UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Gestión desatendida y remota del software instalado en equipos de un dominio de Active Directory

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Carlos Salgado Bartolomé

Tutor: Mario Gómez Martinez

Curso 2016-2017

Resumen

Este proyecto consiste en el desarrollo de una aplicación software para gestión la instalación, actualización y desinstalación desatendida de diferentes aplicaciones en equipos de un entorno empresarial con dominio de Active Directory y sistemas de escritorio Windows.

Esto se solventa creando una aplicación de gestión centralizado que administra terminales remotas de PowerShell y realiza instalaciones de paquetes de software de forma similar al apt-get de linux.

Aprovechar este misma aplicación para tener un Inventario de ordenadores, características, software instalado y software disponible.

Palabras clave: Active Directory, PowerShell, Chocolatey, WinRM, Instalación

Resum

Aquest projecte consisteix en el desenvolupament d'una aplicació software per a gestionar l'instal·lació, actualització i desinstal·lació desatesa de software en equips d'un entorn empresarial amb domini d'Active Directory i sistemes d'escriptori Windows.

Creant un programa de gestió centralitzat que administra terminals remotes de PowerShell i realitza instal·lacions de paquets de software de manera similar al apt-get de linux.

A més, aprofitarem aquest mateix programa per tenir un Inventari d'ordinadors, característiques, programari instal·lat i programari disponible.

Paraules clau: Active Directory, PowerShell, Chocolatey, WinRM, Instalació

Abstract

This project consists of developing an aplication to solve the unattended software installation, updating and management on computers in a business environment with Active Directory domain and Windows desktop systems.

Creating a centralized management program that manages remote PowerShell terminals and installs software packages like the command apt-get in linux.

Take advantage of this same program to have an inventory of computers, features, software installed and available software.

Key words: Active Directory, PowerShell, Chocolatey, WinRM, Installation

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Sobre las instalaciones y su problemática

Todos sabemos instalar software en un sistema operativo Windows 10 a través de un instalador. Si nos preguntasen cuál es la mejor manera de instalar nuevo software en nuestro ordenador, todas las respuestas serían buenas con tal de cumplir el objetivo.

Esto que parece realmente simple, cuando se tratan cientos o miles de ordenadores como administrador de sistemas, pasa de ser algo simple a algo realmente complejo; No todas las soluciones son buenas soluciones, hay que tener muchos factores en cuenta y los principales de todos: el tiempo que se pierde del administrador, del usuario y el factor humano a la hora de cometer errores.

El objetivo de un administrador de sistemas es que nadie pierda tiempo con problemas técnicos en sus equipos, garantizar que todos los usuarios tengan **la mayor cantidad de tiempo activo disponible** sin que el propio administrador deje de estar disponible para monitorizar o atender posibles emergencias o averías críticas.

Sobre la infraestructura

Típicamente en las empresas y concretamente en la empresa en la que se realiza este estudio, está implementado un sistema de control, gestión y monitorización de usuarios y equipos conocido como Active Directory.

Active Directory es un servicio de directorio en una red distribuida de ordenadores, esto quiere decir que es un servicio que almacena y organiza información sobre los recursos de red disponibles, ya sean ordenadores, usuarios, grupos, políticas u otros recursos.

La ventaja que nos da Active Directory, es que funciona como un DNS y nos permite a través de una cuenta de administrador del dominio, realizar y elevar nuestros permisos en los equipos como administradores locales. Esta es la herramienta fundamental de los administradores de sistemas para poder realizar cambios en los equipos y limitar o prevenir que los usuarios básicos del dominio puedan realizar cambios importantes en la red o en los equipos.

Sobre el problema

La falta de credenciales para los usuarios nos limita en que solamente las personas con cuentas de administrador[1] pueden realizar instalaciones, por lo que tan solo un grupo reducido de personas tienen acceso a realizar estas acciones, y a la hora de actualizar ciertos programas que no se actualizan automáticamente puede suponer un verdadero reto, ya que es escalable al número de equipos que tengan dichos programas instalados.

Una posible solución consistiría en darle a los usuarios privilegios de administración local de sus equipos, así podrían realizar instalaciones, actualizaciones y desinstalaciones, pero esto es por muchos motivos una mala práctica[2].

Otra solución sería no actualizar los programas, pero esto también es una mala práctica, ya que los programas se actualizan por alguna razón, ya sea desde añadir funcionalidades, corregir errores, aumentar la seguridad, etc.

Tras saber que hemos de mantener los programas actualizados y no podemos dar permisos de administrador a los usuarios, nos econtramos con el problema que mostramos en la siguiente sección.

1.1 Motivación o Problema

Nuestro proyecto intenta solventar este problema, basándonos en las buenas prácticas conocidas y aprendidas en la empresa en la que se desarrolla este proyecto[3], de obtener un *sistema de gestión de software* en equipos, que nos permita ahorrar tiempo y problemas, tanto a los usuarios, como a los administradores.

Con *Sistema de gestión de software*, nos referimos a un programa capaz de monitorizar las versiones instaladas en los ordenadores del dominio que se encuentren conectados a la red, y que sea capaz de instalar nuevos programas y actualizar o desinstalar los programas ya instalados en dichos equipos.

Para esto nos encontramos con diversos problemas: no existen sistemas gratuitos de distribución de software que cumplan con las características mínimas exigidas en seguridad, fiabilidad y escalabilidad, que nos permitan realizar las tareas expuestas anteriormente.

De lo que concluimos que para el proyecto se va a tener que desarrollar una herramienta propia adaptada a nuestro entorno.

[1] Ya sean cuentas locales del equipo o del dominio al que están agregados.

[2] estos motivos podrían sintetizarse en dos grupos: tendrían la capacidad de desconfigurar sus propios equipos o tener brechas de seguridad

[3] Laboratirios Sesderma.

1.2 Objetivos

Entre los objetivos de este proyecto, podemos diferenciar los siguientes apartados:

- Aprender a desarrollar una aplicación de gestión en un entorno real.
- Investigar diferentes alternativas de gestión de equipos para ser automatizada en la aplicación a desarrollar.
- Investigar diferentes métodos de instalación de software y aplicar el más adecuado a nuestro entorno.
- Aprender sobre las buenas prácticas de administración de sistemas y cumplirlas en nuestra aplicación.
- Desarrollar una aplicación con la que abarcar todos los apartados anteriores y dar solución al problema expuesto.

1.3 Estructura

1. **Capítulo 1: Contextualizaciñon.**

   En este capítulo se describe una introducción al proyecto, un contexto en el que se expone una problemática que se pretende solventar con unos objetivos a cumplir y una estructura para dar al lector a entender los apartados o capítulos de esta memoria.

2. **Capítulo 2: Análisis de aplicaciones, herramientas y tecnologías similares.**

   En este capítulo se explican con detenimiento diferentes herramientas que ya tienen funcionalidades similares a las que se propone en el problema de la introducción, se analizan las propiedades de diferentes tecnologías o herramientas a utilizar en nuestro proyecto y se explica brevemente el por qué de las elecciones tomadas.

3. **Capítulo 3: Especificación, Diseño y Requisitos.**

   En este capítulo se definen más concretamente diferentes herramientas a utilizar, los requerimientos que tendrá nuestra aplicación, y se define una estructura mediante diferentes tipos de diagramas, para especificar elementos a desarrollar en el siguiente capítulo.

4. **Capítulo 4: Desarrollo de protoripo.**

   En este capítulo se expone la metodología empleada para la realización del prototipo, se muestran capturas de pantalla del prototipo, se evalúa con

diferentes métricas y se comprueba si se cumplen los requerimientos definidos en el apartado anterior. Además, se realiza una valoración sobre el propio prototipo y si merece la pena continuar con él como una solución viable.

5. **Capítulo 5: Mejoras Aplicables al prototipo.**

   En este capítulo se diagnostican bugs encontrados en el prototipo y se proponen y explican diferentes mejoras aplicables al propio prototipo.

6. **Capítulo 6: Conclusiones.**

   En este capítulo se realizan valoraciones sobre los aprendizajes que se han realizado durante el proyecto y la carrera, y se explican brevemente diferentes metodologías utilizadas que se han aprendido en la carrera.

7. **Apéndice A: Preparación del entorno.**

   Este apéndice consta de un tutorial que cosiste en preparar el entorno para poder implementar nuestro proyecto en un entorno empresarial.

8. **Apéndice B: Seguridad**

   Este apéndice trata sobre poblemas de seguridad que podrían derivarse de un mal uso de nuestro proyecto[4], y cómo podrían solventarse los mismos con buenas prácticas.

[4] O en el caso de la aplicación resultante de este proyecto, trata de la seguridad relativa a las herramientas usadas (Chocolatey).

CAPÍTULO 2
Análisis de aplicaciones, herramientas y tecnologías similares.

En este capítulo se pretende hacer un análisis de las herramientas disponibles, evaluar sus características, analizar las ventajas y desventajas de las mismas e identificar las funcionalidades que queremos que tenga nuestra solución.

2.1 Productos similares

Para obtener una mejor idea de las funcionalidades que queremos en la aplicación, conviene revisar diferentes aplicaciones que tengan usos similares.

2.1.1. PDQDeploy + PDQInventory

La suma de estos dos productos de la misma empresa (PDQ) es lo que realmente nos inspira para darle las funcionalidades que acabará teniendo el proyecto final que se quiere desarrollar.

PDQDeploy es un programa que permite instalar programas y parches a través de paquetes en una red corporativa. Estos paquetes se crean a mano eligiendo los componentes que se quiere instalar y los argumentos/parámetros que le quiere pasar, no necesita un agente remoto en cada equipo.

De este programa se toma el concepto de instalar software a través de paquetes.

PDQInventory este programa escanea los ordenadores que haya en la red o Active Directory y obtiene su hardware, software, configuraciones de Windows. Tiene más funcionalidades como Wake On Lan y un escritorio remoto.

De este programa se obtiene la idea de crear un inventario de software en equipos.

Ambos programas están pensados para usarse conjuntamente para crear un gestor de software en equipos similar al que se desea crear en este proyecto, pero estos son programas de pago con licenciamiento de mil euros anuales[1].

[1] A fecha de Febrero de 2017.

2.1.2. manage engine (Service Desk Plus)

Este programa gestiona diferentes campos de tecnologías de información, gestión de Active Directory, tickets, MDM, redes, servidores, seguridad y gestión de aplicaciones. Se pueden hacer despliegues de software, pero este no tiene paquetes predefinidos, está mejor pensado para automatizar el despliegue de parches puntuales más que para hacer un sistema de gestión de software, ya que no tiene forma de administrar el software ya instalado en las máquinas, ni gestionar las versiones instaladas en las mismas.

Este programa da la impresión de caer varios pasos atrás de las soluciones de PDQ en cuanto a gestión de software, no resulta como solución viable.

2.1.3. conclusiones sobre las aplicaciones del mercado

Tras analizar el mercado de este tipo de aplicaciones podemos concluir que son similares entre ellas en concepto.

Se pueden diferenciar tres áreas funcionales:

- Una parte de gestión de **ordenadores**. Es necesario conocer de antemano los ordenadores en la red, el software ya instalado y otras características que nos puedan ser de utilidad, como sistema operativo o arquitectura de 32,64 bits.

- Una parte de gestión de **paquetes**. Gestionar el software que se va a instalar a través de paquetes predefinidos. Estos pueden ser de diferentes tipos: exes, msi configurados por parámetros o scripts que automatizan los anteriores.

