
UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

INGENIEROS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE MÁSTER

MÁSTER EN INGENIERÍA DE TELECOMUNICACIONES

# Instalación desatendida de Windows 10/11 en la red del grupo de investigación LPI

Autor:

**D. Jesús Alonso García**

Tutor:

**Dr. D. Federico Simmross Wattenberg**

Valladolid, Junio 2025

| | |
|---|---|
| TÍTULO: | **Instalación desatendida de Windows 10/11 en la red del grupo de investigación LPI** |
| AUTOR: | **D. Jesús Alonso García** |
| TUTOR: | **Dr. D. Federico Simmross Wattenberg** |
| DEPARTAMENTO: | **Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática** |

**Tribunal**

| | |
|---|---|
| PRESIDENTE: | **Dr. D. Juan Ignacio Asensio Pérez** |
| VOCAL: | **Dr. D. Manuel Rodríguez Cayetano** |
| SECRETARIO: | **Dr. D. Antonio Tristán Vega** |
| P. SUPLENTE: | **Dr. D. Pablo Casaseca de la Higuera** |
| V. SUPLENTE: | **Dr. D. Ignacio de Miguel Jiménez** |
| S. SUPLENTE: | **Dr. D. Miguel Ángel Martín Fernández** |

| | |
|---|---|
| FECHA: | **Junio 2025** |

## Agradecimientos

A mi famila y amigos, por apoyarme durante la elaboración de este proyecto y ayudarme a superar todas las dificultades que han surgido. Sobre todo a mis padres y a mi pareja.

A mi tutor Federico Simmross Wattenberg, por la orientación proporcionada para que el proyecto sea un éxito y por la ayuda proporcionada.

## Resumen

Este Trabajo de Fin de Máster se centra en el diseño e implementación de un entorno totalmente automatizado para la instalación del sistema operativo Windows 10/11. Este aplicativo está orientado a ser desplegado en uno de los laboratorios de investigación de la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid.

Para su desarrollo se ha empleado un servidor PXE, compuesto por otros dos servidores, DHCP y TFTP, para proporcionar los ficheros de prearranque, para posteriormente cargar un entorno de preinstalación. También se ha empleado un servidor HTTP Apache2 para ofrecer al usuario los ficheros esenciales de la imagen Windows PE. Finalmente, se ha empleado un servidor Samba, para gestionar la compartición de los ficheros de la imagen ISO de Windows, junto al fichero de respuestas que automatiza el proceso de instalación, al mismo tiempo que se logra gestionar los accesos a las nuevas máquinas que se instalen el nuevo sistema operativo.

Este aplicativo reduce significativamente el tiempo de despliegue del software, reduce los errores humanos, centraliza la gestión de las máquinas de la red y da una solución a medida que cubre una necesidad clave del grupo de investigación en el que se desarrolla este proyecto.

## Palabras Clave

Windows, instalación desatendida, Windows PE, PXE, DHCP, TFTP, Samba, Directorio Activo

## Abstract

This Master's Thesis focuses on the design and implementation of an environment for the unattended installation of the Windows 10/11 operating system. The system is intended to be deployed in one of the research laboratories of the School of Telecommunications Engineering at the University of Valladolid.

For its development, a PXE server was used, composed of two additional servers, DHCP and TFTP, to provide the pre-boot files required to subsequently load a preinstallation environment. An Apache2 HTTP server was also employed to deliver the essential Windows PE image files to the client. Additionally, a Samba server was implemented to manage the sharing of the Windows ISO image files, along with the answer file that automates the installation process, while also allowing centralized access management for newly installed machines.

This solution significantly reduces software deployment time, minimizes human error, centralizes the management of networked machines and provides a custom-made solution that cover a key necessity of the investigation group in witch this project is developed.

## Keywords

Windows, unattended installation, Windows PE, PXE, DHCP, TFTP, Samba, Active Directory

# Índice general

# Índice de figuras

# Índice de tablas

# Índice de Scripts

# Capítulo 1

# Introducción

Este primer capítulo pretende servir al lector como una introducción al proyecto que se ha desarrollado, así como sus objetivos, métodos y resultados obtenidos.

El presente Trabajo de Fin de Máster se ha llevado a cabo bajo la tutorización del Dr. D. Federico Simmross Wattenberg, perteneciente al Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática de la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid.

