- Debe almacenar en una base de datos la información recopilada sobre las máquinas identificadas, las credenciales de usuario obtenidas y los saltos realizados.
- Debe poseer una interfaz que pemita visualizar un grafo que represente en sus nodos las máquinas identificados, los usuarios a los cuales se han extraído credenciales y los saltos que se han realizado entre equipos. En el grafo se debe poder visualizar los caminos de saltos seguidos para acceder a las distintas máquinas de la red.

1.4. Glosario de términos

A continuación se presenta una lista de términos propios del proyecto con sus definiciones que se usarán a lo largo del presente documento, por lo que se recomienda su lectura antes de proceder con la del resto del documento.

- **Acción**: Cada uno de los trabajos solicitados por un usuario a un agente; se incluyen la ejecución de comandos y scripts y la subida y bajada de archivos.
- **ACP**: Se trata de las siglas de Protocolo Agente-Controlador (Agent-Controller Protocol). Este protocolo es el empleado por el subsistema Agente y el subsistema Controlador para comunicarse entre ellos.
- **Active Directory**: Active Directory o Directorio Activo es el servicio de directorios implementado por Microsoft para controlar de manera centralizada a los usuarios y máquinas de uno o varios dominios. Se entiende como red de Active Directory aquella red de equipos que están administrados por el servicio de directorios de Active Directory. Es decir que se encuentran en un bosque de dominios de Active Directory.
- **Agente**: En este documento se utilizará este término para referirse a 2 entes similares. El término agente o agentes se refiere a los procesos que se ejecutan en los ordenadores ajenos al del usuario y que se comunican con el controlador. Por otro lado el subsitema Agente, o Agente, es el que define el comportamiento que tendrán los agentes desplegados.
- **Bosque**: Conjunto de dominios donde se encuentra un dominio principal y dominios derivados de este o subdominios.
- **Controlador de agentes**: Subsistema que permite al usuario comunicarse con los agentes y que almacena en una base de datos la información transmitida por estos. Cuando se utilice el término Controlador en este documento, salvo que se exprese lo contrario, se estará haciendo referencia al Controlador de agentes.
- **Domain Controller**: El Controlador de dominio o Domain Controller es un servidor que almacena la base de datos de objetos de Active Directory y permite centralizar la administración de usuarios de una red de ordenadores.
- **Desplegar agente**: Instalar y ejecutar un nuevo agente en un ordenador.
- **Dominio**: Conjunto de ordenadores que mantienen una relación de confianza y que son administrados por un mismo servidor (Domain Controller) o conjunto de servidores.

- **Expansión**: Es el proceso por el cual los agentes se expanden por los ordenadores de una red de directorio activo, mediante los sucesivos despliegues de agentes en nuevas máquinas.
- **Salto**: Un salto entre máquinas es el despliegue de un nuevo agente, por parte del agente en la máquina origen del salto, en una máquina objetivo del salto a la que se tiene acceso.
- **Stager**: Pequeña parte del subsitema Agente que se ejecuta en una máquina ajena y que se encarga de solicitar y descargar del Controlador el código necesario para ejecutar un agente en el equipo.
- **Interfaz de datos**: Interfaz gráfica que ofrece la funcionalidad de consulta de la base de datos donde se almacenan los datos recogidos por los agentes. No se corresponde con la interfaz ofrecida por el Controlador.
- **Ordenador controlado**: Es todo aquel ordenador en el cual se haya desplegado un agente y por tanto en el que el auditor puede ejecutar acciones, a través de dicho agente.

1.5. Aclaraciones generales

En esta sección se muestran algunas aclaraciones que permiten entender el resto del documento.

En las secciones principales del documento todo el producto objetivo se representará como un solo sistema unificado, no obstante debido a que, durante la realización del proyecto se decidió dividir el producto en 2 herramientas diferenciadas y complementarias, se presentan separados en los anexos los manuales y licencias de ambas herramientas. Las herramientas obtenidas tras la realización del proyecto son las siguientes:

- **VanaX**: Compuesta por los subsistemas Controlador y Agente. Es la herramienta destinada a desplegar y controlar los agentes durante el proceso de expansión, así como a almacenar los datos recogidos por los agentes.
- **TrueniX**: Compuesta por el subsistema Interfaz de datos. Es la herramienta destinada a ofrecer una visualización gráfica de los datos recogidos durante el proceso de expansión.

1.6. Organización del documento

El presente documento se encuentra organizado en los siguientes capítulos:

- **Capítulo 2 - Análisis de requisitos**: Se presenta los casos de uso y requisitos que definen las necesidades que debe cubrir el sistema.
- **Capítulo 3 - Gestión del proyecto**: Se muestran los diferentes aspectos a gestionar del proyecto. El alcance, la metodología empleada, los riesgos, el tiempo y los costes.
- **Capítulo 4 - Diseño**: Se presenta el diseño del sistema y cada uno de sus componentes, indicándose también las tecnologías empleadas para la implementación.
- **Capítulo 5 - Pruebas**: Se describen las pruebas realizadas para llevar a cabo la aceptación del producto.
- **Capítulo 6 - Conclusiones y posibles ampliaciones**: Se exponen las conclusiones obtenidas durante el desarrollo del proyecto, así como futuras ampliaciones de funcionalidades que se podrían realizar.
- **Anexo A - Manuales técnicos**: Se presentan los manuales que describen las características de las herramientas que se deben conocer para mantenerlas y ampliarlas.
- **Anexo B - Manuales de usuario**: Se exponen los manuales de las herramientas que describen la instalación y el uso de las mismas.
- **Anexo C - Licencias**: Se presentan la licencias de las herramientas desarrolladas.

Capítulo 2

Análisis de requisitos

En el presente capítulo se exponen las características y funcionalidades que debe presentar el producto final.

Para ello se definen los casos de uso que describen la funcionalidad esperada y los requisitos que se deben satisfacer.

Secciones del capítulo:

1. **Casos de uso**
2. **Análisis de requisitos del software**

2.1. Casos de uso

Los casos de uso son una técnica ampliamente utilizada para ayudar en la definición de los requisitos de un proyecto, ya que permiten definir las necesidades de un sistema de una forma sencilla. En ellos se describe cómo se deben realizar las interacciones entre el sistema objetivo y los usuarios u otros sistemas ajenos de manera informal y simple, sin entrar en detalles técnicos, con lo que se consigue una representación del sistema comprensible, tanto por el cliente, como por el equipo de desarrollo.

2.1.1. Descripción de los actores

Como se ha comentado anteriormente, los casos de uso muestran las interacciones del sistema con agentes externos, que se denominan actores. Los actores representan a los usuarios o sistemas de terceros que interactuan con el sistema que se desarrollará en el proyecto.

Una buena definición de los actores que interactúan con el sistema facilita la comprensión del mismo y permite al equipo de desarrollo diseñarlo e implementarlo teniendo presente a que tipo de público se dirige.

A continuación se describen los actores que interactuán con el sistema:

Auditor de seguridad

Actor que interactúa con el Controlador de los agentes y la Interfaz de datos. Es un usuario con conocimientos técnicos.

Se debe apuntar que debido a la existencia de un solo actor, es decir, un solo usuario que utilice el sistema, a lo largo de esta memoria al utilizar el término usuario se estará haciendo referencia al auditor de seguridad que trabaja con el sistema.

2.1.2. Descripción de casos de uso

Durante las entrevistas realizadas con el cliente se establecieron los casos de uso requeridos por el sistema. Gracias a esto se pudieron identificar los diferentes

subsistemas encargados de la interacción con el usuario:

- **Controlador de agentes**: Subsistema capaz de otorgar al usuario el control de los agentes desplegados.
- **Interfaz de datos**: Subsistema que permite la visualización de manera gráfica de los datos recogidos por los agentes.

A continuación se muestra el formato seguido para la especificación de los casos de uso mostrados en las siguientes secciones:

Identificador - Nombre del caso de uso

Descripción: Descripción breve del caso de uso.

Actores: Actores que pueden llevar a cabo la interacción.

Casos de uso relacionados: Casos de uso que se relacionan con el actual.

Relevancia: La importancia que tiene el caso de uso:

- **Esencial**: Se requiere implementar el caso de uso para el correcto funcionamiento del sistema.
- **Deseado**: La no implementación del caso de uso no impide funcionar correctamente al sistema, pero añade mucho valor al mismo.
- **Opcional**: La implementación aporta valor al sistema pero es prescindible.

Precondiciones: Especifican el estado del que se supone que parte el sistema.

Postcondiciones: Especifican el estado en que se encuentra el sistema tras llevarse a cabo la correcta interacción.

Escenario Principal: Secuencia de acciones que se llevan a cabo en una interacción correcta.

Escenarios Alternativos: Acciones que se realizan en caso de errores en la secuencia principal.

Se debe apuntar que, en cualquier campo de este o siguientes formularios, se especificará N/A en caso de que no exista o no se consideré necesario especificar el valor.

Subsistema: Controlador de agentes

CU101 - Desplegar agente

Descripción: El Controlador debe generar el código Powershell que tras ejecutarse en la terminal Powershell de una máquina Windows instale en esta un agente.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene el código para desplegar un nuevo agente.

Escenario Principal:

1. Se ejecuta el comando de generación de código especificando la IP a la que se debe conectar el agente.
2. Se genera el código.

Escenarios Alternativos:

1. Se especifica mal la IP y el Controlador devuelve un mensaje de error.

CU102 - Listar agentes

Descripción: El Controlador muestra una lista de los agentes en ejecución, incluyendo los detalles de cada uno de ellos. Como mínimo se incluirán el identificador del agente y el nombre del ordenador controlado.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene la lista de agentes.

Escenario Principal:

1. Se ejecuta el comando para listar agentes.
2. Se muestra el listado con los agentes.

Escenarios Alternativos: N/A

CU103 - Seleccionar agente

Descripción: El Controlador debe permitir al usuario seleccionar un agente específico para interactuar con él.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene acceso al menú de acciones del agente.

Escenario Principal:

1. Se ejecuta el comando de selección de agente especificando el identificador del agente deseado.
2. Se accede al menú para interactuar con el agente.

Escenarios Alternativos:

1. Se especifica un identificador de agente incorrecto y el sistema devuelve un mensaje de error.

CU104 - Eliminar agente

Descripción: El Controlador debe permitir al usuario eliminar un agente específico.

Actores: Auditor de seguridad.

Casos de uso relacionados: Incluye CU103

Relevancia: Esencial

Precondiciones: El usuario se encuentra en el menú del agente que desea eliminar.

Postcondiciones: Se elimina el agente.

Escenario Principal:

1. Se ejecuta el comando de eliminación de agente.
2. El Controlador envía un mensaje de eliminación al agente.
3. El agente es eliminado.

Escenarios Alternativos: N/A

CU105 - Ejecutar comando en agente

Descripción: El Controlador debe permitir al usuario enviar comandos a un agente para que este los ejecute y devuelva la respuesta.

Actores: Auditor de seguridad.

Casos de uso relacionados: Incluye CU103

Relevancia: Esencial

Precondiciones: El usuario se encuentra en el menú del agente al que desea enviar comandos.

Postcondiciones: Se obtiene el resultado de los comandos ejecutados.

Escenario Principal:

1. Se ejecuta el comando de ejecución de comandos en agente especificando lo que se desea ejecutar.
2. Se envía un mensaje al agente con los comandos para ejecutar.
3. El agente ejecuta los comandos.
4. El agente devuelve el resultado de la ejecución.
5. Se muestran los resultados al usuario.

Escenarios Alternativos:

1. El agente trata de ejecutar un comando y se produce un error que se envía al Controlador.
2. Se muestra el error al usuario.

CU106 - Ejecutar script en agente

Descripción: El Controlador debe permitir al usuario enviar un script de Powershell a un agente para que este lo ejecute y devuelva la respuesta.

Actores: Auditor de seguridad.

Casos de uso relacionados: Incluye CU103

Relevancia: Deseado

Precondiciones: El usuario se encuentra en el menú del agente al que desea enviar el script.

Postcondiciones: Se obtiene el resultado del script ejecutado.

Escenario Principal:

1. Se ejecuta el comando de ejecución de script en agente especificando la ruta del script que se desea ejecutar.
2. Se envía un mensaje al agente con el contenido del script para ejecutar.
3. El agente ejecuta el script.

4. El agente devuelve el resultado de la ejecución.
5. Se muestran los resultados al usuario.

Escenarios Alternativos:

Escenario Alternativo **A**:

1. Se especifican una ruta de script errónea y el Controlador muestra un mensaje de error.

Escenario Alternativo **B**:

1. El agente trata de ejecutar un script y se produce un error que se envía al Controlador.
2. Se muestra el error al usuario.

CU107 - Descargar archivo

Descripción: El Controlador debe permitir al usuario descargar un archivo desde un ordenador controlado.

Actores: Auditor de seguridad.

Casos de uso relacionados: Incluye CU103

Relevancia: Deseado

Precondiciones: El usuario se encuentra en el menú del agente de cuyo ordenador quiere descargar el archivo.

Postcondiciones: Se guarda el archivo en el ordenador local.

Escenario Principal:

1. Se ejecuta el comando de descarga de archivo especificando la ruta del archivo que se desea descargar.

2. Se envía un mensaje al agente indicando la ruta del archivo.
3. El agente devuelve el archivo.
4. El Controlador almacena el archivo descargado e informa al usuario.

Escenarios Alternativos:

1. El agente no es capaz de encontrar el archivo en la ruta especificada y devuelve un error al Controlador.
2. Se muestra el error al usuario.

CU108 - Subir archivo

Descripción: El Controlador debe permitir al usuario subir un archivo a un ordenador controlado.

Actores: Auditor de seguridad.

Casos de uso relacionados: Incluye CU103

Relevancia: Deseado

Precondiciones: El usuario se encuentra en el menú del agente a cuyo ordenador quiere subir el archivo.

Postcondiciones: Se guarda el archivo en el ordenador donde se encuentra el agente.

Escenario Principal:

1. Se ejecuta el comando de subida de archivo especificando la ruta del archivo que se desea subir y la ruta donde se desea guardar en el ordenador remoto.
2. Se envía un mensaje al agente con el archivo y la ruta donde se desea guardar.
3. El agente devuelve un mensaje de éxito.
4. El Controlador informa al usuario de que se ha subido el archivo con éxito.

Escenarios Alternativos:

Escenario Alternativo **A**:

1. Se especifican una ruta local de archivo errónea y el Controlador muestra un mensaje de error.

Escenario Alternativo **B**:

1. El agente trata de almacenar el archivo en una ruta inválida y se produce un error que se envía al Controlador.
2. Se muestra el error al usuario.

CU109 - Listar credenciales

Descripción: El Controlador debe permitir al usuario listar las credenciales, esto es, las contraseñas obtenidas por los agentes durante el proceso de expansión.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene el listado de credenciales descubiertas.

Escenario Principal:

1. Se ejecuta el comando de listado de credenciales.
2. Se muestran las credenciales al usuario.

Escenarios Alternativos: N/A

CU110 - Listar IPs

Descripción: El Controlador debe mostrar las interfaces de red del equipo local y sus correspondientes IPs con objeto de facilitar al usuario la configurarión del código de instalación del agente.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Opcional

Precondiciones: N/A

Postcondiciones: Se muestran las interfaces de red e IPs del sistema.

Escenario Principal:

1. Se ejecuta el comando de listado de IPs.
2. Se muestran las interfaces de red e IPs al usuario.

Escenarios Alternativos: N/A

CU111 - Listar acciones de agente

Descripción: El Controlador debe mostrar la lista de acciones que el usuario ha solicitado a un agente, incluyendo los detalles necesarios para describir cada una de ellas.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: El usuario se encuentra en el menú del agente cuyas acciones desea listar.

Postcondiciones: Se muestran las acciones solicitadas al agente.

Escenario Principal:

1. Se ejecuta el comando de listado de acciones.
2. Se muestran las acciones del agente.

Escenarios Alternativos: N/A

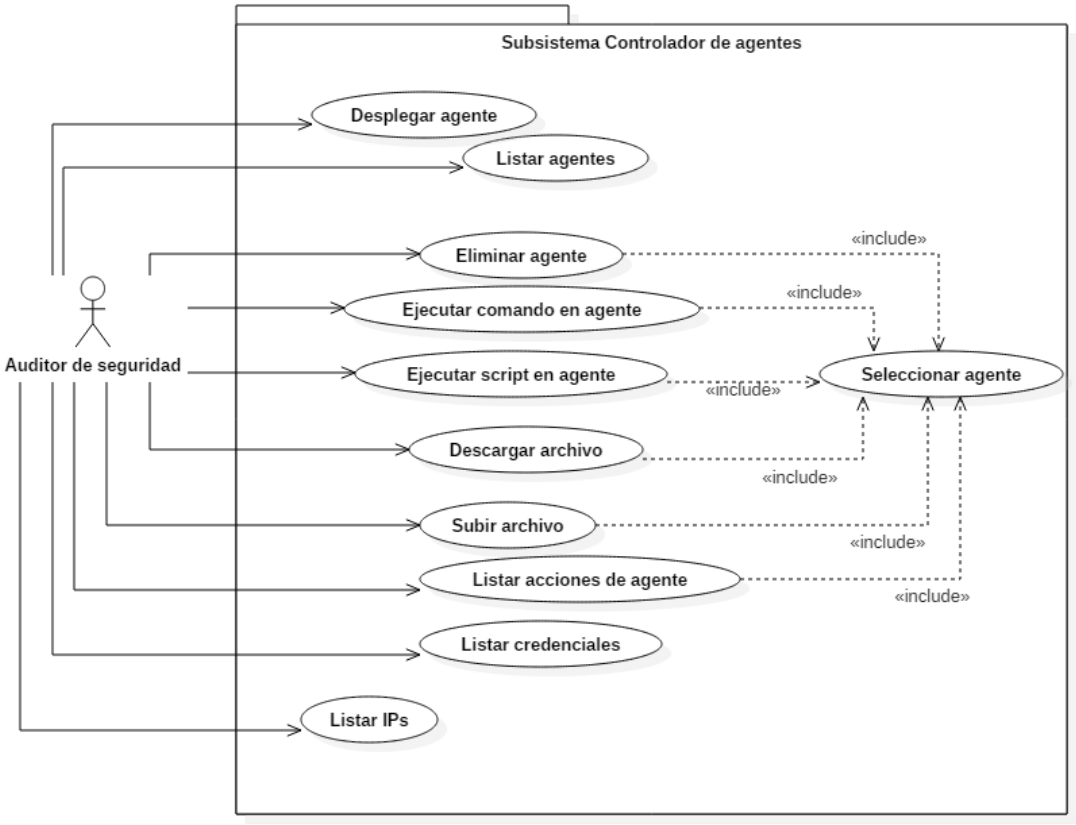

Figura 2.1: Casos de uso del subsistema *Controlador de agentes*

Subsistema: Interfaz de datos

CU201 - Realizar consulta personalizada

Descripción: La Interfaz de datos debe permitir al usuario realizar una consulta personalizada para obtener los datos que este desee, y mostrarlos en forma de grafo.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene una representación en forma de grafo de los datos consultados.

Escenario Principal:

1. El usuario inserta la consulta.
2. La interfaz sistema muestra el grafo con los datos seleccionados.

Escenarios Alternativos:

1. Se especifica una consulta errónea y se devuelve un error al usuario.

CU202 - Consultar agentes

Descripción: La Interfaz de datos debe permitir al usuario mostrar los agentes, los ordenadores controlados por estos y el dominio al que pertenecen dichas máquinas.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene una representación en forma de grafo de los agentes, ordenadores y dominios.

Escenario Principal:

1. El usuario selecciona la opción para ver los agentes.
2. La interfaz muestra el grafo con los agentes, sus ordenadores y los dominios.

Escenarios Alternativos: N/A

CU203 - Consultar credenciales

Descripción: La Interfaz de datos debe permitir al usuario obtener los usuarios cuyas credenciales hayan sido conseguidas a lo largo del proceso de expansión y el ordenador/dominio (según de trate de usuarios locales o de dominio) al que pertenezcan.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene una representación en forma de grafo de los usuarios y ordenadores/dominios.

Escenario Principal:

1. El usuario accede al menú de consultas.
2. El usuario selecciona la opción para ver las credenciales.
3. La interfaz muestra los usuarios cuyas credenciales se han obtenido, relacionándolos con sus respectivos ordenadores/dominios.

Escenarios Alternativos: N/A

CU204 - Consultar saltos entre ordenadores

Descripción: La Interfaz de datos debe permitir al usuario mostrar todos los ordenadores descubiertos del bosque y los saltos realizados entre estos por los agentes.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene una representación en forma de grafo del bosque, dominios, ordenadores y saltos.

Escenario Principal:

1. El usuario accede al menú de consultas.
2. El usuario selecciona la opción para ver los saltos entre ordenadores.
3. La interfaz muestra los ordenadores descubiertos y los saltos que los agentes han realizado entre dichas máquinas.

Escenarios Alternativos: N/A

CU205 - Consultar miembros de un grupo del dominio

Descripción: La Interfaz de datos debe permitir al usuario mostrar los miembros de un determinado grupo de dominio. Se entiende por miembros todos aquellos otros usuarios o grupos que pertenezca al grupo de manera directa o transitiva.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Deseado

Precondiciones: N/A

Postcondiciones: Se obtiene una representación en forma de grafo del grupo y sus miembros (usuarios/grupos).

Escenario Principal:

1. El usuario accede al menú de consultas.
2. El usuario selecciona la opción para ver los miembros de un grupo de dominio.
3. El usuario selecciona el grupo de dominio.
4. La interfaz muestra miembros del grupo.

Escenarios Alternativos: N/A

CU206 - Consultar usuarios de un dominio

Descripción: La Interfaz de datos debe permitir al usuario mostrar todos los usuarios descubiertos de un determinado dominio.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene una representación en forma de grafo del dominio y los usuarios.

Escenario Principal:

1. El usuario accede al menú de consultas.
2. El usuario selecciona la opción para los ver usuarios de dominio.
3. El usuario selecciona el dominio.
4. La interfaz muestra los usuarios del dominio seleccionado.

Escenarios Alternativos: N/A

CU207 - Consultar ordenadores de un dominio

Descripción: La Interfaz de datos debe permitir al usuario mostrar todos los ordenadores descubiertos de un determinado dominio.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: N/A

Postcondiciones: Se obtiene una representación en forma de grafo del dominio y los ordenadores.

Escenario Principal:

1. El usuario accede al menú de consultas.
2. El usuario selecciona la opción la opción para ver los ordenadores de un dominio.
3. El usuario selecciona el dominio.
4. La interfaz muestra los ordenadores del dominio seleccionado.

Escenarios Alternativos: N/A

CU208 - Consultar grupos de un dominio

Descripción: La Interfaz de datos debe permitir al usuario mostrar todos los grupos descubiertos de un determinado dominio.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Deseado

Precondiciones: N/A

Postcondiciones: Se obtiene una representación en forma de grafo del dominio y los grupos.

Escenario Principal:

1. El usuario accede al menú de consultas.
2. El usuario selecciona la opción para ver los grupos de un dominio.
3. El usuario selecciona el dominio.
4. La interfaz muestra los grupos del dominio seleccionado.

Escenarios Alternativos: N/A

CU209 - Consultar datos de agente

Descripción: La Interfaz de datos debe permitir al usuario mostrar los datos de un determinado agente.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: En pantalla se muestra un grafo con nodos de agentes.

Postcondiciones: Se obtienen los datos del agente.

Escenario Principal:

1. El usuario selecciona un nodo, del grafo, que representa a un agente.
2. La interfaz muestra los datos del agente seleccionado.

Escenarios Alternativos: N/A

CU210 - Consultar datos de ordenador

Descripción: La Interfaz de datos debe permitir al usuario mostrar los datos de un determinado ordenador.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: En pantalla se muestra un grafo con nodos de ordenadores.

Postcondiciones: Se obtienen los datos del ordenador.

Escenario Principal:

1. El usuario selecciona un nodo, del grafo, que representa a un ordenador.
2. La interfaz muestra los datos del ordenador seleccionado.

Escenarios Alternativos: N/A

CU211 - Consultar datos de usuario de dominio

Descripción: La Interfaz de datos debe permitir al auditor de seguridad mostrar los datos de un determinado usuario de dominio.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Esencial

Precondiciones: En pantalla se muestra un grafo con nodos de usuarios de dominio.

Postcondiciones: Se obtienen los datos del usuario.

Escenario Principal:

1. El usuario selecciona el nodo, del grafo, que representa a un usuario de dominio.
2. La interfaz muestra los datos del usuario de dominio seleccionado.

Escenarios Alternativos: N/A

CU212 - Buscar por nombre

Descripción: La Interfaz de datos debe permitir al usuario mostrar los datos de un determinado elemento (agente, ordenador o usuario del dominio) especificando su nombre.

Actores: Auditor de seguridad.

Casos de uso relacionados: N/A

Relevancia: Deseado

Precondiciones: N/A

Postcondiciones: Se obtienen los datos del elemento.

Escenario Principal:

1. El usuario introduce el nombre del elemento buscado.
2. La interfaz muestra el nodo del elemento con sus datos.

Escenarios Alternativos: N/A

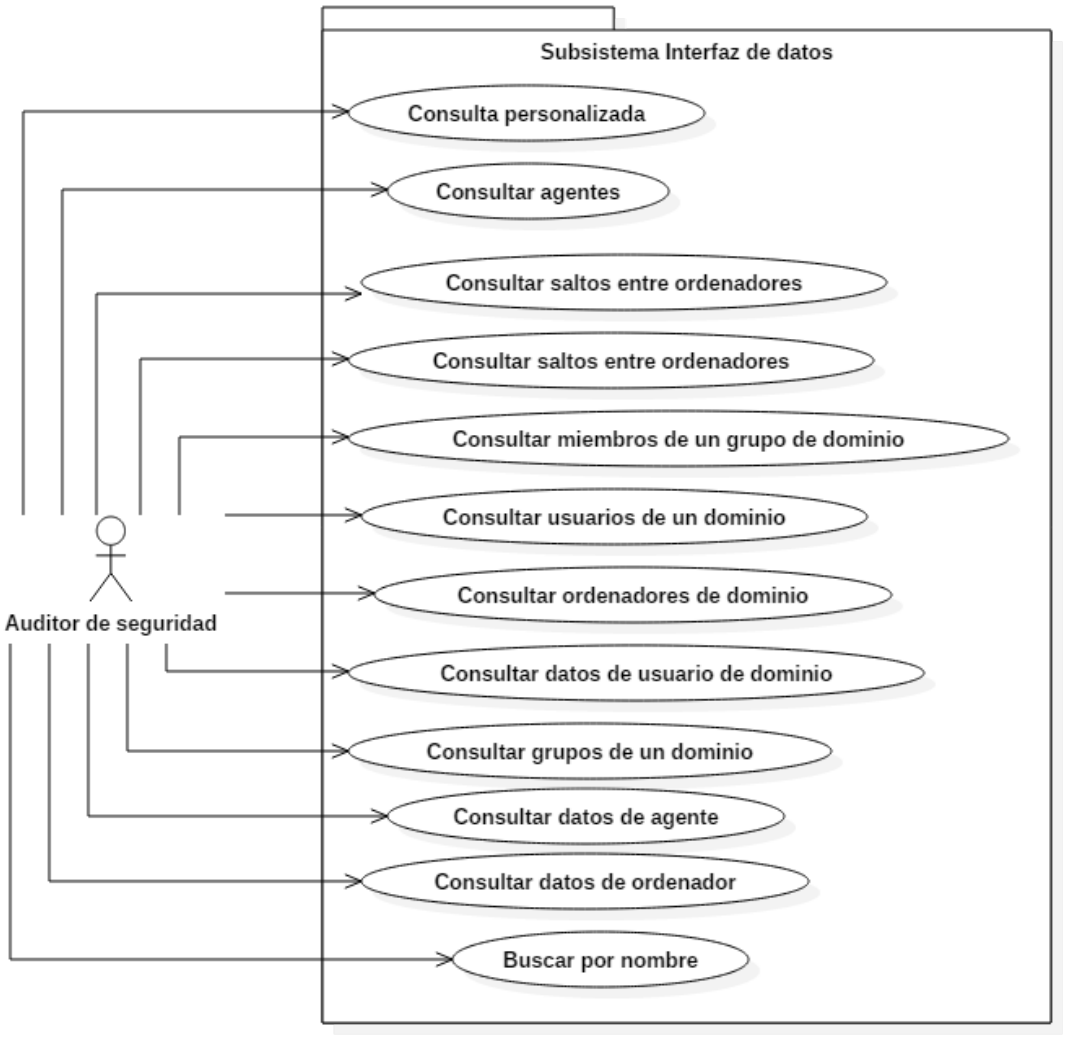

Figura 2.2: Casos de uso del subsistema *interfaz de datos*

2.2. Análisis de requisitos del software

La especificación de requisitos permite establecer las limitaciones, características y funcionalidades con las que debe contar el producto final de una forma completa e inequívoca permitiendo el correcto desarrollo del proyecto.

Los requisitos especificados a continuación se dividen en los siguientes tipos:

- **Restricciones de diseño**: Son limitaciones o condiciones que se imponen sobre el producto debido a las condiciones de aplicación del mismo.
- **Requisitos funcionales**: Referentes a las funcionalidades que debe implementar el producto. En este caso los requisitos funcionales vienen definidos por los casos de uso.
- **Requisitos de interfaz**: Definen las características con las que debe contar el producto para permitir a los usuarios o a otros sistemas mantener una interacción óptima con este.
- **Requisitos de calidad**: Especifican las exigencias que debe cumplir el producto para alcanzar un nivel de calidad adecuado.
- **Requisitos del proyecto**: Establecen una serie de condiciones que deben cumplirse en el desarrollo del proyecto.
- **Requisitos de evolución**: Indican que características o exigencias debe cumplir el producto para facilitar la ampliación de las características del mismo en un futuro.
- **Requisitos de seguridad**: Requisitos que establecen los aspectos de seguridad que debe cumplir un sistema.

Para definir cada requisito se seguirá el siguiente esquema:

Identificador - Nombre del requisito

Descripción: Descripción breve del requisito.

Relevancia: Cada requisito tiene uno de los siguientes niveles de relevancia:

- **Esencial**: Se requiere el requisito para conseguir el objetivo del proyecto.
- **Deseado**: El requisito mejora en gran medida la calidad del proyecto aunque no es estrictamente necesario implementarlo.

- **Opcional**: El requisito mejora ligeramente la calidad del proyecto pero no es en absoluto necesario.

Validación: Medida para comprobar que se ha cumplido el requisito.

2.2.1. Restricciones de diseño

En este apartado se especifican las restricciones de diseño, que imponen limitaciones sobre el producto debido a las condiciones de aplicación del mismo.

Las restricciones de diseño se identifican con la notación RDn, donde n es el número asociado a la restricción o requisito.

RD01 - Powershell versión 2

Descripción: Los agentes del sistema deben ser programados en el lenguaje Powershell de manera que se puedan ejecutar en la versión 2 o superiores de este lenguaje.

Relevancia: Esencial

Validación: Al ejecutar un agente en una terminal con Powershell versión 2 no se debe obtener ningún error relacionado con la interpretación del código.

RD02 - Independencia de módulos externos de Powershell

Descripción: Algunos módulos externos pueden ampliar las funcionalidades del lenguaje Powershell, sin embargo no existe ninguna garantía de que en los ordenadores controlados se encuentren instalados estos módulos, por ello las funcionalidades de los agentes no deben depender de módulos externos de Powershell.

Relevancia: Esencial

Validación: Ejecución sin errores de un agente en un ordenador sin módulos externos de Powershell instalados.

RD03 - Ejecución en memoria

Descripción: El software que deja archivos en disco es más fácilmente identificable mediante el software antivirus, por ello es deseable que la ejecución del agente se produzca completamente en memoria.

Relevancia: Deseado

Validación: Se debe ejecutar el agente en un ordenador comprobando que no escribe ningún archivo en el disco.

RD04 - Máximo de bytes por mensaje

Descripción: Es deseable que ninguno de los mensajes provenientes o dirigidos hacia el agente ocupe más de 1MB para evitar el consumo de un gran ancho de banda; si un mensaje supera 1MB debe fraccionarse en mensajes más pequeños que no superen este tamaño.

Relevancia: Deseado

Validación: Se deben estudiar las comunicaciones realizadas por el agente a la hora de transmitir una tasa de datos superior a 1MB y comprobar que los mensajes no superan ese tamaño.

RD05 - 1 agente por ordenador

Descripción: No se debe permitir la ejecución de más de 1 agente en un ordenador controlado ya que se podría producir un uso excesivo de recursos de la máquina.

Relevancia: Esencial

Validación: Ejecutar 2 agentes en un mismo ordenador y tras un minuto comprobar que solamente se mantiene en ejecución 1 agente.

RD06 - Comunicación mediante HTTP

Descripción: En muchas empresas existen cortafuegos que restringen las co-

nexiones entrantes y salientes entre la red interna e internet, sin embargo las conexiones salientes de la red al puerto 80 son normalmente permitidas ya que se trata del puerto usado por las aplicaciones web. Teniendo esto en cuenta los agentes deben conectarse por el puerto 80 al Controlador utilizando para ello el protocolo HTTP.

Relevancia: Esencial

Validación: Ejecutar un agente y comprobar que todas sus comunicaciones salientes van destinadas al puerto 80 y a la IP de la máquina donde se está ejecutando el Controlador, cumpliendo con la especificación del protocolo HTTP.

RD07 - Datos de expansión en base de datos

Descripción: El Controlador debe almacenar en una base de datos la información recogida por los agentes durante la expansión.

Relevancia: Deseado

Validación: Comprobar desde la base de datos que durante una expansión se va almacenando la información recogida por los agentes.

RD08 - Ausencia de base de datos

Descripción: El Controlador deberá ser operativo sin necesidad de hallarse comunicado con la base de datos donde se almacena la información de la expansión, aunque esto implique perder los datos de agentes y credenciales en caso de reinicio del Controlador.

Relevancia: Deseado

Validación: Comprobar que tras iniciarse el Controlador sin conexión a la base de datos es igualmente posible desplegar agentes.

RD09 - Autoelimininación de agente

Descripción: En caso de que un agente pierda la comunicación con el Controlador, tras 10 minutos debe terminar su propio proceso.

Relevancia: Esencial

Validación: Comprobar que un agente en ejecución se termina tras 10 minutos sin conexión con el Controlador.

RD10 - Propagar agentes

Descripción: Un agente debe ser capaz de desplegar nuevos agentes en equipos remotos.

Relevancia: Esencial

Validación: Verificar que un agente capaz de conectarse a un equipo remoto puede ejecutar el código necesario para desplegar un nuevo agente en dicho equipo.

RD11 - Obtener credenciales

Descripción: El agente debe poder extraer las contraseñas de los usuarios con sesión iniciada en la máquina en la que reside cuando cuenta con permisos de administrador en ella.

Relevancia: Esencial

Validación: Iniciar sesión con un usuario en una máquina donde resida un agente con sesión de administrador y comprobar que se envía dicha contraseña al Controlador.

RD12 - Detectar equipos en red de Active Directory

Descripción: El agente debe tener medios para identificar ordenadores que se encuentren en la red de Active Directory.

Relevancia: Esencial

Validación: Ejecutar un agente en un ordenador que se encuentre en la misma red de Active Directory que otros y comprobar que el agente es capaz de detectarlos.

RD13 - Obtener información básica del ordenador

Descripción: El agente debe poder recopilar información sobre el ordenador en el que se encuentra. En concreto debe recuperar la siguiente información:

- Nombre del equipo
- Dominio del equipo
- Sistema operativo del equipo

Relevancia: Esencial

Validación: Ejecutar un agente en un equipo y verificar que el Controlador almacena la anterior información del equipo en la base de datos.

RD14 - Obtener información adicional del ordenador

Descripción: El agente debe poder recopilar, además de la información básica, los siguientes datos el ordenador en el que se encuentra:

- Usuarios del equipo
- Grupos del equipo
- Interfaces de red del equipo incluyendo la MAC e IP asociadas a cada una

Relevancia: Deseado

Validación: Ejecutar un agente en un equipo y verificar que el Controlador almacena la anterior información del equipo en la base de datos.

RD15 - Obtener información básica de Active Directory

Descripción: El agente debe poder recopilar la siguiente información, a mayores de los equipos, del bosque de dominios de Active Directory donde se encuentra su ordenador controlado:

- Dominios
- Servidores de los dominios
- Grupos administradores de los dominios
- Usuarios administradores de los dominios

Relevancia: Esencial

Validación: Ejecutar un agente en un equipo de un dominio y verificar que el Controlador almacena la anterior información del equipo en la base de datos.

RD16 - Comprobar permisos de un agente

Descripción: El agente debe disponer de algún método para identificar los permisos de ejecución con los que cuenta, ya que algunas funcionalidades solo pueden ser ejecutadas si se cuenta con permisos de administrador en el ordenador controlado.

Relevancia: Esencial

Validación: Desplegar agente y verificar que informa al Controlador de los permisos que tiene.

RD17 - Sistema funcional en Windows

Descripción: El producto debe poder ejecutarse un sistema operativo Windows.

Relevancia: Esencial

Validación: Verificar que es posible ejecutar el producto en un sistema operativo Windows.