- Una parte de gestión de **despliegue**. Para saber cómo realizar una instalación en un ordenador remoto de forma desatendida y cómo obtener resultados que puedan ser gestionados automáticamente.

Habiendo diferenciado la solución en estas tres partes, habrá que buscar tecnologías que nos puedan servir para gestionar cada una de ellas.

2.2 Selección de tecnologías de control disponibles

Ahora vamos a buscar la tecnología de control de ordenadores necesaria para nuestra aplicación. Puntos a tener en cuenta pueden ser:

- Facilidad de despliegue.
- Facilidad de obtención de feedback de las gestiones.
- Seguridad.
- Requisitos previos para el funcionamiento.

2.2.1. Telnet

Telnet es el nombre de un protocolo de red para conectarse a otra máquina para manejarla remotamente como si estuviéramos sentados delante de ella.

Ventajas: Telnet es un protocolo estandar y se puede utilizar con prácticamente la totalidad de los ordenadores de una empresa y cumple las necesidades de control y procesamiento.

Desventajas: En un principio no parece seguro, ya que se envía todo como texto plano, incluido las posibles credenciales necesarias para realizar la instalación, lo cual puede suponer varios agujeros de seguridad importantes en nuestra empresa[2].

2.2.2. GPO (Group Policy Object)

GPO es una característica de Windows; Es un conjunto de reglas que controlan el entorno de trabajo de cuentas de usuario y cuentas de equipo, administran la gestión centralizada y configuración de sistemas operativos, aplicaciones y configuración de los usuarios en un entorno de Active Directory.

Ventajas: está diseñado para integrarse dentro del dominio de Active Directory empresarial y ya hay muchos ejemplos y guías de distribución software via GPO, *usando esto se aprovecharía la infraestructura de Active Directory ya disponible para estructurar y organizar los equipos conectados a la red.* También tendría un sistema totalmente desatendido y automatizado, por lo que no gasta tiempo de administrador en los despliegues.

Desventajas: no tendría un buen sistema de gestión para el software ya instalado, ni de los errores de instalación, a través de logs de eventos que irían rellenandose automáticamente. Otra desventaja sería tener diferentes programas instalados en diferentes equipos, se tendría que generar una estructura de arbol muy compleja en la que se asignen paquetes de software diferentes prácticamente por cada usuario.

[2] Puede suponer varios agujeros de seguridad importantes en nuestra empresa al tener que usarse credenciales de administrador de dominio para realizar las instalaciones, si se obtuviesen estas credenciales, todo el dominio correría un grave peligro.

2.2.3. PSExec (PsTools)

PsExec es una suite de programas/comandos que permiten ejecutar comandos en ordenadores remotos.

Ventajas: está diseñado para lo que queremos específicamente: ejecutar un programa en un ordenador a través de una única línea de comandos y devuelve el log de la instalación.

Desventajas: al igual que telnet, manda la contraseña en texto plano, por lo que usar esto es una mala práctica para administrar contraseñas de administrador[3].

2.2.4. WinRM + PowerShell

Windows Remote Management es la implementación de Microsoft del protocolo *WS-Management Protocol*, un protocolo simple de acceso a objetos *(SOAP)-based* que permite la interoperatibilidad entre hardware y sistemas operativos, de un modo no conflictivo con el firewall.

***Windows PowerShell** es una interfaz de consola (CLI) con posibilidad de escritura y unión de comandos por medio de instrucciones (scripts en inglés). Es mucho más rica e interactiva que sus predecesores, desde DOS hasta Windows 7. Esta interfaz de consola está diseñada para su uso por parte de administradores de sistemas, con el propósito de automatizar tareas o realizarlas de forma más controlada.*[4]

Las características de administración remota de PowerShell están administradas por el protocolo *WS-Management Protocol* y el servicio *Windows Remote Management (WinRM)* que lo implementa en Windows.

Ventajas: PowerShell no trabaja con texto, trabaja con objetos lo cual es una gran ventaja, nos permite ejecutar y administrar cualquier equipo Windows de nuestra red que es exactamente lo que queremos.

Desventajas: hay que habilitar el servicio de WinRM en todos los ordenadores del dominio que queramos administrar.

2.2.5. Conclusión de control

Tras analizar los puntos a tener en cuenta para la selección de teconolgías de control, PowerShell parece ser la mejor manera de gestionar ordenadores de forma remota, ya que este nos proporciona información en tiempo real, a diferencia de la GPO, y está diseñado para ser seguro, a diferencia de Telnet o PsExec.

[3] Ya se explica bevemente en telnet el por qué de esto.

[4] https://es.wikipedia.org/wiki/Windows_PowerShell

2.3 Selección de tecnologías de Instalación disponibles

Una vez decidido que PowerShell y WinRM es la mejor combinación de herramientas para los despliegues de software a realizar, ahora se tiene que ver qué tipo de tecnología se puede usar usar para realizar la instalación de los paquetes. Requisitos a tener en cuenta pueden ser:

- Facilidad de crear paquetes desatendidos.
- Facilidad para obtener feedback de las instalaciones.
- Garantías de funcionalidad.
- Velocidad de instalación / Carga de instalación.
- Seguridad.

2.3.1. MSI (Windows installers)

Windows Installer, previamente conocido como Microsoft Installer (MSI), es un motor para la instalación, mantenimiento y eliminación de programas en plataformas Microsoft Windows.

Los paquetes MSI se definen como instaladores de Microsoft, son paquetes de software que contienen la información necesaria para automatizar su instalación, minimizando la intervención manual del usuario, ya que toda la información iría contenida en el propio archivo o fichero en formato .msi. [5]

Este es el primer método en el que se podría pensar, ya que los MSI son paquetes especialmente diseñados para el objetivo del proyecto: instalación desatendida de software en masa con una línea de comando.

El mayor problema es encontrar los parámetros de cada .msi y realizar las pruebas para comprobar la correcta funcionalidad de los paquetes. Esto mismo escalado a la cantidad de software que hay en una empresa, escalado a la cantidad de actualizaciones que hay al año, podría considerarse que no es el mejor método a aplicar ya que requeriría mucho tiempo para desarrollar los paquetes y pruebas, sobre todo comparado con el resto de tecnologías detalladas a continuación.

Tampoco tiene el mejor sistema de feedback, ya que los propios paquetes MSI no tienen una interfaz estandarizada de retorno.

2.3.2. Ninite free

Ninite.com[6] es una web de la cual se descargan ejecutables que automatizan la instalación de paquetes de software elegidos previamente en la web, de forma desatendida, sin toolbars y gratuitamente.

[5] https://es.wikipedia.org/wiki/Windows_Installer

[6] En su versión gratuita se ofecen estas características, la versión de pago ofrece más funcionalidades.

Ninite parece a simple vista, una de las mejores soluciones que se podrían aplicar, ya que no ocuparía tiempo en la creación de paquetes que se quieran instalar. Instalar y actualizar se hace de la misma forma, no instala toolbars ni ningún tipo de malware y garantiza que la instalación sea correcta.

Entre los problemas de Ninite (al menos en su versión gratuita) se encuentra que los paquetes disponibles están muy limitados en cantidad, no se pueden gestionar paquetes propios, se tienen que obtener de su repositorio en la nube (lo cual supone una sobrecarga de la red) y una vez se llega a realizar pruebas, es bastante lento y pesado para el sistema.

Tampoco ofrece un buen sistema de feedback de instalación para ser automatizado a través de una aplicación, ya que este feedback sólo se encuentra en la interfaz de usuario.

2.3.3. Chocolatey Open Source

Chocolatey es un gestor de paquetes de instalación de software inspirado en el *apt-get* de linux, con su propio repositorio de paquetes alimentado por la comunidad y las mismas empresas de software que crean su propio "*choco package*".

Se define a sí mismo como *"The package manager for Windows, Chocolatey - Software Management Automation*".

Las ventajas que ofrece Chocolatey son tener paquetes ilimitados (alrededor de 5000 paquetes de instalación), poder crearse paquetes propios, poder acceder a base de datos propia en vez de a su repositorio en la nube. Es altamente personalizable y al realizar pruebas es bastante mas ligero y rápido que ninite.

La mayor contra es que algunas aplicaciones podrían contener malware/adware oculto, al tratarse de paquetes generados por la comunidad. Pero por otro lado, Chocolatey conoce este defecto y contrasta con más de 50 antivirus sus paquetes, además de registrar el nombre de los publicadores de los paquetes. [7]

2.3.4. Conclusión de Instalación

Tras analizar y realizar pruebas superficialmente, Chocolatey parece la forma más cómoda de alcanzar la solución que queremos, ya que este tiene un método de retorno estandarizado de paquetes, ya ofrece su propia gestión de paquetes instalados, e instalar nuevos paquetes es tan simple como ejecutar remotamente una línea de comando.

Este cumple cuatro de los cinco requisitos propuestos para la selección de método:

- **Facilidad de crear paquetes desatendidos**: la mayoría ya están creados.
- **Facilidad para obtener feedback de las instalaciones**: es el mejor feedback de todos lor propuestos.

[7] Más adelante en el Apéndice B de esta memoria, se explica más sobre los detalles de seguridad de Chocolatey, el cual, es la herramienta que se utilizará para el desarrollo de este proyecto.

- **Garantías de funcionalidad**: no es el que mejor garantiza las instalaciones, no obstante no es habitual que falle.
- **Velocidad de instalación / Carga de instalación**: no es el más rápido, pero tiene una velocidad asumible y es el que menos sobrecarga el procesador. [8]

Respecto a la **Seguridad**: convendría analizar la fuente y revisar el análisis de antivirus cuando se vaya a instalar un nuevo paquete. En un principio no han ocurrido eventos graves respecto a esto, dado que Chocolatey tiene que dar la aprobación del paquete antes de publicarlo y distribuirlo en su repositorio (chocolatey.org). [7]

2.4 Conclusiones sobre el Análisis

Tras realizar el análisis de las aplicaciones, herramientas y tecnologías disponibles, sabemos que queremos una aplicación de escritorio centralizada que pueda:

· **Inventariar ordenadores** y su software de forma desatendida.

· **Administrar estos ordenadores** a través de WinRM + PowerShell.

· **Instalar, actualizar, gestionar paquetes de Chocolatey** a través de WinRM.

A continuación se **especificará** cómo implementar esta solución.

[8] Más tarde en esta memoria se descubre que sí es el sistema más rápido, a simple vista no lo parecía, pero tras realizar pruebas se demuestra que es un sistema sorprendentemente rápido.

CAPÍTULO 3
Especificación, Diseño y Requisitos

Este capítulo trata sobre toda la planificación necesaria para empezar a desarrollar la solución al problema planteado en el primer capítulo. Se diseñará y especificarán las propiedades que este debe cumplir: técnica, funcional, visual, estructural y arquitectónicamente. A continuación expondremos las tecnologías a utilizar para este desarrollo.

3.1 Tecnologías a utilizar

3.1.1. Windows SDK

Windows tiene un *namespace* que es *System.Management.Automation*, y es el root-namespace del motor de comuicación de C# con PowerShell. Este contiene clases, enumeraciones e interfaces hechas para la invocación del mismo. Es el namespace que se utiliza para crear Cmdlets y PSCmdlets en Microsoft.

En resumidas palabras, el SDK de Windows ofrece una librería que permitirá usar elementos ya existentes para crear un motor de ejecución de PowerShell, así que parte de este proyecto consistirá en encapsular en nuestro código las utilidades ya implementadas de este motor.