Este proyecto comenzó en febrero de 2025 y se ha llevado a cabo en el laboratorio 25 de la escuela, con materiales tales como un PC, cables de red, monitores, teclados y ratones. Además, se ha desarrollado gracias al continuo contacto con el tutor a través de correo electrónico.

## 1.1. Contexto y motivación

La creciente cantidad de máquinas de trabajo que se utilizan en entornos de investigación, desarrollo y docencia y que necesitan estar conectadas a una misma red, crea la necesidad de estar mantenidas y actualizadas de una manera más centralizada. Este enfoque es esencial no solo para mantener un entorno homogéneo y funcional, sino también para garantizar que todos los dispositivos estén actualizados y alineados con las últimas políticas de seguridad, compatibilidad y rendimiento.

Uno de los retos más comunes en este tipo de entornos es la migración o actualización masiva de sistemas operativos (SO), especialmente cuando se manejan múltiples dispositivos de red que deben funcionar de forma sincronizada y bajo los mismos estándares de configuración.

El avance de de las tecnologías de instalación ha permitido reemplazar los métodos tradicionales, que requerían intervención manual equipo por equipo, por soluciones mucho más eficientes. Entre estas, se encuentran las instalaciones a partir de imágenes de sistema previamente configuradas y almacenadas localmente o en un medio de instalación USB/DVD. No obstante, en busca un nivel de automatización y escalabilidad superior al llevar este tipo de instalaciones, las realizadas en red destacan notablemente.

Tradicionalmente, la instalación y configuración de los sistemas operativos en los equipos de estos laboratorios se ha realizado de forma completamente manual. Este método presentaba múltiples inconvenientes: requería por parte del administrador un elevado consumo de tiempo, un consumo alto de recursos humanos y un elevado riesgo de cometer errores o inconsistencias entre diferentes estaciones de trabajo. Este enfoque no solo afecta negativamente a la eficiencia del personal técnico, sino que también puede traducirse en tiempos de inactividad prolongados y una experiencia desigual entre los usuarios.

En este proyecto, el método empleado será la instalación en red, mediante el uso del protocolo PXE acompañado de otros tales como DHCP o HTTP. No solamente la instalación en red es importante, sino que también lo es su automatización. Este concepto se denomina instalación desatendida. La instalación desatendida de sistemas operativos es un método de instalación que permite la implementación automatizada de software en múltiples dispositivos sin intervención manual.

Para el caso que se presenta, surge una necesidad de automatizar en todos los dispositivos del laboratorio 25 de la Escuela Superior de Ingenieros de Telecomunicación la instalación de Windows 10/11 Education actualizado. Además, debe contar con todas las herramientas necesarias para desarrollar el trabajo de manera adecuada y respetar las políticas de usuario para que usuarios no autorizados no puedan acceder al sistema, así como configurar la posible presencia del sistema de ficheros compartidos o distribuidos.

Las motivaciones principales por las que se ha escogido este método de instalación son las grandes ventajas que presenta. Estas son:

- **Eficiencia operativa y escalabilidad:** al llevar a cabo una instalación en red de manera desatendida, permite que se puedan llevar a cabo de manera paralela en distintos dispositivos a la vez, minimizando al mismo tiempo los errores de configuración y el tiempo de instalación cuando se trata de instalaciones en varios dispositivos.
- **Flexibilidad y uniformidad:** permite configurar de una forma personalizada a gusto de las necesidades de la organización en la que se encuentra desplegado de manera uniforme. De igual manera, si se desea hacer un cambio, simplemente ha de hacerse solamente una vez y, posteriormente, replicarlo.

- **Mejora en la seguridad y gestión centralizada:** gracias a la centralización del software, se garantiza que todas las máquinas cumplan con las configuraciones y políticas de seguridad establecidas por la organización de la que forman parte. La compatibilidad con Arranque Seguro (*Secure Boot*) y autenticación en red permite un control más estricto sobre los dispositivos autorizados a instalar el sistema operativo.

## 1.2. Objetivos

El objetivo principal de este Trabajo Fin de Máster es el despliegue y configuración de un servicio que permita llevar a cabo la instalación de Windows 10/11 de manera desatendida en las máquinas de trabajo del laboratorio 25 de investigación de la Escuela de Ingenieros de Telecomunicación de la Universidad de Valladolid. A lo largo de la memoria se encuentra la descripción completa de cada uno de los pasos seguidos para conseguir el objetivo planteado.