RD18 - Agente en un único fragmento

Descripción: El código del agente debe ser lo suficientemente pequeño como para no necesitar su fragmentación a la hora de enviarlo, esto es, debe ocupar menos de 1MB.

Relevancia: Deseado

Validación: Verificar que el archivo que contiene el código del agente ocupa menos de 1MB.

2.2.2. Requisitos funcionales

Los requisitos funcionales definen las funcionalidades que debe implementar el producto.Estos se identifican con la notación RFn, donde n es el número asociado al requisito.

Debido a que la descripción de los requisitos funcionales ya viene expresada en los casos de uso y para evitar una redundancia que podría crear problemas a lo largo del proyecto se ha decidido simplemente establecer como alias para cada caso de uso el formato anteriormente expresado para los requisitos funcionales, de modo que el requisito RFx sea una referencia al caso de uso CUx. Para poner un ejemplo, si en el presente documento se referencia el requisito RF3, para ver la especificación de dicho requisito se debe consultar el caso de uso CU3.

2.2.3. Requisitos de interfaz

En este apartado se especifican los requisitos de interfaz, que definen las características con las que debe contar el producto para permitir a los usuario o a otros sistemas mantener una interacción óptima con este.

Los requisitos de interfaz se identifican con la notación RIn, donde n es el número asociado al requisito.

RI01 - Consola de comandos en el Controlador

Descripción: La interfaz entre el Controlador de agentes y el usuario será una interfaz de línea de comandos ya que el perfil de usuario al que va dirigido el aplicativo está acostumbrado a tratar con este tipo de interfaz. Además esta

proporciona una mayor agilidad y flexibilidad a la hora de manejar varios agentes.

Relevancia: Esencial

Validación: Verificar que el Controlador ofrece una interfaz de línea de comandos para operar.

RI02 - Interfaz de datos en grafo

Descripción: La interfaz usada para representar la información de la base de datos deberá mostrar el conjunto de datos en forma de grafo, haciendo más simple la visualización de las relaciones entre los diferentes entes como pueden ser ordenadores, agentes o usuarios.

Relevancia: Esencial

Validación: Verificar que los datos se muestran en forma de grafo.

2.2.4. Requisitos de calidad

En este apartado se detallan los requisitos de calidad, que especifican las exigencias que debe cumplir el producto para alcanzar un nivel de calidad adecuado.

Los requisitos de calidad se identifican con la notación RCn, donde n es el número asociado al requisito.

RC01 - Ayuda en comandos

Descripción: Se debe incluir un comando de ayuda que liste los comandos disponibles en el Controlador de agentes, asimismo cada comando que se ejecute en el Controlador debe tener una opción de ayuda que explique en que consiste el comando y que parámetros acepta.

Relevancia: Deseado

Validación: Verificar que todos los comandos y menús del Controlador implementan una ayuda.

RC02 - Idioma inglés

Descripción: El idioma empleado en las interfaces del aplicativo debe ser el inglés.

Relevancia: Deseado

Validación: Verificar que el idioma utilizado en las interfaces es el inglés.

2.2.5. Requisitos del proyecto

En este apartado se especifican los requisitos del proyecto, que establecen una serie de condiciones que deben cumplirse en el desarrollo del mismo.

Los requisitos del proyecto se identifican con la notación RPn, donde n es el número asociado al requisito.

RP01 - Software libre

Descripción: El producto objetivo del proyecto deberá tener una licencia de software libre.

Relevancia: Esencial

Validación: Comprobar que en la documentación del producto se incluye la licencia de software libre.

2.2.6. Requisitos de evolución

En este apartado se especifican los requisitos que indican las características o exigencias que debe cumplir el producto para facilitar la ampliación de las características del mismo en un futuro.

Los requisitos del evolución se identifican con la notación REn, donde n es el número asociado al requisito.

RE01 - Protocolo de comunicación independiente del lenguaje

Descripción: Se debe establecer un protocolo de comunicación entre el agente y el Controlador que sea independiente del lenguaje de programación empleado en ellos, especialmente el del agente, con vistas a poder crear agentes, en un futuro, en otros lenguajes de programación que puedan comunicarse con el Controlador

sin necesidad de implementar otro protocolo.

Relevancia: Esencial

Validación: Comprobar que el protocolo de comunicación se puede implementar desde cualquier lenguaje de programación.

RE02 - Protocolo de comunicación extensible

Descripción: Se debe definir un protocolo de comunicación que permita en un futuro añadir nuevas funcionalidades como el cifrado de las comunicaciones sin tener que preocuparse del contenido de los mensajes.

Relevancia: Esencial

Validación: Verificar que el protocolo es flexible y permite extender sus funcionalidades.

2.2.7. Requisitos de seguridad

En este apartado se especifican los requisitos del proyecto que establecen los aspectos de seguridad que debe cumplir el sistema.

Los requisitos de seguridad se identifican con la notación RSn, donde n es el número asociado al requisito.

RS01 - Restricción a subredes del dominio

Descripción: Un comportamiento no deseado por parte de los agentes es que estos sean desplegados automáticamente, es decir, el nuevo agente es desplegado por otro agente, en ordenadores externos a la red del bosque del dominio donde se desplegó el primer agente.

Relevancia: Esencial

Validación: Desplegar agentes y verificar que no se despliegan en ningún ordenador externo al bosque de dominios.

RS02 - Contraseña de Agente

Descripción: El Controlador solamente debe aceptar peticiones de aquellos agentes que incluyan en dicha petición una contraseña válida, indicada por el usuario.

Relevancia: Esencial

Validación: Desplegar agentes con contraseña válida e inválida y verificar que solamente aquellos con contraseña válida son capaces de comunicarse con el Controlador.

Capítulo 3

Gestión del proyecto

Se presentan en el siguiente capítulo diferentes aspectos y elementos que ha sido necesario planificar, gestionar y revisar a lo largo del proyecto. Entre ellos el alcance, la metodología de ciclo de vida seguida, los riesgos, el tiempo, los costes y los elementos de la configuración.

Secciones del capítulo:

1. **Alcance del proyecto**
2. **Metodología de desarrollo**
3. **Gestión de riesgos**
4. **Planificación temporal**
5. **Gestión de la configuración**
6. **Gestión de costes**

3.1. Alcance del proyecto

La definición del alcance del proyecto permite establecer que es lo que se espera obtener tras la realización del proyecto. El alcance de un proyecto se debe tener en cuenta a la hora de realizar la planificación de un proyecto ya que permite establecer el límite entre lo que es necesario hacer en un proyecto y lo que no, evitando que se destine tiempo a tareas cuyo objetivo es la realización de acciones que se escapan de lo establecido para el proyecto e invirtiendo este tiempo en la ejecución de tareas que son relevantes para conseguir el producto final.

3.1.1. Definición del alcance

El producto final del proyecto será un sistema que permitirá desplegar agentes en máquinas Windows pertenecientes a una red de Active Directory y tener un control remoto sobre estos agentes, permitiendo al usuario, que se encontrará en la máquina donde está instalado el programa de control de los agentes, ejecutar comandos en aquellas máquinas donde estos hayan sido desplegados.

Por otro lado cada uno de estos agentes debe tener la capacidad de descubrir otros ordenadores dentro del mismo dominio y de obtener las credenciales almacenadas en la máquina huésped, todo ello de forma automática sin necesidad de que intervenga el usuario. Usando esta información deben ser capaces de desplegar nuevos agentes en otras máquinas del dominio para continuar la expansión.

Por último se debe almacenar en una base de datos toda la información relacionada con los ordenadores controlados por los agentes, las credenciales y los saltos realizados entre máquinas. Se debe además proveer al usuario de una interfaz que permita realizar consultas sobre la base de datos y que presente la información en formato de grafo mostrando las relaciones entre los equipos.

3.1.2. Criterios de aceptación

Para considerar el producto aceptable este debe cumplir con todos los requisitos previamente acordados cuya relevancia tenga la categoría Esencial, pasando para ello todas las pruebas definidas para comprobar que se han cumplido estos requisitos.

Sin embargo se espera también que el producto cumpla también todos aquellos requisitos de categoría Deseado que sea posible implementar dentro del tiempo de proyecto preestablecido, que en este caso son 410 horas.

3.1.3. Entregables del proyecto

Una vez terminado el proyecto, el cliente deberá haber recibido los siguientes 4 entregables:

- **Código fuente del software**: El código fuente del producto producido a lo largo del proyecto.
- **Memoria del trabajo**: El presente documento donde se recoge toda la información de la gestión del proyecto y diseño del producto.
- **Manual de usuario**: Un manual que explica como instalar y manejar el software para todo aquel que necesite usarlo.
- **Manual técnico**: Un manual donde se describe toda la información necesaria para mantener o ampliar las funcionalidades del producto.

3.1.4. Exclusiones del proyecto

Debido a que el proyecto debe contar con una licencia de software libre quedan excluidas de uso en este todas aquellas librerías cuyas restricciones de licencia no permita contar al producto con una licencia de software libre.

3.1.5. Restricciones del proyecto

En este proyecto existen las siguientes restricciones:

- El tiempo máximo de realización es de 410 horas.
- El lenguaje de programación para implementar los agentes debe ser Powershell.

3.2. Metodología de desarrollo

Una de las fases cruciales de un proyecto es la elección de una metodología de desarrollo o ciclo de vida que marcará como se organizará y desenvolverá el proyecto, conllevando una mala decisión posibles sobrecostes, retrasos e incluso la propia suspensión del proyecto.

Una vez conocidos los requisitos iniciales del cliente y definido el alcance del proyecto, se deben tener en cuenta estos datos para escoger una metodología de desarrollo que se ajuste a las necesidades y el entorno de trabajo y tecnológico en el que se va a desenvolver el producto final.

La primera decisión que se debe tomar a la hora de escoger una metodología es cual de las dos vertientes principales, predictivas o ágiles, se va a seguir:

- Las **metodologías predictivas** se basan en realizar una planificación y un diseño iniciales del proyecto y el producto, respectivamente, y seguir está línea durante el resto del proyecto. Esto es lo óptimo cuando los requisitos no varían o varían muy ligeramente a lo largo del proyecto, pero presenta problemas en caso de tener requisitos cambiantes ya que esto obliga a volver a la fase de diseño y planificación para adaptar el producto, produciendo una pérdida importante de tiempo.

- Por otro lado la premisa de las **metodologías ágiles** es que los requisitos cambian a lo largo del tiempo, por tanto no es posible establecer un diseño inicial definitivo y se suelen dividir en fases que incorporan sus subfases de diseño, desarrollo y pruebas. Además en este tipo de metodologías es esencial el contacto con el cliente para mantener al día los requisitos y necesidades del proyecto.

En este caso, se seguirá una metodología ágil por las siguientes razones:

- Se prevé que los requisitos iniciales cambien a lo largo del proyecto debido a que el campo de la seguridad informática es altamente variable.

- El personal no cuenta con demasiada experiencia en el área de conocimiento del proyecto, en concreto el de las tecnologías a utilizar.

- El cliente cuenta con bastante disponibilidad para responder a las dudas sobre los requisitos que se puedan dar a lo largo del proyecto.

3.2.1. Scrum

Entre las metodologías ágiles disponibles se ha elegido Scrum, que es una metodología ampliamente utilizada actualmente y que permite al proyecto ajustarse a las necesidades del cliente.

Scrum divide el proyecto en 3 fases diferenciadas:

- **Primera fase**: En esta fase se establecen los requisitos iniciales, se realiza una planificación inicial del proyecto y se realiza un diseño base de la arquitectura del software a desarrollar.
- **Segunda fase**: Esta es la fase en la cual se procede con el desarrollo del software. Esta fase se divide en sprints. Un sprint es cada uno de los ciclos incrementales por los que pasa el desarrollo del producto, en cada sprint se van añadiendo nuevas características a este de forma que al terminar el último sprint se obtenga el producto funcional final con todas sus características implementadas. Los sprints tienen una duración de entre 2 y 4 semanas, que viene fijada por el equipo de trabajo y se va ajustando a lo largo del proyecto siguiendo las necesidades del equipo de trabajo. Cada sprint tiene las siguientes etapas:
  1. **Planificación**: Se especifica y planifica el trabajo que se realizará en el sprint. Se identifican las tareas a realizar y se diseñan las pruebas que debe pasar el producto al final del sprint.
  2. **Ejecución**: Se llevan a cabo las tareas definidas en la planificación. Se toma nota del tiempo de realización de cada tarea y se compara con su estimación para tenerse en cuenta en ocasiones futuras.
  3. **Revisión**: Se revisa el producto creado junto con el cliente. Para esto se realizan las pruebas de aceptación y se escuchan las sugerencias, opiniones y peticiones que desee formular el cliente respecto del producto.
- **Tercera fase**: Se finaliza el proyecto, se completa la documentación del proyecto junto con los manuales técnicos y del usuario. Se realiza un revisión del proyecto y se deja constancia de las lecciones aprendidas.

3.3. Gestión de riesgos

Una de las principales tareas que se debe abordar en la gestión de un proyecto es la gestión de riesgos. Es imprescindible llevar a cabo la identificación de los eventos que pueden amenazar a los activos de un proyecto con objeto de anticiparse a ellos y reducir su probabilidad de aparición e impacto en la medida de lo posible, para evitar así el fracaso del proyecto.

La gestión de riesgos está compuesta por las siguientes cuatro etapas:

1. **Identificación de riesgos**: Fase inicial donde se identifican los posibles riesgos que pueden amenazar a los activos del proyecto.
2. **Análisis de riesgos**: Etapa en que se estudia la probabilidad de aparición y el impacto de cada uno de los riesgos identificados.
3. **Planificación de riesgos**: Se definen las estrategias o planes a seguir para evitar la aparición de los riesgos o minimizar su impacto.
4. **Supervisión de los riesgos**: En esta fase se lleva a cabo la revisión de los riegos especificados para el proyecto, con objeto de identificar riesgos cambiantes, la aparición de nuevos riesgos o las condiciones que disparan los planes de contingencia definidos para cada riesgo.

Los riesgos identificados en este proyecto se incluyen en una de las siguientes categorías:

- **Riesgos de gestión del proyecto**: Riesgos que se originan durante la etapa de gestión del proyecto.
- **Riesgos de disponibilidad**: Relacionados con la imposibilidad de disponer de un activo necesario para el avance del proyecto.
- **Riesgos de desarrollo**: Riesgos que se pueden manifestar en la etapa de implementación y desarrollo del producto.
- **Riesgos de personal**: Relacionados con el personal del proyecto.

Una vez identificados los riesgos y antes de iniciar la etapa de análisis de riesgos se hace necesario definir las métricas que se utilizarán para definir tanto la probabilidad de aparición del riesgo como el impacto que tendrán los mismos en caso de materializarse.

Se muestran a continuación las métricas empleadas para medir la probabilidad de aparición de un riesgo:

Muy baja: Menos de un 10 % de probabilidad de que el riesgo se manifieste.

Baja: Entre un 10 % y un 25 % de probabilidad de que el riesgo se manifieste.

Media: Entre un 25 % y un 50 % de probabilidad de que el riesgo se manifieste.

Alta: Entre un 50 % y un 75 % de probabilidad de que el riesgo se manifieste.

Muy alta: Más de un 75 % de probabilidad de que el riesgo se manifieste.

Seguidamente se muestran las métricas seguidas para clasificar el impacto de los riesgos:

Insignificante: Menos de un 5 % de repercusión en plazo/esfuerzo/coste.

Tolerable: Entre un 5 % e un 10 % de repercusión en plazo/esfuerzo/coste.

Serio: Entre un 10 % e un 20 % de repercusión en plazo/esfuerzo/coste.

Catastrófico: Más de un 20 % de repercusión en plazo/esfuerzo/coste.

Una vez se han establecido las métricas de probabilidad de aparición e impacto se pueden combinar para identificar el nivel de exposición de un riesgo, que nos permite conocer que riesgos son más peligrosos y por tanto merecen una mayor atención durante el proyecto.

Se muestran en la tabla 3.1 los niveles de exposición manejados en este proyecto.

		Probabilidad				
		Muy Alta	Alta	Media	Baja	Muy Baja
Impacto	Catastrófico	Muy Alta	Alta	Alta	Media	Media
	Serio	Alta	Alta	Media	Media	Baja
	Tolerable	Alta	Media	Media	Baja	Baja
	Insignificante	Media	Media	Baja	Baja	Muy Baja

Tabla 3.1: Nivel de exposición por probabilidad e impacto

Para especificar cada riesgo se seguirá el esquema siguiente:

Identificador - Nombre de riesgo

Descripción: Descripción breve del riesgo.

Probabilidad: La probabilidad de que el riesgo se manifieste:

- Muy baja
- Baja
- Media
- Alta
- Muy alta

Impacto: El impacto que tiene el riesgo sobre el proyecto:

- Insignificante
- Tolerable
- Serio
- Catastrófico

Exposición: El nivel de exposición en función de la probabilidad y el impacto según se muestra en la tabla 3.1.

Medidas de prevención: Medidas a aplicar para evitar la aparición del riesgo y minimizar el impacto.

Medidas de corrección: Medidas a aplicar tras la manifestación del riesgo.

3.3.1. Riesgos de gestión del proyecto

En este apartado se especifican los riesgos que pueden manifestarse a consecuencia de fallos durante la gestión del proyecto, debido, principalmente, a estimaciones erróneas sobre el mismo que se puedan llevar a cabo en esta fase.

RG01 - Alcance inadecuado

Descripción: Se especifican demasiadas funcionalidades con respecto del tiempo disponible para la realización del proyecto.

Probabilidad: Media

Impacto: Serio

Exposición: Media

Medidas de prevención: Invertir un extra de tiempo en la definición del alcance del proyecto e identificar la importancia de cada funcionalidad, diferenciando aquellas que son necesarias de las que pueden ser opcionales.

Medidas de corrección: Descartar las funcionalidades no esenciales que no han podido ser llevadas a cabo en el tiempo del proyecto.

RG02 - Incumplimiento de los plazos

Descripción: Debido a la falta de experiencia es posible que la estimación de los plazos de las actividades no sea realista, llevando consigo la necesidad de un aumento de tiempo no planificado necesario para realizar el proyecto.

Probabilidad: Media

Impacto: Serio

Exposición: Media

Medidas de prevención: Realizar una sobrestimación de tiempo en las tareas del proyecto.

Medidas de corrección: Incrementar la carga de trabajo.

RG03 - Especificación de requisitos errónea

Descripción: Se establecen unos requisitos que no satisfacen las necesidades del cliente.

Probabilidad: Alta

Impacto: Serio

Exposición: Alta

Medidas de prevención: Repasar los requisitos con el cliente una vez estos estén especificados y todavía no se haya comenzado la fase de diseño.

Medidas de corrección: Rediseñar el sistema en la medida en que el cambio en los requisitos lo requiera.

RG04 - Fallos en el diseño

Descripción: Un fallo en la etapa de diseño puede dificultar e incluso hacer inviables las tareas de implementación haciendo que sea necesario más invertir más tiempo del planeado en implementar o rediseñar el proyecto.

Probabilidad: Media

Impacto: Serio

Exposición: Media

Medidas de prevención: Aplicar un diseño modular que evite que los fallos de diseño se expandan más allá del módulo donde se encuentran. Una vez diseñado el software revisar el diseño completamente antes de proceder con la implementación.

Medidas de corrección: Estudiar como afecta el fallo a la implementación y rediseñar el área afectada en caso de ser necesario.

3.3.2. Riesgos de disponibilidad

Los riesgos de esta sección están relacionados con la posibilidad de que un activo no esté disponible en algún momento del proyecto, impidiendo el avance del mismo.

RG05 - Fallo en el material de trabajo

Descripción: Un fallo en el material de trabajo puede ocasionar que este quede inutilizado, impidiendo que el personal pueda trabajar.

Probabilidad: Baja

Impacto: Serio

Exposición: Media

Medidas de prevención: Comprobar que el material de trabajo se encuentra en perfecto estado al inicio del proyecto y repararlo o cambiarlo en caso de detectarse algún deterioro.

Medidas de corrección: Disponer temporalmente de otra herramienta de trabajo mientras se repara la original.

3.3.3. Riesgos de desarrollo

En esta sección se especifican los riesgos que pueden manifestarse durante las distintas etapas de desarrollo del proyecto.

RG06 - Pérdida de los ficheros del proyecto

Descripción: Un fallo en el dispositivo donde se almacena el código fuente del software o la documentación del trabajo puede suponer la pérdida de todo el trabajo realizado.

Probabilidad: Baja

Impacto: Catastrófico

Exposición: Media

Medidas de prevención: Mantener una copia del trabajo en un servicio de almacenamiento en la nube.

Medidas de corrección: Recuperar la copia más reciente del trabajo.

3.3.4. Riesgos de personal

A continuación se muestra la especificación de los riesgos que están relacionados con los problemas que pudieran afectar al personal del proyecto.

RG07 - Desconocimiento en las tecnologías

Descripción: La falta de experiencia con las tecnologías empleadas en el proyecto por parte del personal puede llevar a realizar estimaciones poco precisas en fase de planificación del proyecto haciendo que no se destine el tiempo necesario a la realización de algunas tareas pudiendo atrasar todo el proyecto.

Probabilidad: Media

Impacto: Serio

Exposición:Media

Medidas de prevención: Realizar una fase de formación inicial que permita conocer las tecnologías empleadas en el proyecto.

Medidas de corrección: Suprimir funcionalidades opcionales relacionadas con la tecnología desconocida que permita recuperar la pérdida de tiempo en el proyecto.

RG08 - Baja del personal del proyecto

Descripción: La pérdida temporal de recursos humanos durante el proyecto puede conllevar un aumento de tiempo en la realización del mismo.

Probabilidad: Media

Impacto: Tolerable

Exposición:Media

Medidas de prevención: Realizar una sobrestimación del tiempo del proyecto para compensar tiempo perdido por baja. Establecer un entorno de trabajo agradable y que evite problemas de salud.

Medidas de corrección: Aumentar la carga de trabajo para compensar la pérdida de tiempo.

3.4. Planificación temporal

Una de las tareas fundamentales al inicio de un proyecto es establecer que tareas se van a desarrollar en él y estimar el tiempo de duración de estas.

Al inicio de un proyecto es complicado realizar una estimación precisa sobre el tiempo que abordará la realización del mismo, y, aunque es bastante probable que la planificación inicial sufra cambios a lo largo del tiempo, se hace necesaria realizarla para establecer una línea base que nos permitirá anticipar la duración total del proyecto y el porcentaje que se le debería dedicar a cada uno de los grandes grupos de tareas dentro del tiempo total.

3.4.1. EDT: Estructura de descomposición del trabajo

La EDT permite visualizar de manera sencilla y jerárquica que tareas es necesario llevar a cabo para conseguir terminar con éxito el proyecto y obtener el producto deseado.

Se muestra la EDT de este proyecto en la figura 3.1.

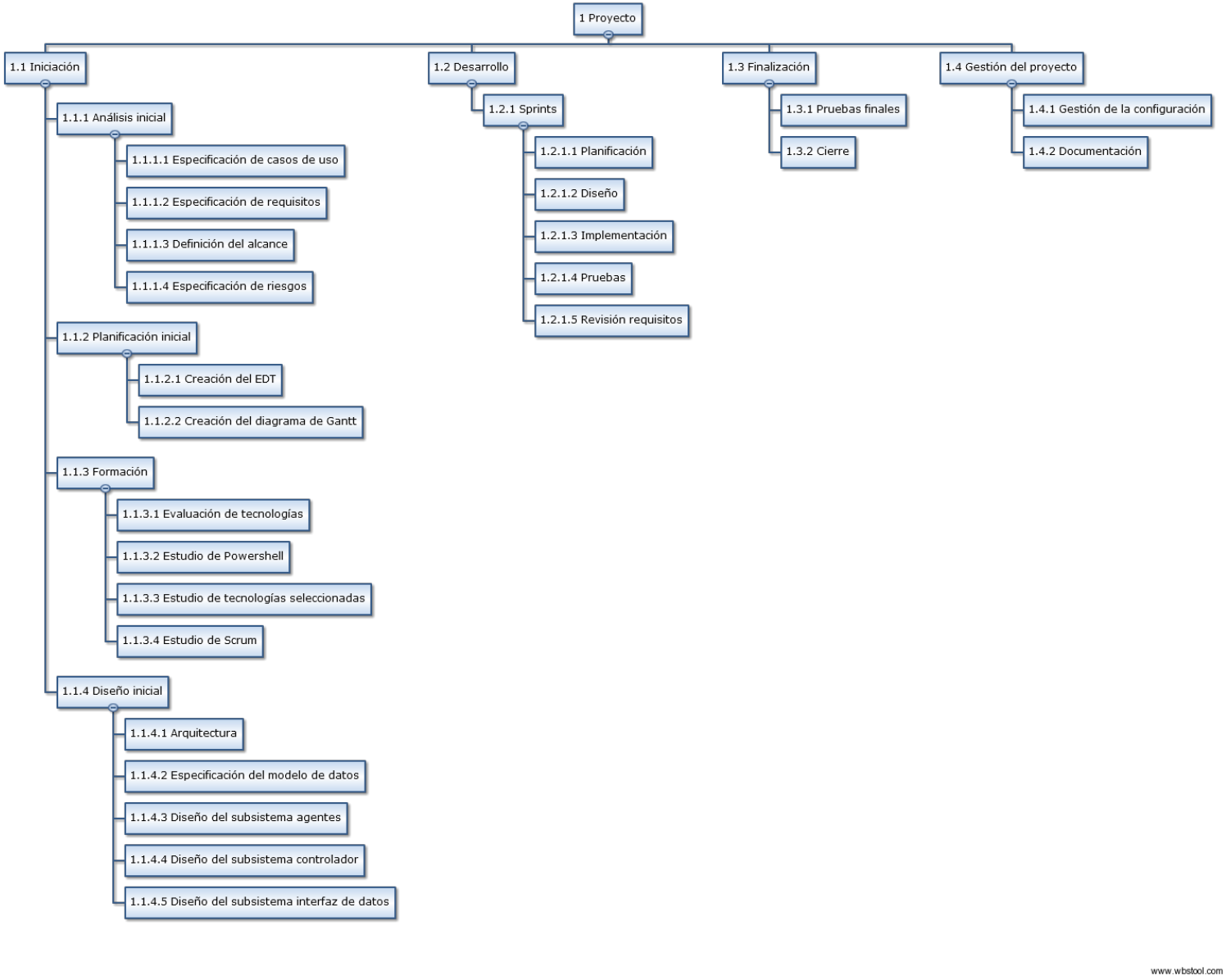

Figura 3.1: EDT del proyecto

3.4.2. Restricciones temporales

Para llevar a cabo la realización de la planificación de este proyecto se han tenido en cuenta las siguientes restricciones:

- La duración ideal del proyecto son 400 horas y el máximo permitido son 410 horas.
- El horario laboral del personal es de 4 horas al día, de 8:00 a 12:00, de lunes a viernes.
- El personal no trabajará en los siguientes períodos:
  - El mes de Agosto de 2016
  - La semana que comienza el 5 de Diciembre de 2016
  - Los siguientes días:
    + El 12 de Octubre de 2016
    + El 1 de Noviembre de 2016
    + El 26 de Diciembre de 2016
    + El 2 de Enero de 2017
- La metodología Scrum establece que el proceso de planificación de un sprint no debe abarcar más de una jornada laboral.
- Scrum indica que cada sprint debe tener una duración de entre 2 y 4 semanas.

3.4.3. Planificación

La planificación del proyecto se resume en el diagrama de Gantt mostrado a continuación, sin embargo se deben tener en cuenta los siguientes aspectos a la hora de visualizar el diagrama:

- En la planificación se pueden apreciar 2 roles distintos: Director del proyecto y Analista/Programador. Si se observa el diagrama de Gantt se puede ver que estos 2 roles no colaboran en ninguna tarea, es decir, no participan a la vez en ningún caso. Esto es debido a que como el equipo está compuesto solamente de una persona (que trabajaría con una carga del 100 %) que cumple ambos roles, en cada tarea, por claridad, solo se ha incluido el rol principal que se debe asumir, aunque existan ciertas tareas como puede ser la especificación de requisitos donde ambos roles colaborarían.

- Existen grupos de tareas en el Gantt donde varias tareas se podrían llevar a cabo simultáneamente en caso de disponer de un equipo de trabajo, como podrían ser las tareas de diseño inicial. Sin embargo debido a que el proyecto solo lo realiza una persona, esta solo podrá estar realizando una tarea al mismo tiempo, por ese motivo todas las tareas tiene la restricción de no poder empezar hasta que la tarea anterior termine.

- El grupo de *tareas transversales* está compuesto por aquellas tareas que no entran en ninguna fase específica ya que deben realizarse a lo largo de todo el proyecto. Como estas tareas deben ser realizadas continuamente y por todo el equipo de trabajo (o roles en este caso) no se le a asignado ningún recurso concreto en el Gantt por claridad.

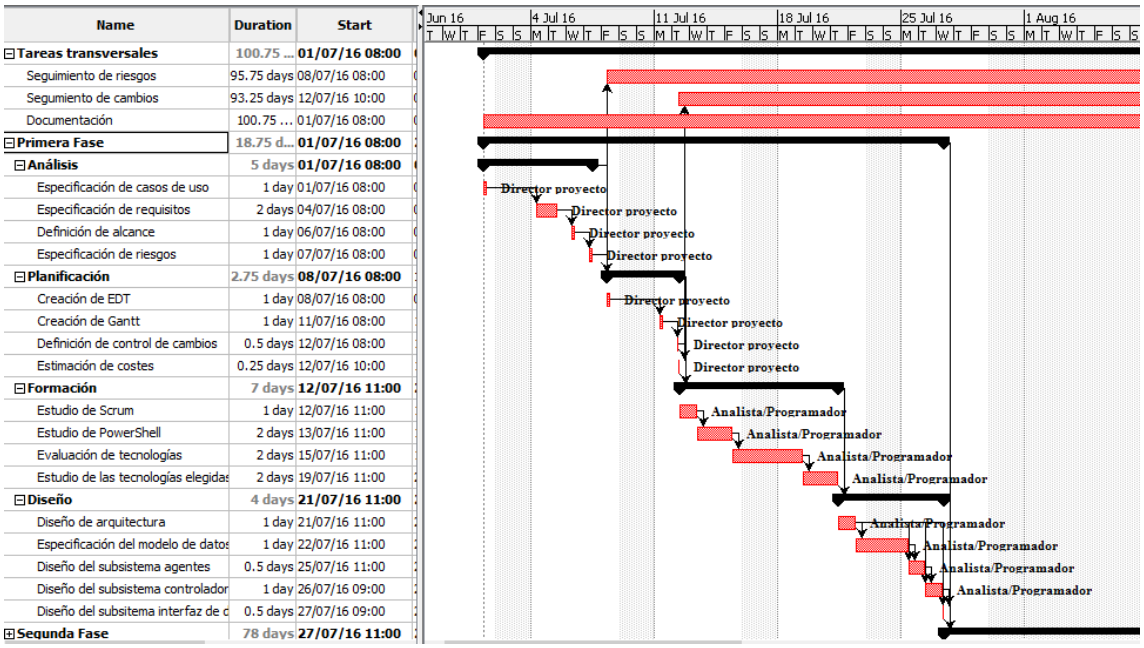

Figura 3.2: Gantt - Primera Fase e inicio de las tareas transversales

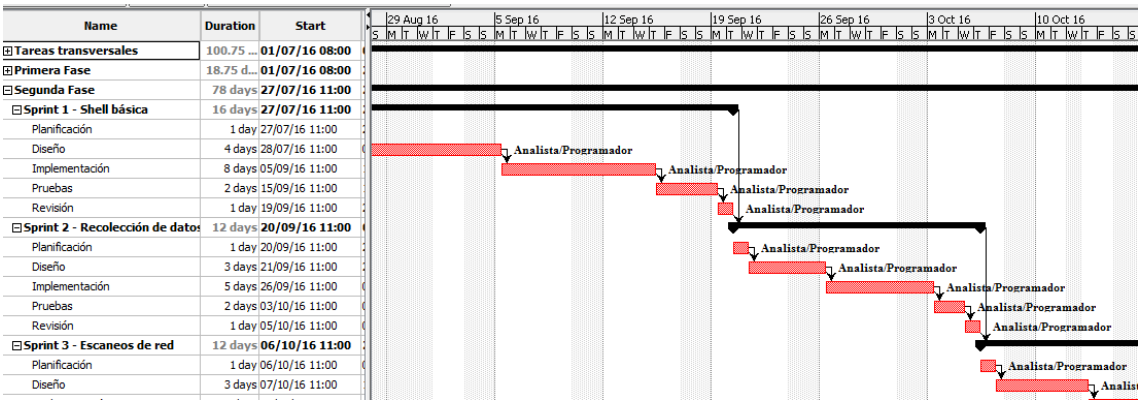

Figura 3.3: Gantt - Sprints 1 y 2 de la Segunda Fase

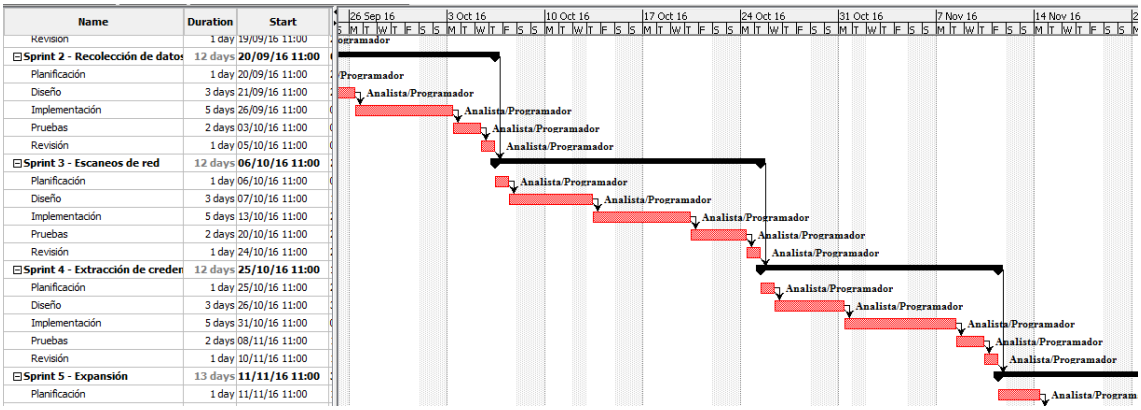

Figura 3.4: Gantt - Sprints 3 y 4 de la Segunda Fase

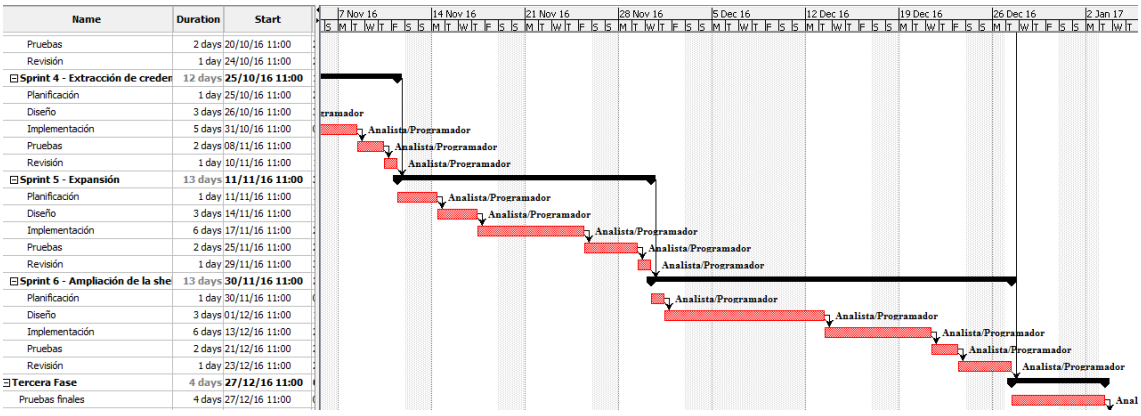

Figura 3.5: Gantt - Sprints 5 y 6 de la Segunda Fase

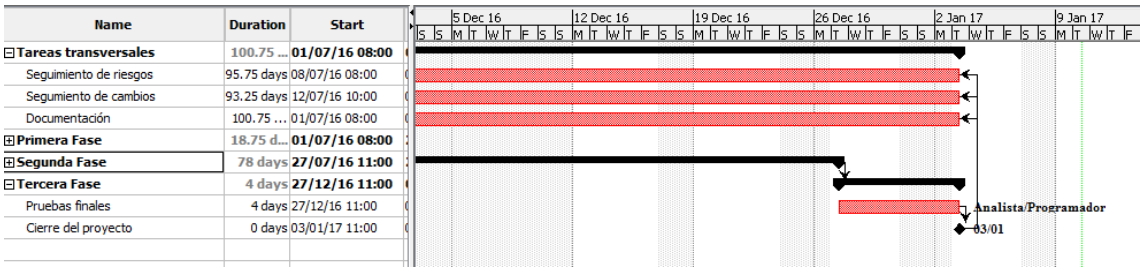

Figura 3.6: Gantt - Tercera Fase y terminación de las tareas transversales

Planificación de Sprints

A pesar de que el diagrama de Gantt ofrece un buen resumen de la mayoría de las tareas del proyecto, no queda demasiado claro que se ha hecho en cada uno de los sprints de la segunda fase, por lo que se describirán a continuación con algo más de detalle:

Sprint 1 - Shell básica

Descripción: En este primer sprint se trató de diseñar e implementar la estructura del agente y el protocolo de comunicación Agente-Controlador. Además se implementó la funcionalidad de ejecutar comandos en el ordenador controlado por el agente y de eliminar a un agente manualmente.