3.1.2. WPF

Este punto se resume a elegir ente *WPF* y *Windows Forms*, se ha escogido WPF por estos puntos:

- La capacidad de **personalización**, en la cual *Forms* se queda corto.
- La capacidad de ***Enlace de datos***, mediante la cual puedo enlazar elementos de la interfaz a elementos de la lógica y estos se comunican las modificaciones a través de eventos sin tener que implementar manualmente esa lógica.
- Microsoft ha dejado de sacar actualizaciones de *Windows Forms* para centrarse activamente en el desarrollo de WPF. De esto podemos prever que WPF será más fácilmente un estándar de mercado a corto plazo.

3.2 Requerimientos Funcionales

RF.01 Conexión con ordenadores de la red:

Poder crear un canal seguro de conexión y crear remotamente una terminal de PowerShell en segundo plano con capacidad de administración.

RF.02 Instalación de agente en terminales:

Configuración de los ordenadores para aceptar scripts externos e instalación del agente de Chocolatey en las terminales que van a ser administradas.

RF.03 Obtención por red de información en terminales:

Obtener a través de la red información como nombres de ordenador, para ayudar a tomar decisiones administrativas.

RF.04 Inventariado de los terminales:

Tener una lista actualizada de todos los ordenadores que hay en la red, ordenadores que tengan el agente instalado.

RF.05 Inventariado del software disponible:

Tener una lista local del repositorio de paquetes disponible de Chocolatey para poder conocer el software que hay disponible para instalar en los equipos.

RF.06 Gestión del software en los terminales:

Obtener la lista del software instalado y su versión de cada ordenador del inventario.

RF.07 Creación de paquetes de software:

Poder crear paquetes personalizados de software para agilizar y estandarizar el software que hay distribuido por los equipos.

RF.08 Distribución desatendida de software:

Poder realizar instalaciones o actualizaciones desatendidas de software a través de la red.

3.3 Requerimientos No Funcionales

Sobre los requerimientos no funcionales tenemos que encontrar un equilibrio entre ellos, esperando a necesitar realizar pruebas para poder calibrar un equilibrio.

RNF.01 Evitar sobrecargar la red:

Evitar tener la red corporativa sobrecargada es el Requerimiento no funcional más prioritario, ya que una aplicación no debe evitar que el resto de aplicaciones que utilicen la red dejen de funcionar o no funcionen apropiadamente. Es menos prioritario la actualización de software que el funcionamiento habitual de la empresa.

RNF.02 El despliegue tiene que ser ágil:

Sería conveniente realizar los despliegues en paralelo, sin que eso afecte a la sobrecarga de la red (RNF01).

RNF.03 El inventario tiene que estar actualizado:

Para tener las últimas versiones de software disponible conviene tener los ordenadores actualizados, hacerlo de forma rutinaria, cíclica y consumiendo pocos recursos. Ya que se pueden instalar componentes de forma separada a la misma instancia del programa de gestión. Esto tampoco tiene una prioridad crítica, pero sí podría necesitarse de forma ágil puntualmente.

3.4 Requisitos de instalación previos para el funcionamiento

3.4.1. Dominio de Active Directory

Se necesita una infraestructura de Active Directory, con todas las terminales que se pretendan administrar agregadas al dominio, esto viene de la mano con un servidor DNS que conozca los nombres de dominio de los equipos y su dirección IP.

3.4.2. WinRM configurado

Se necesita tener activado el servicio de WinRM en los equipos que se pretenda administrar y se necesita que esté configurado para aceptar solicitudes de administración remota por el equipo que vaya a usarse para enviar las peticiones.

3.4.3. Instalar Chocolatey, política de scripts

Chocolatey se instala a través de un script en una web.

Esto se puede ejecutar con una sola línea de código de PowerShell, en una terminal con la variable de sistema *ExecutionPolicy*[1] ajustado al menos en *RemoteSigned*:

iex ((New-Object System.NET.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

[1] https://technet.microsoft.com/es-es/library/ee176961.aspx

3.5 Diagramas arquitectónicos

A continuación se mostrará una gráfica, la figura 3.1 [2], en el que se muestran los diferentes dominios que existirán en nuestra solución y una breve explicación de las conexiones o comunicaciones que habrán entre ellos. Con este diagrama se pretende dar una idea general de cómo funcionará nuestra aplicación.

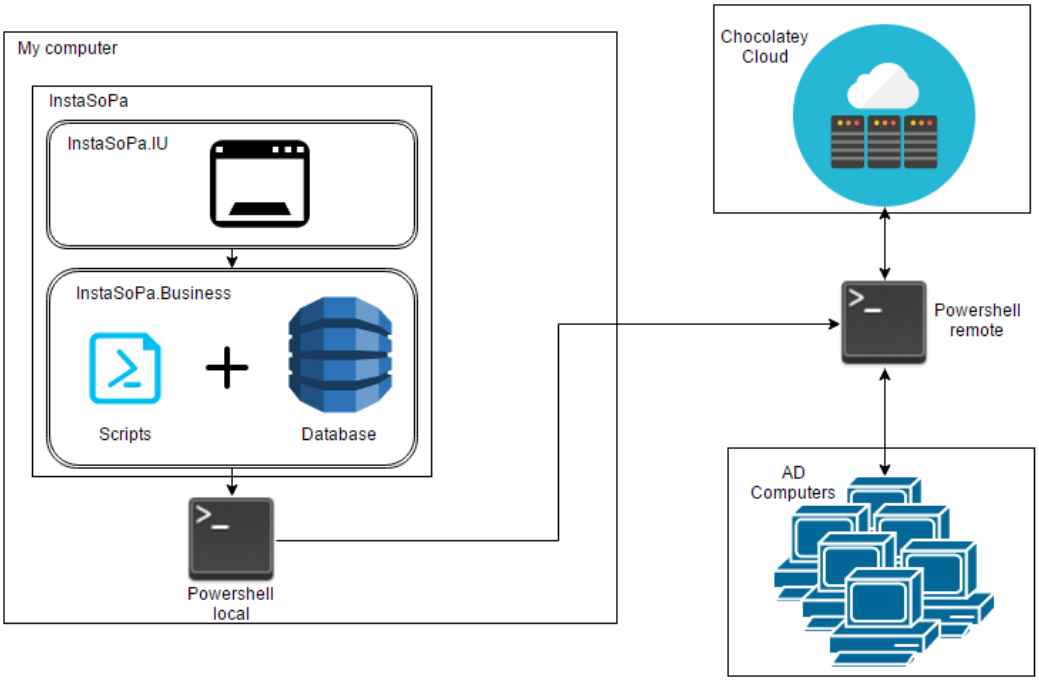

Figura 3.1: Especificación: Arquitectura y Dominios

- **InstaSoPa:** Este será el nombre que le hemos dado a la aplicación, viene de **Insta**lación de **So**ftware por **Pa**quetes. Este dominio comprende todo el ensamblaje a desarrollar para la funcionalidad que se busca.
- **PowerShell local/remote:** Estas son las terminales de PowerShell en segundo plano, transparentes para los usuarios, que realizarán la tarea de comunicación mútua y comunicación entre nuestra aplicación, la nube de Chocolatey y el sistema operativo de los usuarios.
- **Chocolatey Cloud:** El repositorio de paquetes de chocolatey.org del cual nosotros descargaremos los paquetes para instalarlos en las terminales.
- **AD Computers:** AD son las siglas de **A**ctive **D**irectory, Representa todas las terminales que queremos administrar con nuestra aplicación.

[2] Pasado a limpio, ya que esto se realizó con bolígrafo y papel en su momento.

3.6 Diseño de interfaces

En esta sección se muestran bocetos de las ventanas que se prevee que tenga la aplicación y sus funcionalidades. De la aplicación podemos abstraer tres funcionalidades independientes, las cuales separaremos en tres pestañas/ventanas respectivamente:

3.2 Ordenadores a administrar.

3.3 Gestión de packs.

3.4 Despliegue de aplicaciones.

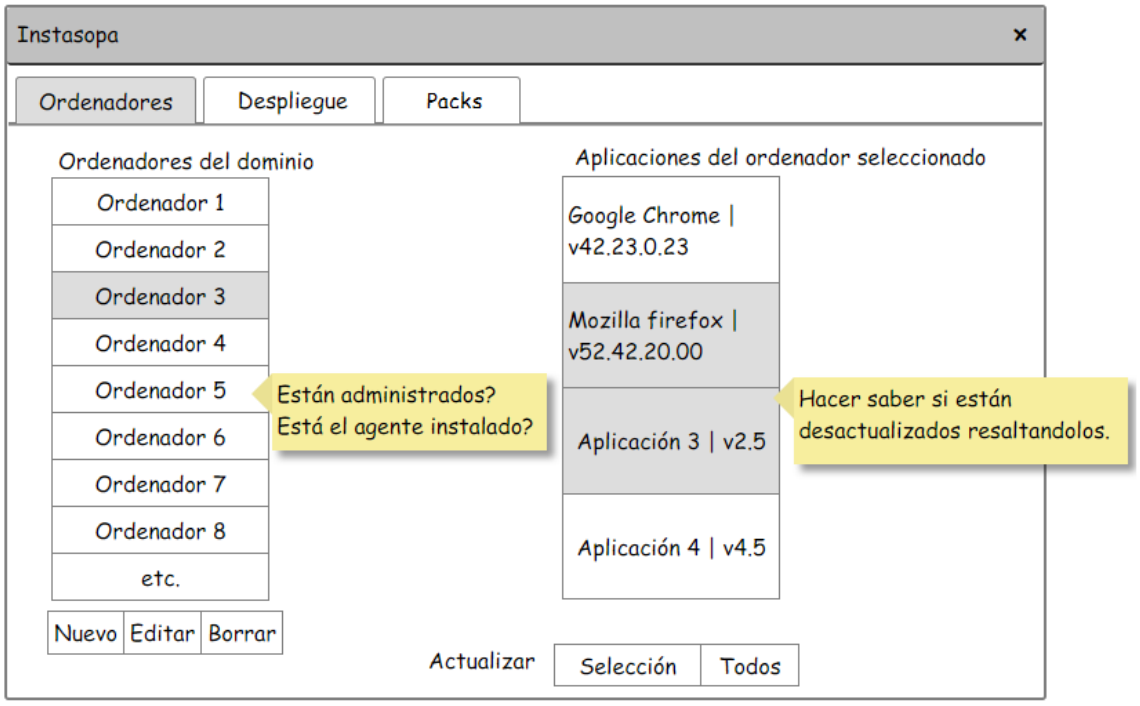

Figura 3.2: Diseño de interfaces: Ordenadores a administrar

- **Ordenadores del dominio:** Son los ordenadores que queremos administrar con nuestra aplicación, en concreto estos son ordenadores conectados actualmente que tienen instalado el agente de Chocolatey.
- **Aplicaciones del ordenador seleccionado:** Son las aplicaciones instaladas mediante el agente de Chocolatey en el ordenador de dominio seleccionado actualmente. Esta lista debería reflejar el nombre de la aplicación, la versión y si se encuentra desactualizada.
- **Nuevo/Editar/Borrar:** Estos botones son para agregar, editar o borrar ordenadores de la lista de ordenadores del dominio.
- **Actualizar Selección/Todos:** Estos botones sirven para actualizar las aplicaciones seleccionadas o actualizar todas las aplicaciones desactualizadas respectivamente.