La solución tecnológica planteada debe permitir al usuario que, tras arrancar por red, pueda elegir entre una variedad amplia de sistemas operativos, el que más le convenga. El menú debe permitir navegar al usuario dentro de las opciones que se le presenten, y se debe habilitar la opción extra que permita instalar Windows 10/11.

El proceso de instalación desatendida del sistema operativo Windows 10 y el del 11 son muy similares y su comportamiento es casi idéntico. Es por ello por lo que, para una mayor claridad a la hora de llevar a cabo las explicaciones oportunas y con vista a futuras actualizaciones del software, me centraré en explicar más en profundidad la instalación de Windows 11.

La solución tecnológica debe ofrecer al usuario una configuración del sistema operativo estándar para todo el equipo de trabajo y la posibilidad de cancelar la instalación en cualquier momento. Además, la configuración del sistema operativo debe seguir las reglas de seguridad de la organización a la que pertenece, teniendo en cuenta las normativas y políticas de usuario y posibles sistemas de almacenamiento compartidos o distribuidos.

La solución tecnológica debe ser totalmente desatendida, es decir, debe completarse de manera exitosa sin interrupciones y sin que el usuario interactúe directamente con el instalador del sistema operativo.

## 1.3. Fases y métodos

Para el desarrollo del presente proyecto se han seguido los siguientes pasos:

1. **Adquisición de conocimientos sobre el arranque por red:** el primer paso de todos es la documentación sobre todo lo necesario para que el proceso de arranque en red funcione de manera correcta. Teniendo el conocimiento previo de que este proceso se lleva a cabo mediante el uso del protocolo PXE, se procede a la lectura de toda la documentación sobre el funcionamiento y configuración de este protocolo para su despliegue en la parte servidora.
2. **Adquisición de conocimientos sobre la automatización de Windows:** partiendo de un conocimiento nulo sobre el tema, se expanden los conocimientos sobre el sistema operativo y profundiza en su proceso de instalación. A partir de esto, se selecciona la herramienta Windows ADK (*Assessment and Deployment Kit*) [1], que junto a extensiones supletorias permite crear el fichero de respuestas a partir de una imagen ISO de referencia.
3. **Adquisición de conocimientos de gestión de dominios:** el aplicativo no debe ser accesible para todo el mundo, por lo que se debe implantar un método de restricción de usuarios. Esto puede llevarse a cabo a partir de la creación, configuración y gestión de un dominio dedicado para la red del laboratorio.
4. **Recopilación de requisitos de los usuarios:** previo a la configuración del servidor y con el conocimiento ya adquirido, se procede a la toma de requisitos y necesidades de los usuarios. Esto se traduce principalmente en la configuración del idioma, cantidad y nombres de usuarios e instalación de aplicaciones de trabajo.
5. **Configuración del servidor:** el punto de partida de este proyecto es un servidor PXE ya preexistente, desplegado en el laboratorio de investigación y con ofertas de instalación de solamente versiones Linux. Sin embargo, aunque no se parte completamente de cero, es necesario un estudio completo sobre los servicios ofertados y se deben reconfigurar de nuevo todos los servicios, para así adaptarlos a las necesidades operativas de este proyecto. Estas modificaciones parten desde la reconfiguración de las interfaces de red y el despliegue de los servicios en estas interfaces, hasta la implementación de nuevos servicios.
6. **Creación del fichero de respuestas:** para que Windows 11 se instale de manera desatendida, es decir, sin intervención directa de ningún administrador de la red, se debe configurar un fichero de respuestas. Este fichero se envía a la máquina cliente para que siga todas las instrucciones ahí indicadas durante el proceso de instalación. Los requisitos anteriormente recogidos se encuentran en este fichero.
7. **Despliegue en entorno controlado:** tras la configuración del servidor, se realiza un despliegue en un entorno controlado. Las primeras pruebas se llevarán a cabo en un etorno de virtualización, empleando la herramienta VMWare. Posteriormente se desplegará en un entorno físico controlado. Dicho entorno consta del PC del desarrollador de este proyecto, con una máquina virtual servidora, junto a un PC perteneciente al grupo de investigación en donde se desplegará la solución tecnológica.