Requisitos relacionados: RD04 RD06 RF102 RF103 RF104 RF105 RE01 RE02

Sprint 2 - Recolección de datos

Descripción: En este sprint se añadió en el agente funciones para la recogida de datos del ordenador controlado y de Active Directory. Por otro se definió el modelo de datos para guardar esta información en la base de datos. Por último se implementaron las funcionalidades necesarias para visualizar esta información en la Interfaz de datos.

Requisitos relacionados: RD13 RD14 RD15 RF202 RF206 RF207

Sprint 3 - Escaneos de red

Descripción: El objetivo de este sprint fue desarrollar los algoritmos y funcionalidades que permiten a los agentes escanear la red de Active Directory en busca de nuevos equipos. Por otro lado se incluyó la funcionalidad para listar acciones de agentes en el Controlador. Finalmente se añadieron las restantes consultas de para la visualización de datos del bosque de dominio en la Interfaz de datos.

Requisitos relacionados: RD12 RF111 RS01 RF205 RF208 RF209 RF210 RF211

Sprint 4 - Extracción de credenciales

Descripción: En el siguiente sprint se implementaron las funcionalidades que permiten al agente extraer las credenciales de usuarios de los ordenadores controlados. Por otro lado se ha definido el modelo de datos para almacenar dicha información en la base de datos y se han implementado las funcionalidades necesarias en la Interfaz de datos para mostrar las credenciales descubiertas.

Requisitos relacionados: RD11 RF109 RF203

Sprint 5 - Expansión

Descripción: En este sprint se implementó en el agente la funcionalidad para verificar que credenciales tienen permisos de administrador en equipos remotos y desplegar en ellos nuevos agentes. Por otro lado se ha incorporado al modelo de datos los saltos entre ordenadores controlados, implementándose la visión de esta información en la Interfaz de datos. En última instancia se ha implementado la funcionalidad para producir el código de instalación de agentes en el Controlador.

Requisitos relacionados: RD05 RD10 RF101 RF204

Sprint 6 - Ampliación de la shell

Descripción: En este último incremento se incorporaron la posibilidad de ejecutar nuevas acciones en los agentes, como la ejecución de scripts y la subida y bajada de archivos. Por otro lado se ha habilitado la búsqueda personalizada y

por nombre desde la Interfaz de datos. Además se implementó en el Controlador la función para listar IPs

Requisitos relacionados: RF106 RF107 RF108 RF110 RF201 RF212

3.5. Gestión de la configuración

La gestión de la configuración es el proceso que se ocupa de mantener la integridad de los distintos elementos a lo largo del proyecto.

Es imprescindible evitar los cambios no controlados en los elementos de configuración del proyecto, como pueden ser el código fuente del software que se desarrolla o la documentación de gestión del proyecto, a fin de mantener estos consistentes los unos con los otros y con las necesidades del proyecto. Por ello, identificar los elementos de la configuración y definir como se controlarán los cambios en estos es una tarea de realización obligatoria en las primeras etapas de un proyecto.

3.5.1. Elementos de la configuración

El primer paso en la gestión de la configuración es identificar que elementos deben ser gestionados en el proyecto.

En el caso de este proyecto existen 2 elementos principales de los que debe controlarse la integridad:

- El código fuente del software a desarrollar.
- La documentación asociada al proyecto.

3.5.2. Gestión del código fuente

Para mantener el código fuente del proyecto controlado y documentado la mejor opción es optar por mantener este gestionado por un software de control de versiones, en este caso se ha optado por Git[3]. Además para proteger el código frente a posibles pérdidas el servidor de versiones que se utilizará estará alojado en la plataforma GitHub[2].

El modo de proceder para actualizar las versiones del código será el siguiente, cada vez que se cumpla una tarea o se añada una funcionalidad nueva al código se procederá a realizar un *commit* en local, también se hará así cuando se realice un cambio importante aunque no se hayan añadido funcionalidades o se repare un error substancial en el código. Además en cada *commit* se incluirá una descripción significativa del cambio que se ha llevado a cabo.

Por otro lado, para subir los cambios al servidor o repositorio remoto se realizará un *push* al final de cada jornada o en caso de que se realicen 2 *commits* desde el *push* anterior.

3.5.3. Gestión de la documentación

En el caso de la documentación del proyecto podría ser provechoso utilizar también para este elemento un software que controlase los cambios, esto es, indicase que miembros del personal los han realizado y se permitiese aceptar o declinar dichas modificaciones. Sin embargo al tratarse este de un proyecto desarrollado por una sola persona, y al ser esta la encargada tanto de realizar como de evaluar dichos cambios se supone más conveniente suprimir este proceso por comodidad y utilizar una herramienta como Dropbox[4] que actualice los cambios automáticamente y mantenga las versiones anteriores de los documentos para permitir revertir los cambios realizados en ellos en caso de encontrarse un error tras aplicarlos.

Como añadido se debe decir que Dropbox mantiene los documentos en un servidor remoto sincronizado con las carpetas locales de la máquina de trabajo, evitando así que en caso de pérdida de los datos locales no se perdiesen las modificaciones realizadas en la documentación.

3.6. Gestión de costes

En esta sección se presenta la gestión de costes, que permite determinar los costes que conlleva la realización del proyecto. Para ello es necesario revisar distintos factores, como el personal y los recursos software y hardware que se necesitan para desarrollar el producto.

3.6.1. Recursos humanos

En el proyecto actual solamente se cuenta con un trabajador que desempeñará diferentes roles en el proyecto, contando cada uno de estos roles con un sueldo diferente. Los sueldos medios para los diferentes roles/empleos en la provincia de A Coruña, según los datos recogidos en los portales web InfoJobs[7], Tecnoempleo[8] y Experteer[6] son los siguientes:

- **Director de proyecto**: 31000€ al año / 15,50€ por hora.
- **Analista/Programador**: 21000€al año / 10,50€ por hora.

De acuerdo con la planificación el trabajador realiza 31 horas de trabajo como *Director de proyecto* y 370 como *Analista/Programador*. Si se realizan los cálculos se obtienen los siguientes costes:

- **Director de proyecto**: 15,50€/hora * 31 horas = 480,50€
- **Analista/Programador**: 10,50€/hora * 370 horas = 3885€

3.6.2. Recursos hardware

Para la realización de este proyecto se necesita un ordenador con unos requisitos altos de potencia en términos de memoria RAM, en concreto 16GB, debido a que se necesita simular con máquinas virtuales de Windows un dominio de Active Directory para realizar pruebas de expansión con el software. Teniendo en cuenta esto el precio del ordenador se eleva apróximadamente a los 1100€ según la media de precios observada en PCComponentes[5].

No obstante el ordenador podrá ser reutilizado para la realización de otros trabajos, por tanto solamente se aplica un porcentaje del precio total como costes del proyecto.

Si se tiene en cuenta que la vida útil de un ordenador de trabajo suele ser de 3 años y este proyecto dura, contando los períodos no laborables, 6 meses, se asumen como costes 185€.

3.6.3. Recursos software

Debido a la necesidad de almacenar el software del proyecto en un repositorio privado de Github, se debe pagar una cuenta con un costo de 6,65€[2] al mes.

Por otro lado para desarrollar el controlador de los agentes se utilizará el IDE de programación WebStorm, cuya licencia tiene un precio de 129€[1] al año que se debe pagar íntegra aunque la duración del proyecto no se prolongue tanto.

3.6.4. Resumen de costes

Teniendo en cuenta los costes mencionados y que la duración del proyecto se han calculado a continuación los costes del mismo:

Concepto	Cantidad	Coste unitario	Coste total
Director de proyecto	31 horas	15,50€	480,50€
Analista/Programador	370 horas	10,50€	3885€
Ordenador	1 unidad	185€	185€
Cuenta GitHub	7 meses	6,65€	46,55€
Licencia WebStorm	1 año	129€	129€
Total			4726,05€

Tabla 3.2: Resumen de costes

Capítulo 4

Diseño

Se aborda en este capítulo el diseño del sistema, para lo que se empieza mostrando la arquitectura general del sistema. En ella se muestran los distintos subsistemas que se irán detallando en las secciones posteriores del capítulo.

Secciones del capítulo:

1. **Arquitectura**
2. **Modelo de la base de datos**
3. **Subsistema Agente**
4. **Protocolo de comunicación Agente-Controlador**
5. **Subsistema Controlador**
6. **Subsistema Interfaz de datos**

4.1. Arquitectura

La primera tarea que se debe abordar en la fase de diseño de un software es identificar los componentes principales que lo conforman. A estos componentes se los conoce como subsistemas.

En la figura 4.1 se muestra el diagrama de arquitectura del sistema que identifica sus principales subsistemas. Estos se listan a continuación:

- **Agente**: El subsistema Agente es aquel que se ejecuta en los ordenadores remotos recopilando información y credenciales en estos y tratando de encontrar nuevas máquinas para continuar la expansión por la red del dominio, transmitiendo al Controlador la información obtenida en este proceso. A su vez el subsistema Agente debe proporcionar una consola de comandos o shell para que el usuario ejecute los comandos de Powershell que desee en los ordenadores controlados. Al hablar de agentes en este capítulo se estará haciendo referencia a los procesos que implementan al subsistema Agente en los ordenadores remotos.
- **Controlador**: El subsistema Controlador es el encargado de ir almacenando en la base de datos toda aquella información recogida por los agentes referente a máquinas, credenciales y saltos. Además permite al auditor de seguridad comunicarse con estos.
- **Interfaz de datos**: El subsistema Interfaz de datos tiene la misión de leer la información almacenada en la base de datos y presentarla al usuario en un formato comprensible a través de grafos.

Adicionalmente a los subsistemas citados en el diagrama existen dos elementos que permiten interoperar a los componentes entre sí para ofrecer la funcionalidad completa del producto y cuyo diseño también se abordará en el presente documento.

- **Base de datos**: La base de datos permite almacenar la información extraída por los agentes para ser utilizada por la Interfaz de datos y conceder al usuario una visión gráfica del proceso de expansión. En secciones posteriores se mostrará el modelo de datos utilizado para almacenar la información en dicha base de datos.

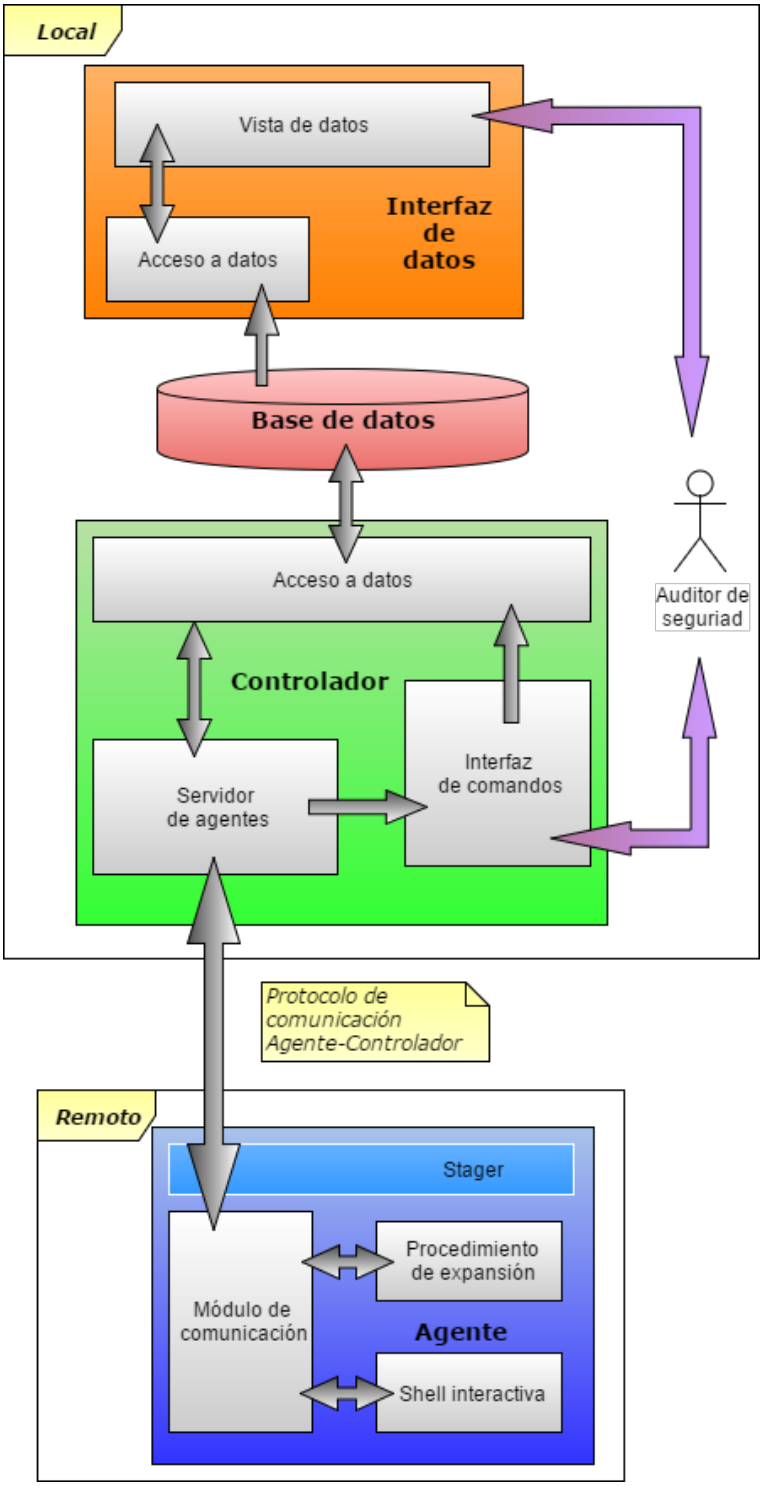

Figura 4.1: Diagrama de la arquitectura del sistema

- **Protocolo de comunicación Agente-Controlador**: El protocolo de comunicación utilizado entre el subsistema Controlador y el subsistema Agente define como se intercambia la información entre estos. En este capítulo se muestra cuales son las diferentes capas de este protocolo y que tipos de mensajes existen en él.

4.2. Modelo de la base de datos

En esta sección se define uno de los elementos más críticos del producto a desarrollar, y es que, en este, como en la mayoría de los productos software, la base de datos juega un papel fundamental en la arquitectura del sistema, siendo la pieza donde se almacenarán, en este caso, los datos recogidos durante el proceso de expansión de los agentes.

Es importante plantear, desde el inicio de la etapa de diseño, el modelo de datos que se empleará en la base de datos, y por consiguiente, en todo el sistema, ya que este determina en gran medida como se realiza el diseño de los distintos subsistemas.

4.2.1. Tecnologías empleadas

Antes de proceder con el diseño de la base de datos se indicará cual será el sistema gestor de base de datos (SGBD) que se empleará para almacenar la información.

Neo4j

Neo4j[15] es una base de datos NoSQL orientada a grafos. En Neo4j, al contrario que en las bases de datos relacionales, no existen las tablas, y la información se almacena siguiendo dos estructuras simples, los nodos, que almacenan la información de los entes o activos, y las relaciones, que permiten visualizar las conexiones entre los nodos, lo cual se ajustan al modelo requerido en este proyecto.

4.2.2. Modelo Entidad-Relación

A continuación, en la figura 4.2, se presenta el modelo Entidad-Relación que muestra los distintos elementos de la base de datos y las relaciones existentes entre ellos.

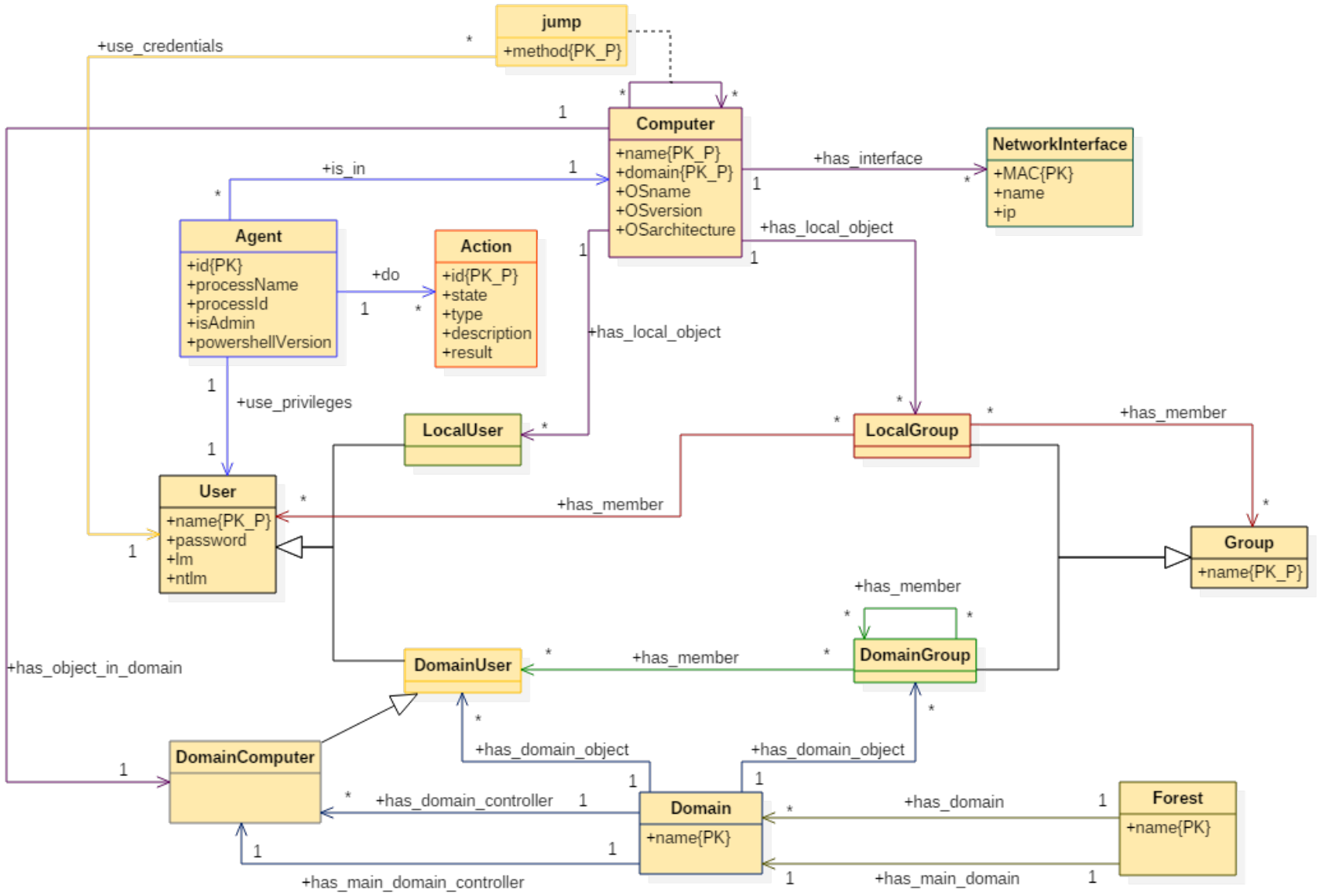

Figura 4.2: Modelo Entidad-Relación de la base de datos

Se deben realizar unas cuantas aclaraciones sobre lo representado en el diagrama Entidad-Relación:

A pesar de que no parezca lógico en un primer momento, *DomainComputer* hereda de *DomainUser* debido a que así se hace en las clases de Active Directory [35], ya que una cuenta de ordenador puede contar con los mismos permisos que un usuario, y por tanto cuenta también con credenciales.

El servidor principal de un dominio (relación *has_main_domain_server*) es uno de los servidores del dominio (relación *has_domain_server*). Del mismo modo, el dominio principal de un bosque de dominios (relación *has_main_domain*) es uno de los dominios del bosque (relación *has_domain*).

En un modelo Entidad-Relación puede ser contraproducente utilizar el mismo nombre para distintas relaciones, debido a que estas pueden derivar en tablas en una base de datos relacional. Sin embargo en este caso, al utilizarse para la implementación una base NoSQL basada en grafos, en el diagrama se muestran varias relaciones que utilizan el mismo nombre (en este caso *has_domain_object*, *has_local_object* y *has_member*) para dejar claro que aunque la relación se de entre distintas entidades, esta sigue siendo la misma, es decir, tiene el mismo significado.

4.2.3. Definición de las entidades

En este apartado se definen las distintas entidades del modelo Entidad-Relación para dejar claro su significado.

Por otro lado debido a que la base de datos utilizada no es relacional sino basada en grafos, no se realiza la transformación del modelo Entidad-Relación a las tablas del modelo Relacional. Como consecuencia de esto se especifican en este apartado todas las transformaciones que requiera el modelo de cara a su implementación.

- **Agent**: Representa al agente que se encuentra desplegado en una máquina y el contexto en que este se ejecuta. Se debe señalar que el campo *isAdmin* no señala si el usuario al que pertenece el proceso del agente es administrador, sino que indica si el propio proceso tiene privilegios de administrador, ya que es posible para un usuario administrador ejecutar un proceso sin esos privilegios.
- **Action**: Identifica las acciones realizadas por los agentes desplegados y que son ordenadas por el usuario.

- **Computer y DomainComputer**: Estas entidades representan dos facciones distintas de una misma máquina. Por un lado *Computer* trata de representar al ordenador como elemento físico, y por otro lado *Computer-Domain* identifica al objeto de Active Directory que representa al ordenador en un dominio. Aunque en el modelo, por claridad, se representen separadas, en la práctica se implementan como una sola entidad ya que de otra forma se crearía redundancia en los datos, como se puede deducir a partir de la relación (*has_object_in_domain*) con multiplicidad 1:1 que mantienen.

- **NetworkInterface**: Esta interfaz representa cada una de las interfaces de red que puede poseer una máquina. A pesar de que este dato no se utiliza en el sistema, se almacena para usos posteriores o por si el auditor desea conocerlo a través de consultas personalizadas en la Interfaz de datos.

- **Forest**: Representa al bosque de dominios de Active Directory donde se realiza la expansión.

- **Domain**: Esta entidad identifica a cada uno de los dominios del bosque que contienen los distintos usuarios, grupos y equipos.

- **User, DomainUser y LocalUser**: En *User* se indican los datos comunes que se deben recoger para todos los tipos de usuarios (incluido *Domain-Computer*). En el caso del atributo *name*, que es una clave parcial, se debe decir que para los *LocalUser* se combina con la clave primaria de la entidad *Computer* con la que estén relacionados. En el caso de la entidad *Domain-User* se hace lo equivalente con el *Domain* con el que se relacione. De cara a la implementación se elimina la entidad *User* y las entidades derivadas incorporan los atributos definidos en esta.

- **Group, DomainGroup y LocalGroup**: Al igual que sucede con los usuarios, los grupos de dominio utilizan como parte de su clave el nombre de dominio al que pertenezcan y los locales toman la clave del ordenador en que se encuentren. Así mismo desaparece en la implementación la entidad *Group* y las entidades hijas incorporan el atributo *nombre*.

- **jump**: Esta relación representa los saltos dados entre los ordenadores por los agentes. Como clave primaria de esta relación se tiene las claves primarias de los ordenadores y el usuario involucrados junto con el método utilizado para ejecutar código remoto.

4.3. Subsistema Agente

En este apartado se profundiza en el diseño del subsistema Agente. Como se ha visto, este subsistema es el único que se ejecuta en remoto, es decir, en máquinas ajenas a las del auditor de seguridad, por lo que a la hora de ser diseñado cuenta con algunas limitaciones especiales, que se comentan a lo largo de esta sección.

4.3.1. Tecnologías empleadas

Para realizar un buen diseño se debe tener en cuenta las tecnologías que se utilizarán para implementar el subsistema Agente. En este apartado se presentan dichas tecnologías.

Powershell

Debido a que así se establece en los requisitos, concretamente en el requisito RD01, el lenguaje de programación en que se implementarán los agentes será Powershell[9].

Powershell es el lenguaje de programación utilizado por la consola de Windows que recibe el mismo nombre. Powershell contrasta con otros lenguajes utilizados en terminales, como bash, ya que en vez manejar cadenas de texto como estos, utiliza el framework .NET para poder manejar objetos. En el contexto de Powershell a las funciones se las denomina *cmdlets*.

La instalación básica de Powershell, que se encuentra por defecto en todos los sistemas de Microsoft posteriores a Windows Server 2008 proporciona las *cmdlets* más comunes para interactuar con el sistema operativo, añadiendo en cada nueva versión de Powershell nuevas *cmdlets* respecto a la anterior. Para ampliar estas funcionalidades también es posible para un usuario de la terminal Powershell importar nuevos módulos o crear en la misma consola sus propias *cmdlets*.

En este proyecto a consecuencia de los requisitos RD01 y RD02 los agentes solo pueden utilizar las cmdlets que se incluyen en la instalación básica de la versión 2 de Powershell con el objetivo de operar en el mayor número de sistemas operativos posibles.

Powershell ISE

En los sistemas Windows en los cuales está presente Powershell se encuentra instalado el IDE de programación Powershell ISE, que permite desarrollar y ejecutar los scripts de Powershell de manera rápida y sencilla. En este proyecto ha sido la herramienta utilizada para realizar la implementación de los agentes.

VirtualBox

VirtualBox[16] es una herramienta de virtualización que permite instalar sistemas operativos invitados dentro del sistema operativo que se ejecuta en la máquina física. Incluido entre estos Virtualbox permite virtualizar los distintos sistemas operativos Windows y desplegar redes que permitan conectar estos sistemas. Por este motivo se ha usado este software para construir un laboratorio donde se desplegará una red de Active Directory para realizar las pruebas de expansión con los agentes.

4.3.2. Arquitectura de Agente

En la figura 4.1 se ha visto a alto nivel como se estructura el subsistema Agente, sin embargo este diagrama solo muestra las partes más relevantes de este, tanto para su funcionamiento, como para su comunicación con el Controlador. En esta sección se mostrará la arquitectura del subsistema Agente más detalladamente.

En la figura 4.3 se pueden observar tres grupos de elementos diferenciados:

- **Hilos**: Para realizar su función el agente debe seguir varias líneas de ejecución que le permitan realizar el proceso de expansión a la vez que proporciona al usuario la capacidad de realizar acciones en el equipo controlado. Se debe notar que salvo el hilo de iniciación, el resto de hilos muestran una flecha que une su final con su principio indicando que se ejecutan en forma de bucle hasta que el agente termine su proceso. Esto es debido a que el agente realmente es un demonio que se ejecuta continuamente en segundo plano en aquel ordenador donde se despliegue.

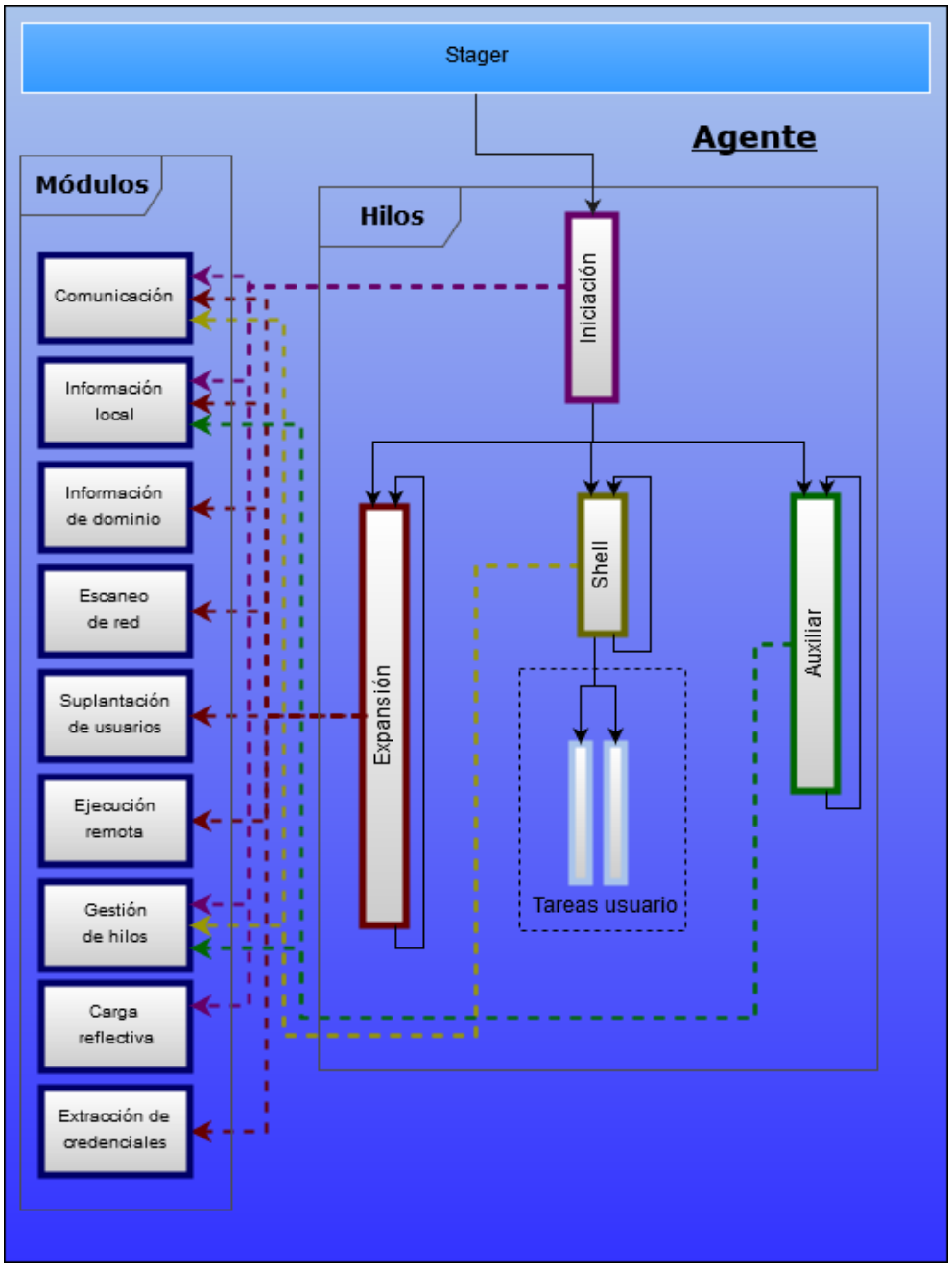

Figura 4.3: Diagrama de la arquitectura del subsistema Agente

- **Módulos**: Los módulos agrupan las funciones necesarias para que el agente pueda operar. Los hilos durante su ejecución se sirven de estos módulos para realizar sus respectivas tareas.
- **Stager**: El stager es un pequeño componente que tiene la tarea de desplegar un agente en el equipo en que se ejecuta.

Por otro lado el diagrama de arquitectura del subsistema Agente también se muestra que módulos son invocados por cada hilo para realizar su función.

En las secciones siguientes se muestra que funciones incluye cada módulo, cual es la secuencia de acciones que sigue cada hilo y que función cumple el stager.

4.3.3. Módulos del Agente

En esta sección se describen los módulos incluidos en el subsistema Agente que se han mostrado en la figura 4.3 y que son utilizados por los hilos de ejecución.

En la figura 4.4 se muestran, con más detalle, los módulos del susbistema Agente, los cuales serán descritos a continuación. Se debe comentar que los formatos de los nombres de las funciones o *cmdlets* incluidas en el agente siguen la reglas de nombrado de Powershell[34] que establece que la estructura del nombre debe ser Verbo-Nombre. Sin embargo debido a que el software de modelado UML no acepta el carácter "-"se suprime este quedando los nombres mostrados en las figuras como VerboNombre.

Módulo de comunicación

En el módulo de comunicación se encuentran todas aquellas funciones que permiten a los agentes comunicarse con el Controlador, empleando para ello el protocolo ACP, que se describe en secciones posteriores de la presente documentación.

En este módulo se pueden encontrar 4 grupos diferentes de funciones:

- **HTTP**: Funciones para enviar los datos a través de HTTP, en este caso *Send-HTTPRequest*.
- **JSON**: Las funciones orientadas a convertir los datos a formato JSON, en este caso *ParseTo-Json*. Se debe decir que a partir de Powershell versión 3 se incorporan funcionalidades nativas para la transformación de los datos

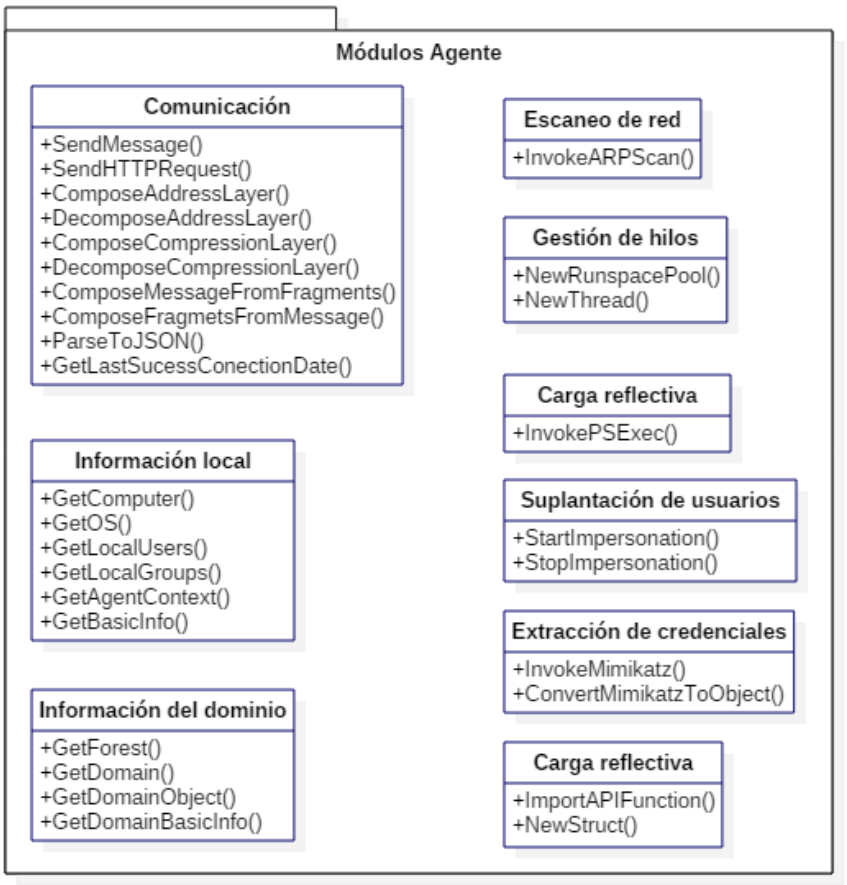

Figura 4.4: Módulos incluidos dentro del subsistema Agente

al formato JSON y viceversa. Sin embargo es necesario implementar esta función para soportar Powershell versión 2.

- **Información de comunicación**: Funciones orientadas a ofrecer información sobre las comunicaciones al agente, en este caso *Get-LastSucessConectionDate*, que permite obtener la fecha y hora en la que se produjo la última conexión del agente.

- **Capas del protocolo**: El resto de las funciones del módulo están destinadas a ensamblar y desensamblar los paquetes de las diferentes capas del protocolo ACP.

Módulo de información local

En este módulo se encuentran las funciones que permiten al agente recolectar información sobre el entorno local, es decir, sobre la máquina controlada. Dentro de los datos proporcionados por este módulo se encuentran los siguientes:

- **Dominio**: Datos relacionados con el dominio, entre ellos el nombre del equipo, el nombre del dominio al que pertenece el equipo y el rol del equipo dentro del dominio, que según lo establecido por Microsoft[36] puede ser: Standalone Workstation, Member Workstation, Standalone Server, Member Server, Backup Domain Controller o Primary Domain Controller.
- **Sistema operativo**: Nombre, versión y arquitectura del sistema operativo.
- **Usuarios locales del equipo**: Nombres de los usuarios locales.
- **Grupos locales del equipo**: Nombres de los grupos locales y sus miembros, que pueden ser grupos o usuarios locales o de dominio.
- **Interfaces de red**: Información sobre las interfaces de red como su nombre, MAC e IP.

Módulo de información de dominio

En el módulo de información de dominio se encuentran funciones que permiten realizar consultas contra el directorio del dominio y extraer los siguientes datos:

- **Dominios**: Los dominios que se encuentran en el bosque de dominios del ordenador controlado.
- **Usuarios de dominio**: Los usuarios de los dominios de Active Directory.
- **Grupos de dominio**: Los grupos de dominio y sus miembros asociados, que pueden ser otros grupos o usuarios de dominio.
- **Equipos**: Los ordenadores pertenecientes a los dominios de Active Directory, entre los cuales es posible identificar los servidores principales.

Módulo de escaneo de red

En el módulo de escaneo de red residen las funciones para llevar a cabo escaneos de red y realizar el descubrimiento de nuevas máquinas. Este módulo solamente cuenta con la función *Invoke-ARPScan*, que realiza peticiones utilizando el protocolo ARP para descubrir nuevas máquinas en el segmento de red, aunque tal vez en un futuro sea posible ampliarlo para dotar al agente de nuevas técnicas de escaneo.