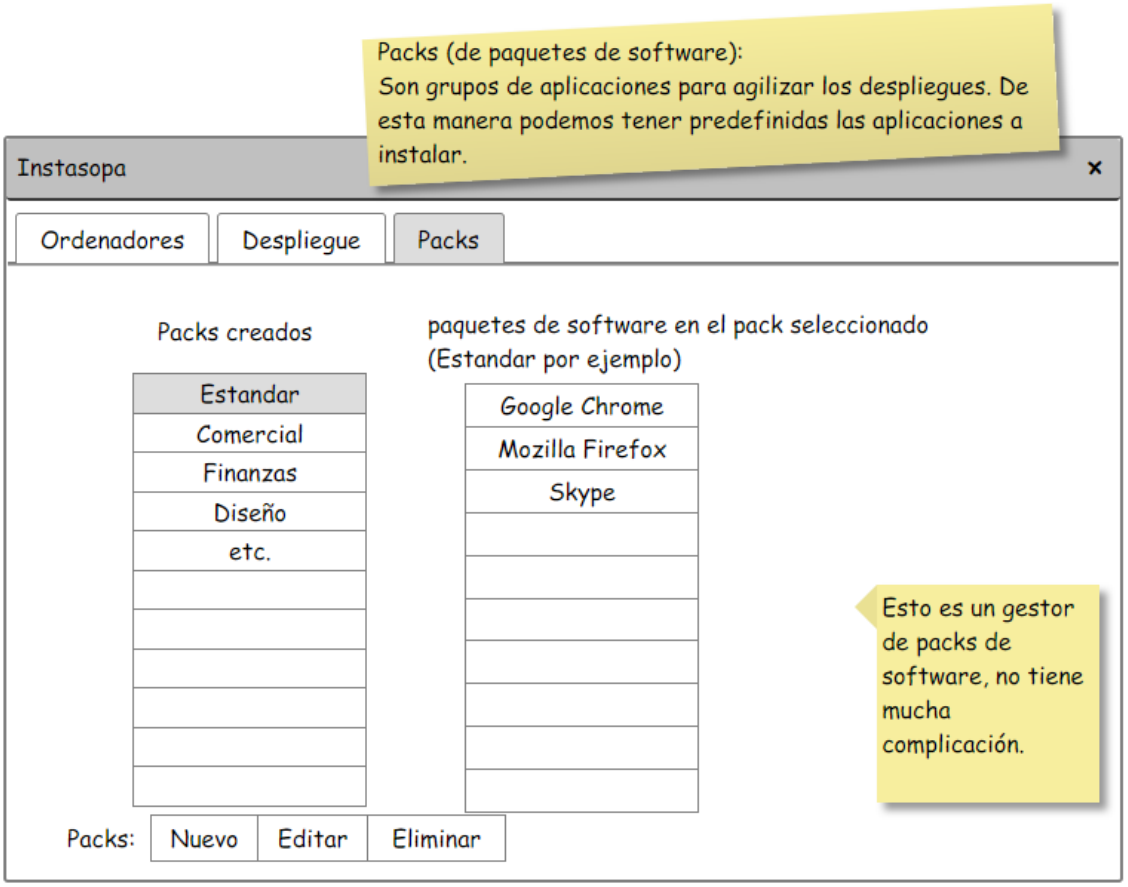

Figura 3.3: Diseño de interfaces: Gestión de Packs

- **Packs Creados:** Estos son los Packs (packs de paquetes) creados por el usuario. El uso de Packs está pensado para contener varios paquetes de chocolatey.org, de tal forma que se estandarice el software a instalar en los despliegues sucesivos.
- **Paquetes de software en el pack seleccionado:** Esta es la lista de paquetes de software que contiene el Pack seleccionado en el momento.
- **Nuevo/Editar/Eliminar:** Estos son botones para gestionar la creación, modificación o eliminación de Packs.

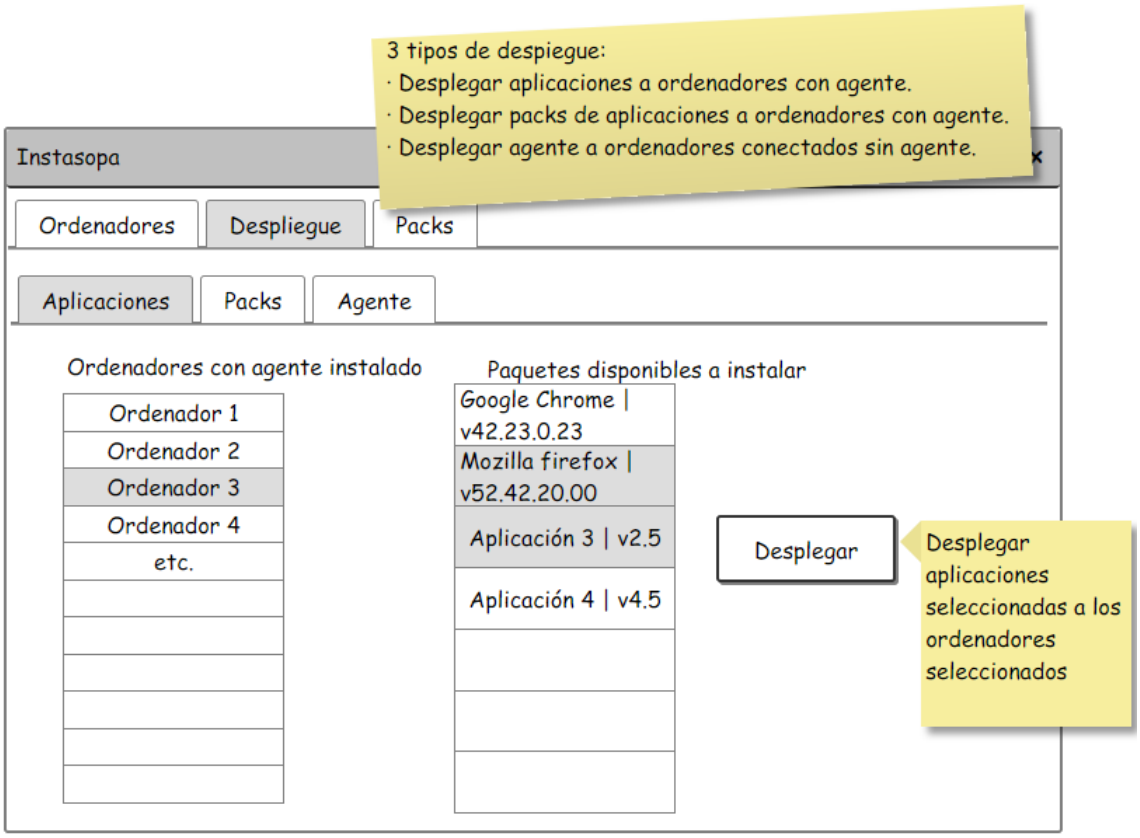

Figura 3.4: Diseño de interfaces: Despliegue de aplicaciones

- **Subpestaña: Despliegue → Aplicaciones:** En esta subpestaña se realiza el despliegue de las aplicaciones seleccionadas en los ordenadores seleccionados.
- **Subpestaña: Despliegue → Packs:** En esta subpestaña se realiza el despliegue de los Packs seleccionados en los ordenadores seleccionados.
- **Subpestaña: Despliegue → Agente:** En esta subpestaña se realiza el despliegue del agente en los ordenadores seleccionados de una lista específica de ordenadores conectados sin el agente instalado.

3.7 Diagrama de elementos del prototipo

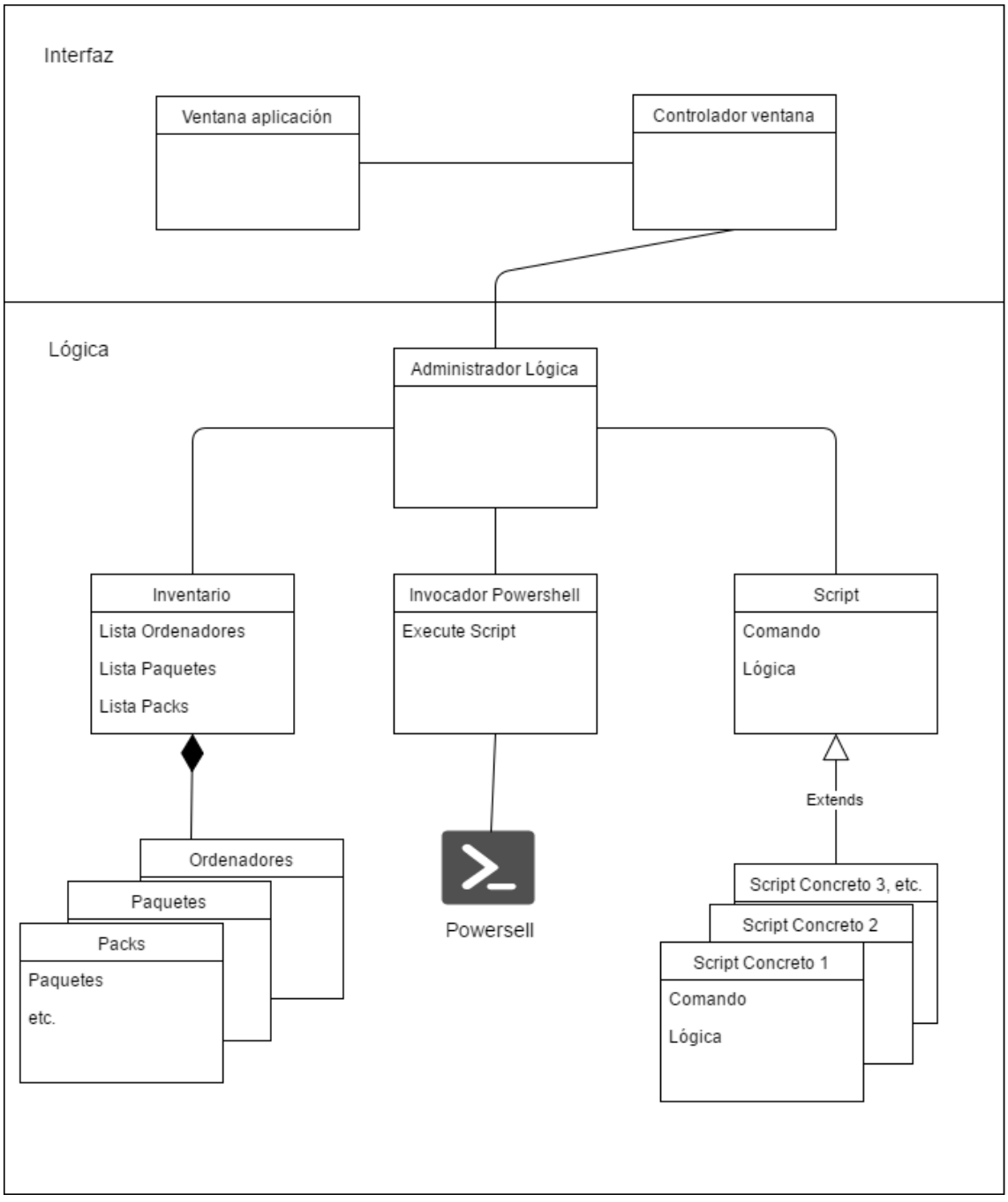

Figura 3.5: Diagrama de clases prototipo

- **Venatana aplicación:** Esto contendrá la interfaz de usuario de la aplicación, siguiendo el patrón "*Model View ViewModel*", este sería nuestro *View*.
- **Controlador ventana:** Es una clase que tendrá la lógica de la IU y realizará la comunicación con el *Model* (Administrador lógica), siguiendo el patrón "*Model View ViewModel*", este sería nuestro *ViewModel*.
- **Administrador Lógica:** Esta clase crea una capa de abstracción de la lógica de la aplicación de tal manera que se centraliza toda la lógica en esta clase, en el patrón "*Model View ViewModel*", este sería nuestro *Model*.
- **Inventario:** Esta clase contiene todos los objetos variables de nuestra aplicación: ordenadores, paquetes y packs. Sigue un patrón *singleton* para garantizar unicidad en nuestros datos.
- **Pack/Paquete/Ordenadore:** Estas clases encapsulan la información de los datos a administrar en nuestra aplicación: nombres, versiones y las interconexiones entre ellos mismos. Por ejemplo: los paquetes instalados en un ordenador concreto o los paquetes contenidos en un pack.
- **Invocador PowerShell:** Esta clase contiene toda la lógica para crear una o varias instancias de PowerShell y realizar la comunicación entre el dominio de nuestra aplicación y el dominio propio de Windows:

  system.management.automation.
- **Script:** Esta clase abstracta define una interfaz para los Scripts Concretos. Esta clase a su vez se usa de variable para contener cualquier Script Concreto.
- **Script Concretos (varios):** Estas clases encapsulan toda la lógica para crear nuestros comandos de PowerShell, obtener el valor de retorno y cualquier otra lógica que ataña a cada script en particular.