8. **Despliegue de la solución:** finalmente, cuando se comprueba el correcto funcionamiento de todo el sistema, se monta la configuración anterior en el servidor del laboratorio preexistente a este proyecto. Tras el despliegue, se debe comprobar el correcto abastecimiento a los usuarios y funcionamiento del sistema.

## 1.4. Requisitos del cliente

Para instalar Windows 11 mediante PXE, los dispositivos cliente que vayan a utilizar este servicio deben cumplir ciertos requisitos de *hardware* que garantizan no solo la compatibilidad con el proceso de arranque en red, sino también un rendimiento y seguridad óptimos del sistema operativo una vez instalado.

Los requisitos mínimos de *hardware* son:

- **Procesador:** Windows 11 requiere de un procesador de 64 bits con una velocidad mínima de 1 GHz y al menos dos núcleos [2]. Esto incluye tanto procesadores tradicionales como sistemas en un chip (SoC), y asegura que el SO pueda manejar de una manera eficiente multitareas.
- **Memoria RAM:** la memoria RAM necesaria para que funcione de manera fluida y sin que los procesos del sistema operativo mermen el rendimiento es de 4 GB.
- **Almacenamiento:** el dispositivo se estima que debe contar con un mínimo de 64 GB de espacio libre en su sistema de almacenamiento. Este espacio no está planteado solamente para la instalación inicial del sistema operativo, sino que también se tienen en cuenta futuras actualizaciones e instalación de aplicaciones que el usuario requiera utilizar. Cabe destacar que, en un futuro, este requisito necesite ser superior.
- **Tarjeta gráfica:** la tarjeta gráfica debe ser compatible con DirectX 12 o versiones posteriores y contar con un controlador WDDM 2.0 o superior [2].
- ***firmware*:** el dispositivo cliente requiere de un *firmware* UEFI [2] y compatible con *Secure Boot*. UEFI reemplaza al antiguo BIOS, ofreciendo una interfaz más moderna y capacidades avanzadas de gestión de *hardware*. Por otro lado, *Secure Boot* es una característica de seguridad que garantiza que el dispositivo arranque utilizando únicamente *software* de confianza del fabricante, protegiendo el sistema contra *malware* y otros programas maliciosos durante el proceso de arranque.
- **TPM (Módulo de Plataforma Segura):** debe ser mínimo la versión 2.0. Un chip TPM es un procesador criptográfico seguro diseñado para proporcionar funciones de seguridad basadas en *hardware*, como el cifrado de datos y la gestión de claves criptográficas [3].

Además, incluye varios mecanismos de seguridad físicos para que ningún tipo de *software* malicioso no pueda alterar las funciones de seguridad del TPM.

- **Conexión a Internet:** la conectividad a Internet es necesaria para descargar actualizaciones y para usar algunas características adicionales. Ese no es en sí un requisito obligatorio pero sí recomendable.
- **Tarjeta de red:** para la instalación mediante PXE, el dispositivo debe estar equipado con una tarjeta de interfaz de red (NIC) que sea compatible con PXE. Esta tarjeta permite al dispositivo conectarse a la red y comunicarse con el servidor PXE, para descargar los ficheros necesarios durante el proceso de arranque e instalación del sistema operativo. Es fundamental que la NIC soporte arrancar en red y esté configurada correctamente con el *firmware* UEFI, para permitir el arranque desde el servidor PXE.

Cumplir con estos requisitos de *hardware* es esencial para garantizar la instalación exitosa de Windows 11 en las máquinas de laboratorio mediante el uso del protocolo PXE. Estos requisitos no solo aseguran la compatibilidad del dispositivo con el proceso de arranque en red, sino que también proporcionan una base sólida para un rendimiento óptimo y eficiente del nuevo SO. Por lo tanto, antes de proceder con la instalación, se debe asegurar que todos los componentes del *hardware* cumplan con las especificaciones mencionadas, realizando los cambios de componentes en aquellos equipos que no las cumplan.

## 1.5. Estructura de la memoria

A continuación se indica la estructura de esta memoria correspondiente al Trabajo Fin de Máster. En esta memoria, se podrán encontrar seis capítulos, siendo el primero una pequeña introducción y descripción del proyecto que se ha llevado a cabo.