Módulo de suplantación de usuarios

Una de las habilidades que requiere en el agente durante proceso de expansión es la de suplantar a otros usuarios una vez se hayan obtenido sus credenciales para realizar acciones con los privilegios de estos.

En este módulo se incluyen las funciones *Start-Impersonation* y *Stop-Impersonation* que permiten a los agentes suplantar a otros usuarios durante un período de su ejecución con el fin de utilizar los permisos de estos usuarios para desplegar nuevos agentes en nuevas máquinas.

Módulo de ejecución remota

Para llevar a cabo el proceso de expansión los agentes deben tener la capacidad de ejecutar código en equipos remotos con la finalidad de desplegar nuevos agentes en las máquinas de la red. Las funciones para realizar la ejecución remota de código se encuentran en este módulo, sin embargo solamente se encuentra disponible el método de ejecución remota PSExec (basado en el funcionamiento de la herramienta PSExec[21]) y se esperan que en un futuro se añadan nuevos métodos.

Módulo de gestión de hilos

Este módulo permite al agente la creación y gestión de hilos para realizar tareas en paralelo. Las dos funciones *New-RunspacePool* y *New-Thread* permiten crear nuevos runspaces e hilos, respectivamente, facilitando la realización de tareas simultáneas en la ejecución del agente.

Módulo de carga reflectiva

Este módulo le permite al agente incorporar funciones de la API de Windows[37] que por defecto no son accesibles desde Powershell. Cuenta con dos funciones principales *Import-APIFunction*, que permite importar las funciones de la API de Windows y *New-Struct* que permite definir estructuras que se usan como parámetros en las funciones importadas.

Para acceder a a las funcionalidades de la API de Windows este módulo incorpora la librería PSReflect[38]. Para incorporar esta librería a los agentes se debe incrustar su código fuente junto con los demás módulos del subsistema Agente.

Módulo de extracción de credenciales

Este módulo incorpora funcionalidades para extraer las credenciales que se encuentran almacenadas en el ordenador controlado. Para realizar la extracción de las credenciales se utiliza la herramienta Mimikatz[39], que ha sido portada a Powershell mediante la librería Invoke-Mimikatz[40].

Por tanto, la función *Invoke-Mimikatz* es la encargada de invocar a la herramienta Mimikatz para que recolecte las credenciales almacenadas. Además, se incluye la función *Convert-MimikatzToObject* que permite al agente transformar la cadena de texto que devuelve Mimikatz en objetos de Powershell, con la información de las credenciales, más fácilmente manejables.

La utilidad Mimikatz no se incluirá en el cuerpo del agente debido a que ocupa un tamaño de disco mayor de 1MB, lo que impide incluirlo en un solo fragmento de mensaje (RD18). Por este motivo los agentes se encargarán de descargar esta herramienta del Controlador e importarla dinámicamente durante el tiempo de ejecución para poder utilizarla.

4.3.4. Hilos del Agente

En esta sección se muestran los diferentes diagramas de secuencia que detallan como es la ejecución de cada uno de los hilos que componen el subsistema Agente.

Como se ha comentado anteriormente, el subsistema Agente es un demonio que se ejecuta en segundo plano dentro de un ordenador ajeno al del auditor de seguridad. Este demonio tiene dos funcionalidades principales:

- Recolectar información sobre la máquina en la cual se ejecuta el agente y el bosque de dominios donde se encuentra dicha máquina, y descubrir credenciales almacenadas y nuevas máquinas para continuar con el proceso de expansión de los agentes por la red del bosque de dominios. De esta tarea se encargará el hilo *Expansión*.

- Permitir la ejecución de comandos, en la máquina controlada, que envíe el auditor de seguridad. Esta funcionalidad se incluye en el hilo *Shell*, que creará los hilos que se encargarán de ejecutar las acciones que solicite el auditor de seguridad.

Además de los comentados se incluye el hilo *Auxiliar* para realizar tareas secundarias que permitirán al agente operar adecuadamente, como la actualización de parámetros que pueden variar con el tiempo.

Por último se encuentra el hilo *Iniciación* que se ejecuta al iniciarse el agente. Este hilo se encarga de configurar los parámetros del agente, registrarlo en el Controlador e iniciar los otros tres hilos que realizan las funciones principales.

En la figura 4.5 se muestra el esquema de los hilos del agente. Se debe apreciar que salvo el hilo *Iniciación*, el resto muestran una flecha que sale por debajo de la figura del hilo y vuelve al inicio, simbolizando que al ser el agente un demonio, estos hilos se ejecutan continuamente hasta que se termine el proceso, bien sea por que el usuario así lo decida o por que se haya perdido la comunicación con el Controlador.

A continuación se muestran los diagramas de secuencia que representan el comportamiento de los distintos hilos cuya ejecución conforma el comportamiento del subsistema Agente.

El objetivo de los diagramas de secuencia posteriores en esta sección no es mostrar la ejecución de un caso de uso, que es lo que habitualmente se representa utilizando este tipo de gráfico, sino mostrar como es el comportamiento de distintos hilos del agente. Por esto, y con motivo de mantener la claridad en los diagramas, los elementos con línea de tiempo en estos serán los hilos del subsistema Agente y no así sus módulos, indicándose el nombre de estos entre paréntesis al final de cada acción en que el hilo correspondiente se sirva de uno de ellos. También se muestra la línea de tiempo del Controlador para mostrar las interacciones con este.

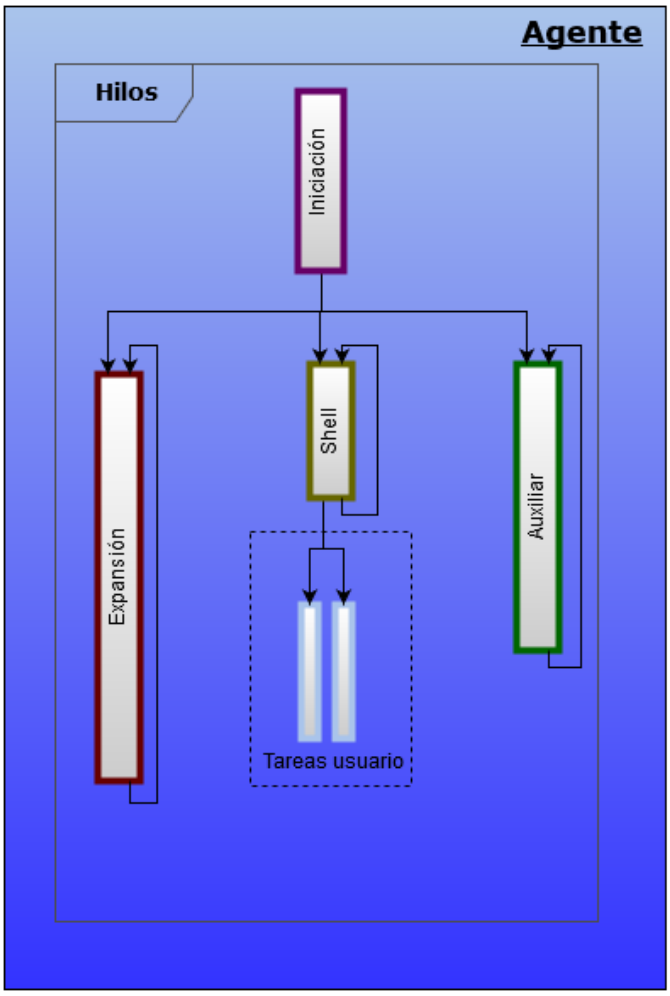

Figura 4.5: Hilos principales del subsistema Agente

Hilo de iniciación

En la figura 4.6 se muestra el diagrama de secuencia que describe el comportamiento del hilo *Iniciación* del subsistema Agente.

El hilo *Iniciación* se ocupa en primer lugar de cargar las funcionalidades de la API de Windows utilizadas por los módulos del agente. A continuación se encarga de establecer y configurar distintos parámetros del agente, como puede ser la IP del Controlador y la contraseña necesaria para interactuar con este.

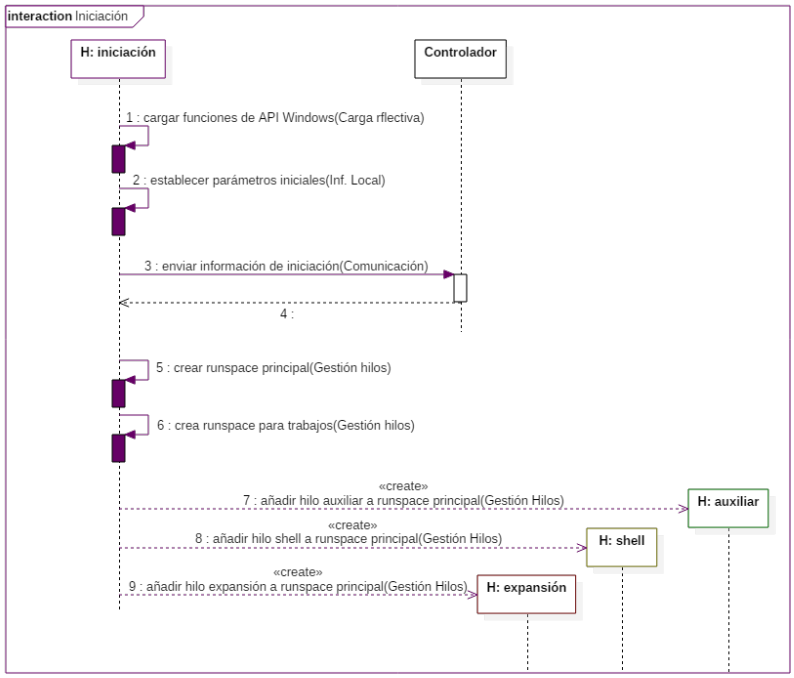

Figura 4.6: Diagrama de secuencia del hilo *Iniciación* del agente

Una vez establecida la configuración inicial el agente envía al Controlador su información inicial, como el nombre del equipo controlado y la información de como se ha realizado el salto para llegar a dicha máquina, incluyendo el método, credenciales y equipo origen del salto.

Una vez realizado este paso pueden darse dos eventos. En caso de que el Controlador verifique que el agente es el único desplegado en la máquina, le devuelve a este un identificador que será utilizado a partir de ese punto por el agente para identificarse. La otra posibilidad es que otro agente se encuentre ya desplegado en esa máquina, de modo que el Controlador responda con un mensaje para que este nuevo agente termine su proceso.

Por último, en caso de que el agente continue, este crea dos *runspaces*, el primero almacenará los hilos principales y el segundo servirá para alojar los hilos de trabajos encargados de ejecutar las acciones que el usuario envíe desde el Controlador. Una vez creados los *runspaces*, se inician los 3 hilos principales del agente.

Hilo Shell

En la figura 4.7 se muestra el diagrama de secuencia que describe el comportamiento del hilo *Shell* del subsistema Agente.

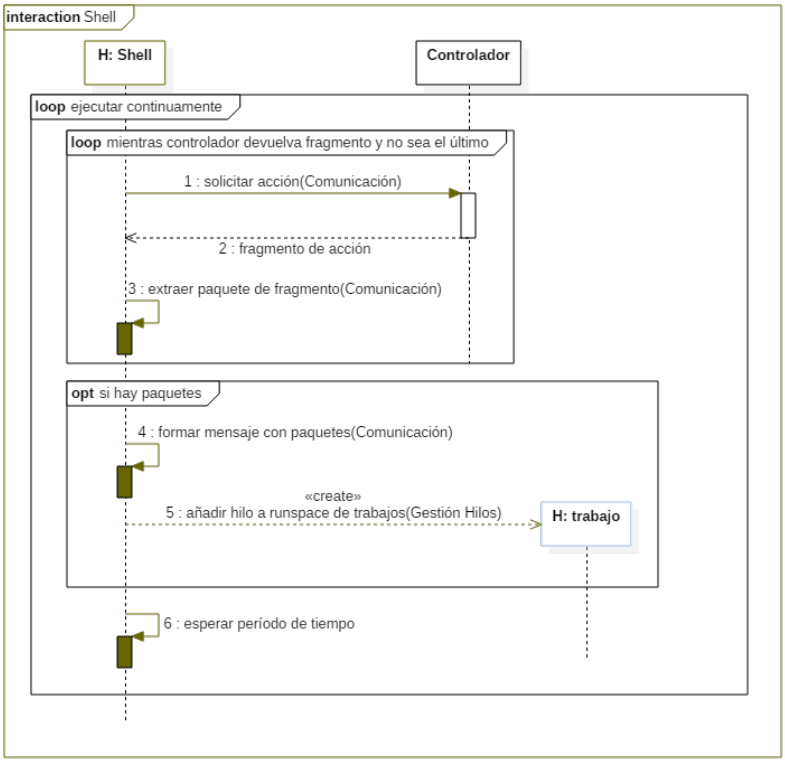

Figura 4.7: Diagrama de secuencia del hilo *Shell* del agente

Este es el hilo encargado de proporcionar al auditor de seguridad ejecución de acciones en la máquina controlada. Para ello el hilo *Shell* pregunta periódicamente al Controlador si dispone de alguna acción para el agente, y en caso afirmativo, recibe desde el Controlador el mensaje (fragmentado si supera el tamaño máximo establecido) que contiene la información necesaria para llevar a cabo dicha acción. Una vez se haya recibido el mensaje, se crea un nuevo hilo de trabajo que realice la acción indicada en el mismo.

Se debe notar que en un sistema convencional utilizar una arquitectura de publicador-suscriptor habría sido una mejor opción que una arquitectura basada en *pulling*

por parte del cliente (subsistema Agente) contra el servidor (subsistema Controlador) como el que se utilizará en esta ocasión, sin embargo debido al contexto de los agentes, que se ejecutan en un equipo, y pudiera ser una red, ajenos a la máquina del auditor de seguridad, es posible que un firewall restrinja las conexiones entrantes a la red impidiendo que un protocolo publicador-suscriptor pueda funcionar correctamente.

Hilo Auxiliar

En la figura 4.8 se muestra el diagrama de secuencia que describe el comportamiento del hilo *Auxiliar* del subsistema Agente.

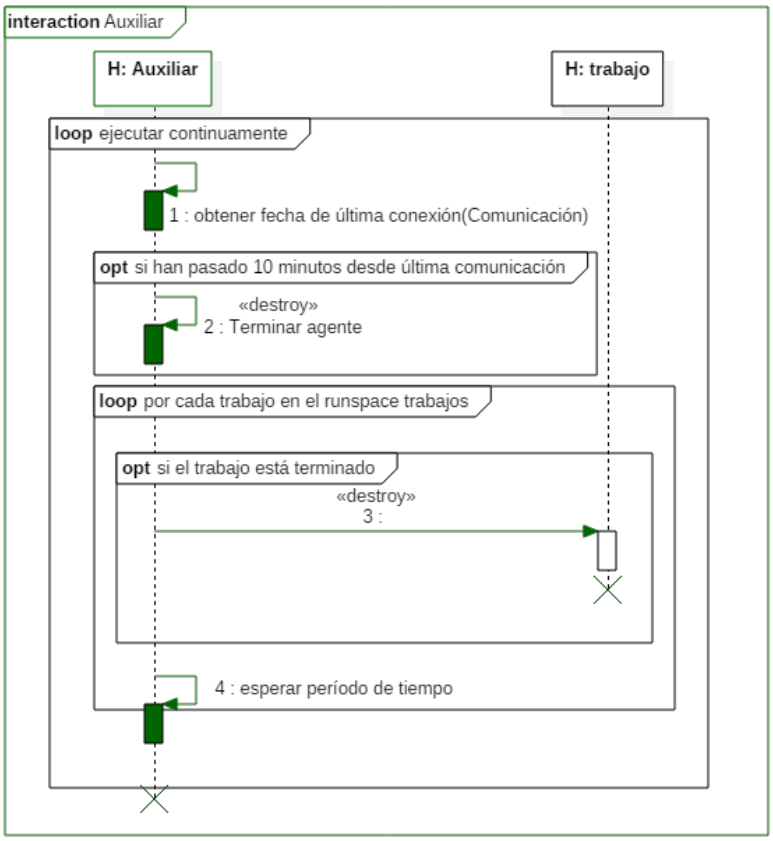

Figura 4.8: Diagrama de secuencia del hilo *Auxiliar* del agente

El funcionamiento del hilo *Auxiliar* se compone de dos funciones principales que se ejecutan periódicamente.

- Verificar el tiempo que ha transcurrido desde la última conexión exitosa con el servidor y en caso de que supere los 10 minutos, terminar el proceso del agente.
- Revisar el *runspace* de trabajos y eliminar de él aquellos hilos que hayan terminado su ejecución.

Hilo Expansión

En las siguientes figuras, 4.9, 4.10 y 4.11, se representa el funcionamiento del hilo *Expansión* del subsistema Agente.

La ejecución del hilo *Expansión* como se puede observar en las figuras anteriores se divide en 6 partes:

- **Recolección de información local**: En esta parte el agente recoge datos del ordenador controlado, donde se encuentran entre otros el nombre de la máquina, el sistema operativo, los usuarios y los grupos locales. Una vez recogida esta información esta se envía al Controlador.
- **Recolección de información del dominio**: En esta fase el agente consulta el directorio de Active Directory para obtener información sobre los dominios, los servidores, los usuarios y los grupos administradores del bosque de dominios donde se encuentra el ordenador controlado. Cuando se obtienen los datos estos son enviados al Controlador.
- **Extracción de credenciales**: Esta es la etapa en que el agente extrae, utilizando para ello la herramienta Mimikatz, las credenciales almacenadas en el equipo. Lo primero que debe hacer el agente es cargar la utilidad Mimikatz si esta no se encontraba previamente cargada. Una vez esta esté

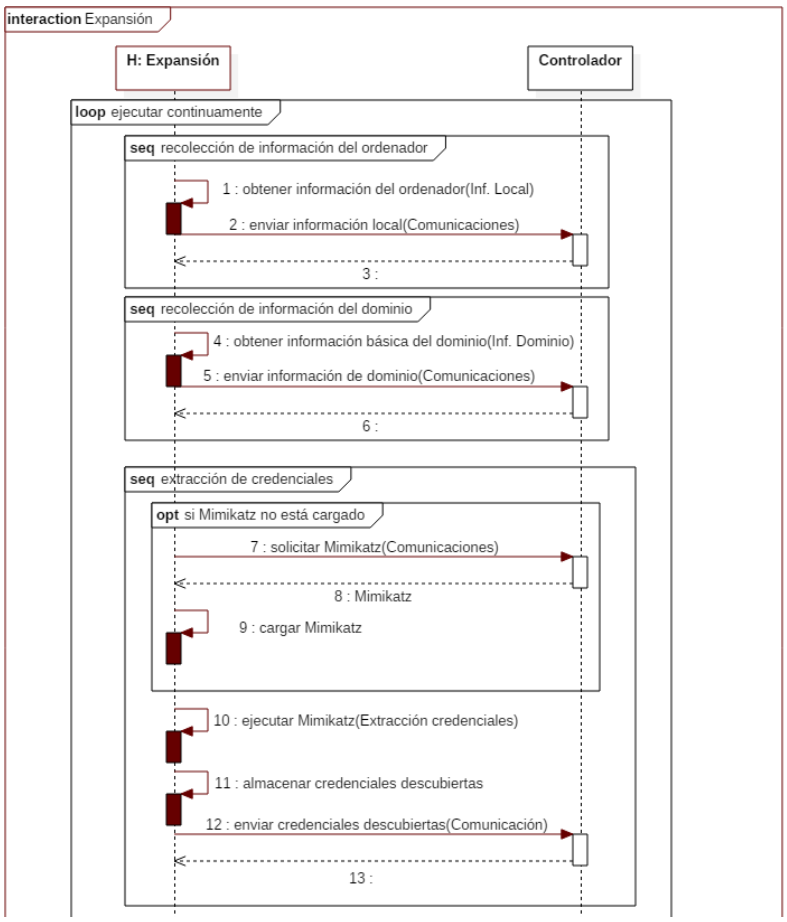

Figura 4.9: Diagrama de secuencia del hilo *Expansión* del agente - Parte 1

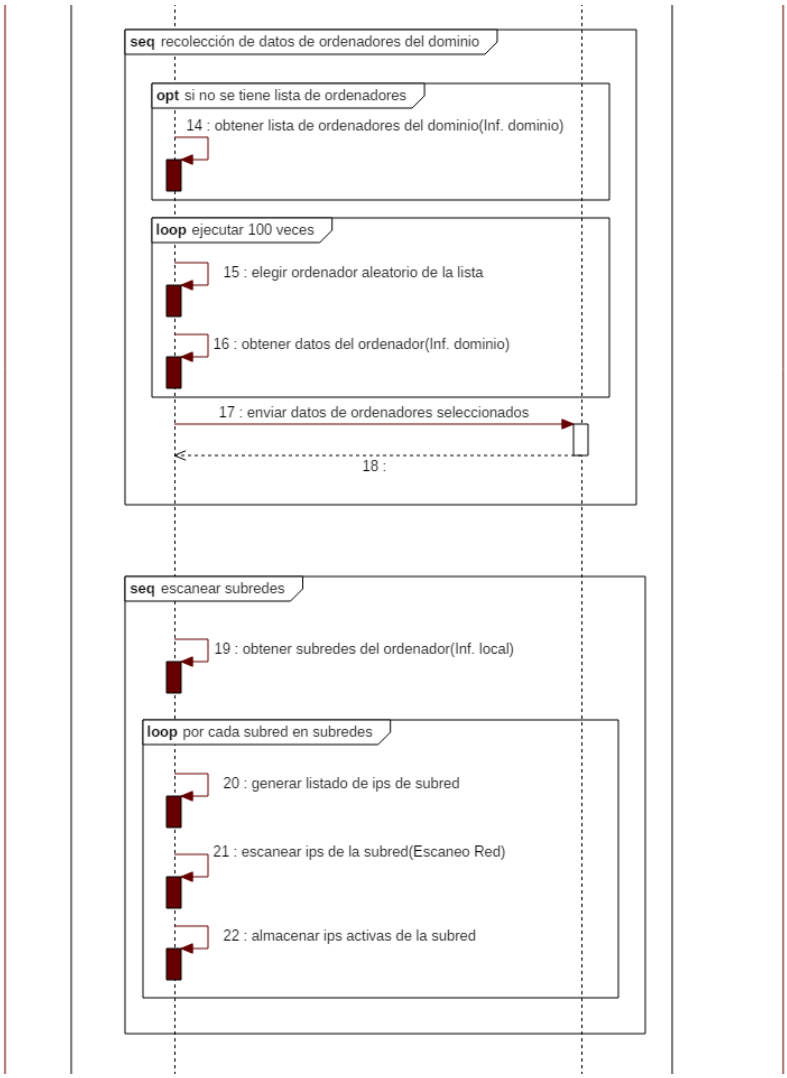

Figura 4.10: Diagrama de secuencia del hilo *Expansión* del agente - Parte 2

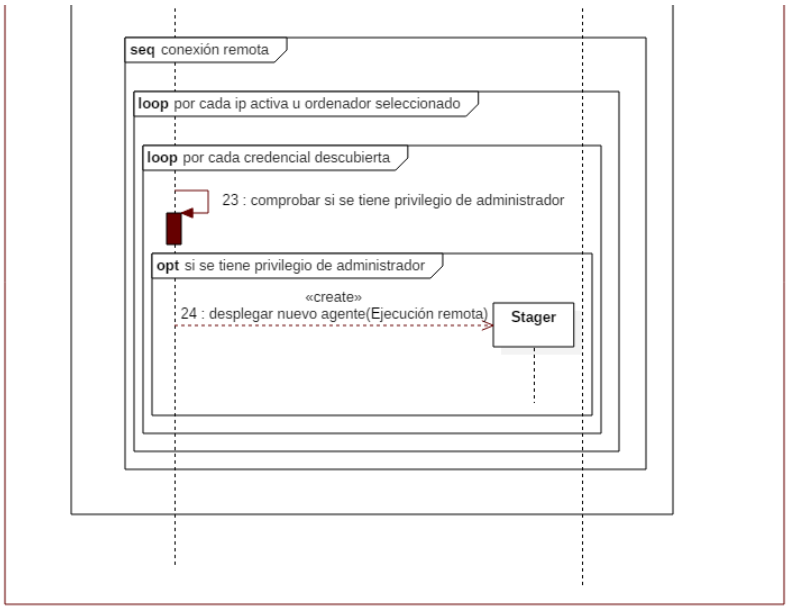

Figura 4.11: Diagrama de secuencia del hilo *Expansión* del agente - Parte 3

disponible se puede ejecutar para extraer las credenciales, enviar estas al Controlador y almacenarlas en la memoria del agente para utilizarlas posteriormente.

- **Recolección de datos de ordenadores del dominio**: En esta parte, si no se ha hecho previamente, el agente realiza una consulta ligera contra el controlador de dominio para extraer una lista de nombres de máquinas (se considera una consulta ligera ya que solamente se solicita el nombre de la máquina sin más datos adicionales, por lo que se transmite poca información). Luego, en cada iteración del hilo *Expansión*, se escogen 100 nombres de equipos, se pide información más detallada de cada uno de ellos con consultas más pesadas (ya que en ellas se transmitirán más datos de cada máquina), se envían los datos recogidos al Controlador y se almacenan los nombres para realizar una comprobación posterior de credenciales.

- **Escaneo de subredes**: El agente comprueba en que subredes se encuentran el ordenador controlado y realiza un escaneo de cada una de ellas. En el diagrama se ha realizado una simplificación, pero en caso de que el rango de las IPs de una subred sea superior a 255, se escanearán rangos de 255 IPs en sucesivas iteraciones del hilo *Expansión* hasta lograr escanear el rango completo de IPs de la red. Cuando se haya realizado el escaneo de las IPs, se almacenan aquellas que se hubiesen encontrado activas para comprobar contra ellas las credenciales obtenidas.

- **Conexión remota**: En la última etapa de la iteración el agente comprueba, para cada máquina o IP almacenada, todas las credenciales que se hayan obtenido para verificar si en algún equipo se consiguen privilegios de administrador. En caso afirmativo el agente ejecutará el código necesario para desplegar un nuevo agente en la máquina remota.

Una vez explicado el diagrama se deben realizar ciertos apuntes el procedimiento de este hilo:

En primer lugar, en las fases de recolección de datos del dominio y ordenadores del bosque de dominios, aunque se podrían solicitar todos los datos y ordenadores del bosque, esta operación llevaría un período de tiempo considerable evitando que el hilo pudiese realizar las otras fases, por lo que se sigue la aproximación de recoger, en el primer caso, solo los datos más esenciales del bosque, y en el segundo solo 100 ordenadores por iteración del hilo.

En segundo lugar, a pesar de que en el diagrama se muestre que se realizarán todas las fases en cada iteración, el agente deberá tener un contador para cada fase de forma que estas solo se realicen en ciertas iteraciones, ya que ciertas fases como el escaneo de subredes, de realizarse repetidamente podrían levantar las sospechas de los sistemas defensivos de la red de ordenadores donde se está intentando realizar la expansión.

4.3.5. Stager

Durante la planificación inicial del proyecto se estableció que una vez encontradas unas credenciales que concediesen privilegios de administrador en una máquina remota, el agente debería ejecutar en ella el código correspondiente a un nuevo agente.

Sin embargo durante la implementación se encontró un problema relacionado con los parámetros del ejecutable Powershell, ya que este no permitía pasarle por parámetro comandos con un número de caracteres superior a 12259 al crear un nuevo proceso de Powershell, lo cual es necesario para desplegar un agente.

El problema tenía dos posibles soluciones:

- Implementar un agente en un número de caracteres inferior a 12259.

- Implementar una pequeña porción de código inferior a 12259 caracteres que se conectase al Controlador y descargarse y ejecutase en la máquina remota el agente.

Debido a que era inviable implementar en tan pocos caracteres la funcionalidad del agente, se optó por la segunda opción, crear un *Stager* que una vez ejecutado descargase el código del agente y lo ejecutase.

Por otro lado se aprovechó la necesidad de crear el Stager para establecer este como el código de instalación que el usuario debe ejecutar para desplegar un agente.

A continuación, en la figura 4.12, se muestra el diagrama de secuencia del *Stager*.

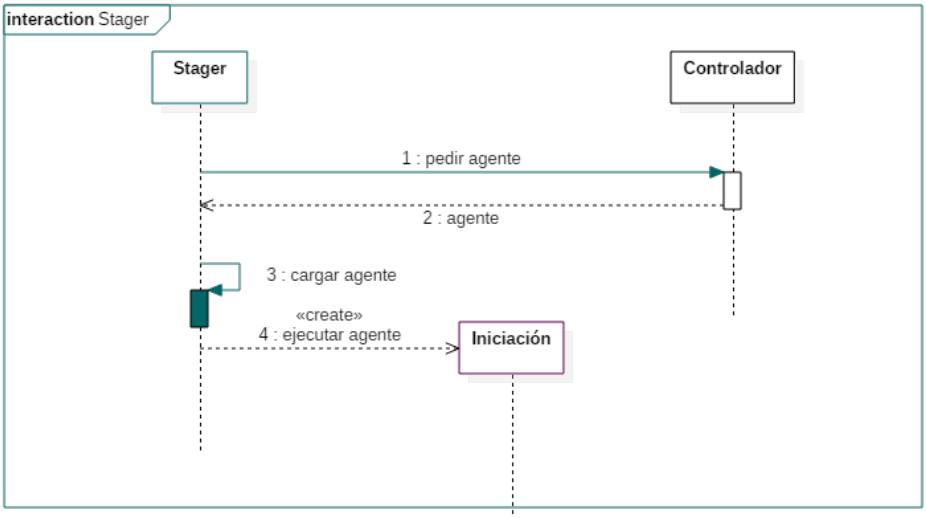

Figura 4.12: Diagrama de secuencia del *Stager*

4.4. Protocolo de comunicación Agente-Controlador

En esta sección se explica como se realiza la comunicación entre el subsistema Agente y el subsistema Controlador.

4.4.1. Tecnologías empleadas

Se explican en primer lugar las tecnologías que se usan y en las que se basa el protocolo diseñado para la comunicación entre el Controlador y los agentes.

JSON

JSON[11] o JavaScript Object Notation es un lenguaje para el intercambio de información que se caracteriza, al contrario que XML, por su estructura minimalista basada en la declaración de objetos utilizada por Javascript, tal y como su propio nombre indica.

Se utilizará JSON como formato para el intercambio de datos complejos entre los agentes y el controlador. Se ha elegido este formato debido a que la estructura de JSON ocupa menos bytes que XML y permite reducir el tamaño de los mensajes entre los agente y el Controlador, facilitando la consecución del requisito RD04.

HTTP

El protocolo HTTP[12] (HyperText Transfer Protocol) es utilizado en la web para transmitir las páginas HTML a los usuarios, entre otro tipo de ficheros. Es un protocolo orientando a una arquitectura cliente-servidor. En el se definen diferentes métodos para que los clientes realicen peticiones y asimismo se especifican las posibles respuestas que puede proporcionar un servidor.

En este caso se utilizan los siguientes dos métodos de HTTP:

- **GET**: Es el método utilizado para solicitar un recurso.
- **POST**: Se utiliza para transmitir datos que se almacenarán en el servidor.

Por otro lado se emplean las siguientes respuestas:

- **OK**: Código 200. Respuesta estándar para las solicitudes realizadas con éxito.
- **Not Found**: Código 404. Respuesta utilizada cuando el recurso solicitado no pudo ser encontrado.

El protocolo HTTP es un protocolo sin estado, pero permite el uso de cookies que pueden ser usadas para mantener información sobre el estado del cliente.

4.4.2. Uso del protocolo HTTP

El protocolo base sobre el que se implementa la comunicación entre los agentes y el Controlador será HTTP, ya que es el protocolo utilizado en la web y evitará que salte la alarma de los sistemas de detección de intrusiones que podrían estar desplegados por la red de Active Directory.

Dentro del protocolo HTTP se utilizarán los métodos GET y POST, ya que son los métodos más comunes en aplicaciones web para solicitar y enviar datos, respectivamente.

El método GET será el utilizado en los siguientes casos:

- Cuando el Stager quiera descargar el código del agente. En este caso el GET se realizará solicitando el path */agent*.
- En las peticiones de solicitud de comando que el agente realizará durante la ejecución del hilo *Shell*. La solicitud en este caso se realizará sobre el directorio raíz / del servidor web incluido en el Controlador.

En el resto de los casos, en los cuales el agente enviará datos al Controlador se usará el método POST.

De manera habitual el Controlador responderá a las peticiones del agente con la respuesta OK, ya que es la habitualmente utilizada en la web a pesar de que se puedan producir errores. La única excepción será en el caso de que el Controlador no disponga de acciones cuando el agente las solicite, en este caso la respuesta será Not Found.

Por otro lado los agentes enviarán en cada petición una cookie denominada *ID* que contendrá el identificador del agente con el objetivo de que el servidor pueda identificarlo. El Stager no enviará esta cookie por no disponer de identificador.

Además se enviará tanto en los agentes como Stagers una cookie denominada *Key* con una contraseña que permita verificar al servidor que se trata de una conexión válida.

4.4.3. Protocolo Agente-Controlador

Una vez definido como se utilizará HTTP para la transmisión de mensajes se necesita definir que formato seguirá el contenido enviado en los mensajes HTTP entre el Controlador y el agente, en concreto las peticiones POST y las respuestas del Controlador, ya que el método GET no transporta contenido en sus mensajes.

El contenido de los mensajes HTTP vendrá dado por lo que denominaremos a partir de este punto protocolo Agente-Controlador o ACP (Agent-Controller Protocol).

El protocolo ACP se compone de 2 capas diferenciadas e independientes:

- **Capa de aplicación**: En esta capa se define e identifica cada uno de los mensajes.
- **Capa de fragmentación**: Esta capa permite fragmentar los mensajes para que los mensajes HTTP no superen un tamaño máximo e identificar los fragmentos de una misma secuencia.

A continuación se definen cada una de las capas que componen el protocolo ACP.

Capa de aplicación

Como se ha comentado previamente la capa de aplicación es la que incluye el mensaje que se trata de enviar. Los paquetes de esta capa incluyen una cabecera que permite identificar el tipo de mensaje.

En la figura 4.13 se presenta la estructura de un paquete de la capa de aplicación.

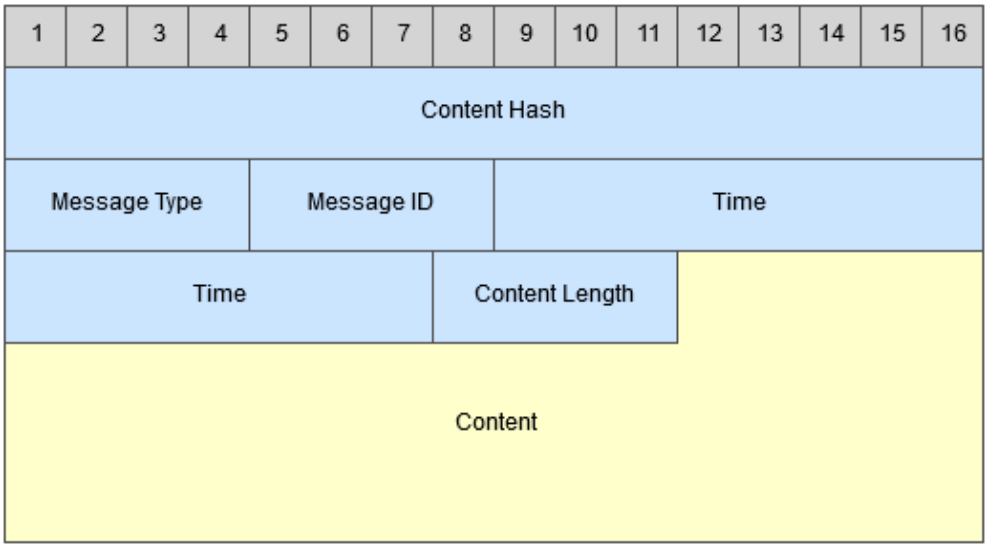

Figura 4.13: Paquete de la capa de aplicación del protocolo ACP

La cabecera de los paquetes de la capa de aplicación cuenta con los siguientes campos:

- **Content Hash (16 bytes)**: Este campo se obtiene a partir de aplicarle el algoritmo MD5 al contenido del mensaje. Sirve para validar la integridad del contenido.
- **Message Type (4 bytes)**: Identifica el tipo de contenido que se envía. Posteriormente se indican los tipos de mensajes existentes.
- **Message ID (4 bytes)**: Un identificador para permitir identificar un mensaje. La respuesta de un mensaje deberá llevar el mismo ID que el mensaje de solicitud.
- **Time (17 bytes)**: La fecha y hora en que se envió el mensaje siguiendo el formato YYYYMMDDhhmmssfff donde YYYY es el año, MM el mes, DD el día, hh la hora, mm los minutos, ss los segundos y fff los milisegundos. La fecha se debe especificar en formato UTC[42].
- **Content Length (4 bytes)**: El tamaño en bytes del contenido del mensaje.

Además en la capa de aplicación se definen diferentes tipos de mensajes que se pueden intercambiar. Los mensajes se dividen en dos tipos principales: mensajes de agente y mensajes de servidor.