CAPÍTULO 4
Desarrollo de Prototipo

Para la implementación de esta aplicación, teniendo en cuenta que no tenemos un cliente ajeno, se ha utilizado un desarrollo evolutivo, más concretamente el desarrollo de un prototipo que nos permita ver sin demasiado esfuerzo cuales son los puntos fuertes y débiles de la aplicación especificada previamente.

4.1 Metodología empleada

Para el desarrollo de la aplicación, no sólo del prototipo, se ha utilizado un tipo de **desarrollo iterativo / incremental**.

Ya que el proyecto tiene un objetivo muy claro y conciso[1], se puede realizar esta funcionalidad brevemente en un prototipo y sobre las conclusiones halladas en este prototipo ir añadiendo funcionalidades colaterales como implementar diferentes formas de agregar ordenadores, almacenar o cargar diferentes perfiles de usuario, implementar métodos más completos para procesar el retorno de las peticiones de WS-Management, etc.

4.2 Desarrollo del prototipo

A continuación vamos a desarrollar el prototipo, el cual presentaremos y evaluaremos en las siguientes secciones de este capítulo, esta sección es breve en la memoría, pero es la más larga en tiempo real del proyecto.

[1] Instalar software en equipos remotos.

4.3 Prototipo funcional

En esta sección se mostrará la interfaz del prototipo desarrollado, explicaremos brevemente los puntos fuertes y débiles encontrados en la aplicación.

4.3.1. Interfaz

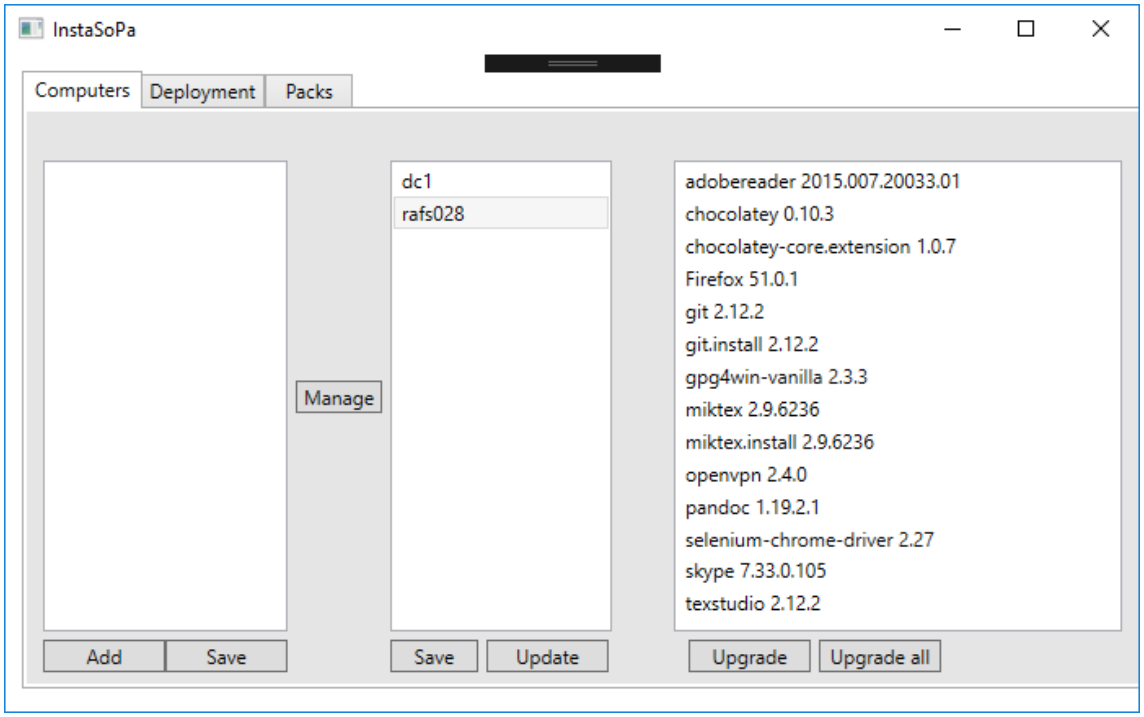

Figura 4.1: Prototipo: Pestaña ordenadores

Esta es la pestaña usada para administrar ordenadores y su software instalado, todas las funcionalidades de ver los programas instalados con su versión respectiva, actualizar un paquete en concreto o actualizar todos funcionan correctamente.

Tras realizar pruebas como usuario, se pueden encontrar varios fallos de diseño como:

- Se necesitan más métodos para agregar ordenadores, añadir ordenadores por string está bien, pero no es eficiente para agregar en masa.
- Añadir una barra clásica de "Archivo", para poder cargar listas de ordenadores o guardar un log en formatos xml o xslx con los ordenadores y su software
- Añadir un sistema de respuesta, para tener información de los fallos que han podido ocurrir al realizar una actualización o si se han completado correctamente.
- Añadir un sistema para poder instalar el agente desde esta ventana y unificar la lista de ordenadores con agente y la lista de ordenadores sin agente en una lista que los diferencie.

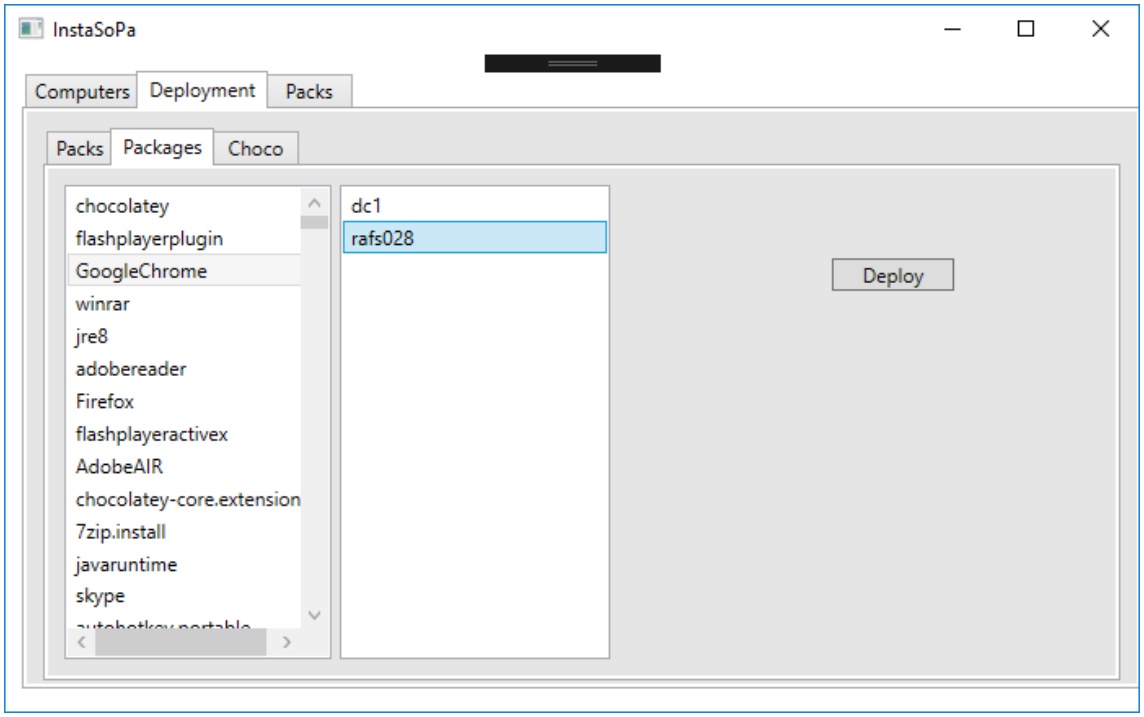

Figura 4.2: Prototipo: Pestaña despliegue

Esta es la pestaña usada para realizar despliegues de paquetes, packs o el propio agente. Se selecciona el paquete/pack y se selecciona el ordenador y se le da al botón *Deploy*.

Tras realizar pruebas como usuario, se pueden encontrar varios fallos de diseño como:

- Está mal aprovechado el espacio, se podría aprovechar la zona de la derecha para añadir otras funcionalidades de la interfaz.
- Añadir una barra clásica de "Herramientas", para poder cargar diferentes configuraciones de cómo realizar despliegues de paquetes, si se desea en paralelo, que congestionaría más la red, o realizar en serie, que nos permite métodos más simples de obtener información de vuelta.
- Añadir un sistema de respuesta, para tener información de los fallos que han podido ocurrir al realizar un despliegue o si se han completado correctamente.
- La subpestaña de distribuir el agente es redundante, se podría poner ambas subpestañas "Packs" o "Packages".