En el **capítulo 2** se lleva a cabo una explicación sobre todas las tecnologías empleadas en la elaboración del proyecto. Además, describen dichas tecnologías y se justifica su uso frente a otras opciones.

En el **capítulo 3** se describe el desarrollo de las configuraciones de todos los servicios de red que se prestan en la red del laboratorio, se indica el rol de cada tecnología durante todo el proceso de instalación, y las limitaciones que tienen en el aplicativo.

En el **capítulo 4** se describe el método de diseño y el contenido de los ficheros de configuración más importantes. Estos ficheros son la base de la automatización del proceso de instalación, por lo que se necesita explicar en profundidad todo su funcionamiento.

En el **capítulo 5** se lleva a cabo la presentación de los resultados al desplegar el aplicativo, de manera que se muestra el grado de cumplimiento de los objetivos del proyecto. Asimismo, se muestran los resultados esperados y los procedimientos de despliegue.

Finalmente, en el **capítulo 6** se hace un pequeño resumen de todo lo descrito en la memoria y una pequeña conclusión final sobre el proyecto realizado. También se incluyen líneas futuras para seguir con la evolución del aplicativo y un cierre, junto a las referencias bibliográficas al final de la memoria.

# Capítulo 2

# Tecnologías empleadas

En este capítulo se presentan las principales tecnologías empleadas en el desarrollo de este proyecto. Todas estas tecnologías son clave para cumplir el objetivo final planteado, mejorar la eficiencia durante el desarrollo y proporcionar escalabilidad al sistema.

## 2.1. PXE (*Preboot eXecution Environment*)

La implementación de instalaciones desatendidas en sistemas informáticos modernos requiere la automatización del proceso de arranque y del despliegue de sistemas operativos. Este proceso es posible gracias a la combinación de protocolos estandarizados: PXE, DHCP (*Dynamic Host Configuration Protocol*) y TFTP (*Trivial File Transfer Protocol*).

El protocolo PXE constituye un estándar de la industria diseñado para facilitar el arranque de equipos a través de la red independientemente del sistema operativo instalado [4]. Originalmente fue desarrollado por Intel y Systemsoft en 1999, pero su integración posterior en la especificación UEFI en 2015 consolidó su relevancia en la infraestructura informática moderna [5].

PXE formalizó el proceso de arranque en red de las máquinas, el cual históricamente dependía de protocolos como BOOTP (*Bootstrap Protocol*), DHCP y TFTP. De igual forma, PXE está basado en los estándares DHCP y TFTP. Dicho protocolo permite que una estación de trabajo cliente obtenga una dirección IP y toda su configuración de red mediante DHCP, y posteriormente descargue un fichero de arranque (*bootloader*) desde un servidor TFTP, cuyo nombre y ubicación dentro del sistema de ficheros es indicado por el protocolo DHCP, que está configurado para lanzar un sistema operativo o utilidades de instalación. De igual forma, puede estar configurado para mostrar un menú de navegación para que el cliente elija qué opción le conviene más.

PXE se define técnicamente como un entorno de ejecución de prearranque que permite la iniciación de un equipo cliente y la instalación de un sistema operativo sin la necesidad de medios de almacenamiento locales, utilizando únicamente la interfaz de red [4]. Esta funcionalidad simplifica la implantación de sistemas operativos en múltiples dispositivos, al permitir la gestión centralizada de la configuración en un servidor dedicado. En lugar de arrancar desde un almacenamiento local, el sistema operativo se inicia a partir de un fichero específico que se descarga desde el servidor PXE.

La arquitectura de PXE se fundamenta en un modelo cliente-servidor. El cliente PXE es un *firmware* integrado en la tarjeta de red (NIC) o en el *firmware* del sistema (BIOS o UEFI). Este *firmware* proporciona una pila de red mínima y un protocolo de comunicación que permite solicitar y descargar ficheros desde la red en la fase de prearranque. Por otro lado, el servidor PXE debe estar correctamente configurado para ofrecer servicios de configuración de red al cliente mediante el protocolo DHCP (o un *proxy* DHCP), previamente establecida por el administrador, y servicios de transferencia de ficheros, comúnmente a través de TFTP, aunque también se pueden emplear otros protocolos más robustos como HTTP (el protocolo HTTP está pensado para el envío de un mayor volúmen de datos, en cambio el protocolo TFTP no). Para este proyecto se empleará junto a DHCP el protocolo TFTP.