Mensajes de Agente Los mensajes de Agente son aquellos que son transmitidos desde el subsistema Agente hacia el subsistema Controlador. Se listan a continuación los distintos tipos de mensaje.

- **Módulo**: Número 202002. Para solicitar el módulo que se especifica en el cuerpo del mensaje.
- **Información máquina**: Número 203001. Se envía la información referente a la máquina local en formato JSON.
- **Información usuarios**: Número 203003. Se envía la información referente a los usuarios junto con sus credenciales en formato JSON.
- **Información dominio**: Número 203011. Se envía la información referente al dominio en formato JSON.
- **Información de conexión**: Número 203004. Se envía la información referente a como se ha desplegado el agente en formato JSON.
- **Descargar archivo**: Número 204002. Se transmite el archivo descargado desde el agente.
- **Error**: Número 205001. Para enviar los resultados de error.
- **Terminación**: Número 206066. Se envía cuando el agente va a terminar su ejecución.
- **Resultado**: Número 207001. Para enviar el resultado de una petición.
- **Éxito**: Número 207002. Mensaje sin contenido que indica éxito en la operación solicitada.

Mensajes de Controlador De modo contrario a los tipos de mensajes de agente, los tipos de mensajes del Controlador son aquellos que son transmitidos desde el Controlador hacia el agente. Se listan a continuación los distintos tipos de mensaje.

- **Comando**: Número 101001. En el cuerpo se especifica el comando que debe ejecutar el agente.
- **Script**: Número 102001. En el cuerpo se especifica el script que debe ejecutar el agente.
- **Subir fichero**: Número 104001. Se envía el fichero que debe almacenar el agente.

- **Descargar fichero**: Número 104002. Se indica el fichero que debe ser descargado por el agente.
- **Error**: Número 105001. Para enviar mensajes de error.
- **Inicio**: Número 106001. En el cuerpo se especifica el ID del agente.
- **Terminación**: Número 106066. Se envía para terminar la ejecución de un agente.

Respecto del contenido de los mensajes de la capa de aplicación la codificación usada para transmitir la información es UTF-8, salvo en el caso en que se transmitan archivos. En este caso estos se transmitirán en formato binario para no alterar el contenido del mismo.

Capa de fragmentación

La capa de fragmentación es la que se ocupa de particionar los mensajes para que cada paquete no ocupe más de lo establecido, en este caso, 1MB. Para conseguir esto se deben añadir campos en la cabecera de cada fragmento que permitan en el receptor reconstruir el paquete original a partir de los fragmentos.

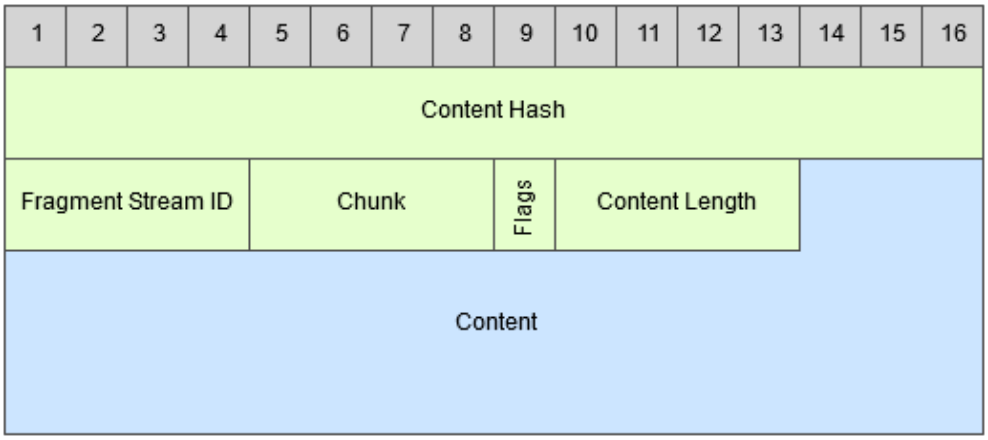

Figura 4.14: Paquete de la capa de fragmentación del protocolo ACP

En la figura 4.14 se muestra la cabecera de los paquetes de la capa de fragmentación, que cuenta con los siguientes campos:

- **Content Hash (16 bytes)**: Este campo se obtiene a partir de aplicarle el algoritmo MD5 al contenido del mensaje. Sirve para validar la integridad del contenido.
- **Fragment Stream ID (4 bytes)**: Identificador compartido por todos los fragmentos de un mismo mensaje que los identifica como miembros de este.
- **Chunk (4 bytes)**: Número que permite identificar la posición del fragmento dentro de la cadena de fragmentos que componen el mensaje.
- **Flags (1 byte)**: Permite definir condiciones en los mensajes. De momento la única flag existente es la de último fragmento. Para establecer esta flag se debe poner el valor a 1.
- **Content Length (4 bytes)**: El tamaño en bytes del contenido del fragmento.

El método de partición de un paquete de la capa de aplicación será comprobar si este ocupa más de lo establecido, y en ese caso partir dicho paquete en el número de los fragmentos necesario para que ninguno de los fragmentos supere el máximo establecido. Luego a cada uno de estos fragmentos se le añade la cabecera de fragmentación para que el receptor pueda reconstruir el paquete.

4.5. Subsistema Controlador

El segundo de los subsistemas es el Controlador, que cumple con dos funciones esenciales. Por una parte permite al Auditor de Seguridad interactuar con los agentes desplegados y por otra es el encargado de almacenar en la base de datos toda aquella información que los agentes recojan durante el proceso de expansión, permitiendo de esta forma que el susbsistema Interfaz de datos pueda posteriormente consultar y representar la información.

4.5.1. Tecnologías empleadas

Se muestran primeramente las tecnologías empleadas para implementar el subsistema Controlador, ya que, aunque el diseño sea independiente de la tecnología empleada, tenerla en cuenta puede ahorrar gran esfuerzo a la hora de tomar decisiones en esta fase.

Javascript

Javascript[10] es un lenguaje de programación interpretado, creado en 1995 por NetScape y que cumple con el estándar ECMAScript. Es un lenguaje mayoritariamente orientado a aplicaciones web, normalmente usado en la parte del cliente. Sin embargo es posible utilizarlo para programar la lógica de los servidores.

En este proyecto el lenguaje Javascript será utilizado en el Controlador de agentes, debido a su alta compatibilidad con JSON, y en la Interfaz de datos ya que en él se basan las tecnologías usadas para crear la interfaz.

NodeJS

NodeJS[14] proporciona un entorno para ejecutar código Javascript principalmente orientado a construir servidores, pero no limitado a ello. NodeJS permite la ejecución de funciones asíncronas en un mismo hilo, por lo que es posible atender a varios clientes concurrentemente sin tener costes computacionales asociados a los cambios de contexto. Además al ejecutar todo el proceso en un mismo hilo no es necesaria la utilización de mutexes para restringir el acceso a variables, facilitando la implementación.

En este proyecto se implementará el controlador de agentes utilizando Javascript junto con NodeJS, ya que proporciona una manera sencilla de construir un

servidor web y manejar concurrentemente a los distintos agentes sin tener que preocuparse por condiciones de carrera.

WebStorm

WebStorm es un IDE de programación, creado por la empresa JetBrains, orientado a desarrollo web. Entre otras funcionalidades permite desarrollar aplicaciones utilizando NodeJS, por lo que será la herramienta utilizada para programar el Controlador de agentes.

4.5.2. Arquitectura del Controlador

En este apartado se muestra la arquitectura general del subsistema Controlador.

En la figura 4.15 se pueden diferenciar tres componentes principales dentro del subsistema Controlador, cada uno con sus correspondientes módulos. Del mismo modo es posible apreciar la relación de invocación entre los diferentes módulos, siendo el módulo invocado el señalado por las flechas discontinuas.

Respecto de las relaciones de invocación, tal vez las más importantes sean las dadas entre componentes. Se puede observar en el diagrama que los tres componentes tienen en común un módulo llamado Exportador. Este módulo será el encargado de exportar las funciones necesarias de cada componente para que los demás puedan interactuar con este. De esta forma se mantiene la independencia entre componentes y la modularidad del subsistema, permitiendo que en un futuro si por ejemplo se quisiera sustituir la interfaz (la terminal, no confundir con Interfaz de datos) solamente fuese necesario sustituir, en ese caso concreto, el componente Interfaz de comandos.

A continuación se realiza una breve descripción de los componentes, que serán explicados con mayor detalle en las secciones posteriores:

- **Servidor de agentes**: El Servidor de agentes, al que a partir de este punto denominaremos Servidor, es el componente encargado de realizar la comunicación con los agentes desplegados y mediar entre estos y el resto del Controlador. En él se implementa el protocolo ACP.

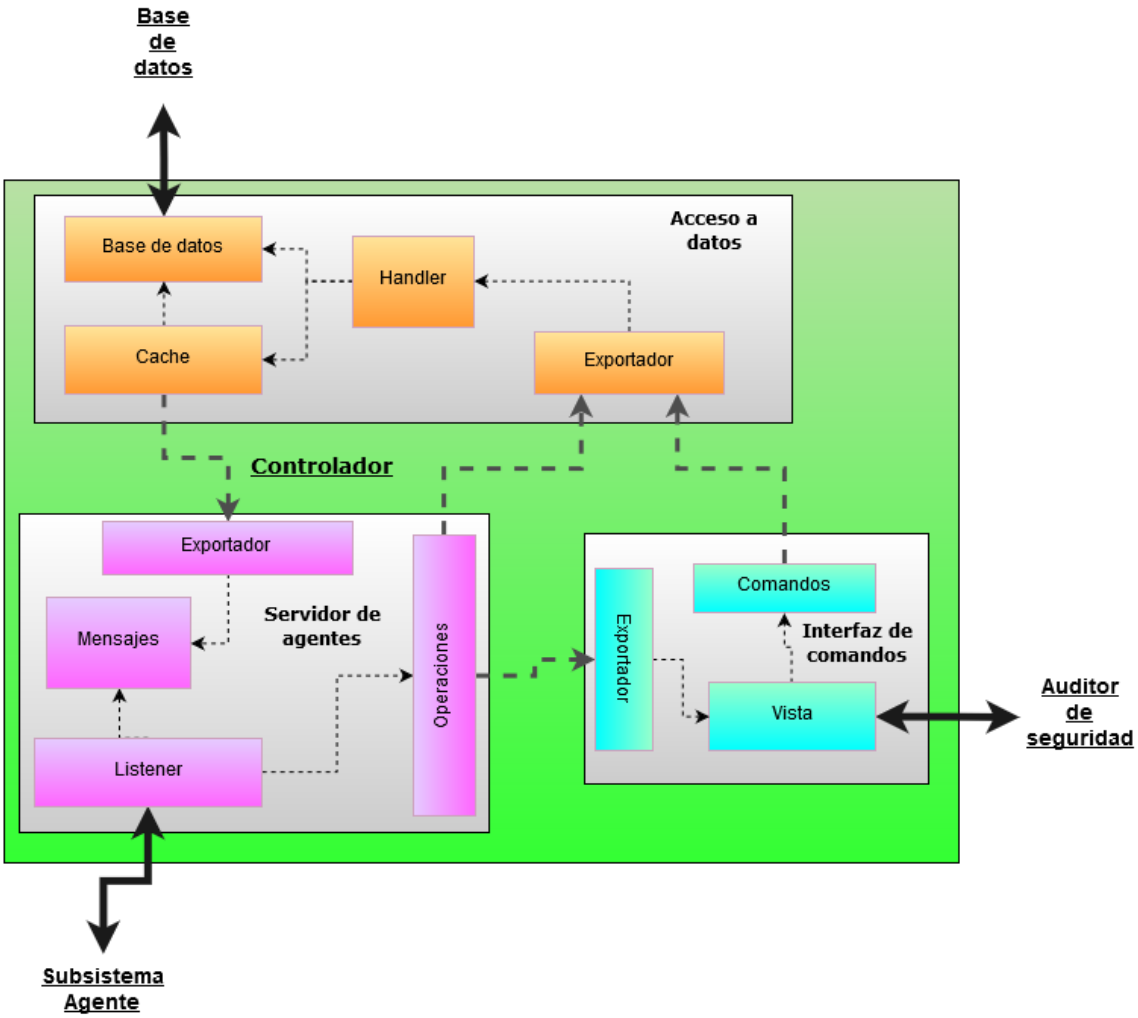

Figura 4.15: Arquitectura del subsistema Controlador

- **Interfaz de comandos**: La Interfaz de comandos, denominada a partir de este punto Terminal, se encarga de presentar al usuario una consola de comandos con la que poder interactuar con los agentes desplegados.

- **Acceso a datos**: Este es el componente donde se manejan todos los datos del Controlador, permitiendo a los demás componentes interactuar con ellos y proporcionando una caché para evitar tener que depender de la base de datos en caso de que esta no se encuentre disponible. Posteriormente en el presente documento se denominará a este componente Manejador.

4.5.3. Servidor de agentes

En este apartado se describirá la estructura interna del componente Servidor, que permitirá a los agentes ponerse en contacto con el Controlador.

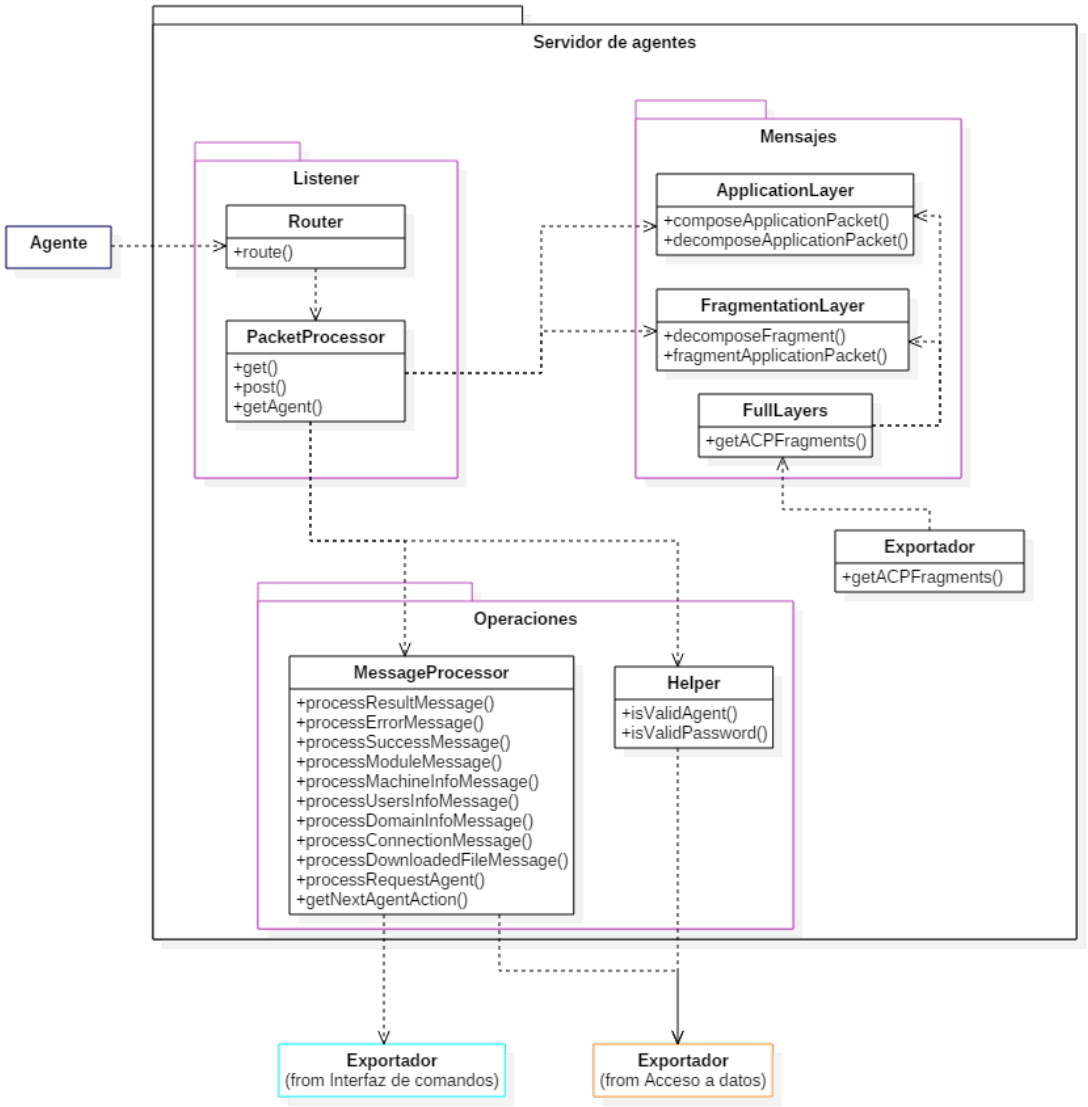

Figura 4.16: Arquitectura del componente Servidor de Agentes del subsistema Controlador

En la figura 4.16 se muestran los subcomponentes (se denominará de esta forma a los paquetes de módulos para evitar su confusión con los paquetes referidos a los mensajes del protocolo ACP) y módulos que conforman el Servidor. Se puede ver la interacción realizada entre los diferentes módulos que conforman el servidor.

A continuación se explican los diferentes módulos de cada uno de los subcomponentes y luego se hará un pequeño resumen de la interacción entre estos.

Listener

El subcomponente Listener es el encargado de procesar los mensajes recibidos desde el subsistema Agente y de enviar a este la respuesta adecuada de vuelta.

Los módulos de este subcomponente son los siguientes:

- **Router**: Encargado de examinar los paquetes HTTP e invocar la correspondiente función en el módulo PacketProcessor según el tipo de paquete que se reciba.
- **PacketProcessor**: Este es el módulo donde se gestionarán los paquetes del protocolo ACP, permitiendo distinguir cada tipo de paquete recibido. Para ello se utilizarán las funciones del subcomponente Mensajes, que permitirán manejar estos paquetes. Una vez identificado el mensaje del agente se invocará la operación consecuente del módulo MessageProcessor. Además en este módulo se deben implementar una serie de listas que permitan almacenar los fragmentos recibidos y una vez obtenidos todos los de un mensaje, formar dicho mensaje.

Operaciones

Una vez procesados e identificados los mensajes enviados por el agente, se deben realizar acciones en consecuencia con el tipo de dichos mensajes, como almacenar la información enviada o informar al usuario de un error producido por la ejecución de un comando en un ordenador controlado. El subcomponente Operaciones implementará las funciones que serán ejecutadas cuando se identifique un mensaje.

Los módulos de este subcomponente son los siguientes:

- **MessageProcessor**: Este módulo implementará todas aquellas funcionalidades que permitan procesar el contenido de los diferentes tipos de mensaje

enviados por los agentes.

- **Helper**: Se encuentran en este módulo las funciones auxiliares que se necesiten para el procesamiento de los paquetes.

Mensajes

Como se ha visto en secciones anteriores, el subsistema Agente y el subsistema Controlador se comunican utilizando el protocolo ACP. El subcomponente Mensajes implementará las funciones necesarias para permitir al Controlador manejar los paquetes de este protocolo.

Los módulos de este subcomponente son los siguientes:

- **ApplicationLayer**: Proporciona funcionalidades para ensamblar y desensamblar los paquetes de la capa de aplicación del protocolo ACP.
- **FragmentationLayer**: Permite manejar los paquetes de capa de fragmentación del protocolo ACP.
- **FullLayer**: Permite componer y fragmentar los mensajes utilizando las funcionalidades de las otras clases de este módulo. No se implementa una función que obtenga el mensaje a partir de los fragmentos debido a que este es un proceso que se irá realizando a medida que llegan dichos fragmentos desde el agente y para ello el Listener interactuará directamente con las capas.

Exportador

Este módulo permitirá exportar funciones que serán utilizadas por otros componentes. En este caso las funciones exportadas serán las referentes al protocolo de mensajes ACP.

El motivo de exportar las funciones de creación de fragmentos del protocolo ACP es el siguiente: cuando el usuario indique al Controlador que debe enviar una acción como ejecutar un comando a un agente, el Manejador almacenará dicha acción y deberá componer los fragmentos que deben ser enviados en el momento que el agente solicite la acción. Para componer dichos fragmentos, el Manejador debe poder tener acceso a las funciones del módulo FullLayers, de lo cual se encarga el actual módulo de exportación.

Interacciones

En este apartado se ilustrarán algunas interacciones realizadas entre el subsistema Agente y el subcomponente Listener para mostrar de forma gráfica como se realiza la comunicación con el agente desde la perspectiva del Controlador.

Se verán tres tipos distintos de interacción. El primero será la petición del código del agente por parte del Stager, a continuación se ilustrará como se realiza el *pulling* del agente solicitando acciones para llevar a cabo y por último se pondrá un ejemplo de envío de información desde el agente. Se omite de estos diagramas los pasos de comprobación respectivos de las cookies enviadas y se asume que son correctas con el objetivo de centrarse en la funcionalidad descrita.

El la figura 4.17 se presenta la secuencia que se lleva a cabo cuando el Stager solicita el código de agente para ejecutarlo en el ordenador controlado. En este caso no se utilizan las funcionalidades del protocolo ACP ya que el propio Stager no utiliza este protocolo debido a que sus funciones son reducidas. En esta ocasión simplemente se realiza una petición GET a la ruta */agent* y el Controlador devuelve el código del agente. Para esto el Controlador lee el fichero de implementación del agente y transfiere su contenido al Stager.

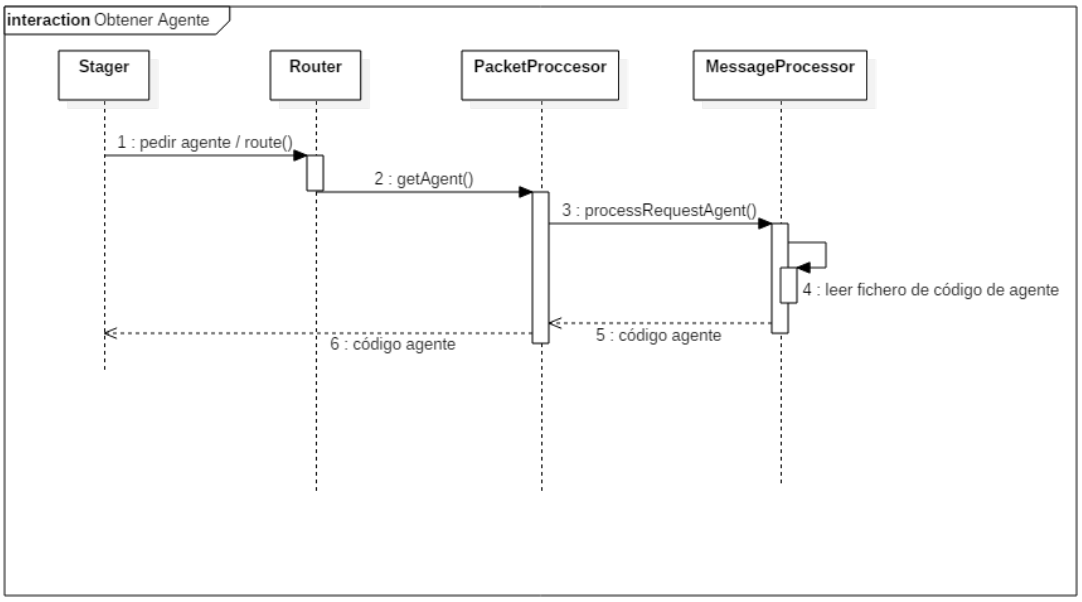

Figura 4.17: Diagrama de secuencia de la petición de un agente

La secuencia de interacciones que realiza el Controlador cuando un agente solicita una acción se muestra en la figura 4.18. Se puede comprobar que una vez realizada esta petición, el servidor consulta al Manejador para obtener el fragmento(protocolo ACP) de la acción que se debe enviar (dichos fragmentos son generados cuando el usuario solicita la ejecución de la acción). El agente debe obtener todos los fragmentos para poder realizar la acción, por lo que esta secuencia podría darse varias veces antes de que un agente obtuviese completamente el paquete que contiene la acción que debe ejecutar.

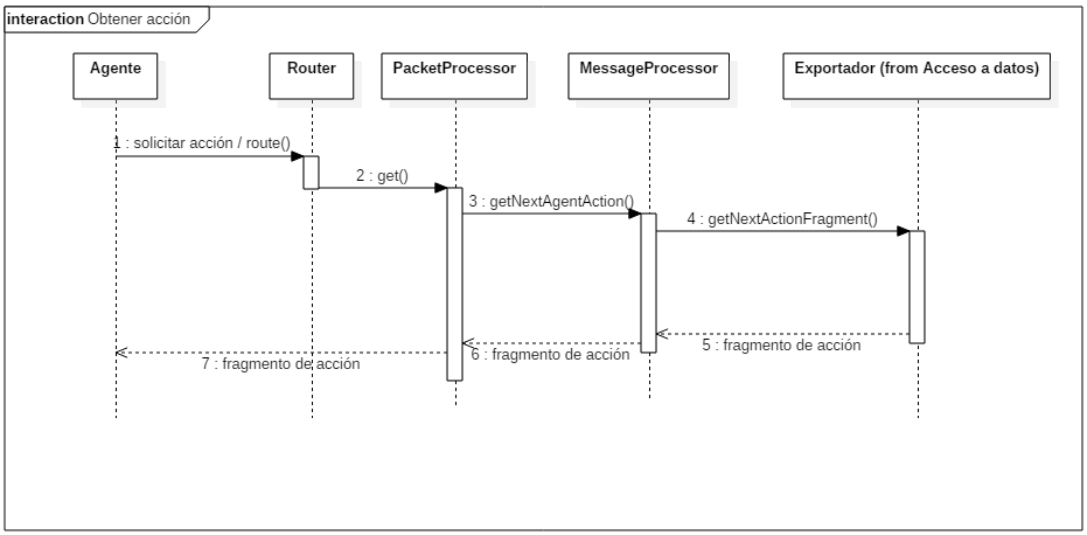

Figura 4.18: Diagrama de secuencia de la petición de una acción para el agente

En último lugar, para ejemplificar los mensajes en los que un agente envía información al Controlador, en los cuales es utilizado el método POST del protocolo HTTP, se muestra en la figura 4.19 los pasos seguidos cuando se recibe la información relativa a una nueva conexión, asumiéndose que el nuevo agente es el único desplegado en su equipo. En este caso una vez recibida la información se le pasa al Manejador para que la almacene y se envía a la terminal un mensaje para indicarle al usuario que un nuevo agente ha sido desplegado. En el paso de descomponer el fragmento se asume que la información está totalmente contenida dentro de un único fragmento. En caso contrario, en este paso habría que devolverle al agente la confirmación de que se ha recibido el fragmento y esperar los siguientes hasta obtenerlos todos para pasar al siguiente paso, que es el de componer el mensaje a partir del contenido de los fragmentos.

De este último diagrama se puede extraer el funcionamiento general de cada uno de los mensajes en que se envía información al Controlador, la única diferencia entre los distintos mensajes de información será la operación a ejecutar una vez recibido el mensaje. Todas las funciones implican el almacenamiento de datos aunque solamente algunas, como las de ejecución de acciones solicitadas por el usuario o el despliegue de un nuevo agente, se enviarán a la pantalla del terminal para no entorpecer al usuario que se encuentre introduciendo comandos.

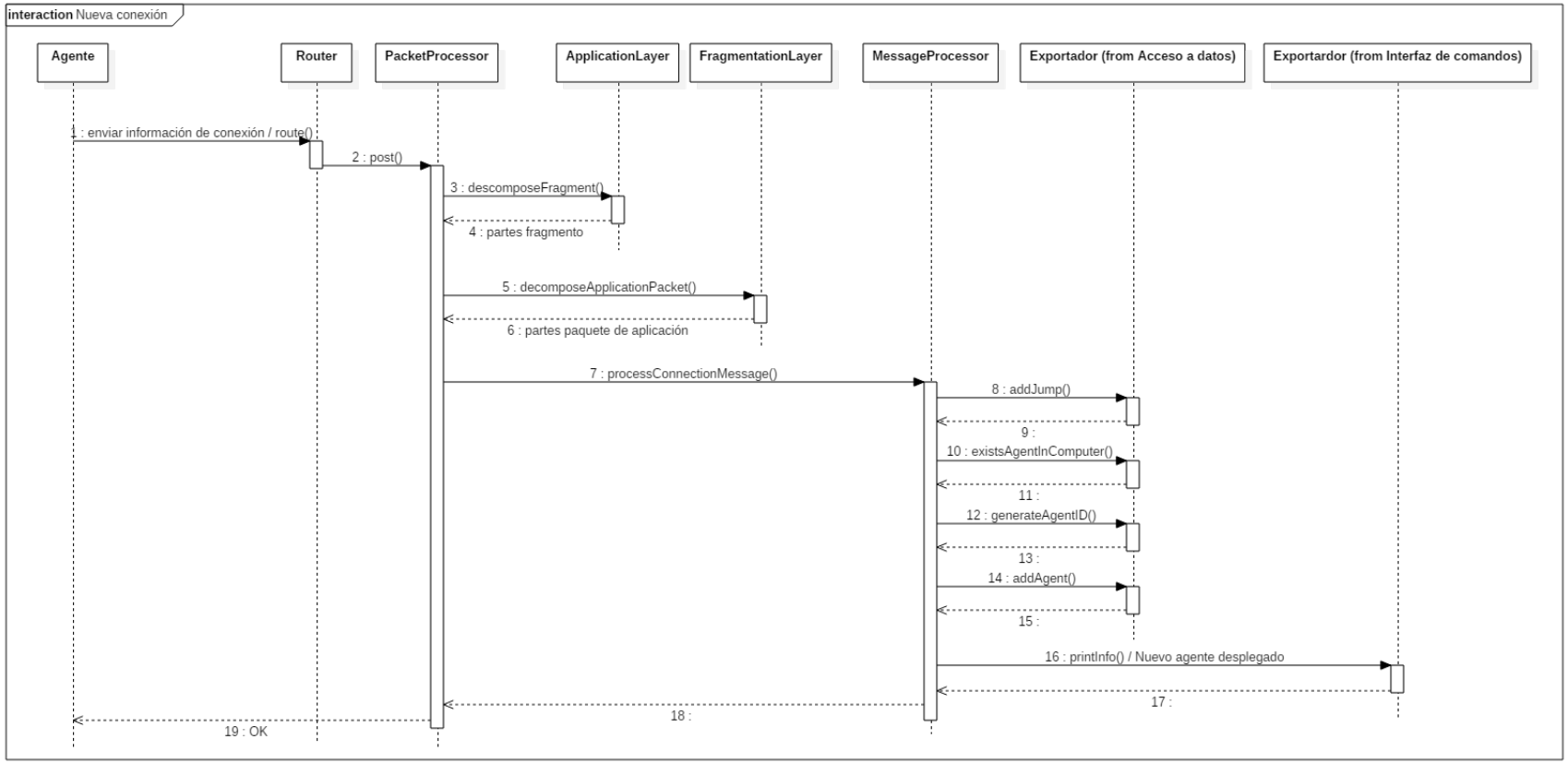

Figura 4.19: Diagrama de secuencia del envío de información de conexión desde un agente

4.5.4. Interfaz de comandos

En este apartado se presenta el componente Interfaz de Comandos, o Terminal para abreviar, que será el encargado de presentar al usuario un menú de comandos que le permita interaccionar con los agentes a través del Controlador.

La línea de comandos ofrecida por este componente se dividirá en los siguientes menús:

- **Menú principal**: Es el menú con el que se iniciará la línea de comandos. Permitirá realizar las funcionalidades generales del Controlador como listar los agentes o ver la ayuda. También permitirá seleccionar un agente para saltar al menú de agente.
- **Menú de Agente**: Este menú ofrecerá los comandos para interaccionar con el agente seleccionado, como ejecutar comando o terminar agente. También permitirá ejecutar las funciones del menú principal.

La arquitectura de este componente se muestra en la figura 4.20 y a continuación se procederá con la explicación de cada uno de sus subcomponentes. Una vez explicados se procederá a mostrar algunos detalles visuales que se tendrán en cuenta a la hora de presentar la información al usuario a través de la terminal. Una guía de los comandos utilizados en la terminal se ofrece en el manual de usuario incluido en los anexos de la presente documentación.

Comandos

En este subcomponente se encontrarán los módulos encargados de realizar las funciones indicadas por el usuario en la línea de comandos.

Los módulos de este subcomponente son los siguientes:

- **Helper**: Este módulo incorporará funciones para extraer y clasificar los argumentos indicados en la línea de comandos y que puedan ser utilizados por las diferentes funciones que ejecutan los comandos.
- **MainCommands**: Este módulo incorporará las funciones que podrán ser ejecutadas desde el menú principal, y por consiguiente desde el menú de agente.
- **AgentCommands**: Se implementarán en este módulo aquellas funcionalidades que serán ejecutadas por el usuario dentro del menú de agente.

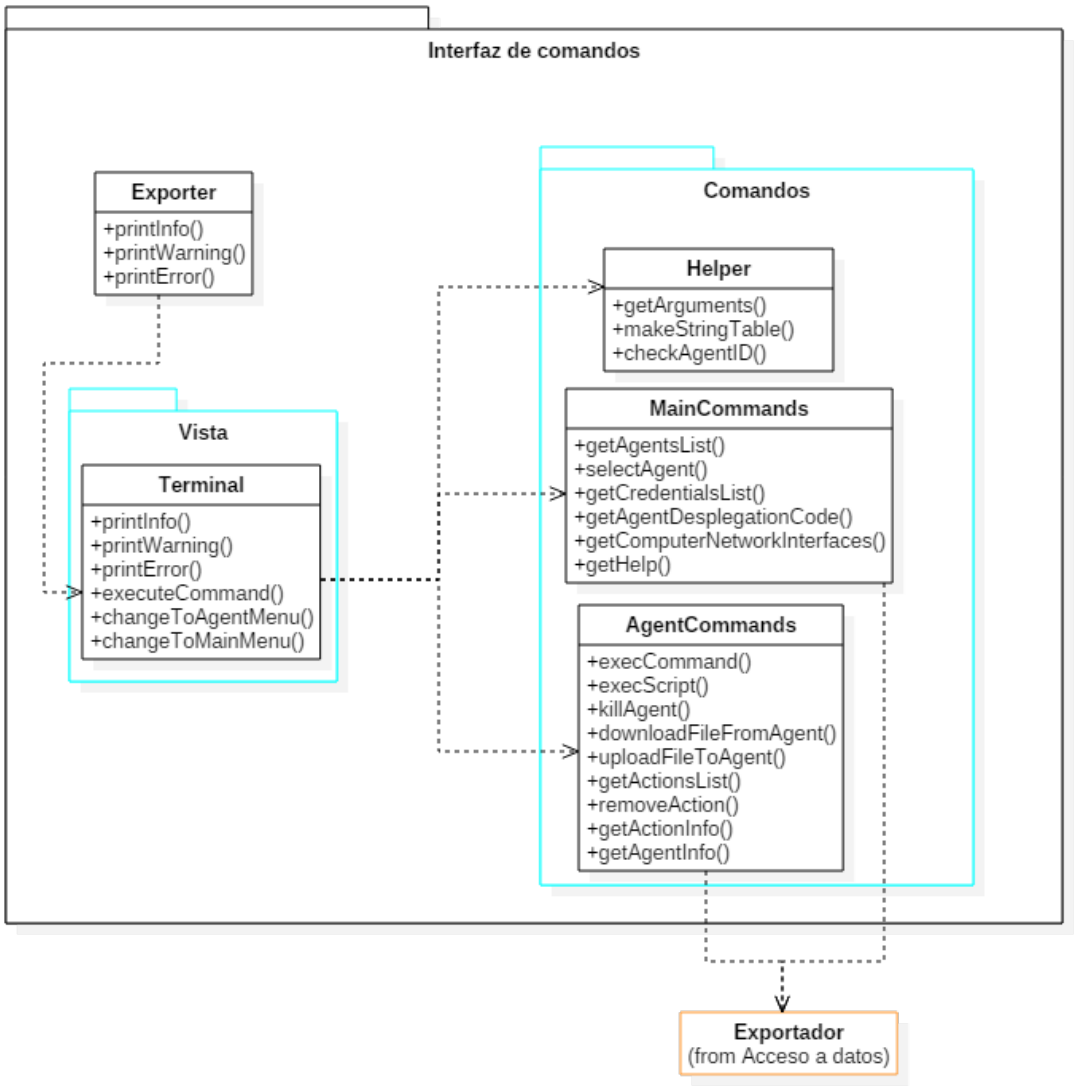

Figura 4.20: Arquitectura del componente Interfaz de Comandos del subsistema Controlador

Vista

El subcomponente Vista será en encargado de presentar al usuario la consola de comandos. Esto implica que será el encargado de transmitirle a este los mensajes provenientes de los demás subcomponentes del Controlador y por consiguiente, los mensajes provenientes de los agentes.

Por otro lado este subcomponente se encargará de recibir la entrada del usuario

y en función de ella, invocar a las distintas funcionalidades del subcomponente Comandos.

Este subcomponente incluye el siguiente módulo:

- **Terminal**: En este módulo se encontrarán las funciones necesarias para recibir la entrada del usuario y mostrar los mensajes de salida por pantalla.

Exportador

Este módulo se encargará de exportar las funciones necesarias para que los demás subcomponentes del Controlador puedan enviar mensajes al usuario.