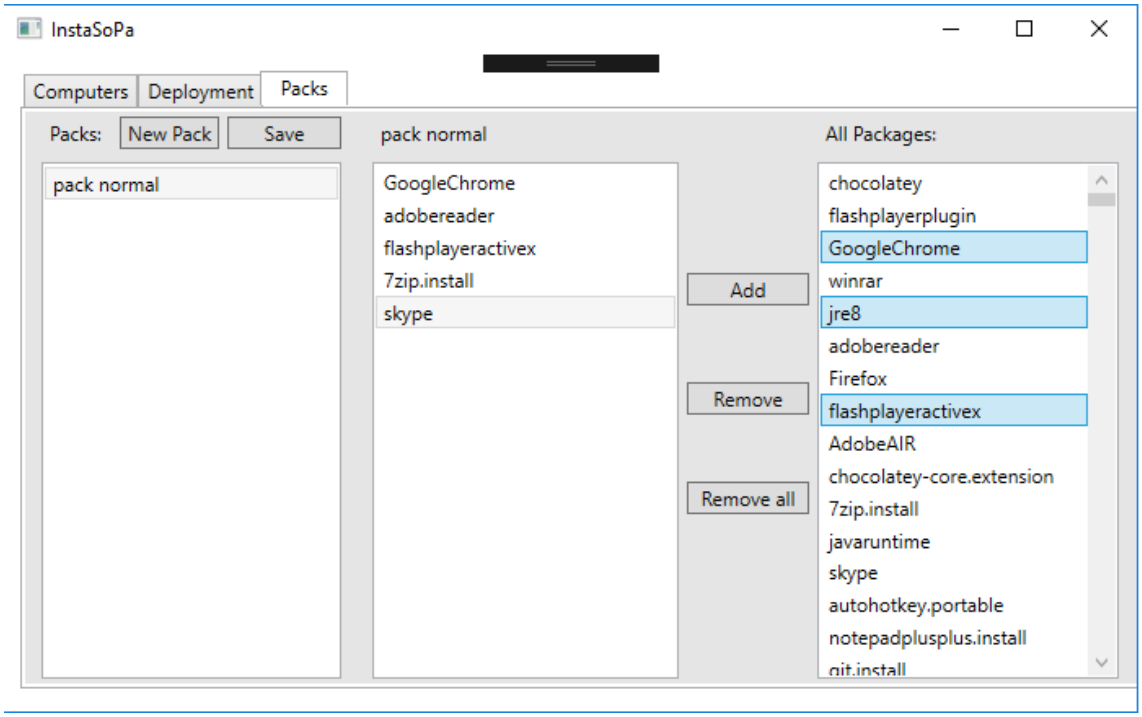

Figura 4.3: Prototipo: Pestaña packs

Esta es la pestaña usada para crear nuevos packs, ver o modificar los packs existentes.

Tras realizar pruebas como usuario, se pueden encontrar varios fallos de diseño como:

- Se necesitan más controles para crear packs, sólamente están implementados *New Pack* y *Save*, se debería de implementar también un botón de borrar.
- Añadir una barra clásica de ´ Archivo", para poder guardar la configuracion o exportar/importar packs existentes.

Esta es a primera vista la pestaña que menos modificaciones necesite.

4.3.2. Funcionalidad y requerimientos

Para la evaluación de este prototipo se han cumplido todos los requerimientos previos:

- **RF.01 Conexión con ordenadores de la red**
- **RF.02 Instalación de agente en terminales**
- **RF.03 Obtención por red de información en terminales**
- **RF.04 Inventariación de los terminales**
- **RF.05 Inventariación del software disponible**
- **RF.06 Gestión del software en los terminales**

- **RF.07 Creación de paquetes de software**
- **RF.08 Distribución desatendida de software**
- **RNF.03 El inventario tiene que estar actualizado**

4.3.3. Métricas

Métricas temporales de despliegue

Para realizar las métricas temporales de despliegue, se han escogido tres ordenadores, cada uno mejor que el anterior con diferencias significativas en procesador o disco duro y se han cronometrado los tiempos de instalación de diferentes aplicaciones. Con esto queremos evaluar si los tiempos de instalación son aceptables para el requerimiento no funcional: **RNF.02 El despliegue tiene que ser ágil**.

Tabla 4.1: Métricas temporales: Ordenador 1

	Ordenador 1
Manufacturador	ASUSTeK COMPUTER INC.
Modelo	X550CL
Procesador	i5-3337U CPU @ 1.80GHz
Memoria	8GB DDR3
Disco	HDD 5200RM
Sistema Operativo	Windows 10 64bits

Tabla 4.2: Métricas temporales: Ordenador 2

	Ordenador 2
Manufacturador	Dell Inc.
Modelo	OptiPlex 3040
Procesador	i5-6500 CPU @ 3.20GHz
Memoria	8GB DDR3
Disco	HDD 7200RM
Sistema Operativo	Windows 10 64bits

Tabla 4.3: Métricas temporales: Ordenador 3

	Ordenador 3
Manufacturador	Dell Inc.
Modelo	OptiPlex 3040
Procesador	i5-6500 CPU @ 3.20GHz
Memoria	8GB DDR3
Disco	SSD 540Mb/s
Sistema Operativo	Windows 10 64bits

Se ha decidido instalar 7zip, VLC y Ccleaner ya que son programas que se usan en la empresa en la que se realizarán estas pruebas, estos son los tiempos cronometrados de instalación en sendos equipos:

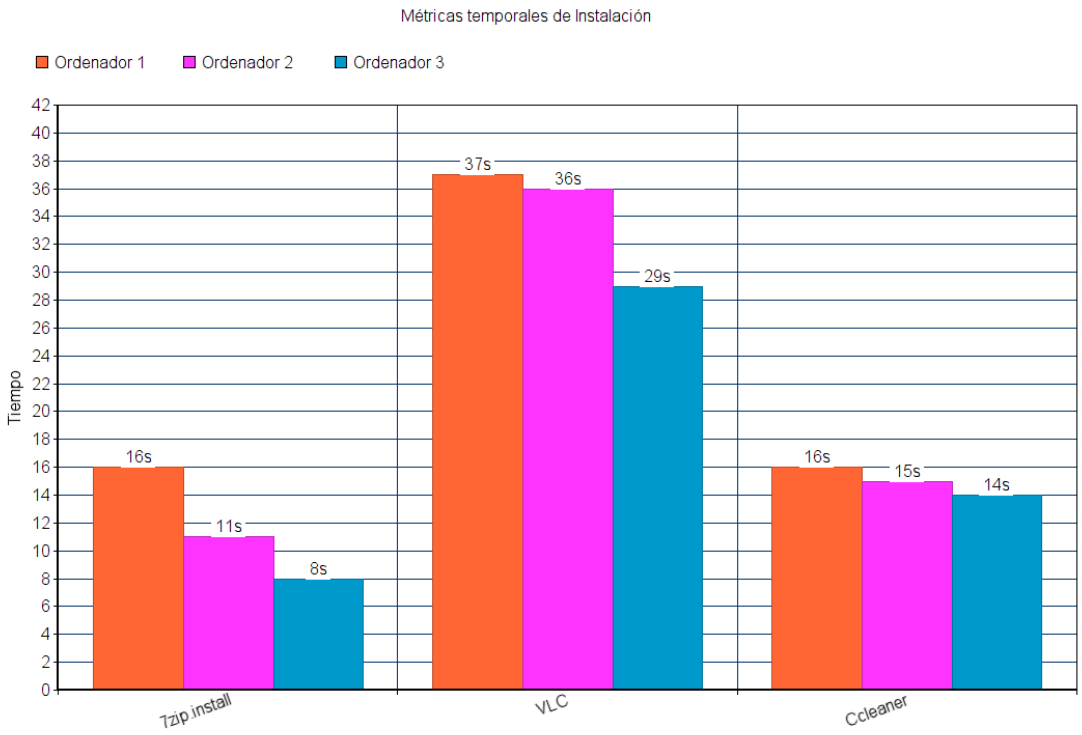

Figura 4.4: Métricas: Tiempos de despliegue

Tras analizar los tiempos podemos concluir que el tiempo de instalación es altamente satisfactorio, damos por válido el requerimiento no funcional: **RNF.02 El despliegue tiene que ser ágil**

Métrica de memoria de despliegue

Estra métrica evalúa el coste por despliegue en MB de memoria, esto nos servirá para tener una idea aproximada de cuanto consumo de memoria tendrá tener un número definido de despliegues en paralelo.

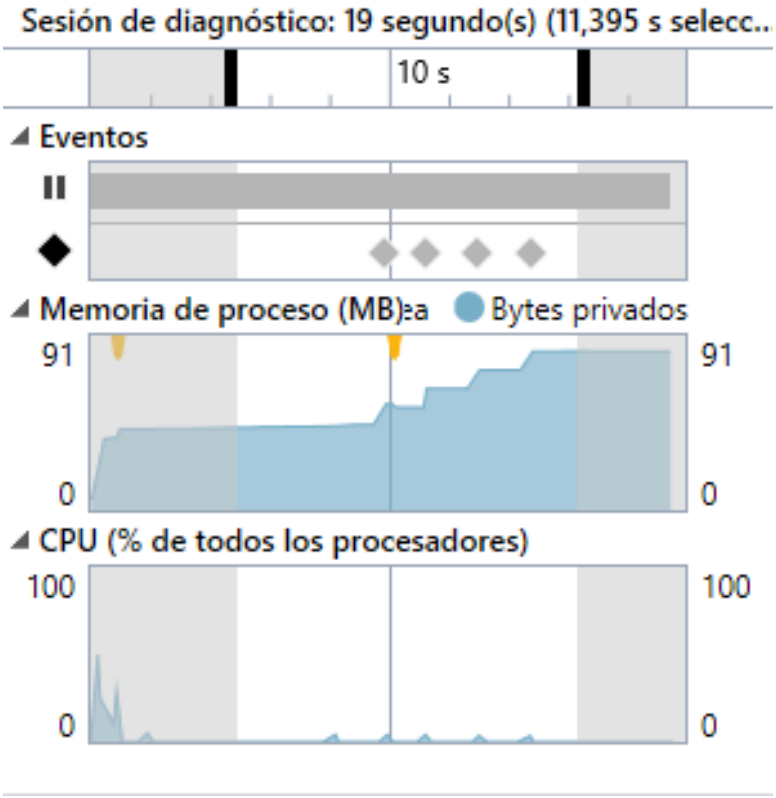

Figura 4.5: Métricas: Memoria usada por despliegues en paralelo

Para realizar esta métrica, arrancamos la aplicación y se hicieron 4 despliegues consecutivos, esto internamente generó 4 instancias del motor de PowerShell de system.management.automation.

En la gráfica podemos apreciar el coste base de la aplicación en MB (alrededor de 50MB) y cómo se han cargado los 4 despliegues consecutivamente hasta alcanzar los 91MB de memoria ocupada.

$$91MBtotales = 50MBbase + 4 * (X)MBdespliegue$$

$$41MB/4 = (X)MBdespliegue$$

$$\approx 10MBdespliegue$$

Tras analizar los resultados obtenidos, podemos observar que el coste de cada despliegue por separado ronda los 10MB, suponiendo que el número de conexiones paralelas que usa Microsoft con las PSSsessions es 32, esto ocuparía $50MB + 32 * 10MB = 370MB$ de consumo de memoria con la paralelización máxima por defecto que ofrece microsoft. Este coste nos parece aceptable.

Métrica de memoria de consumo de ancho de banda

Esta métrica no ha podido realizarse correctamente en este proyecto ya que el entorno de pruebas que se ha usado para realizarse es un entorno empresarial complejo, con un firewall por dónde pasa el tráfico de cientos de usuarios.

El consumo de ancho de banda dentro de la subred se podría considerar despreciable para el nivel de paralelismo que queremos que soporte nuestra aplicación, pero el consumo con el WAN no es despreciable, el problema de realizar esta métrica, es que con las herramientas de análisis de tráfico disponibles para nuestro firewall, no nos permite hacer un estudio correcto de el consumo de nuestra aplicación, debido a la cantidad de ruido que generan el resto de cientos de usuarios.

Podría llegar a realizarse una medida, pero excede con creces las capacidades administrativas para poder preparar el entorno y las capacidades técnicas para poder evaluarlo, así que el requisito no funcional: **RNF.01 Evitar sobrecargar la red**, no se podrá evaluar de forma objetiva, consideramos este requisito como válido ya que no hubo problemas de saturación de la red mientras se llevaba a cabo el desarrollo y pruebas de este proyecto a falta de métricas más formales.

Métrica temporal y de memoria de arranque

Para esta métrica vamos a usar la herramienta de diagnóstico incorporada en Visual Studio que muestra una gráfica de la memoria utilizada a lo largo del tiempo.