El proceso PXE se divide en tres fases principales: descubrimiento de la red, oferta del servidor y transferencia y ejecución del fichero de arranque.

### 1. Descubrimiento de la red

Durante el arranque, un cliente configurado para utilizar PXE como método de inicio, examina su configuración de arranque en la BIOS/UEFI y, si el método de arranque por red está seleccionadao como el primer método a emplear, el *firmware* del cliente inicializa su interfaz de red. Una vez inicializada, el cliente PXE inicia una comunicación DHCP especial (con modificaciones) para localizar un servidor de arranque y, al mismo tiempo, obtener los parámetros de red esenciales para su comunicación en la red local. Esta petición de difusión DHCPDISCOVER tiene la cabecera modificada con campos adicionales para identificar el tipo de arquitectura y otros parámetros específicos de PXE [6].

Las extensiones de la cabecera son fundamentales para un servidor PXE reconozca esta petición como una solicitud de arranque PXE. Dichas extensiones son:

- **Opción 60 (*Vendor Class Identifier*):** el cliente indica al servidor mediante el uso de esta opción en la cabecera del mensaje que está iniciando un arranque mediante PXE y necesita hacer uso de esos servicios. Esto lo indica añadiendo el valor literal "`PXEClient`" a dicha opción.

- **Opción 93 (*Client System Architecture*):** el cliente puede indicarle al servidor mediante el uso de esta opción en la cabecera el tipo de arquitectura física que tiene, como por ejemplo x86 o x64.
- **Opción 94 (*PXE Interface Layer*):** el cliente puede indicar al servidor información adicional sobre la interfaz de red que emplea, así como de todas las capacidades del *firmware* utilizando esta opción en la cabecera del mensaje [7].

**2. Oferta del servidor**

Tras recibir la petición DHCPDISCOVER del cliente, el servidor PXE (o *proxy* DHCP) envía al cliente no solo una oferta con los parámetros de red básicos (dirección IP, máscara de subred, servidor DNS...), sino que proporciona la información adicional solicitada. Esta información adicional es, como se ha mencionado anteriormente, el nombre del fichero de arranque y el servidor TFTP donde se aloja el fichero. Es importante destacar que el servidor de arranque puede ser el mismo que el servidor DHCP o un sistema independiente dentro de la red.

Este mensaje de respuesta DHCPOFFER que envía el servidor también incluye modificaciones en la cabecera con las siguientes opciones:

- **Opción 66 (*TFTP Server Name*):** Mediante el uso de esta opción en la cabecera del mensaje, el servidor indica al cliente PXE el nombre o la dirección IP del servidor TFTP donde se aloja el fichero de arranque.
- **Opción 67 (*Bootfile Name*):** El servidor indica al cliente PXE al emplear esta opició en la cabecera del mensaje el nombre del fichero de arranque que debe solicitar y la ubicación dentro del sistema de ficheros del servidor [6].

Cuando el cliente recibe esta información, solicita la descarga del fichero.

**3. Transferencia y ejecución del fichero de arranque**

Con toda la información obtenida, el cliente PXE inicia una sesión con el servidor TFTP indicado y solicita el fichero de arranque (definido en la opción 67de DHCP). Aunque TFTP es limitado en términos de rendimiento y seguridad (ver sección 2.3), resulta adecuado para esta primera etapa de arranque, en la cual se busca maximizar la compatibilidad con la mayor cantidad de equipos y minimizar al mismo tiempo la complejidad del sistema.

El servidor, tras recibir la petición del cliente, responde transfiriendo el fichero de arranque al cliente PXE, el cual normalmente cargará dicho fichero en la memoria RAM. El fichero

transferido mediante TFTP típicamente contiene un cargador de arranque, que inicia el proceso de instalación del sistema operativo o presenta al usuario un menú de opciones de arranque. Este proceso puede implicar la descarga adicional de ficheros de configuración y ficheros ejecutables. En el segundo caso, el menú resultante permite al usuario o al sistema automatizado la capacidad de seleccionar la modalidad de arranque o instalación que mejor se ajuste a sus necesidades.