Detalles visuales

En esta sección se describirá la forma en que se mostrarán los distintos elementos de la interfaz de comandos. Al no tratarse de una interfaz gráfica el grado de manipulación sobre la presentación se encuentra bastante más limitado, no obstante sí se deben diseñar algunos elementos, como las tablas o listas que se usarán para presentar al usuario los datos de objetos como agentes o usuarios.

Lo primero será decidir como se mostrará la línea de comandos. En este caso al existir dos menús distintos se debe incluir un elemento diferenciador que permita al usuario identificar en qué menú se encuentra, y en caso del menú de agentes, a qué agente se encuentra manipulando.

Menú principal:

```
>
```

Menú Agente:

```
Agente1 >
```

El otro elemento a tratar será el cómo se presentan las listas que ofrecen información de los distintos elementos almacenados. A continuación se mostrará un boceto de cómo se presentarán cada una de ellas.

Lista de agentes:

```
  ID      Computer      User         OS           Process     PS
-------  -----------  ---------  -----------  --------------- ---
Agente1  test\EloyPC  test\Eloy  Windows 7    1337\Powershell  2
Agente2  test\PepePC  test\Pepe  Windows 8.1  0007\Powershell  4
```

Lista de credenciales:

```
  Type      Domain/Computer    User     Password
--------  ------------------  -------  ----------
 Domain    test                Eloy     1234Etse
 Local     test\EloyPC         Admin    4dM1n
```

Lista de acciones:

```
 Id      State       Type             Description           Result
-----  ---------  ------------  ------------------------ ------------
  1      Done       Command       whoami                   test\Eloy
  2      Sended     Down. File    C:\Users\Eloy\pass.txt
```

Lista de interfaces e IPs:

```
 Interface        IP              Mask
-----------  --------------  ---------------
 Ethernet     192.168.0.13    255.255.255.0
 VirtualBox   192.168.56.1    255.255.255.0
```

Existe una limitación a la hora de mostrar el resultado de una acción debido a que si este es, por ejemplo, el resultado de listar un directorio, la salida podría ocupar demasiado espacio y producir como consecuencia una tabla poco legible. Debido a esto se optará únicamente por mostrar resultados que ocupen solamente una línea. Sin embargo se incorporará un comando para visualizar los detalles de una acción. Este comando mostrará los detalles de la acción en el siguiente formato:

```
Id: 4
State: Done
Type: Command
Description: ls
Result:
```

```
    Directory: C:\Users\Eloy\Music


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        09/01/2016     01:38        1655539 Compostela.mp3
-a----        19/08/2015     13:37        3280458 El Serrucho.mp3
-a----        04/07/2016     23:12        3413487 Never Gonna Give You Up.mp3
-a----        25/09/2012     10:27       11728323 Thunderstruck.mp3
```

Se debe apuntar que el resultado del comando se mostrará de la misma forma que lo envíe el agente. A pesar de que en el ejemplo se muestre el listado de archivos organizado, esto se debe a que esa es la forma en que Powershell devuelve el listado.

4.5.5. Acceso a datos

El manejo de los datos es esencial en todas las aplicaciones de software. Definir un buen modelo de control para estos puede facilitar en gran medida las tareas de programación y permitir ofrecer un software de mayor calidad. En este caso el manejo de los datos del Controlador se realiza en el componente, como su nombre indica, Acceso a datos, o Manejador para abreviar.

En este apartado, a diferencia de los anteriores no se describirán los módulos uno por uno, ya que los distintos módulos se basan en los mismos modelos de datos utilizados en la Base de Datos que aparecen en la figura 4.2 y que se detallan en la sección correspondiente. Por este motivo se explicarán los subcomponentes de manera general y se detallarán aquellos elementos que puedan diferir de los modelos mostrados en la base de datos o que presenten alguna peculiaridad.

La estructura del Manejador se puede ver en la figura 4.21, en ella se observan tres subcomponentes principales, sin incluir al Exportador, que, al igual que en los otros componentes permite exportar las funciones del Manejador desde un único punto con objeto de que los componentes restantes puedan acceder a ellas. Dos de los subcomponentes, Cache y Base de datos, permitirán el acceso a los datos de la aplicación, mientras que Handler será el encargado de seleccionar de que punto se obtienen estos datos.

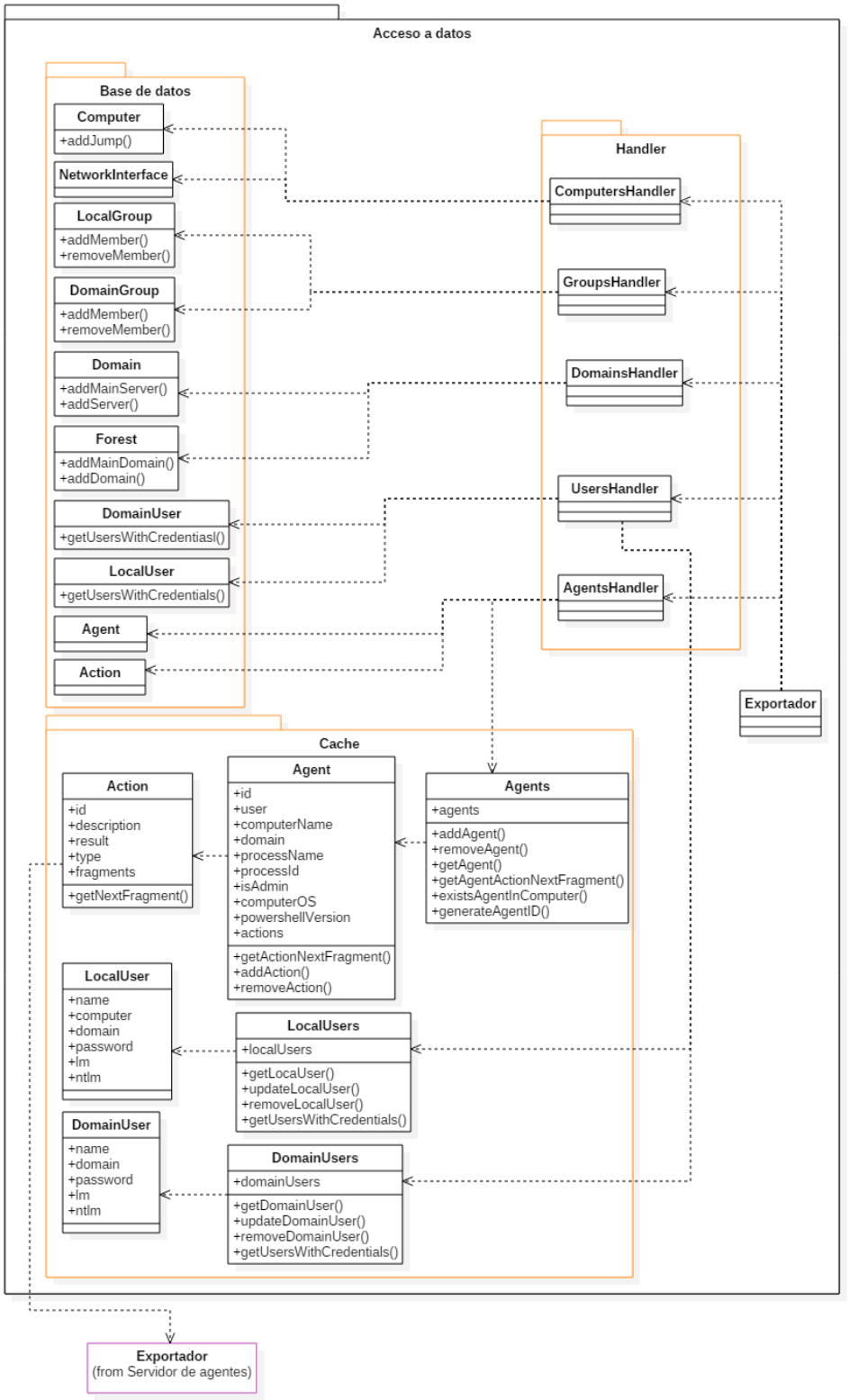

Figura 4.21: Arquitectura del componente Acceso a datos del subsistema Controlador

Se puede observar en el diagrama que ninguno de los módulos presenta demasiadas funciones, esto es porque las funciones CRUD (Create, Read, Update y Delete) se asumen y no se muestran para ahorrar espacio en el diagrama. Entre las funciones CRUD asumidas se encuentra la de devolver todos los objetos de un mismo tipo.

Por otro lado ya que el subcomponente Handler básicamente solo implementará wrappers que se encargarán de elegir si ejecutar las funciones del subcomponente Base de Datos o Cache se entiende que dichos wrappers se llamarán igual que las funciones invocadas y por tanto para mantener más claro el diagrama se omiten dichas funciones en este subcomponente. Del mismo modo el Exportador solamente se encargará de redirigir las llamadas de las funciones hacia las funciones de Handler, con lo que tendrían que volver a copiar el nombre de las funciones en este módulo, aumentando la redundancia y quitando claridad al diagrama, por lo que tampoco se muestran en este las funciones del módulo Exportador.

Base de Datos

En este subcomponente se encontrarán los módulos que permitirán interactuar con los diferentes elementos de la base de datos. En esta sección cuando se quiera hablar del subcomponente se usarán las siglas SBD (Subcomponente Base de Datos) para evitar su confusión con la base de datos con la que interactúa el Controlador a través de este módulo.

Como se ha explicado previamente, todos los módulos del SBD deberán implementar las funciones CRUD y las especificadas en el diagrama para cada uno de ellos. Si se comparan las figuras 4.2, que muestra el modelo Entidad-Relación, y 4.21, que muestra la arquitectura del Manejador, se puede observar que solamente algunas relaciones de la Base de Datos se ven reflejadas en los módulos del SBD. Esto es debido a que muchas de las relaciones son implícitas a la hora de crear los diferentes elementos, especialmente en aquellos casos, como el de los usuarios, que su clave primaria depende del objeto, en este caso ordenador o dominio, con el que se le relaciona. Otro caso es el de los agentes, que a pesar de no tener una dependencia de clave con el ordenador en que se encuentran, cada agente debe estar incluido dentro de un ordenador.

El caso comentado anteriormente no se aplica para los dominios y los bosques, aunque todo dominio pertenece a un bosque, por dos razones. La primera es que esta no es una de las relaciones críticas que se deben modelar en la base de datos, siendo más importantes otras como la dada entre los agentes y los ordenadores o los ordenadores con el dominio. Como consecuencia de esto durante la ejecución del agente la información del dominio se enviará con antelación a la del bosque,

ya que se informará del dominio del agente en el mensaje de conexión o salto mientras que el bosque del dominio será enviado en el mensaje de información de dominios. Esto hace que se necesite una función para añadir a cada bosque, que durante la expansión será solamente una, los dominios que contiene. Del mismo modo sucede con los dominios y sus servidores.

Se debe hacer notar que esta capa añade una importante modularidad y portabilidad al producto ya que, en caso de cambiarse la tecnología de base de datos utilizada en un futuro, solamente se debería modificar este componente para adaptarlo al nuevo sistema gestor de base de datos.

Cache

El subcomponente Cache es el encargado de la gestión de los datos que se guardarán en la memoria del Controlador. Con esto se consiguen dos beneficios, el primero y requerido por los requisitos es permitir al Controlador almacenar y mantener información del proceso de expansión aún en caso de no estar conectado a la base de datos. Por otro lado en caso de caso de tener conexión con la base de datos este subcomponente permite cargar los elementos de esta en memoria para acelerar el acceso a ellos y evitar tener que consultar dicha base de datos, lo cual consume tiempo de computación y ancho de banda en la red.

Algo que se debe mencionar de este subcomponente es que es él único del Controlador que implementará clases en vez de estar compuesto por módulos. Esto es debido a que estas clases representarán los elementos que se deben almacenar en memoria y de las cuales, a diferencia de los módulos en el resto del Controlador, se crearán instancias que variarán de forma notable a lo largo de la ejecución del programa.

En el diagrama de arquitectura del componente Manejador (figura 4.21) se puede observar que este subcomponente no cuenta entre sus clases con todos los elementos que se pueden encontrar en la base de datos, solamente con usuarios, agentes y acciones. Esto es debido a que estos objetos, según los casos de uso (o requisitos funcionales), serán los únicos que manejará el usuario desde la Terminal. La consecuencia de esto es que mucha de la información recogida por los agentes se desechará si el Controlador no se encuentra conectado a la base de datos. Sin embargo, debido al limitado tiempo para realizar el proyecto, esto es asumible ya que no se espera que ese sea el funcionamiento normal del aplicativo.

Por otro lado destaca la dependencia de uso que mantiene la clase Action con el componente Servidor. Esto es debido a que en el momento de crear una nueva acción se construirá y fragmentará el mensaje que se debe enviar al agente para

que realice dicha acción, y por este motivo la clase Action debe poder acceder a las funciones del protocolo ACP que se encuentran en el componente Servidor.

Handler

El subcomponente Handler incorpora módulos que permitirán manejar el flujo de información de la aplicación y dirigirlo hacia la base de datos o hacia la cache del Controlador, para mantener la información en ambas partes actualizada y acceder rápidamente a esta.

Sin embargo como se puede visualizar en la figura 4.21, solamente dos de los módulos del subcomponente Handler permitirán acceder a la cache, siendo el resto dependientes de la base de datos. Esto es debido a que, como se ha comentado en el apartado anterior, el Controlador solamente almacenará en cache aquellos objetos que el usuario pueda manipular desde la línea de comandos. El resto de módulos deben devolver un error cuando no se tenga conexión a la base de datos.

El funcionamiento general de los módulos que pueden acceder a la cache será el siguiente: cuando se inicie el Controlador deben cargar los datos de la base de datos en la cache. Durante la ejecución, en caso de lectura se debe acceder a la cache para coger datos, y en caso de escritura se deben escribir los datos tanto en cache como en la Base de Datos. Este comportamiento cuenta con la desventaja de necesitar almacenar mucha información de la base de datos en la cache, pero esto es admisible ya que no se espera que durante una expansión se genere un número excesivamente grande de datos. Por otro lado se ha elegido este esquema de funcionamiento ya que permite de una forma sencilla mantener actualizados los datos de cache y base de datos, siempre y cuando el Controlador sea el único que inserte datos en la base de datos, lo cual se cumple en este sistema. Además un factor decisivo para optar por este funcionamiento ha sido el limitado tiempo del que se dispone para realizar el proyecto.

4.5.6. Diagramas de secuencia de casos de uso

En el segundo capítulo del presente documento se exponen los casos de uso que el sistema debe poder llevar a cabo. Estos se encuentran divididos en dos partes: los casos del subsistema Controlador de agentes y los del subsistema Interfaz de datos.

En este apartado, una vez se ha visto como está compuesto el Controlador, se procederá a mostrar la secuencia de instrucciones que se debe seguir para los

casos de uso que se desarrollan cuando el usuario interactúa con este subsistema mediante los correspondientes diagramas de secuencia.

Debido a que muchos casos de uso siguen una secuencia de instrucciones bastante similar entre ellos, se agruparán los casos de uso parecidos, se mostrará el diagrama de un caso de uso y se indicará en que difieren los casos del mismo grupo no representados.

Casos de uso: CU101

Se muestran en este apartado el diagrama de secuencia del siguiente caso de uso:

- **CU101**: Desplegar agente

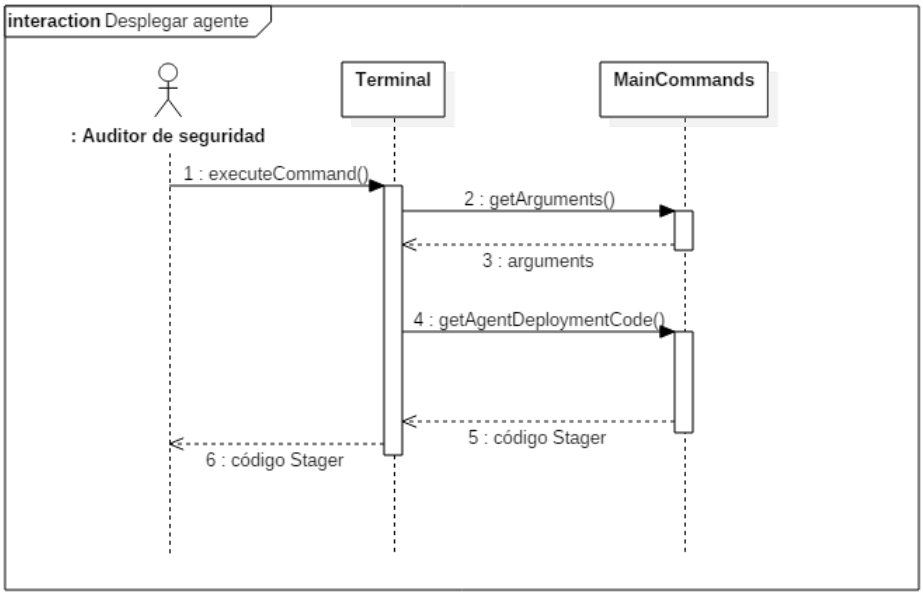

Figura 4.22: Caso de uso - Desplegar agente

En la figura 4.22 se muestra como el Controlador genera el Stager, que se usará para instalar un agente en un ordenador ajeno. Sin embargo este caso de uso no se encuentra completo en esta representación. Una vez que el auditor de seguridad dispone del Stager debe copiar este en la consola de Powershell del ordenador en el cual se quiere desplegar el agente. Esto podría resultar difícil e incluso incomprensible para un usuario que no disponga de ciertos conocimientos técnicos, sin

embargo esto no es un problema en el caso de los auditores de seguridad a los que va orientada esta aplicación, que sí disponen de este conocimiento.

Este caso de uso concluiría una vez ejecutado el Stager en la máquina remota elegida. En la figura 4.12 se muestra la secuencia de instrucciones seguidas por el Stager que concluyen con el agente desplegado.

Casos de uso: CU102, CU109, CU110, CU111

Se muestran en este apartado los siguientes casos de uso:

- **CU102**: Listar agentes
- **CU109**: Listar credenciales
- **CU110**: Listar IPs
- **CU111**: Listar acciones de agente

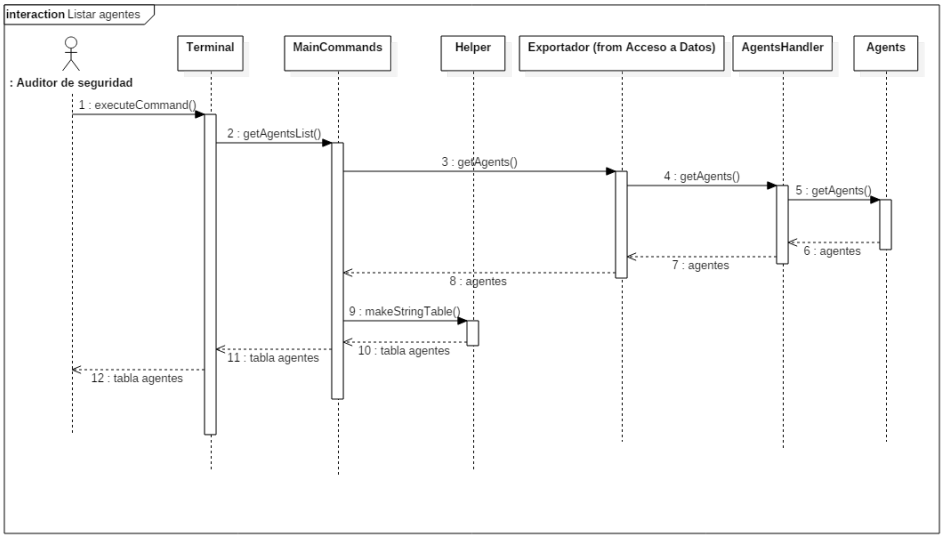

Figura 4.23: Caso de uso - Listar agentes

En la figura 4.23 se muestra la secuencia de instrucciones que seguirá el subsistema Controlador para listar los agentes desplegados (CU102). Como se puede ver el componente Interfaz de comandos solicitará al Manejador los datos de los agentes,

luego realizará una transformación para obtener los datos en formato de tabla para finalmente mostrárselos al usuario.

Los casos de uso CU109 y CU111 se realizarán de manera similar, salvo que en lugar de pedir los datos de los agentes, se solicitarán al Manejador los datos de las acciones o credenciales según convenga. En el caso de las credenciales los datos pedidos serán los de todos aquellos usuarios de los que se haya obtenido la contraseña.

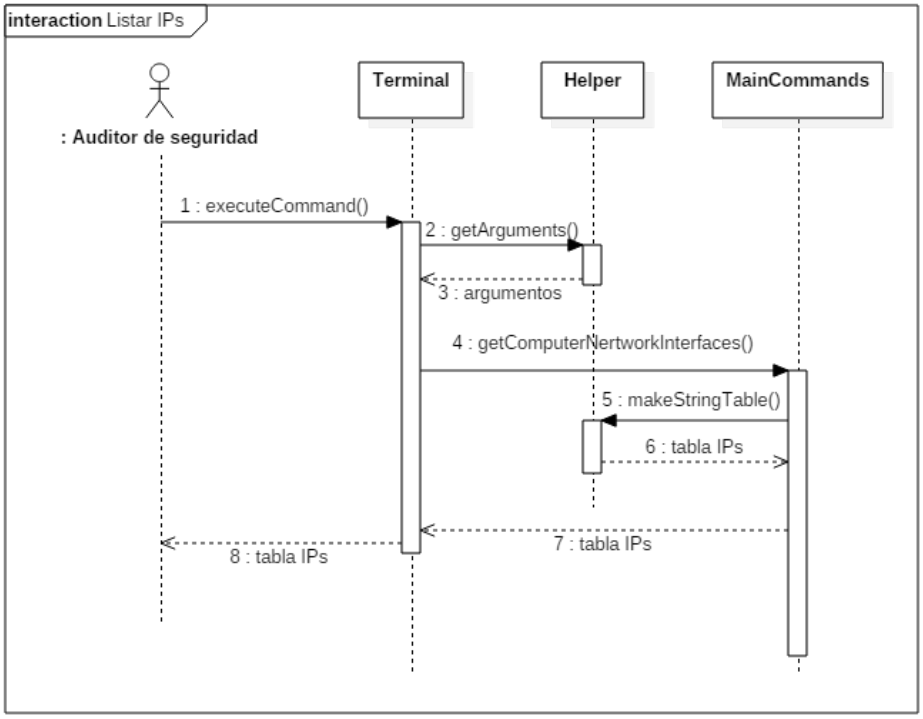

Figura 4.24: Caso de uso - Listar IPs del sistema

Un caso un poco diferente es el CU110, ya que no requerirá de acceso al Manejador de datos. El Controlador simplemente realizará una llamada al sistema operativo para obtener las interfaces y sus respectivas IPs y tras la generación de la tabla con los datos, los devolverá al usuario por la terminal.

Casos de uso: CU103

Se muestran en este apartado el diagrama de secuencia del siguiente caso de uso:

- **CU103**: Seleccionar agente

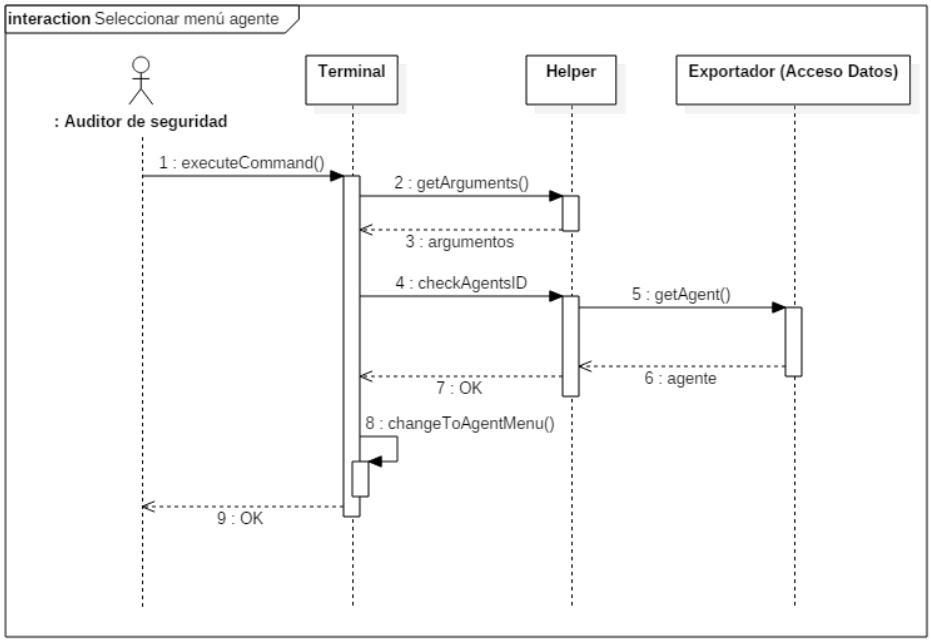

Figura 4.25: Caso de uso - Seleccionar menú de agente

En la figura 4.25 se muestra la secuencia de pasos que se realizan para cambiar de menú. Se comprobará que el identificador de agente es correcto y en ese caso se cambiará al menú del agente deseado por el usuario.

Casos de uso: CU104, CU105, CU106, CU107, CU108

Se muestran en este apartado los siguientes casos de uso:

- **CU104**: Eliminar agente
- **CU105**: Ejecutar comando en agente
- **CU106**: Ejecutar script en agente
- **CU107**: Descargar archivo
- **CU108**: Subir archivo

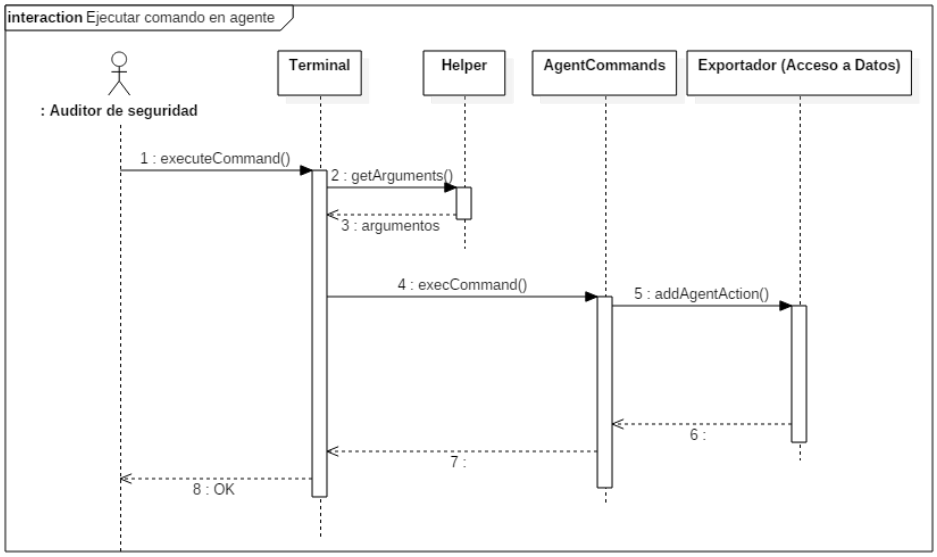

Figura 4.26: Caso de uso - Ejecutar comando en agente

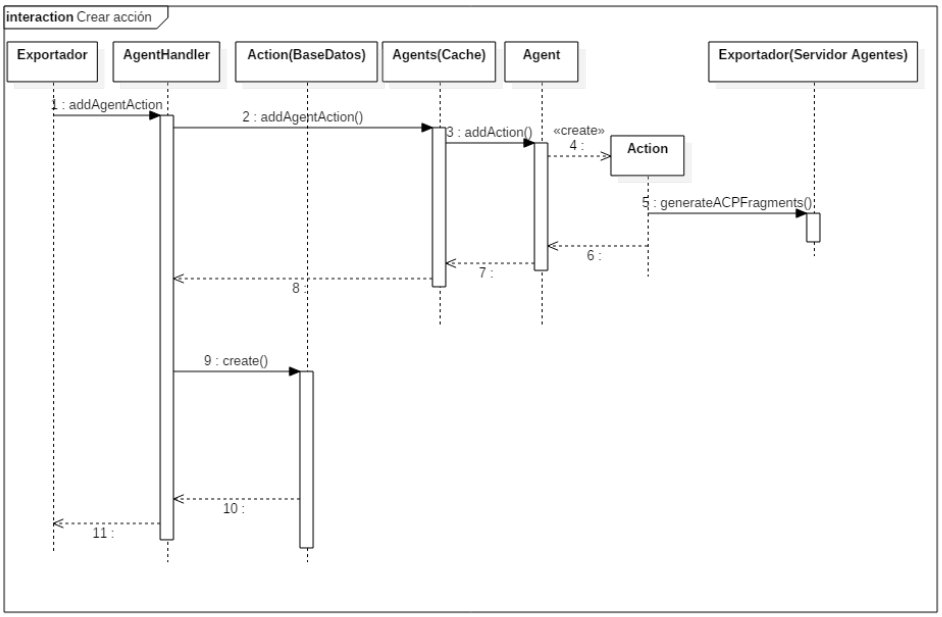

Figura 4.27: Caso de uso - Crear nueva acción para agente

En la figura 4.26 se muestra las acciones que se llevarán a cabo para ejecutar un comando en el agente. Los casos de uso de este apartado tienen en común que para comunicarle al agente lo que se desea de él se deberá añadir crear una acción

que el Controlador se encargará de transmitirle. En la figura 4.27 se muestra como se añade una nueva acción a un agente.

La diferencia entre los distintos casos de uso radica en el comando que se ejecutará en la Terminal, aunque todos ellos tienen un comportamiento similar. Sin embargo en el caso de eliminar agente, la acción de terminación deberá colocarse la primera de la lista de acciones pendientes, con objeto de que sea la siguiente en ejecutarse.

Estos casos de uso no terminan con la definición de una acción, sino que una vez definida esta se debe esperar a que el agente solicite acciones, lo que se puede ver en la figura 4.18, y tras realizarlas que devuelva el resultado, de manera similar a como se muestra en la figura 4.19.

4.6. Subsistema Interfaz de datos

En este apartado se realiza una descripción más en detalle del subsistema Interfaz de datos, que es el encargado de presentar al auditor de seguridad los datos recogidos durante el proceso de expansión de los agentes, mostrando los elementos y relaciones recogidas en la base de datos.

4.6.1. Tecnologías empleadas

Como en apartados anteriores, se indican en este caso las principales tecnologías empleadas para el desarrollo del subsistema Interfaz.

ReactJS

ReactJS[18] o React es una librería de Javascript orientada al desarrollo de interfaces web, especialmente de aquellas que constan de tan solo una página, como es el caso actual.

Los componentes de React suelen estar escritos en JSX. JSX[19] es una extensión que permite definir los distintos componentes en un formato similar al HTML, mediante etiquetas.

Electron

Electron[20] es un framework basado en Nodejs que permite crear aplicaciones de escritorio utilizando tecnologías web, como HTML, CSS y Javascript.

Electron permite implementar toda la interfaz como si de una página web se tratase añadiendo finalmente una capa que simula el navegador web Chromium, que se encarga de renderizar la página web, convirtiendo esta en una aplicación de escritorio.

BloodHound

BloodHound[41] es un software que permite visualizar en forma de grafo algunos elementos y relaciones existentes en una red de Active Directory.

En este proyecto se han reaprovechado componentes de BloodHound para la implementación del subsistema Interfaz de datos. Como consecuencia de esto se ha añadido a la herramienta TrueniX la licencia de software libre GPL, presente en los anexos del documento, ya que es el tipo de licencia poseído por BloodHound.

4.6.2. Arquitectura de Interfaz

El subsistema de Interfaz de datos es el encargado de mostrar una representación gráfica y comprensible de los datos de la red de Active Directory recogidos por los agentes durante el proceso de expansión. En la figura 4.28 se muestra la arquitectura general del subsistema.

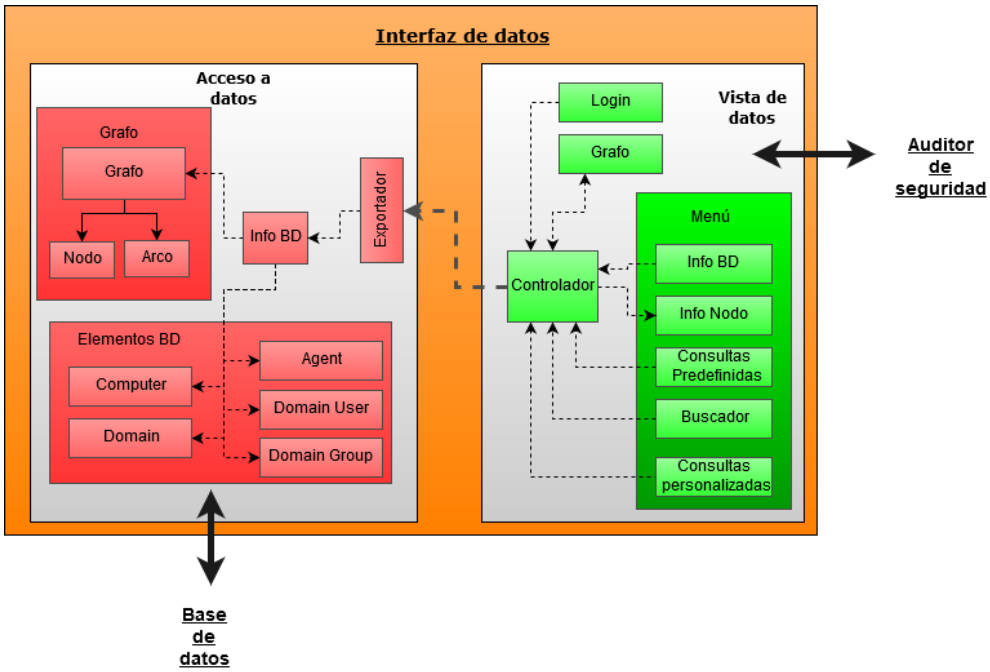

Figura 4.28: Arquitectura del subsistema Interfaz de datos

En la arquitectura es posible distinguir dos componentes principales y diferenciados:

- **Acceso a datos**: Encargado de realizar las consultas a la base de datos y procesar la información. Incluye un módulo Exportador que permite a la Vista interaccionar con este. Se denominará a este componente Acceso en los posteriores apartados de esta sección.
- **Vista de datos**: Responsable de interaccionar con el usuario atendiendo a sus solicitudes y mostrando la información requerida. En este caso se

incluye un módulo Controlador que permite redirigir la salida de datos hacia los diferentes subcomponentes. A partir de este punto para referirse a este componente se utilizará el alias Vista.

En los siguientes apartados se profundiza en los detalles de los subcomponentes y comportamientos de este subsistema.

4.6.3. Acceso a datos

El componente Acceso es el encargado, como se ha explicado anteriormente, de realizar las consultas a la base de datos, recuperar la información de esta y transformarla a un formato que posteriormente permita al componente Vista representarla.

Se muestra en la figura 4.29 la arquitectura del componente Acceso, en la cual se pueden distinguir cuatro elementos principales que se explican a continuación.

Grafo

El subcomponente Grafo es el encargado de realizar las consultas cuyos resultados posteriormente se presentarán en forma de grafo. Consta de un módulo Grafo y dos estructuras, Nodo y Arco.

El módulo Grafo realiza las consultas a la base de datos. Una vez obtenidos los resultados debe procesarlos para extraer los nodos y las relaciones presentes en estos y almacenar la información de estos en las estructuras Nodo y Arco, que se le enviarán al componente Vista para que este pueda realizar una presentación gráfica de los mismos.

Elementos BD

El subcomponente Elementos BD almacena los diferentes módulos que permiten obtener información específica y detallada de cada uno de los elementos presentes en la base de datos.

Como se puede observar en el diagrama de la figura 4.29, cada uno de los módulos de Elementos BD puede extraer, o bien la información detallada de un tipo de nodo, o bien una lista con los identificadores de cada uno de los nodos de la base

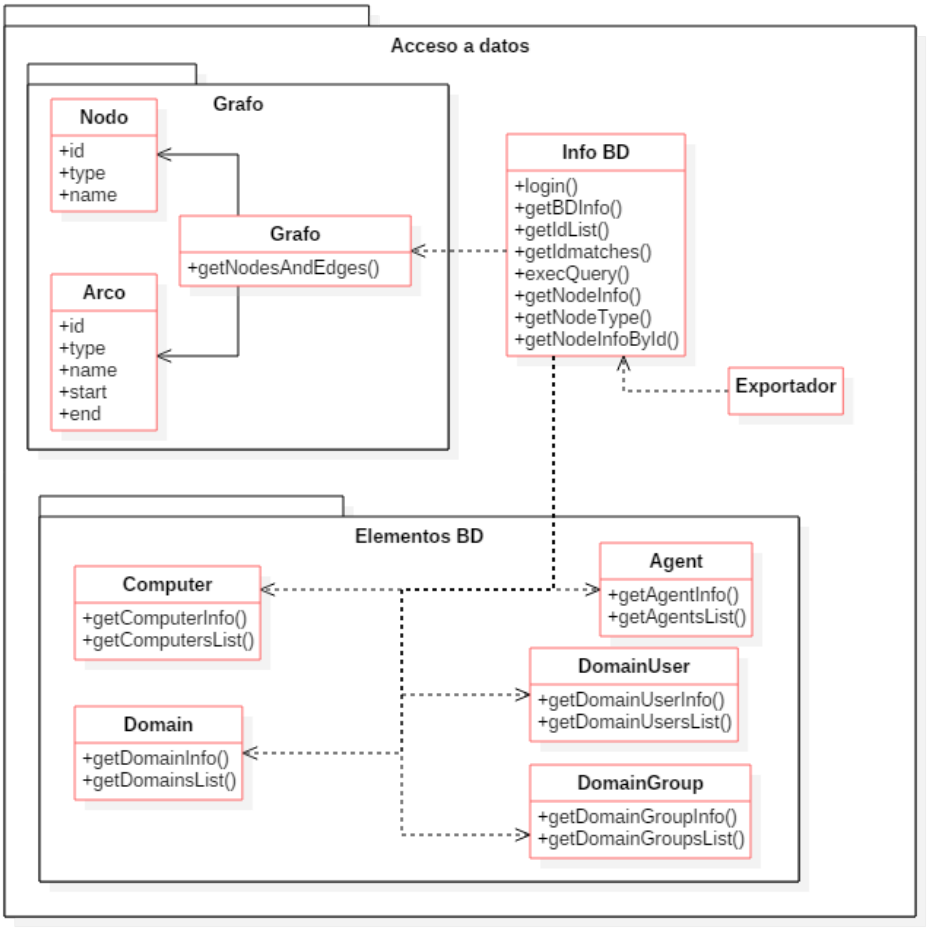

Figura 4.29: Arquitectura del componente Acceso a datos del subsistema Interfaz de datos

de datos. Esto último permite que el usuario pueda elegir, por ejemplo, el grupo para el que desea consultar los miembros.