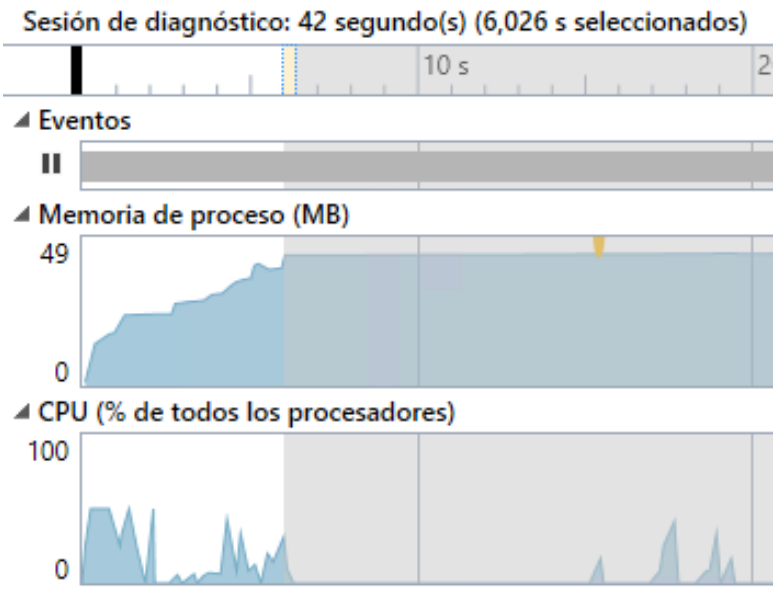

Figura 4.6: Métricas: Tiempo y memoria de arranque

En la gráfica se puede apreciar que el tiempo de carga de la aplicación ronda los 6 segundos y 50MB en memoria, por lo que tampoco hay una necesidad de mejora en esta parte.

4.4 Conclusiones sobre el prototipo

Tras analizar los resultados obtenidos con el prototipo podemos concluir que es una aplicación válida, a nivel técnico es mucho más rápida de lo esperado y

tiene un consumo adecuado de memoria (o al menos es adaptable al entorno), a nivel de interfaz necesita muchas modificaciones, pero esto último es de esperar de un prototipo hecho para probar la funcionalidad de la especificación.

CAPÍTULO 5
Mejoras aplicables al prototipo

5.1 Errores encontrados en la aplicación

5.1.1. Solapación de Actualizaciónes - Instalaciónes

Descripción del error.

Se ha detectado un error de diseño en el que se puede realizar un actualizaciónes e instalaciónes simultáneamente en el mismo equipo, esto puede suponer un problema de malfuncionamiento de las instalaciones.

Reproducción del error.

- Ir a la pestaña de " Despliegue " .
- Seleccionar una aplicación y darle al botón " Despliegue " para instalar una aplicación en un equipo.
- Inmediatamente después volver a darle al botón " Despliegue " con la misma aplicación

¿Por qué ocurre este error?

El sistema que se utiliza para realizar despliegues es una cola que no se atiene a ninguna lógica, se aceptan todos los intentos de despliegues

Posible solución del error.

Crear una clase " Despliegue " que encapsule la lógica de los despliegues, crear una lista en la cual almacenar estos y programar la lógica para que no se realicen despliegues cuando no son deseados, o encapsular un sistema de mensajes de éxito o error.

5.1.2. Apagado del equipo durante la instalación

Descripción del error.

Nuestra aplicación no ofrece ningún sistema de retorno de información o manera de proteger una instalación contra un apagado inesperado del equipo en el que se realiza el despliegue.

Reproducción del error.

Realizar un despliegue de una aplicación y retirar la corriente eléctrica al ordenador objetivo del despliegue antes de que esta acabe.

Posible solución del error.

Existe una herramienta de despliegues de paquetes de Chocolatey llamada Boxstarter, esta herramienta presume en su web de ser " *reboot resilient* " , habría que investigar cómo implementar esta herramienta y usarla con nuestra aplicación.

Web de Boxstarter: http://boxstarter.org/.

5.2 Mejoras

En esta sección se pretenden exponer diferentes mejoras funcionales para nuestra aplicación, la aplicacion de las cuales excede el dominio de este proyecto, pero no su mención o explicación.

5.2.1. Scripts

El sistema de ejecución de comandos de PowerShell está administrado por unas clases que encapsulan estos mismos en su contenido. Una de las posibles mejoras de esta aplicación es poder modificar esos scripts para que se adapten a las necesidades particulares de cada entorno en el que se use, esta mejora permitiría un mayor grado de modularidad, permitiendo ampliar las funcionalidades de la aplicación. Ejemplos de estas mejoras modulares que podría implementar cada usuario con cómo podrían llegar a ser:

- Hacer que las peticiones de Chocolatey no se realicen al repositorio en la nube de Chocolatey si no en un servidor SQL propio configurado como repositorio de paquetes de Chocolatey.
- Almacenar logs personalizados de los resultados de los despliegues en cualquier ruta disponible.
- Limitar o gestionar el uso de ancho de banda.

- El uso de credenciales distintas a las usadas por el usuario que ejecute nuestra aplicación.
- Envío de emails automáticos a las personas cuyos equipos hayan recibido una actualización de software o instalación.
- Otros muchos más usos, esta mejora es increíblemente ampliable.

5.2.2. Servidor SQL de paquetes

Chocolatey tiene la capacidad de trabajar con diferentes fuentes de paquetes, también viene con un repositorio de paquetes por defecto configurado: el repositorio de paquetes de la comunidad que se utiliza en esta apliación. La desventaja que se encuentra es al ser un repositorio de dominio público, este no puede ser confiable al cién por cién. Esto no es un sistema que cumpla los estándares de trabajo empresariales, así que para uso empresarial de Chocolatey, se recomienda crear un repositorio interno de paquetes.

5.2.3. APP a través de portal web

Esta mejora trata de correr la aplicación en un servidor y que esta opere su interfaz a través de un servidor web alojado desde la misma aplicación, de esta forma se podrían realizar instalaciones desde diferentes terminales a la terminal en la que está funcionando la aplicación, por ejemplo, desde cualquier ordenador de la subred del servidor, incluso un ordenador externo a la subred conectado por una red privada virtual (*VPN*) a la misma.

5.2.4. Más información de los ordenadores

Esta mejora consistiría en realizar un inventario de los equipos más exhaustivo, consiguiendo elementos como:

- Modelo de procesador.
- Última dirección IP.
- Capacidad usada y disponible del disco duro.
- Últimos usuarios conectados.
- Aplicaciones instaladas fuera de Chocolatey.
- Otros muchos más usos, esta mejora es muy ampliable.

Realizar esta mejora supone muy poco coste a nivel funcional, ya que solamente supondría unas ligeras modificaciones a las consultas de objetos *WMI* que ya se realizan cuando se conecta con un ordenador, pero supondría más trabajo a nivel de interfaz ya que no hay desarrollado una parte de inventario de ordenadores, o gestión del mismo.

5.2.5. No solamente despliegues Chocolatey

Esta mejora realmente **podría convertirse en un proyecto distinto** que reutilice gran parte del código de este proyecto. Con este proyecto realmente se ha implementado un sistema de despliegue de Scripts de PowerShell a través de WinRM y Active Directory, estos no necesariamente tendrían que ser instalaciones de chocolatey, si no parches, configuraciones u otro tipo de instalaciones (Por ejemplo, de certificados).

Aprovechando esta interfaz gráfica que visibiliza la infraestructura de ordenadores, credenciales y scripts, se podría implementar un sistema de despliegue de cualquier tipo de script a través de la red. Y dado que PowerShell es una herramienta excepcionalmente modular y potente, se podría incluso implementar una versión que mejore GPO o DSC[1] o les de una interfaz gráfica y una serie de herramientas para autoconfigurarse.

La alcanzabilidad de este nuevo proyecto sería a simple vista inmesurable dentro de un dominio de Active Directory con equipos Windows.

[1] Desired State Configuration, es una herramienta de PowerShell para configurar declarativamente los servidores de una dominio

CAPÍTULO 6
Conclusiones

Para finalizar este proyecto, se han alcanzado todos los objetivos esperados, solventando el problema y dando una nueva perspectiva a la forma de distribuir software por una red empresarial.

La solución alcanzada es satisfactoria, pero sigue sin poder ser una aplicación comercial completa que pueda satisfacer estándares empresariales, por ejemplo, por comparar con una empresa que ofrece la misma gama de servicios[1], esta empresa tiene actualmente treinta-y-una personas a su cargo plenamente dedicadas a esta aplicación, personas con años de experiencia y escogidas para llevar adelante la comercialización de estos productos, no se puede competir realmente ofreciendo toda la variedad de funcionalidades y servicios que estos ofrecen, así que quizá una mejor solución empresarial sería comprar su software que desarrollarlo.

Eso sí, educacionalmente ha sido de gran valor y ha cambiado la forma de organizar las actualizaciones de software en la empresa en la que se ha desarrollado este proyecto.

6.1 Aportes de la carrera

En esta sección se comentan aportes directos de la carrera que pueden ser claramente identificables dentro de este proyecto o memoria.

6.1.1. Metodologías: Gestión de proyectos

Se pueden apreciar los conocimientos prácticos obtenidos en esta rama de asignaturas en la propia estructura que se ha seguido para realizar este proyecto, entre ellas se puede diferenciar claramente cinco etapas muy similares a las etapas de desalloro software[2] estudiadas en la carrera, pero orientadas al proyecto en sí, no únicamente al propio software.

- Etapas de análisis de viabilidad:

[1] PDQ en https://www.pdq.com/about-us/

[2] https://es.wikipedia.org/wiki/Desarrollo_por_etapas

Esta es la etapa en la que se investigaron productos similares y se encontraron tecnologías viables para poder implementar la solución.

- Planificación:

  Aquí es dónde se especificó qué se pretendía hacer con el proyecto, su alcance, requerimientos y la preparación del entorno y diagramas para definir qué se iba a desarrollar.

- Ejecución:

  Esta etapa no está directamente reflejada en este documento, es en la que se desarrolló el proyecto.

- Seguimiento y control:

  Esta etapa se ve reflejada en los capítulos de Desarrollo del prototipo y Mejoras aplicables al prototipo, en los cuales se analizan los resultados obtenidos en el proyecto y se realizan labores de mantenimiento como búsqueda de errores, mejoras aplicables o defectos de funcionalidad o interfaz. En esta etapa también se controla si se han alcanzado los requerimientos expresados en la etapa de Planificación.

- Cierre:

  Esta es la etapa en la que se elabora esta memoria y se documentan los procedimientos que se han seguido para la realización del proyecto, esta etapa no quiere decir un cierre propio del software, si no un cierre a esta etapa de presentar una memoria de trabajo de fin de grado.

6.1.2. Metodologías: Ingeniería del Software

En la rama de ingeniería del software se aprendieron diferentes técnicas que han sido aplicadas en el proyecto, como las etapas del desarrollos software que se pueden apreciar en la subsección anterior, el desarrollo de software dividido en capas dividiendo la arquitectura por funcionalidades, o diferentes patrones de diseño que sirvieron para tener un código de calidad, como lazy initialization, object pool o el patrón estado. El establecimiento de requerimientos funcionales y no funcionales visto en esta memoria y la manera de escribir código limpio, siguiendo un patrón constante para escribir clases, métodos, variables públicas o privadas, o la estructura dentro de cada función.