Completada la descarga, el fichero es ejecutado desde la memoria RAM del cliente por el *firmware* PXE. Este cargador es el encargado de mostrar por pantalla un menú interactivo, para que el cliente elija el método de arranque, o iniciar la carga de un sistema operativo desde la red de manera automática, entre otras posibles opciones.

## 2.2. DHCP (*Dynamic Host Configuration Protocol*)

DHCP es un protocolo de red que permite a los servidores asignar automáticamente información de configuración IP a los clientes en una red, incluyendo dirección IP, máscara de subred, puerta de enlace y servidores DNS, entre otros muchos posibles detalles. Este protocolo simplifica y centraliza la administración de todas las redes, sobre todo cuando su tamaño es considerable, evitando la necesidad de configurar manualmente cada dispositivo [8].

Al igual que el protocolo PXE, DHCP tiene una arquitectura cliente-servidor, en donde un cliente identificado típicamente con su dirección MAC única solicita los datos de red a un servidor previamente configurado por el administrador del sistema. Esta comunicación se lleva a cabo mediante paquetes UDP a través de los puertos 67 para el servidor y el 68 para el cliente.

DHCP surge de la necesidad de solucionar los problemas derivados de la gestión de grandes entornos de red con una gran cantidad y variedad de dispositivos, sobre todo de dispositivos que no se encuentran constantemente conectados a la red. El precursor de DHCP es el protocolo BOOTP, cuya función era permitir a un dispositivo de red, como un ordenador de sobremesa, obtener todos parámetros de red de un servidor BOOTP durante el proceso de arranque. DHCP lo sustituye, mejorando el dinamismo y la variedad de opciones a la hora de llevar a cabo la asignación de parámetros [9].

Además de llevar a cabo una asignación automática de direcciones IP y asociarlas a una máquina, DHCP permite reutilizar dichas direcciones IP entre varias máquinas, siempre y cuando no se encuentren conectadas al mismo tiempo. Si ocurriese esta situación, DHCP asignaría una dirección IP libre, asegurando al mismo tiempo que cada máquina reciba una dirección IP única dentro de la red.

Esta gestión centralizada ayuda a los administradores de sistemas a configurar redes una

única vez, desde uno o varios servidores. Esto otorga una gran flexibilidad a la red, ya que facilita el movimiento de dispositivos entre redes sin necesidad de reconfigurarlos manualmente.

El proceso de asignación de una dirección IP y los demás parámetros de red consta de una secuencia de cuatro mensajes (ver figura 1, denominada "DORA":

- **DHCP Discover:** cuando un cliente arranca y no tiene configurado manualmente una dirección IP, emite un mensaje DHCPDISCOVER en modo difusión, ya que no conoce la dirección IP del servidor, en busca de un servidor DHCP o un proxy DHCP. Este mensaje puede incluir modificaciones para obtener datos adicionales, por ejemplo durante un arranque por PXE.
- **DHCP Offer:** al recibir el mensaje, el servidor DHCP encargado de la red envía un mensaje DHCPOFFER con destino la dirección MAC del cliente. Este mensaje contiene: una dirección IP que el servidor está dispuesto a ofrecer al cliente, la máscara de subred, la duración del arrendamiento (el tiempo de validez de la dirección IP) y la dirección IP del servidor DHCP que realiza la oferta. Además, pueden incluirse otros detalles.
- **DHCP Request:** una vez que el cliente recibe la oferta del servidor, este puede rechazarla o aceptarla, enviando de vuelta un mensaje DHCPDECLINE o DHCPREQUEST respectivamente. Cabe destacar que el mensaje DHCPREQUEST puede utilizarlo un cliente que ya posee una dirección IP para solicitar la renovación o extensión de su validez.
- **DHCP Acknowledge:** el servidor DHCP, al recibir el mensaje DHCPREQUEST, envía un mensaje de confirmación al cliente DHCPACK, finalizando así la comunicación entre ambos. A partir de ese momento, el cliente ya puede utilizar la dirección IP.