Info BD

Info BD es el módulo central del componente Acceso y se encarga de realizar consultas a la base de datos para obtener información general de esta.

Por otro lado, en caso de recibir consultas específicas a los datos, se encargada de estudiar la consulta y redirigirla hacia el subcomponente correspodiente que debe resolverla.

Exportador

Permite exportar las funciones que serán utilizadas desde el componente Vista. En principio se exportan todas las funciones indicadas de Info BD. Debido a esto y para no añadir redundancia y mantener la claridad del diagrama de la figura 4.29 no se vuelven a indicar en el módulo Exportador todas las funciones.

4.6.4. Vista de datos

Uno de los puntos donde el usuario interacciona con el sistema es el componente Vista de datos. Este componente se encarga de recoger las interacciones del usuario y realizar peticiones al componente Acceso, que le devolverá los datos que Vista debe presentar por pantalla al usuario.

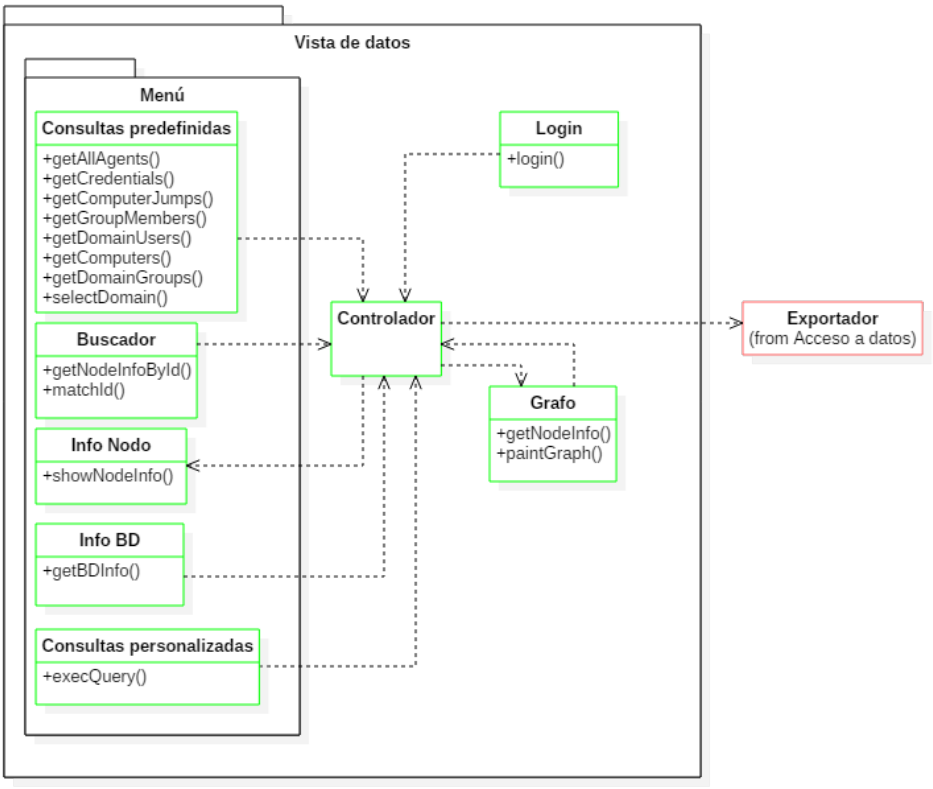

Figura 4.30: Arquitectura del componente Vista de datos del subsistema Interfaz de datos

En la figura 4.30 se muestran los distintos subcomponentes de Vista. Cada uno

de ellos, a excepción del módulo Controlador, se encargan de la representación visual de cada uno de los elementos de la interfaz, que es descrita en un apartado posterior.

A continuación se procede a describir los elementos de este componente.

Login

El módulo Login es el encargado de recoger la información inicial para realizar la autenticación contra la base de datos, de forma que se pueda establecer una conexión con esta para la consulta de datos.

En este caso, a diferencia del subsistema Controlador, la autenticación con la base de datos se realiza de manera interactiva ya que esta es necesaria para el funcionamiento de este subsistema, a diferencia del subsistema Controlador, donde las credenciales se especifican en un fichero (como se explica en el anexo de la presente documentación) para evitar solicitarlas incluso cuando la base de datos se encuentre desconectada.

Grafo

El módulo Grafo, como su nombre indica, es el responsable de representar la información consultada en forma de grafo. Para esto recibe los nodos y arcos que deben ser mostrados y se encarga de la definición de sus atributos físicos como posición o color.

Por otro lado es el encargado de invocar las consultas sobre los datos de los nodos que el usuario seleccione.

Menú

El subcomponente Menú contiene a los módulos, de menor categoría en la representación gráfica que el módulo Grafo, que permiten al usuario interactuar con la interfaz.

Se excluye el módulo Login ya que este se encarga de la autenticación previa a la interacción objetivo con el usuario.

En este subcomponente se incluyen los siguientes módulos:

- **Consultas predefinidas**: Permite al usuario la realización de una serie de consultas ya preparadas para extraer información útil del proceso de expansión y de la red de Active Directory donde se lleva a cabo este.

- **Consultas personalizadas**: Permite al usuario insertar una consulta en el lenguaje propio de la base de datos, que retornará un grafo que represente a las entidades seleccionadas en la consulta, y sus relaciones.

- **Buscador**: Permite al usuario insertar el nombre identificador de un elemento para consultar su información.

- **Info BD**: Permite consultar información general de la base de datos. En concreto el número de elementos de cada tipo (Agentes, Ordenadores, Usuarios, etc.) que se encuentran almacenados, así como el nombre de usuario y URL utilizados para la conexión con la base de datos.

- **Info Nodo**: Muestra la información detallada del nodo consultado.

Controlador

El módulo controlador es el módulo de gestión principal del componente. Debido a que en muchas ocasiones la entrada y salida de información se realizan utilizando distintos módulos, se ve la necesidad de incluir un módulo que se encargue de dirigir y recibir la información hacia el módulo de representación adecuado.

El módulo controlador se encarga de recibir la entrada del resto de módulos del componente e interactuar con el componente Acceso a datos. Asimismo una vez recibido el resultado de la consulta solicitada, se encarga de transmitir esta información al módulo adecuado para que se encargue de mostrarla por pantalla.

4.6.5. Diagramas de secuencia de casos de uso

En esta sección se presentan los diagramas de secuencia que muestran los flujos de operaciones desarrollados en los casos de uso relacionados con el subsistema Interfaz de datos.

Debido a que algunos casos de uso siguen secuencias de instrucciones similares unos a otros, se han agrupado los casos de uso por parecido. Para cada grupo se muestra un diagrama de ejemplo representando uno de los casos y se indica en difieren los restantes del grupo.

Casos de uso: CU201, CU202, CU203, CU204

Se muestran en este apartado los siguientes casos de uso:

- **CU201**: Realizar consulta personalizada
- **CU202**: Consultar agentes
- **CU203**: Consultar credenciales
- **CU204**: Consultar saltos entre ordenadores

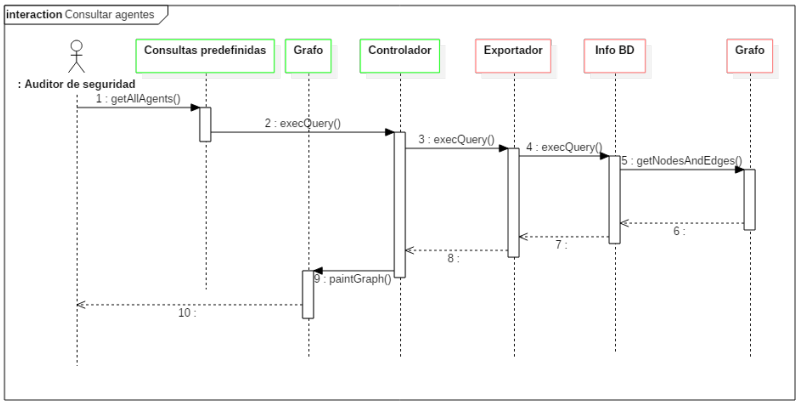

Figura 4.31: Caso de uso - Consultar agentes

En la figura 4.31 se muestra la secuencia de operaciones seguidas en el caso de uso CU202. En este caso el usuario selecciona la consulta para mostrar los agentes, por tanto se envía a la base de datos y con los datos de la respuesta se dibuja el grafo que contiene los nodos de los agentes. En los casos CU203 y CU204 la diferencia radica en la consulta que selecciona el usuario, siendo el procesamiento de esta el mismo que en el caso mostrado.

Sin embargo en el caso CU201 es el usuario el que introduce la consulta, invocando para esto al módulo Consultas personalizadas como se muestra en la figura 4.32.

Casos de uso: CU205, CU206, CU207, CU208

Se muestran en este apartado los siguientes casos de uso:

- **CU205**: Consultar miembros de un grupo del dominio

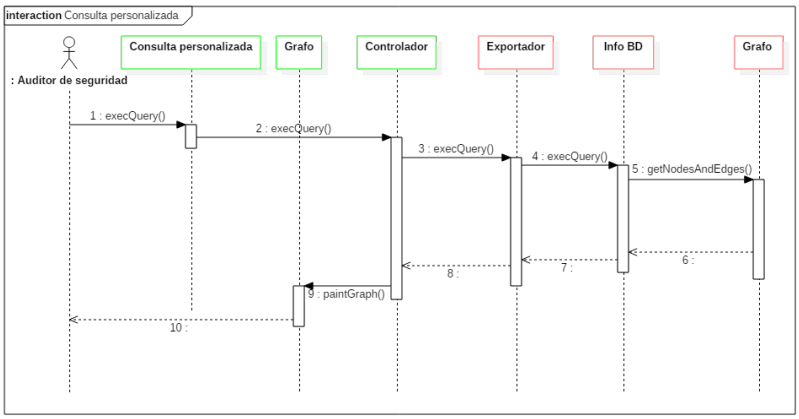

Figura 4.32: Caso de uso - Realizar una consulta personalizada

- **CU206**: Consultar usuarios de un dominio
- **CU207**: Consultar ordenadores de un dominio
- **CU208**: Consultar grupos de un dominio

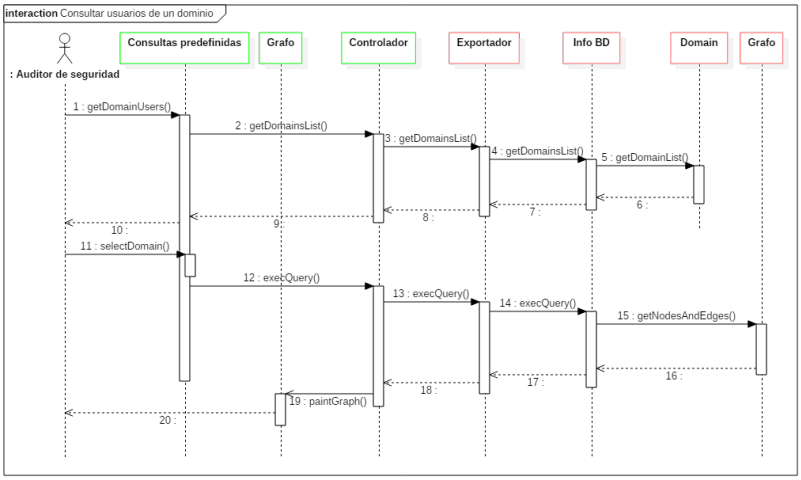

Figura 4.33: Caso de uso - Consultar los usuario de un determinado dominio

Este grupo de casos de uso guardan semejanza con el grupo anterior ya que se trata de consultas predefinidas que el usuario selecciona. Sin embargo como se puede apreciar en la figura 4.33, que representa el caso de uso CU206, en esta ocasión se realiza una operación previa que permita al usuario seleccionar el dominio o grupo del que quiere conocer los miembros. La diferencia de los casos de uso CU207 y CU208 radica en la consulta seleccionada por el usuario. En el

caso de uso CU205 además de cambiar la consulta, el usuario debe escoger un grupo, no un dominio, sobre el que consultar los miembros.

Casos de uso: CU209, CU210, CU211

Se muestran en este apartado los siguientes casos de uso:

- **CU209**: Consultar datos de agente
- **CU210**: Consultar datos de ordenador
- **CU211**: Consultar datos de usuario del dominio

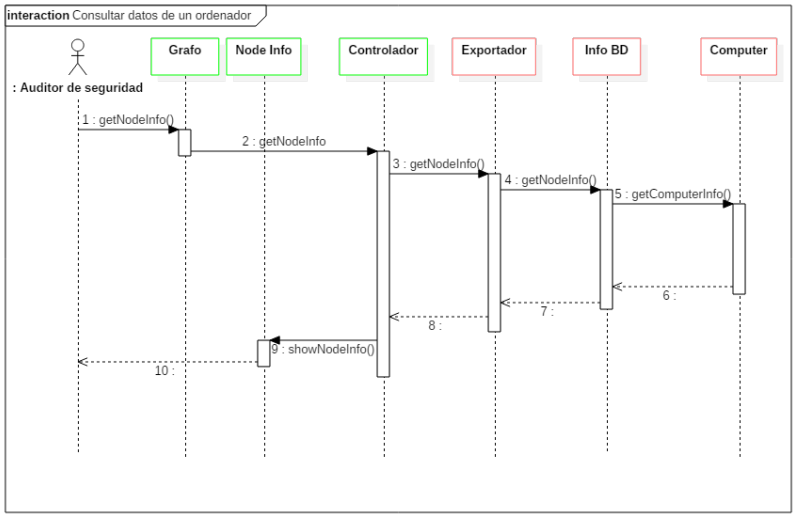

Figura 4.34: Caso de uso - Consultar los datos de un ordenador

En la figura 4.34 se muestra el proceso seguido para obtener los datos de un ordenador a partir de la selección de un nodo del grafo por parte del usuario, lo que corresponde al caso de uso CU210. En los casos de uso CU209 y CU211 una vez llegada la petición a Info BD, este detecta que el tipo de nodo solicitado es un agente o un usuario, por lo que invoca al módulo Agent o DomainUser, respectivamente.

Casos de uso: CU212

Se muestra en este apartado el diagrama de secuencia del siguiente caso de uso:

- **CU212**: Búsqueda por nombre

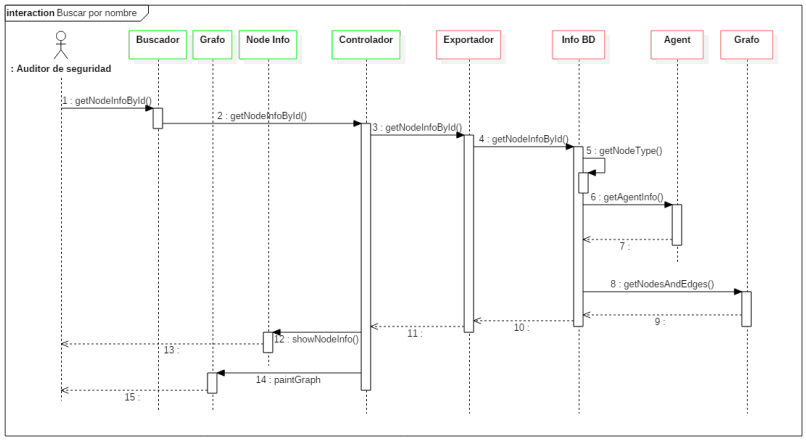

Figura 4.35: Caso de uso - Buscar información de un elemento por el nombre identificativo

En la figura 4.35 se presenta la secuencia de acciones desarrollada para llevar a cabo el CU212. Se muestra que tras introducir el nombre, se consulta la base de datos para obtener tanto el nodo, como los detalles específicos del elemento buscado.

4.6.6. Diseño de la interfaz gráfica

En este apartado se muestra el diseño visual de la interfaz gráfica. El objetivo principal que persigue la interfaz es que el usuario pueda visualizar los datos recogidos por los agentes durante el proceso de expansión en forma de grafo, de una manera intuitiva.

Por tanto, como se muestra en la figura 4.36, la mayor parte del espacio de la ventana se dedica a la representación del grafo, de forma que se pueda mostrar de manera cómoda el mayor número de elementos posible.

Un detalle que puede no apreciarse del todo en la figura 4.36 es que la información de los nodos y la base de datos se mostrará en el mismo componente. El usuario será el encargado de elegir qué datos mostrar seleccionando para ello la pestaña correspondiente.

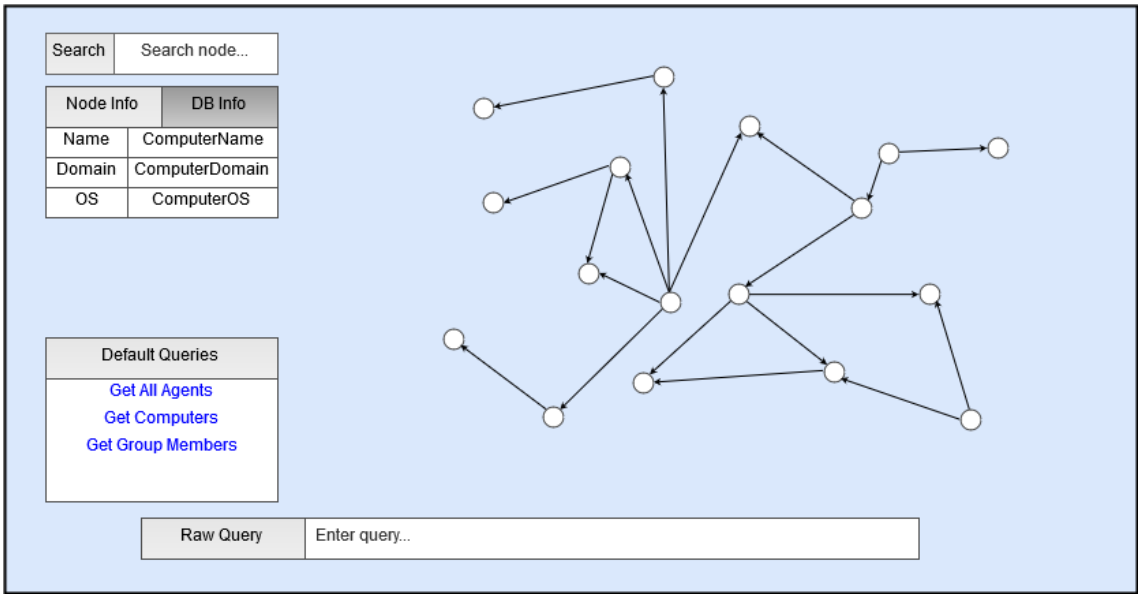

Figura 4.36: Vista general de la interfaz gráfica

Capítulo 5

Pruebas

Durante todo el proyecto se han ido realizando pruebas en la finalización de cada uno de los sprints. Además, una vez alcanzada la fase final del desarrollo del proyecto se han llevado a cabo las pruebas finales. De este modo se ha verificado que el producto cuenta con las funcionalidades deseadas.

En este capítulo se presentan todas las pruebas realizadas que han permitido validar el producto.

Secciones del capítulo:

1. **Pruebas de sprints**
2. **Pruebas finales**
3. **Evaluación de la usabilidad**

5.1. Pruebas de sprints

La última fase realizada en cada uno de los sprints ha sido verificar, junto al cliente, que en dicho sprint se habían implementado correctamente los requisitos funcionales acordados. Además, debido a que existen algunos requisitos de diseño fuertemente relacionados con algunos sprints, también se han incluido estos en las pruebas de final de sprint.

En esta sección se presentan, divididas en sprints, las pruebas que se han llevado a cabo para verificar cada uno de los requisitos.

5.1.1. Sprint 1

PR101 - Listar agentes

Descripción: El Controlador debe permitir al usuario obtener una lista de los agentes desplegados.

Requisitos involucrados: RF102

Validación: Ejecutar el comando de listado de agentes y ver que se obtiene una lista con los detalles de estos.

Estado: Cumplido

PR102 - Seleccionar agente

Descripción: El Controlador debe permitir al usuario acceder al menú de un agente indicando su identificador.

Requisitos involucrados: RF103

Validación: Ejecutar el comando para cambiar de menú al del agente especificado y comprobar que se cambia de menú.

Estado: Cumplido

PR103 - Eliminar agente

Descripción: El Controlador debe permitir al usuario terminar la ejecución del agente seleccionado.

Requisitos involucrados: RF104

Validación: Ejecutar el comando para terminar agente y comprobar que se recibe un mensaje indicando que el agente ha terminado su ejecución. Además comprobar en el ordenador remoto que ha desaparecido el proceso del agente terminado.

Estado: Cumplido

PR104 - Ejecutar comando en agente

Descripción: El Controlador debe permitir al usuario ejecutar un comando de Powershell en el agente seleccionado.

Requisitos involucrados: RF105

Validación: Ejecutar el comando para enviar comandos Powershell al agente seleccionado y comprobar que se recibe un mensaje con la respuesta.

Estado: Cumplido

5.1.2. Sprint 2

PR201 - Consultar agentes

Descripción: La Interfaz de datos debe permitir al usuario ver los agentes, los ordenadores controlados por estos, y el dominio al que corresponden dichas máquinas.

Requisitos involucrados: RF202

Validación: Seleccionar la opción de mostrar agentes en la Interfaz de datos y verificar que se muestran junto con los ordenadores controlados y los dominios a los que corresponden.

Estado: Cumplido

PR202 - Consultar usuarios de un dominio

Descripción: La Interfaz de datos debe permitir al usuario ver qué usuarios pertenecen a un determinado dominio.

Requisitos involucrados: RF206

Validación: Seleccionar la opción de mostrar usuarios de dominio en la Interfaz de datos y verificar que se muestran tanto el dominio como los usuarios.

Estado: Cumplido

PR203 - Consultar usuarios de un dominio

Descripción: La Interfaz de datos debe permitir al usuario ver qué ordenadores pertenecen a un determinado dominio.

Requisitos involucrados: RF207

Validación: Seleccionar la opción de mostrar ordenadores de dominio en la Interfaz de datos y verificar que se muestran tanto el dominio como los ordenadores.

Estado: Cumplido

PR204 - Obtener información básica del ordenador

Descripción: Los agentes deben ser capaces de obtener la siguiente información del ordenador controlado:

- Nombre del equipo
- Dominio del equipo
- Sistema operativo del equipo

Requisitos involucrados: RD13

Validación: Desplegar un agente y confirmar en la base de datos que se añaden los datos básicos del ordenador controlado.

Estado: Cumplido

PR205 - Obtener información adicional del ordenador

Descripción: Los agentes deben ser capaces de obtener la siguiente información del ordenador controlado:

- Usuarios del equipo
- Grupos del equipo
- Interfaces de red , IP y MAC

Requisitos involucrados: RD14

Validación: Desplegar un agente y confirmar en la base de datos que se añaden los datos avanzados del ordenador controlado.

Estado: Cumplido

PR206 - Obtener información básica de Active Directory

Descripción: Los agentes deben ser capaces de obtener la siguiente información del bosque de dominios de Active Directory donde se encuentra el equipo controlado:

- Dominios
- Servidores de dominios
- Grupos administradores de los dominios
- Usuarios administradores de los dominios

Requisitos involucrados: RD15

Validación: Desplegar un agente y confirmar en la base de datos que se añaden los datos básicos del bosque de Active Directory.

Estado: Cumplido

5.1.3. Sprint 3

PR301 - Detectar equipos en red de Active Directory

Descripción: Los agentes deben ser capaces de consultar el directorio de Active Directory y realizar escaneos para determinar si existen equipos activos alcanzables en la red.

Requisitos involucrados: RD12

Validación: Desplegar un agente y confirmar en la base de datos que se añaden nodos de los ordenadores del bosque de Active Directory.

Estado: Cumplido

PR302 - Listar acciones de agente

Descripción: El Controlador debe mostrar la lista de acciones que el usuario ha solicitado a un agente.

Requisitos involucrados: RF111

Validación: Desplegar un agente, solicitar 4 acciones y verificar que se listan todas.

Estado: Cumplido

PR303 - Consultar miembros de un grupo de dominio

Descripción: La Interfaz de datos debe permitir ver al usuario los miembros de un determinado grupo de dominio.

Requisitos involucrados: RF205

Validación: Seleccionar en la Interfaz de datos la opción de consultar miembros de un grupo y verificar que se muestra el grupo y sus miembros.

Estado: Cumplido

PR304 - Consultar grupos de un dominio

Descripción: La Interfaz de datos debe permitir al usuario ver qué grupos pertenecen a un determinado dominio.

Requisitos involucrados: RF208

Validación: Seleccionar la opción de mostrar grupos de dominio en la Interfaz de datos y verificar que se muestran tanto el dominio como los grupos.

Estado: Cumplido

PR305 - Consultar datos de agente

Descripción: La Interfaz de datos debe permitir al usuario ver los datos de un agente seleccionado.

Requisitos involucrados: RF209

Validación: Seleccionar un nodo de agente del grafo y verificar que la Interfaz de datos muestra sus datos.

Estado: Cumplido

PR306 - Consultar datos de ordenador

Descripción: La Interfaz de datos debe permitir al usuario ver los datos de un

ordenador seleccionado.

Requisitos involucrados: RF210

Validación: Seleccionar un nodo de ordenador del grafo y verificar que la Interfaz de datos muestra sus datos.

Estado: Cumplido

PR307 - Consultar datos de usuario de dominio

Descripción: La Interfaz de datos debe permitir al auditor de seguridad ver los datos de un usuario de dominio seleccionado.

Requisitos involucrados: RF211

Validación: Seleccionar un nodo de usuario de dominio del grafo y verificar que la Interfaz de datos muestra sus datos.

Estado: Cumplido

5.1.4. Sprint 4

PR401 - Listar credenciales

Descripción: El Controlador debe permitir al usuario obtener una lista de las credenciales obtenidas por los agentes.

Requisitos involucrados: RF109

Validación: Ejecutar el comando de listado de credenciales y verificar que se muestra una lista con las credenciales obtenidas.

Estado: Cumplido

PR402 - Consultar credenciales

Descripción: La Interfaz de datos debe permitir consultar los usuarios cuyas credenciales hayan sido obtenidas por los agentes durante el proceso de expansión, junto con sus ordenadores o dominios asociados.

Requisitos involucrados: RF203

Validación: Seleccionar la opción de mostrar credenciales y verificar que se muestran los usuarios con credenciales descubiertas junto con sus ordenadores y dominios asociados.

Estado: Cumplido

PR301 - Obtener credenciales

Descripción: El agente debe ser capaz de extraer las credenciales de los usuarios con sesión iniciada en la máquina.

Requisitos involucrados: RD11

Validación: Desplegar agente, iniciar sesión con un usuario en el ordenador controlado y verificar en la base de datos que se han añadido las credenciales de la cuenta.

Estado: Cumplido

5.1.5. Sprint 5

PR501 - Desplegar agente

Descripción: El Controlador debe permitir al usuario obtener el código Powershell necesario para desplegar un agente en un ordenador remoto.

Requisitos involucrados: RF101

Validación: Ejecutar el comando para generar código de despliegue y verificar que tras ejecutar el código generado aparece en la terminal de comandos del

Controlador un mensaje indicando la creación de un nuevo agente.

Estado: Cumplido

PR502 - Propagación de agentes

Descripción: Un agente, con unas credenciales válidas, debe tener la capacidad de desplegar nuevos agentes en equipos remotos.

Requisitos involucrados: RD10

Validación: Desplegar un agente en un ordenador donde se encuentren unas credenciales que permitan conectarse a otro ordenador de la red y verificar que se despliega un nuevo agente en dicho ordenador.

Estado: Cumplido

PR503 - Consultar saltos entre ordenadores

Descripción: La Interfaz de datos debe permitir al usuario ver los ordenadores del bosque y los saltos realizados por los agentes entre estos.

Requisitos involucrados: RF204

Validación: Seleccionar la opción de mostrar saltos entre ordenadores en la Interfaz de datos y verificar que se muestran los saltos, ordenadores y dominios.

Estado: Cumplido

5.1.6. Sprint 6

PR601 - Ejecutar script en agente

Descripción: El Controlador debe permitir al usuario ejecutar un script de Powershell en el agente seleccionado.

Requisitos involucrados: RF106

Validación: Ejecutar el comando para enviar script Powershell al agente seleccionado y comprobar que se recibe un mensaje con la respuesta.

Estado: Cumplido

PR602 - Descargar archivo

Descripción: El Controlador debe permitir al usuario descargar un archivo desde el ordenador controlado por el agente seleccionado.

Requisitos involucrados: RF107

Validación: Ejecutar el comando para descargar archivo del ordenador en el que se encuentre el agente seleccionado y comprobar que se recibe un mensaje con la respuesta y se ha transferido el archivo al ordenador local.

Estado: Cumplido

PR603 - Subir archivo

Descripción: El Controlador debe permitir al usuario subir un archivo al ordenador controlado por el agente seleccionado.

Requisitos involucrados: RF108

Validación: Ejecutar el comando para subir archivo al ordenador en el que se encuentre el agente seleccionado y comprobar que se recibe un mensaje con la respuesta. Verificar que se ha creado el fichero en la máquina controlada.

Estado: Cumplido

PR604 - Listar IPs

Descripción: El Controlador debe permitir al usuario obtener una lista de las interfaces de red y sus correspondientes IPs de la máquina local donde se ejecuta.

Requisitos involucrados: RF110

Validación: Ejecutar el comando de listado de IPs y verificar que se muestra una lista de las interfaces e IPs.

Estado: Cumplido

PR605 - Consulta personalizada

Descripción: La Interfaz de datos debe permitir realizar una consulta personalizada utilizando el lenguaje de consulta correspondiente y devolver los datos consultados.

Requisitos involucrados: RF201

Validación: Ejecutar una consulta utilizando el lenguaje de consulta y verificar que se presentan en pantalla los nodos y relaciones consultados.

Estado: Cumplido

PR606 - Buscar por nombre

Descripción: La Interfaz de datos debe permitir al usuario ver los datos de un nodo indicando su nombre identificativo.

Requisitos involucrados: RF212

Validación: Introducir el nombre identificativo de un nodo y verificar que la Interfaz de datos muestra sus datos.

Estado: Cumplido

5.2. Pruebas finales

Durante la realización de la validación de los sprints, a pesar de que solamente se han realizado pruebas válidas para verificar la correcta implementación de los requisitos funcionales, también se ha ido haciendo seguimiento con el cliente de los aspectos no funcionales de la aplicación, especialmente en aquellos relacionados con el sprint que se hubiera realizado.

Sin embargo, no ha sido hasta la fase final del proyecto cuando se han realizado las pruebas definitivas para obtener la verificación de que se han cumplido los requisitos no funcionales del proyecto, ya estos deben ser evaluados para el conjunto global del proyecto.

Se presentan a continuación las pruebas llevadas a cabo para validar los requisitos no funcionales.

PR701 - Powershell versión

Descripción: Los agentes deben ejecutarse correctamente en las versiones 2 y superiores de Powershell.

Requisitos involucrados: RD01

Validación: Desplegar un agente en una máquina con Powershell versión 2 y verificar que no se produce ningún error debido a la interpretación del código.

Estado: Cumplido

PR702 - Sin módulos externos de Powershell

Descripción: Los agentes deben ejecutarse correctamente en un equipo que no tenga instalado ningún módulo que amplíe las funcionalidades de Powershell.

Requisitos involucrados: RD02

Validación: Desplegar un agente en una máquina sin módulos externos instalados y verificar que tras ejecutarse 3 iteraciones del hilo expansión sigue funcionando correctamente.

Estado: Cumplido

PR702 - Ejecución en memoria

Descripción: Los agentes deben ejecutarse totalmente en memoria sin crear archivos en disco, a excepción de aquellos subidos por el usuario.

Requisitos involucrados: RD03

Validación: Desplegar un agente y verificar que su proceso no crea ningún archivo.

Estado: Cumplido

PR703 - Máximo de bytes por mensaje

Descripción: Ninguno de los mensajes enviados por agentes o Controlador debe superar 1MB.

Requisitos involucrados: RD04

Validación: Desplegar un agente, descargar un archivo que ocupe más de 1MB, subir un archivo de más de 1MB y verificar que en ningún caso se ha mandado un mensaje con un tamaño superior a 1MB.

Estado: Cumplido

PR704 - 1 agente por ordenador

Descripción: No se puede permitir la ejecución de más de 1 agente por máquina.

Requisitos involucrados: RD01

Validación: Desplegar 2 agentes en un mismo ordenador y verificar tras 1 minuto que el proceso de uno de los agentes ha desaparecido.

Estado: Cumplido

PR705 - Comunicación por HTTP

Descripción: Los agentes y el Controlador deben comunicarse utilizando el protocolo HTTP.

Requisitos involucrados: RD06

Validación: Desplegar un agente y verificar que tras ejecutarse 3 iteraciones del hilo expansión todos los mensajes entre agente y Controlador se ha transmitido mediante HTTP.

Estado: Cumplido

PR706 - Datos de expansión en base de datos

Descripción: La información referente a la expansión debe almacenarse en una base de datos.

Requisitos involucrados: RD07

Validación: Desplegar un agente y verificar que tras ejecutarse 1 iteración del hilo expansión se han almacenado los datos recogidos por el agente en la base de datos.

Estado: Cumplido

PR707 - Ausencia de base de datos

Descripción: El Controlador debe funcionar sin necesidad de estar conectado a la base de datos.

Requisitos involucrados: RD08

Validación: Desconectar la base de datos, arrancar el Controlador, desplegar un agente y verificar que es posible interactuar con él y listar las credenciales obtenidas.

Estado: Cumplido

PR708 - Autoeliminar agente

Descripción: Un agente debe eliminarse tras no conseguir contactar con el Controlador en un período de tiempo.

Requisitos involucrados: RD09

Validación: Desplegar un agente, luego desconectar el controlador y comprobar tras 10 minutos que el agente ha dejado de funcionar.

Estado: Cumplido

PR708 - Agente en un único fragmento

Descripción: El código del agente debe poder transmitirse en un único fragmento, por lo que debe ocupar menos de 1MB.

Requisitos involucrados: RD18

Validación: Verificar que el código del agente no ocupa más de 1MB.

Estado: Cumplido

PR709 - Ayuda en comandos

Descripción: Cada comando del Controlador debe proporcionar una ayuda que ofrezca una descripción del comando y modo de empleo.

Requisitos involucrados: RC01

Validación: Ejecutar la ayuda de todos los comandos de la terminal del Controlador y verificar que existe.

Estado: Cumplido

PR710 - Idioma inglés

Descripción: Las interfaces del sistema deberán utilizar el inglés para los textos.

Requisitos involucrados: RC02

Validación: Revisar todos los comandos y pantallas de las interfaces y verificar que la información se presenta en inglés..

Estado: Cumplido

PR711 - Software libre

Descripción: El software objetivo del proyecto deberá contar con licencia de software libre.

Requisitos involucrados: RP01

Validación: Comprobar que en la documentación se presenta una licencia de software libre.

Estado: Cumplido

PR712 - Protocolo de comunicación independiente del lenguaje

Descripción: El protocolo de comunicación entre agente y Controlador debe ser independiente del lenguaje de programación.

Requisitos involucrados: RE01

Validación: Analizar el protocolo diseñado y verificar que es posible implementar en otros lenguajes de programación.

Estado: Cumplido

PR713 - Protocolo de comunicación extensible

Descripción: El protocolo de comunicación entre agente y Controlador debe ser lo suficientemente flexible para añadirle nuevas propiedades en un futuro.

Requisitos involucrados: RE01

Validación: Analizar el protocolo diseñado y verificar que sería posible añadirle un capa de cifrado en un futuro.

Estado: Cumplido

PR714 - Restricción a subredes de dominio

Descripción: No se debe permitir que los agentes se expandan más alla del bosque de dominio.