6.2 Valoración personal

Considero que este proyecto ha sido muy enriquecedor a nivel personal, ya que no ha sido un proyecto hecho para cumplir unas métricas preestablecidas de una asignatura, ha sido un proyecto puramente personal que tras ver que era viable y podría convertirse en un proyecto de fin de carrera, me ha dado la motivación para aplicar los conocimientos adquiridos a lo largo de la carrera y dedicarme durante todo el curso lectivo a ir poco a poco definiendo lo que ahora podría decir que es el primer proyecto serio que he realizado.

La mayoría de aportes de la carrera se pueden apreciar de forma totalmente indirecta, ya que esta no ha aportado todos los conocimientos técnicos necesarios para este proyecto si no más bien, **una forma concreta de pensar como ingeniero informático**, aprendiendo a descomponer problemas complejos en problemas más sencillos de resolver, formándome de forma autodidacta con la información técnica necesaria y enseñándome a sintetizar conceptos mentales en piezas de información con las que poder trabajar en un ordenador, como por ejemplo, pensar en programación orientada a objetos o en la etapa de análisis de este proyecto y a saber que iba a necesitar una tecnología de gestión y otra de instalación, ya que estos eran dos problemas distintos.

6.3 Agradecimientos

Quisiera dejar un breve agradecimiento a todos los profesores de la carrera que me han ayudado a aprender y formarme como ingeniero informático, y a Laboratorios Sesderma, por formarme como administrador de sistemas y permitirme desarrollar este proyecto con ellos usando toda su red empresarial a mi disposición.

Bibliografía

[1] WinRM, WS-Management, PowerShell, Etapas desarrollo software Consultado en https://en.wikipedia.org/.

[2] Instalaciones con Chocolatey. Consultado en https://chocolatey.org/.

[3] Instalaciones con Ninite. Consultado en https://ninite.com/.

[4] MSDN, documentación sobre la programación con PowerShell y system.management.automation. Consultado en https://msdn.microsoft.com/en-us/library/system.management.automation(v=vs.85).aspx.

[5] Documentación sobre C#. Consultado en https://stackoverflow.com/.

[6] Cursos sobre visual studio en la Microsoft Virtual Academy. Consultado en https://mva.microsoft.com/product-training/visual-studio-courses.

[7] PDQDeploy y PDQInventory. Consultado en https://www.pdq.com/.

[8] Manage Engine. Consultado en https://www.manageengine.com/es/.

[9] Diagramas y Bocetos. Creados en https://www.draw.io/.

APÉNDICE A
Preparación del entorno

Este apéndice consta de la preparación de un entorno para implementar esta herramienta, a continuación se desarrollaran los puntos necesarios para que funcione correctamente nuestra aplicación en un entorno y se da un ejemplo de despliegue para poder reproducir paso por paso para prepara.

A.1 Requisitos previos:

- Tener un entorno de Active Directory con cuentas de administrador capaces de realizar cambios en los equipos a administrar, es decir, configurados con permisos de administración local.
- Tener todos los equipos a administrar conectados en la misma subred, que no necesariamente en la misma VLAN.
- Administración DNS de los nombres de los equipos para poder encontrarlos por la red, esto suele estar habilitado a la par con Active Directory.

A.2 Active Directory GPO: Activar el servicio WinRM en dominio

Para que nuestro sistema pueda funcionar en red, hay que habilitar WinRM en todos los ordenadores que se quieran administrar y este viene deshabilitado por defecto en la mayoría de sistemas operativos, la forma más simple de realizar esto es via GPO en dominio.

Podemos distinguir dos tipos de despliegues, el primero, un despliegue ordenador a ordenador [1], o un despliegue más complejo que configuraría automáticamente todos los ordenadores de la red de Active Directory para habilitar WinRM vía GPO.

[1] Esto se realizaría en cada ordenador, abriendo una terminal con privilegios de administrador de PowerShell 2.0 o superior, y escribiendo " *Enable-PSRemoting -force* " y luego escribiendo " y " como respuesta para aceptar que vamos a realizar este cambio.

Con este pequeño tutorial se configura el dominio de formal que WinRM esté siempre habilitado en todos los equipos a los que se le aplique esta política, esta GPO tiene varios puntos donde se puede adaptar a diferentes necesidades.

1. Para empezar este proceso se recomienda crear una nueva GPO.
2. Ir a: Computer - Policies - Windows Components - Windows Remote Management (WinRM) - WinRM Service.
3. Habilitar *Allow remote server management through WinRM*.
4. Especificar " * ", si es posible asignar un rango concreto de ip, asignar en lugar del " * " .
5. A continuación hay que habilitar " Windows Remote Management (WS-Management)" en " Group Policy Preferences Services".
6. Y para finalizar hay que configurar el Firewall de Windows para que acepte WinRM, esto se realiza añadiendo una regla de entrada para habilitar " Windows Remote Management ".

Este tutorial está mejor explicado con imágenes en la página web:

http://www.grouppolicy.biz/2014/05/enable-winrm-via-group-policy/

A.3 Despliegue de .NET 4.6 o Superior

Para que nuestra aplicación pueda funcionar, más concretamente el propio Chocolatey. Es necesario que los ordenadores tengan instalado .NET 2.0, pero se recomienda la última versión del mismo.

Los sistemas Windows vienen desde XP con .NET pre-instalado, para que nuestra aplicación funcione se requiere Windows 7 o superior, y este ya viene con .NET 2.0 pre-instalado.

Esto es interesante porque .NET 2.0 viene ya con herramientas de control remoto de ordenadores con las conexiones *PsSession* en las que se basa nuestra aplicación.

Una de las formas más simples de distribuir la última versión de .NET, es compartir una carpeta en red Samba, permitir el acceso a cualquier usuario y almacenar ahí un ejecutable de Ninite.com que instale automáticamente la última versión de .NET.

A partir de ahí crear un script que en cada ordenador de la red, ejecute el ejecutable almacenado en esa carpeta de red, los ordenadores que ya tengan la última versión no harán nada y los ordenadores que no la tengan se actualizarán.

APÉNDICE B
Seguridad

Este apéndice trata sobre los temas de seguridad que conciernen a nuestra aplicación, esta al ser en resumidas cuentas un gestor muy particular de Chocolatey, tiene todos los problemas de seguridad que pueda tener Chocolatey, estos temas se tratan en su web[1], pero en este trabajo se ofrece un apéndice de seguridad con los aspectos que más nos conciernen:

La seguridad en las aplicaciones empresariales es un punto importante a valorar, nuestra aplicación utiliza una herramienta que puede llegar a ser conflictiva en estos aspectos. Chocolatey no es per sé una herramienta insegura, sin embargo esta puede utilizarse como una herramienta insegura. La mayoría de empresas que utilizan Chocolatey, utilizan repositorios propios, en los cuales se evalúa individualmente qué paquete se está descargando y preparando en el repositorio, o utilizan Chocolatey pro, la versión de pago de Chocolatey que ya tiene una revisión de seguridad constante de los paquetes utilizados.

Lo primero de todo, hay que entender que:

Chocolatey != paquetes de chocolatey.org

B.1 Resumen sobre seguridad y Chocolatey

Chocolatey tiene muchas características sobre seguridad, pueden proveer de información para tomar las mejores decisiones respecto a los paquetes utilizados ya que cada uno de los paquetes utilizados en la comunidad pasa por una sucesión de más de cincuenta antivirus y la propia comunidad suele comentar si ocurre cualquier cosa fuera de lo común, sin embargo, para un análisis más profundo sobre seguridad habría que valorar las diferentes características que ofrece:

- Ofrece la oportunidad de crear nuestro propio repositorio de paquetes: esto se menciona en diferentes apartados de este proyecto, es una clara recomendación y buena práctica, pero añade varios niveles de dificultad al uso actual de la aplicación, ya que habría que configurarla, administrar la base de datos e incorporar e inspeccionar los paquetes.

[1] Web de Chocolatey: https://www.chocolatey.org/docs/security.

- Usar el repositorio de la comunidad, es tan seguro como de seguros sean los paquetes que se están utilizando, estos paquetes se verifican con VirusTotal, tienen un checksum propio y las propias empresas distribuidoras de software habitualmente generan y suben sus propios paquetes de software, pero esto no es una garantía absoluta de seguridad, por lo que para ser extremadamente cautos, habría que ceñirse a otros apartados mencionados en este apéndice.
- Chocolatey ofrece Chocolatey Pro o Chocolatey For Business, que son versiones de pago que instalan una protección contra malware en tiempo real.

B.2 Características de seguridad de Chocolatey

1. Chocolatey reduce los problemas de escalación de privilegios **requiriendo permisos de administrador para ejecutarse**, para añadir el agente a la variable de entorno PATH y para instalar programas en " Archivos de programa " .
2. **Soporta el parámetro de PowerShell -WhatIf**, que nos permite ver sin que realice los cambios, los propios cambios que realizaría en caso de ejecutar el mismo comando sin ese parámetro. (Un ¿qué pasaría si hago esto?)
3. Se pueden realizar checksums en tiempo de ejecución.
4. **Soporta SSL/TLS** en caso de que la aplicación a instalar lo permita.
5. Sólamente se permiten publicar paquetes a través de SSL, para evitar problemas DNS con la API key del repositorio, es decir, suplantación de identidad con la API key.
6. **Se pueden crear paquetes propios**, de tal manera que estos no tengan malware ninguno, Chocolatey ofrece herramientas de pago para crear los mismos.

B.3 Sobre la seguridad de los paquetes de chocolatey.org

Chocolatey tiene un repositorio público de paquetes alimentado por la comunidad, en las cuales particulares y empresas suben paquetes NuGet de instalación de apliacions software, pese a no ser cien por cien seguros, todos los paquetes del repositorio cumplen las siguentes características:

1. Desde 2014, todos los paquetes de chocolatey.org pasan por un proceso de moderación en el que: pasan por varios procesos automatizados de validación y verificación, por un análisis con VirusTotal, un proceso manual de revisión en caso de cualquier conflicto con los análisis anteriores y checksums para las instalaciones. En un futuro se prevee una firma criptográfica para los paquetes.

2. No solamente el paquete, si no el software contenido en el paquete pasa por una segunda revisión de VirusTotal.

3. Algunos usuarios de chocolatey.org, son los propios desarrolladores de software, estos obtienen una marca de *trusted* en la web, por lo que supone un pequeño añadido de confianza.

4. Los paquetes que realizan descargas, se comprueba que la descarga procede del proveedor oficial del software contenido en el paquete, no de una fuente de terceros.

5. Los propios usuarios pueden dejar comentarios en la web de chocolatey.org para avisar de cualquier problema encontrado con los paquetes.

6. Todas las conexiones se realizan mediante SSL/TLS para aumentar la seguridad en la red.

7. Todos los paquetes contienen un checksum que se contrasta contra el checksum almacenado en el repositorio.

8. Si el paquete realiza descagas, también se realizan los checksums de las mismas descargas.

B.4 Conclusiones

Chocolatey se puede considerar una herramienta segura, y realizando ciertas labores que suponen un pequeño trabajo extra de verificación manual, se puede considerar " Seguro " para estándares empresariales de seguridad.

Se podría ampliar este proyecto para trabajar con la versión de pago de Chocolatey, esto mejoraría varios aspectos de seguridad, pero requeriría varios esfuerzos extra por parte del usuario y en cierta manera no es una solución adecuada.