Requisitos involucrados: RS01

Validación: Desplegar agentes en un entorno con máquinas externas al bosque y verificar que estas no son infectadas.

Estado: Cumplido

PR715 - Contraseña de Agente

Descripción: El Controlador solamente debe aceptar peticiones de aquellos agentes que incluyan en dicha petición una contraseña válida.

Requisitos involucrados: RS02

Validación: Desplegar agentes con contraseña válida e inválida y verificar que solamente aquellos con contraseña válida son capaces de comunicarse con el Controlador.

Estado: Cumplido

5.3. Evaluación de la usabilidad

Durante las reuniones finales de cada sprint con el cliente se ha ido evaluando progresivamente la interfaz de la aplicación.

Adicionalmente en las reuniones finales de los sprints 1,3 y 5, se ha contado con 2 usuarios finales a mayores del cliente que también se han encargado de revisar distintos aspectos del sistema, entre ellos las interfaces, tanto la línea de comandos como la interfaz gráfica encargada de mostrar los grafos, permitiendo que el diseño visual de estas se fuera mejorando progresivamente durante la realización del proyecto.

Por último se realizó una evaluación final más estricta para verificar que las interfaces, tanto de comandos como gráfica cumplen adecuadamente las 10 heurísticas de Jakob Nielsen[22].

A continuación se presentan los resultados obtenidos en la evaluación de cada una de las heurísticas:

1. **Visibilidad del estado del sistema**: Se debe mantener a los usuarios informados del estado del sistema, dando una retroalimentación adecuada en un tiempo razonable.
    - Interfaz gráfica: La interfaz muestra la información necesaria para que el usuario conozca el estado del sistema.
    - Consola de comandos: El sistema proporciona los comandos y mensajes adecuados para mantener al usuario informado.
2. **Utilizar el lenguaje de los usuarios**: El sistema debe utilizar el lenguaje natural de los usuarios, con palabras o frases que le sean conocidas, evitando vocabulario técnico propio del sistema y desconocido por el usuario.
    - En ambas interfaces se utiliza un lenguaje con algunos términos técnicos, pero que son comprensibles para los usuarios finales de la aplicación.
3. **Control y libertad para el usuario**: Los usuarios deben tener una forma fácil de salir de funciones del sistema en las que han entrado por error.
    - Interfaz gráfica: No existen funciones de deshacer y rehacer, pero ya que las acciones son inmediatas y concisas es fácil volver a un estado anterior.
    - Consola de comandos: Existen los comandos adecuados para cancelar las acciones solicitadas a los agentes.

4. **Consistencia y estándares**: El lenguaje utilizado por el sistema debe ser el acorde al de la plataforma y ámbito en que está implementado de modo que el usuario no tenga que preguntarse el significado de las palabras y acciones del sistema.

    - En ambas interfaces se utiliza un lenguaje adecuado que permite al usuario reconocer los mensajes del sistema.

5. **Prevención de errores**: Se deben eliminar acciones que puedan llevar al usuario a cometer errores, o advertir sobre el peligro de una acción y preguntar si desea ejecutarla.

    - Interfaz gráfica: No existen acciones que lleven al usuario a un error ya que no se modifica la base de datos.
    - Consola de comandos: Se permite al usuario enviar a los agentes comandos que pueden conducir a error, ya que la peligrosidad solo es posible conocerla en el contexto de un agente y no en el Controlador.

6. **Minimizar la carga de memoria del usuario**: El sistema debe evitar que el usuario deba memorizar información, mostrando lo que necesite para realizar las acciones.

    - Interfaz gráfica: Los elementos se muestran en una misma ventana de modo que el usuario puede consultar toda la información necesaria a la vez.
    - Consola de comandos: El resultado de las acciones se queda grabado en la línea de comandos minimizando la carga de memoria. Los comandos son mnemotécnicos para ayudar al usuario a memorizarlos fácilmente y se dispone de función de autocompletado.

7. **Flexibilidad y eficiencia de uso**: El uso de atajos permiten mejorar la eficiencia de usuarios experimentados sin necesidad de dificultar el uso de usuarios inexpertos.

    - Interfaz gráfica: No existen atajos, sin embargo las acciones son simples por lo que se realizan bastante rápido.
    - Consola de comandos: Es posible navegar por el historial de comandos utilizando las flechas del teclado facilitando la ejecución de comandos previamente insertados.

8. **Diálogos estéticos y diseño minimalista**: La interfaz no debe contener información irrelevante, ya que cada unidad de información disminuye la visibilidad relativa de la información importante.

- Interfaz gráfica: La interfaz se centra en el grafo que es el elemento de mayor interés.
- Consola de comandos: Se muestran los datos necesarios sin excederse, pero no se tiene una función para limpiar los resultados anteriores de la consola.

9. **Ayudar a los usuarios a conocer, diagnosticar y recuperarse de los errores**: Los mensajes de error deben ser claros y sin códigos extraños, permitiendo al usuario identificar perfectamente el error.

   - Interfaz gráfica: Se informa adecuadamente cuando las consultas no obtienen resultado.
   - Consola de comandos: Se muestran los errores en lenguaje natural con una breve descripción técnica para que los usuarios finales de esta aplicación (con conocimientos técnicos) puedan obtener un mayor detalle.

10. **Ayuda y documentación**: Lo ideal es que un sistema sea usable sin documentación, no obstante es posible necesitarla. En ese caso la documentación debe ser fácil de encontrar, clara y concreta.

    - Interfaz gráfica: Los nombres de los elementos permiten identificar la función de cada uno de ellos y a mayores se incluye la documentación de la interfaz en los anexos del presente documento.
    - Consola de comandos: Cada comando y menú cuenta con una opción de ayuda que explica su funcionamiento. Se incluye la documentación en los anexos del presente documento.

En general se puede concluir que la aplicación cumple de manera aceptable con las heurísticas de Nielsen, aunque hay ciertos aspectos que es posible mejorar.

Capítulo 6

Conclusiones y posibles ampliaciones

En este último capítulo se presentan las conclusiones que se han extraído a raíz del desarrollo del proyecto y posteriormente se comentan posibles ampliaciones de funcionalidades que se podrían llevar a cabo en un futuro.

Secciones del capítulo:

1. **Conclusiones**
2. **Ampliaciones**

6.1. Conclusiones

Para finalizar el presente documento, se presentan a continuación una serie de conclusiones obtenidas durante el desarrollo del proyecto.

Se ha creado un sistema capaz de explotar relaciones de confianza entre usuarios y equipos, con el objetivo de detectar vías de elevación de privilegios en una red administrada por el servicio de directorio Active Directory.

Además el sistema permite recopilar toda la información extraída durante este proceso de expansión, de modo que el auditor de seguridad pueda identificar los caminos, de saltos entre máquinas, los cuales permitirían a un atacante acceder a los sistemas críticos de la infraestructura de una compañía, de manera que le sería posible llegar a obtener información corporativa confidencial.

Para ello se ha implementado un sistema basado en agentes que son capaces de trabajar con cierta autonomía, extrayendo, de los equipos y los dominios, información sensible, como credenciales de usuarios, que permiten obtener los privilegios de estos y utilizarlos para desplegar nuevos agentes en distintos equipos de la red.

Por otra parte estos agentes cuenta con un sistema de comunicación y control que permite al auditor mantener el contacto con ellos y solicitarles la realización de acciones en los equipos controlados, como por ejemplo, la ejecución de comandos.

A la vez que se realizan estos trabajos de descubrimiento y explotación de fallos, también se exfiltra toda la información obtenida hacia el equipo del auditor, permitiendo almacenar esta en una base de datos y representar mediante una interfaz gráfica el desarrollo del proceso de expansión de los agentes.

La representación gráfica permite además visualizar el estado de la red ,en la cual se ha realizado la expansión, con el objetivo de ayudar a definir las medidas de seguridad, como la segmentación de redes y privilegios, que se deberían llevar a cabo para mantener protegida la infraestructura de Active Directory.

Respecto del diseño, la modularidad se ha tenido en cuenta durante todo el desarrollo del proyecto, con lo que se ha conseguido producir una herramienta que permite fácilmente el reemplazo de cada uno de sus componentes, haciendo fácilmente posible en un futuro la ampliación de sus capacidades y funcionalidades.

Por último, el hecho de que el sistema haya pasado todas y cada una de las pruebas propuestas, demuestra que se ha desarrollado un producto útil y funcional con el que los auditores de seguridad pueden contar para la realización de un test de intrusión.

6.2. Ampliaciones

Como se ha comentado en el apartado anterior, todavía es posible mejorar el producto. Algunas de las ampliaciones que se podrían realizar serían las siguientes:

- Implementar un sistema de logs en el Controlador que permita a este mantener permanencia de datos cuando no se está conectado a la base de datos.
- Añadir comandos en el Controlador para gestionar grupos de agentes.
- Incluir la funcionalidad en el Controlador de generar archivos con macros que permitan desplegar agentes.
- Añadir a los agentes la funcionalidad de solicitar información de credenciales obtenidas o máquinas descubiertas al Controlador.
- Incluir en los agentes capacidades de persistencia para evitar ser borrados tras un reinicio en el equipo.
- Incluir Scripts predefinidos para ejecutar en un agente.
- Incluir en los agentes la posibilidad de cargar módulos dinámicamente para ampliar sus funcionalidades.
- Desarrollar agentes en otros lenguajes de programación que puedan ejecutarse en máquinas con sistemas operativos linux y macOS.
- Añadir un algoritmo de compresión en el protocolo ACP para permitir el envío de una mayor cantidad de información en cada mensaje.
- Añadir una capa de cifrado en el protocolo ACP para mantener segura la información transmitida.
- Incluir en el protocolo ACP la posibilidad de utilizar tunelización de DNS para transmitir la información.
- Permitir la inserción de datos desde la Interfaz de datos.

Apéndice A

Manuales técnicos

En este anexo se indican todos aquellos aspectos que deben tenerse en cuenta a la hora de realizar el mantenimiento o ampliación de las herramietas. Se presentan en esta sección dos manuales:

- **Manual técnico de VanaX**: Ofrece una descripción de los componentes principales que se deben tener en cuenta a la hora de modificar la herramienta VanaX.
- **Manual técnico de TrueniX**: Muestra los procedimientos que se deben seguir y los componentes que se deben modificar para aumentar las funcionalidades de la herramienta TrueniX.

A.1. Manual técnico de VanaX

En este manual se recogen aquellos estándares y procedimientos que deben respetarse a la hora de realizar cambios en el código de la herramienta VanaX.

A.1.1. Dependencias

Para modificar VanaX se requiere tener instalado el software Neo4j y Nodejs tal y como se indica en el manual de usuario del anexo siguiente.

Además para realizar las pruebas del agente se recomienda contar con la versión 2 de Powershell instalada en el sistema. En el siguiente enlace se ofrece una guía para realizar la instalación:

https://msdn.microsoft.com/powershell/scripting/setup/installing-the-windows-powershell-2.0-engine.

A.1.2. Añadir comandos a los menús

Una de las formas de añadir nuevas funcionalidades a VanaX es incorporando nuevos comandos a los menús de la herramienta.

Para añadir un nuevo comando se debe añadir un fichero a la carpeta *terminal/-commands/mainMenu* o *terminal/commands/agentMenu*, ambas bajo el directorio raíz del código fuente de VanaX, según corresponda a un comando del menú principal o de agente, respectivamente.

En el fichero de especificación de comando se deben exportar los siguientes atributos y métodos:

Atributos

name: String que indica el nombre del comando, en minúsculas.

desc: String que ofrece una pequeña descripción del comando.

usage: String qué indica con que argumentos se debe ejecutar el comando.

Métodos

invoke: Método que acepta un array de strings que serán los argumentos, siendo el primero el nombre del comando. En caso de ser un comando de agente el primer argumento aceptado por *invoke* será una string que representa el ID del agente seleccionado en el menú.

help: Devuelve una string que indica la descripción del comando y su uso. Se debe invocar esté método en caso de que el primer argumento del comando (no el de la función *invoke*) sean las cadenas "help" o "?".

A.1.3. Archivo de Agente y Stager

El archivo que contiene el código del agente es *files/agent.ps1*. Debido a que el agente no puede importar módulos externos, el código íntegro debe encontrase en este fichero. Por otro lado el archivo que contiene la funcionalidad del Stager se encuentra en *files/stager.ps1*. Ambos bajo el directorio raíz de VanaX.

En caso de modificarse alguno de los ficheros debe probarse este en una consola de comandos Powershell v2 para verificar que es compatible con esta versión y las superiores. Además el fichero Stager no debe superar los 12259 bytes, ya que en ese caso no podrá ser invocado desde los parámetros del ejecutable Powershell.

También se debe tener en cuenta que el nombre de las *cmdlets* incorporadas en estos ficheros debe seguir la estructura Verbo-Nombre, y siempre que sea posible se deben utilizar los verbos aprobados por Microsoft. Se incluye la lista de estos verbos en el siguiente enlace: https://msdn.microsoft.com/en-us/library/ms714428(v=vs.85).aspx.

A.2. Manual técnico de TrueniX

En este manual se recoge la información necesaria para continuar el desarrollo de la herramienta TrueniX.

A.2.1. Dependencias

Para modificar TrueniX se requiere tener instalado el software Neo4j y Nodejs tal y como se indica en el manual de usuario del anexo siguiente.

A.2.2. Modo desarrollador

Para evitar tener que ser compilado cada vez que se realiza un cambio, se incluye en TrueniX la posibilidad de ejecutar la herramienta en modo desarrollador, que permite que los cambios realizados en el código se reflejen de manera automática en la interfaz.

Para ejecutar el modo de desarrollador se debe ejecutar el comando *npm run dev* en el directorio raíz que contiene el código fuente de TrueniX.

A.2.3. Añadir nuevas consultas predefinidas

En caso de querer añadirse nuevas consultas para que aparezcan en el menú de TrueniX se debe editar el archivo *src/components/SearchContainer/Tabs/PrebuiltQueries.json*.

En TrueniX existen dos tipos principales de consultas predefinidas. Las consultas directas, en las cuales se realiza la consulta y se devuelven los resultados sin pasos extra, y las consultas que requieren la selección de un elemento previo para ejecutar la consulta. Un ejemplo de estas últimas sería consultar los miembros de un grupo, para lo cual el usuario debe seleccionar el grupo del que quiere obtener los miembros antes de realizar la consulta a la base de datos.

En caso de requerir una consulta directa, simplemente se debe especificar esta en el archivo anteriormente mencionado, indicando para ella un nombre que se le presentará al usuario. Sin embargo en consultas con selección previa, se debe establecer una consulta que extraerá de la base de datos una lista de elementos de entre los cuales el usuario seleccionará uno. El elemento se incluye como

parámetro *result* en la consulta final.

Apéndice B

Manuales de usuario

En este anexo se muestran los manuales de usuario que indican cómo instalar las herramientas en un sistema operativo Windows y qué acciones es posible realizar con ellas.

Se incluyen 3 documentos:

- **Instalación de dependencias**: En este manual, que debe ser leído en primer lugar, se explica como instalar el software externo necesario del que se depende para la correcta ejecución de las herramientas.
- **Manual de usuario de VanaX**: Se muestra en este manual como se debe instalar y utilizar la herramienta VanaX que permite desplegar y controlar a los agentes.
- **Manual de usuario de TrueniX**: Se explica como proceder con la instalación y uso de la herramienta TrueniX utilizada para ver los datos, recogidos por la herramienta VanaX, de manera gráfica.

B.1. Instalación de dependencias

El primer paso antes de proceder con la instalación de las herramientas es realizar la instalación de las dependencias de estas.

Se deben instalar la base de datos Neo4j y el entorno de ejecución Nodejs.

B.1.1. Instalación de Neo4j

Para instalar de la base de datos Neo4j lo primero que se debe hacer es descargar el software. Se debe descargar la versión Community disponible en el siguiente enlace:

https://neo4j.com/download/

Tras descargar el archivo solamente se necesita ejecutarlo, con lo que se inicia una ventana que realiza la guía por el proceso de instalación. Una vez terminado ya se tendrá la Neo4j lista para utilizar.

Una vez iniciada Neo4j aparece una ventana que da a elegir la ruta de directorios de la cual se quiere cargar la base de datos.

En caso de querer crear una nueva base de datos simplemente se debe crear un nuevo directorio y seleccionarlo como origen de la base de datos, Neo4j se ocupará de preparar el directorio para que sea utilizado como una base de datos.

Si por el contrario se desea continuar utilizando una base de datos anterior solamente se debe seleccionar el directorio donde se encuentra la base de datos requerida.

En caso de querer iniciar un nuevo proceso de expansión se recomienda crear una nueva base de datos, esto es, un nuevo directorio.

Una vez elegido el directorio se debe pulsar el botón Start y se arrancará el sistema gestor de base de datos.

Si se accede a través del navegador a la dirección indicada en el panel de Neo4j es posible realizar consultas sobre la base de datos, así como establecer el usuario y contraseña deseados.

B.1.2. Instalación de Nodejs

El primer paso para instalar Nodejs es descargarlo desde la página web oficial en el siguiente enlace:

https://nodejs.org/es/

Para que el sistema funcione correctamente se debe descargar la versión 7, ya que incluye funcionalidades necesarias para la correcta ejecución de la herramientas.

Una vez descargado el archivo de instalación se debe ejecutar y aparecerá una ventana que guiará al usuario en el proceso de instalación. Se deben instalar todos los componentes de Nodejs (opción por defecto). Una vez terminado el proceso ya se tendrá Nodejs instalado en el sistema.

B.2. Manual de usuario de VanaX

B.2.1. Instalación

Para poder ejecutar VanaX se necesita previamente instalar los paquetes necesarios de los que depende la aplicación. Para esto, en la consola de comandos Powershell, es necesario moverse hasta la carpeta que contiene los archivos de VanaX, entre ellos el archivo *package.json* que contiene la lista de los paquetes que se deben instalar.

Para la instalación se debe utilizar el gestor de paquetes npm, que se instala con Nodejs. Para instalar los paquetes se debe ejecutar la instrucción *npm install*.

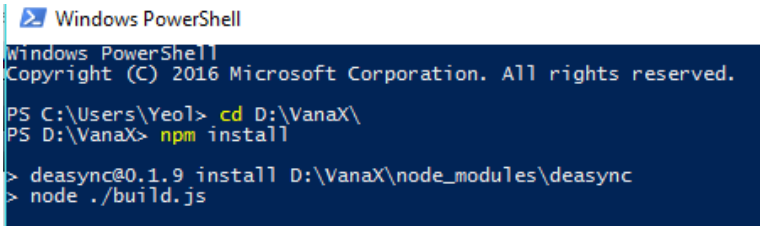

Figura B.1: Instalación de paquetes con la herramienta npm

B.2.2. Uso

Configuración

Una vez se han instalado los paquetes necesarios es posible iniciar la herramienta, sin embargo, antes se deben configurar los parámetros de acceso a la base de datos, que se encuentran en el archivo *globals/config.json*. En este archivo, como se puede apreciar en la figura B.2 se especifican el usuario, contraseña de la base de datos, así como la dirección de esta y el puerto.

Se deben establecer estos valores a los necesarios para conectarse a la base de datos. En principio, si la base de datos Neo4j se ejecuta en la misma máquina que VanaX solamente será necesario ajustar el usuario y contraseña con los que se realizará la conexión.

Figura B.2: Contenido por defecto del archivo config.json

En el archivo *globals/config.json* también se puede indicar qué contraseña debe ser utilizada por los agentes para autenticarse contra el servidor, así como el tamaño máximo (en bytes) que puede ocupar un paquete intercambiado entre el agente y el servidor.

Inicio

Para iniciar la herramienta VanaX se debe ejecutar con el interprete de Nodejs el fichero *app.js* que se encuentra en el directorio raíz. Si los parámetros han sido correctamente configurados el programa se iniciará sin problemas. En caso de que VanaX no detecte la base de datos Neo4j activa o el proceso de autenticación contra esta falle, se mostrará un mensaje advirtiendo al usuario de esta situación.

Comandos

La herramienta VanaX cuenta con 2 menús interactivos, uno principal que permite la realización de las acciones generales, y un menú de agente, que permite interactuar con el agente seleccionado.

Cada menú cuenta con una serie de comandos que se pueden ejecutar. Para cada comando existe una sección de ayuda que se puede ver especificando el

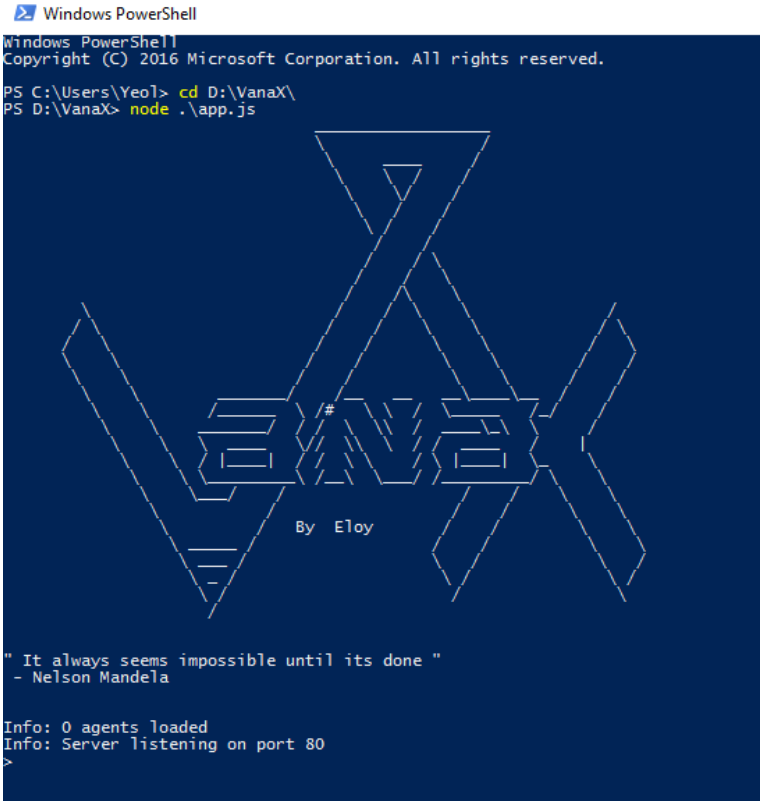

Figura B.3: Inicio de la herramienta VanaX

argumento "?"después del nombre del comando, por ejemplo *listips ?* muestra la descripción y uso del comando *listips*. Además el nombre de los comandos puede ser especificado tanto en letras mayúsculas como minúsculas. A continuación se enumera la lista de comandos disponible en cada menú.

Menú principal

El menú principal se reconoce por que en la línea de comandos solamente aparecerá el símbolo >. Los comandos que se pueden ejecutar son los siguientes:

- **exit**: Termina la ejecución de VanaX. Sin argumentos.
- **geninstallcode**: Se genera el código de instalación de un agente que se encargará de desplegar el agente. Admite como argumentos una IP y un puerto, este último será el 80 en caso de no ser especificado.
- **help**: Lista los comandos disponibles y ofrece una breve descripción de cada uno. Sin argumentos.

- **listagents**: Lista los agentes desplegados, especificando su ID, el ordenador donde se encuentran, el usuario, el PID del proceso y la versión de Powershell en que se ejecutan. Sin argumentos.
- **listcredentials**: Lista a los usuarios de los cuales se ha obtenido la contraseña. Especifica el tipo de usuario (local o de dominio), el nombre del ordenador/dominio al que pertenece el usuario, el nombre de usuario y su contraseña. Sin argumentos.
- **listips**: Lista las IPs de las que dispone la máquina donde se ejecuta VanaX. Se muestra la IP, el nombre de la interfaz, las máscara de red y la dirección MAC. Sin argumentos.
- **select**: Permite seleccionar un agente y acceder a su menú. Acepta como argumento un ID de agente.

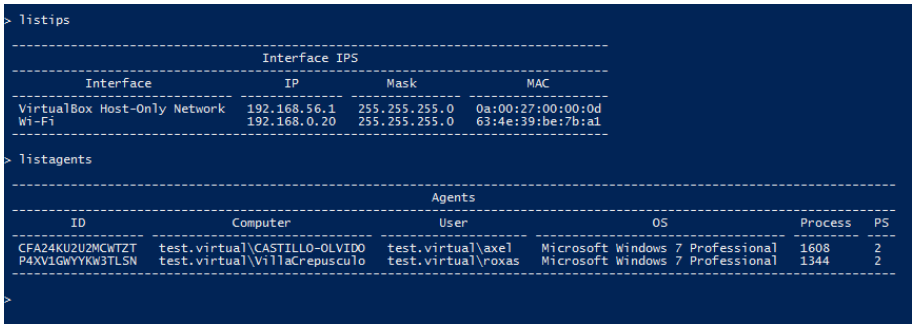

Figura B.4: Ejecución de los comandos *listips* y *listAgents*

Menú de agente

El menú de agente se reconoce por que en la línea de comandos aparecerá el símbolo > acompañado del identificador del agente correspondiente. En este menú es posible ejecutar los comandos del menú principal. Los comandos a mayores que se pueden ejecutar son los siguientes:

- **downloadfile**: Permite enviar al agente una orden para descargar un archivo. Los archivos descargados por los agentes se almacenan en el directorio *globals/downloads* que se encuentra en el directorio raíz de la herramienta VanaX. Admite como argumento la ruta remota (en el ordenador en que se encuentra el agente) del archivo que se desea descargar.
- **exec**: Permite enviar al agente un comando para ejecutar. Admite como argumento el comando que se desea ejecutar.

- **execscript**: Permite enviar al agente un script para ejecutar. Admite como argumento la ruta del script que se desea ejecutar.
- **infoaction**: Muestra información de una acción. Se muestra el identificador, el tipo de acción, que puede ser un comando, un script, una descarga o una subida (de archivo), el estado, que puede ser finalizada (se posee el resultado de la acción), enviada (el agente ha recibido la acción), semi-enviada (el agente ha recibido parcialmente la acción) o en espera (todavía no se ha enviado la acción), una descripción de la acción y el resultado. Admite como argumento un identificador de la acción.
- **kill**: Envía al agente la señal para que termine su ejecución. Sin argumentos.
- **listactions**: Lista las acciones que se han solicitado al agente. Para cada acción se muestra la misma información que el comando *infoaction*. Solamente se muestran los resultados breves, para obtener el resultado completo se puede ejecutar el comando *infoaction*. Sin argumentos.
- **removeaction**: Elimina una acción, siempre que esta no se haya enviado al agente. Admite como argumento un ID de acción.
- **uploadfile**: Permite enviar al agente un archivo que se almacenará en la máquina remota. Admite como argumento la ruta local del archivo a enviar y la ruta remota donde se guardará el archivo.

B.3. Manual de usuario de TrueniX

B.3.1. Uso

Para ejecutar TrueniX no se requiere de ninguna instalación previa, simplemente se debe lanzar el ejecutable correspondiente que se encuentra dentro de la carpeta raíz de la herramienta. Existen 2 versiones, una de 32 bits y otra de 64 bits. Se debe ejecutar la versión que se ajuste a la arquitectura del sistema operativo de la máquina en que se quiere ejecutar.

Una vez ejecutado se presenta la interfaz de login, en la que se deben introducir la URL que especifica la dirección y el puerto en que se encuentra la base de datos Neo4j, el usuario y la contraseña necesarios para autenticarse.

Figura B.5: Panel de login de TrueniX

Una vez se inicia sesión se accede a la pantalla principal en la que se observan distintos elementos que permiten realizar distintas acciones.

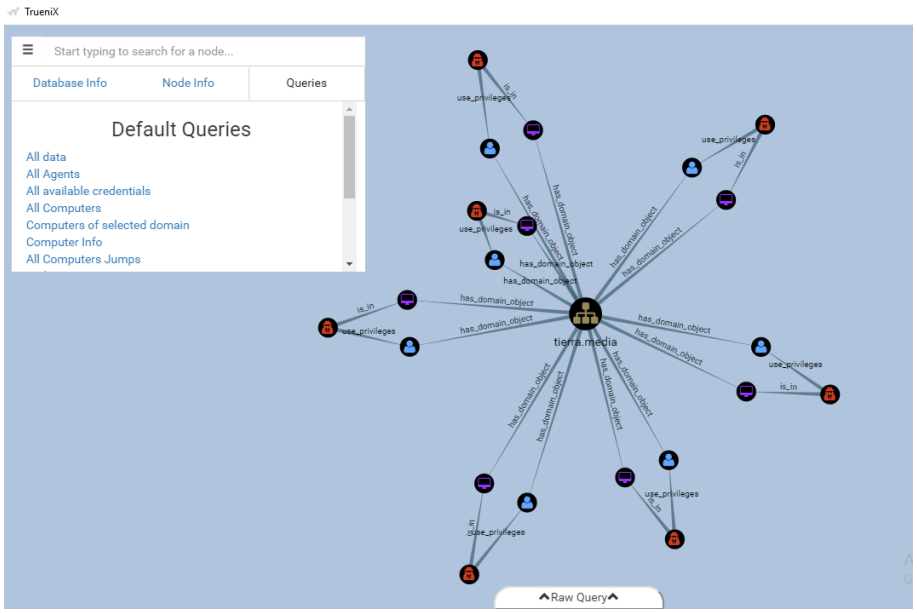

Figura B.6: Ventana principal de TrueniX

A continuación se realiza una descripción de cada una de las acciones que se pueden llevar a cabo en la herramienta TrueniX.

- **Consultas predefinidas**: La pestaña *Queries* del panel principal de TrueniX dispone de una serie de consultas que permiten mostrar distintas vistas de la base de datos en el grafo. A continuación se listan las consultas que se pueden realizar:
  - *All data*: Permite consultar todos los nodos y relaciones de la base de datos.
  - *All agents*: Permite mostrar todos los agentes que se encuentran desplegados, así como los ordenadores y usuarios relacionados con estos. Es la consulta que se realiza al iniciar TrueniX.
  - *All available credentials*: Muestra todos los usuarios de los que se han obtenido credenciales.
  - *All computers*: Muestra todos los ordenadores y los dominios.
  - *Computers of selected domain*: Muestra todos los ordenadores de un dominio.
  - *Computer info*: Muestra las relaciones de un determinado ordenador.
  - *All computers jumps*: Muestra todos los ordenadores y los dominios, así como los saltos realizados por los agentes entre máquinas.

  - *Paths to computers*: Muestra los saltos y ordenadores que han utilizado los agentes para llegar a una determinada máquina.
  - *All domains*: Muestra todos los dominios.
  - *All domain users*: Muestra todos los usuarios de dominio y los dominios.
  - *Domain users of selected domain*: Muestra todos los usuarios de un dominio.
  - *All domain groups*: Muestra todos los grupos de dominio y los dominios.
  - *Domain groups of selected domain*: Muestra todos los grupos de un dominio.
  - *Members of selected domain group*: Muestra todos los miembros, directos y transitivos, de un grupo de dominio.

- **Consultas libres**: La pestaña *Raw Query* de la parte inferior permite desplegar un campo de texto donde se pueden realizar consultas personalizadas en el lenguaje Cypher de Neo4j. En el siguiente enlace se encuentra el manual de Cypher: https://neo4j.com/developer/cypher-query-language/.

- **Seleccionar nodo del grafo**: Es posible seleccionar un nodo del grafo para obtener más información de este. Esta información se mostrará automáticamente en la pestaña *Node Info* del panel principal.

- **Buscar nodos por nombre**: Es posible obtener la información de un nodo a partir de su nombre identificativo introduciendo este en el campo de texto situado en la parte superior del panel principal de TrueniX. Para facilitar se cuenta con función de autocompletado.

- **Obtener estado de la base de datos**: Seleccionando la pestaña *Database Info* del panel principal es posible obtener un resumen del estado de la base de datos. Se incluye en el resumen la URL y usuario utilizados para conectarse a Neo4j, así como una indicación del número de nodos de cada tipo que se encuentran en la base de datos.

Apéndice C

Licencia

C.1. Licencia de VanaX

BSD 3-Clause License

Copyright (c) 2017, Eloy Pérez González

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE

FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

C.2. Licencia de TrueniX

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if

you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its

content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding

Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a

network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright

holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the

patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you

to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
    {one line to give the program's name and a brief idea of what it does.}
    Copyright (C) {year}  {name of author}

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
    {project}  Copyright (C) {year}  {fullname}
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

Bibliografía

[1] WebStorm. Web de JetBrains (*https://www.jetbrains.com/webstorm/*). Consultado en Julio de 2016.

[2] GitHub (*https://github.com/*). Consultado en Julio de 2016.

[3] Git (*https://git-scm.com/*). Consultado en Julio de 2016.

[4] Dropbox (*https://www.dropbox.com/*). Consultado en Julio de 2016.

[5] PCComponentes (*https://www.pccomponentes.com/*). Consultado en Julio de 2016.

[6] Experteer (*https://www.experteer.es/*). Consultado en Julio de 2016.

[7] InfoJobs (*https://www.infojobs.net/*). Consultado en Julio de 2016.

[8] Tecnoempleo (*https://www.tecnoempleo.com*). Consultado en Julio de 2016.

[9] Powershell. Página del sitio web Microsoft (*https://msdn.microsoft.com/en-us/powershell*). Consultado en Julio de 2016.

[10] Javascript. Página de la Wikipedia (*http://es.wikipedia.org*). Consultado en Julio de 2016.

[11] JSON. Página de la Wikipedia (*http://es.wikipedia.org*). Consultado en Julio de 2016.

[12] Hypertext Transfer Protocol. Página de la Wikipedia (*http://es.wikipedia.org*). Consultado en Julio de 2016.

[13] Node.js . Página de la Wikipedia (*http://es.wikipedia.org*). Consultado en Julio de 2016.

[14] Node.js (*https://nodejs.org/*). Consultado en Julio de 2016.

[15] Neo4j (*https://neo4j.com/*). Consultado en Julio de 2016.

[16] VirtualBox (*https://www.virtualbox.org/*). Consultado en Julio de 2016.

[17] Cypher (*https://neo4j.com/developer/cypher-query-language//*). Consultado en Julio de 2016.

[18] React (*https://facebook.github.io/react/*). Consultado en Julio de 2016.

[19] JSX (*https://jsx.github.io/*). Consultado en Julio de 2016.

[20] Electron (*http://electron.atom.io/*). Consultado en Julio de 2016.

[21] PSExec. Web de Microsoft (*https://technet.microsoft.com/en-us/sysinternals/pxexec.aspx*). Consultado en Septiembre de 2016.

[22] Heurísticas de Nielsen. Página de la Wikipedia (*https://es.wikipedia.org/wiki/Heurísticas_de_Nielsen*). Consultado en Julio de 2016.

[23] *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK)*, 3ª edición, Norma Nacional Americana, 2004.

[24] I. Robinson, J. Webber, E. Eifrem, *Graph Databases* , 2ª edición, O'Reilly, 2015.

[25] B. Desmond, J. Richards, R. Allen y A.G. Lowe-Norris, *Active Directory*, 5ª edición, O'Reilly, 2013.

[26] D. Flanagan, *Javascript The Definitive Guide*, 6ª edición, O'Reilly, 2011.

[27] D.M. Clements, *Node Cookbook*, 2ª edición, Packt Publishing, 2014.

[28] P. González Pérez, *Pentesting con Powershell*, 1ª edición, 0xWORD, 2015.

[29] L. Holmes, *Windows Powershell Cookbook*, 3ª edición, O'Reilly, 2013.

[30] B.J.W. Blawat, *Mastering Windows Powershell Scripting*, 1ª edición, Packt Pusblishing, 2015.

[31] M. Russinovich, D.A. Solomon y A. Ionescu, *Windows Internals Part 1*, 6ª edición, Microsoft, 2012.

[32] M. Russinovich, D.A. Solomon y A. Ionescu, *Windows Internals Part 2*, 6ª edición, Microsoft, 2012.

[33] S. Stefanov, *React Up & Running*, 1ª edición, O'Reilly, 2016.

[34] Cmdlets Naming Convention (*https://msdn.microsoft.com/en-us/library/ms714428(v=vs.85).aspx*). Consultado en Julio de 2016.

[35] Active Directory Classes . Web de Miscrosoft (*https://msdn.microsoft.com/en-us/library/ms680938(v=vs.85).aspx*). Consultado en Julio de 2016.

[36] Win32ComputerSystem (*https://msdn.microsoft.com/en-us/library/aa394102(v=vs.85).aspx*). Consultado en Septiembre de 2016.

[37] Use Powershell to interact with the Windows API (*https://blogs.technet.microsoft.com/heyscriptingguy/2013/06/27/use-powershell-to-interact-with-the-windows-api-part-3/*). Consultado en Julio de 2016.

[38] PSReflect (*https://github.com/mattifestation/PSReflect*). Consultado en Septiembre de 2016.

[39] Mimikatz (*https://github.com/gentilkiwi/mimikatz*). Consultado en Septiembre de 2016.

[40] Invoke Mimikatz . Repositorio Empire (*https://github.com/adaptivethreat/Empire/*). Consultado en Septiembre de 2016.

[41] Repositorio BloodHound (ıthttps://github.com/BloodHoundAD/BloodHound). Consultado en Septiembre de 2016.

[42] Tiempo Universal Coordinando . Página de la Wikipedia (*https://es.wikipedia.org/wiki/Tiempo_universal_coordinado*). Consultado en Septiembre de 2016.