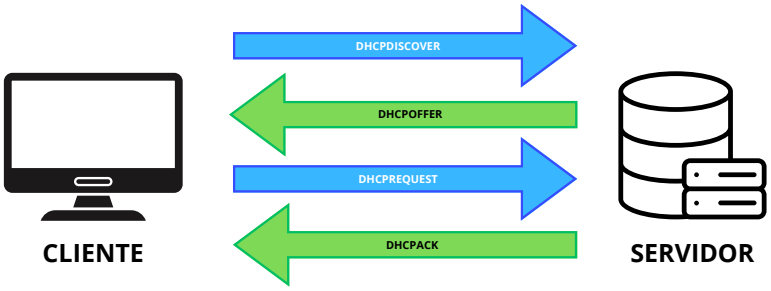


Figura 1: Funcionamiento del protocolo DHCP.

DHCP juega un papel fundamental a la hora de llevar a cabo una instalación desatendida, ya que durante la instalación de Windows 11, el sistema intentará automáticamente obtener una dirección IP y configuración de red. Esto es fundamental para cualquier proceso que requiera comunicación de red durante la instalación, como la descarga de ficheros adicionales, la unión a un dominio de Windows o a un *Active Directory* (Directorio Activo) o la aplicación de configuraciones remotas.

## 2.3. TFTP (*Trivial File Transfer Protocol*)

TFTP es un protocolo simple diseñado para la transferencia de ficheros entre un cliente y un servidor, orientado originalmente a dispositivos sin disco o con recursos limitados. A diferencia del protocolo FTP [10], del cual proviene, TFTP destaca por su pequeño tamaño y su facilidad de implementación, lo que lo hace ideal para entornos donde los recursos son limitados, como en el arranque de dispositivos sin disco por red o la transferencia de ficheros de configuración desde un servidor a dispositivos de red.

TFTP funciona en el puerto 69 y utiliza el protocolo UDP para llevar a cabo sus comunicaciones. Debido a la naturaleza del protocolo UDP de no establecer una conexión previa al intercambio de información, TFTP debe implementar sus propios mecanismos para garantizar la fiabilidad de la transferencia [11]. Esto se logra mediante un sistema de confirmación simple por bloques, en donde la recepción de cada bloque de datos transmitido debe ser confirmada por el receptor. Si un bloque no se confirma dentro de un tiempo establecido, vence su temporizador y se retransmite.

Aparte de la gran ventaja que presenta TFTP por su simplicidad, también son notables grandes limitaciones que hay que tener en cuenta a la hora de emplear este protocolo:

- No consta de ningún mecanismo de autenticación ni cifrado, por lo que los datos pueden ser interceptados, manipulados y leídos por terceros, comprometiendo la privacidad y confidencialidad de la información.
- Está pensado para manejar ficheros de pequeño tamaño (en bloques de 512 bytes), siendo las acciones que recaen sobre ellos son principalmente enviarlos o descargarlos.
- No tiene mecanismos de control de congestión por lo que en situacioens en el que el tráfico de la red es elevado, pueden producirse pérdidas, aumentando de esta manera la latencia y reduciendo el rendimiento.

El funcionamiento de una sesión TFTP es sencilla, ya que solamente existen dos tipos de peticiones que deben enviarse al puerto 69 UDP: escritura (WRQ) y lectura (RRQ). Tras recibir una petición válida, el servidor responde desde otro puerto al cliente para comenzar la comunicación con él.

Tras recibir una petición válida, el servidor responde abriendo un nuevo puerto UDP (diferente del 69) para la transferencia de datos. Este nuevo puerto se utiliza para toda la comunicación posterior entre el cliente y el servidor durante esa sesión de transferencia específica. Una vez que la conexión se establece exitosamente, la transferencia de datos comienza, empleando para ello bloques de datos de tamaño fijo (512 bytes normalmente).

Durante una transferencia mediante UDP puede haber errores. Es por eso que TFTP gestiona los errores mediante un paquete de error. Este paquete puede ser enviado tanto por el servidor como por el cliente para indicar que ha ocurrido un problema durante la transferencia, como por ejemplo “Fichero no encontrado”, “Acceso denegado” u “Operación ilegal”. Cuando un remitente envía un paquete de datos se inicia un temporizador. Si este vence antes de recibir un ACK, se retransmite. Finalmente si se retransmite varias veces de manera fallida, se envía el parquete de error.

Durante una instalación, este servicio cumple un rol fundamental para la transmisión de ficheros de tamaño reducido. Si bien TFTP en sí mismo no es directamente utilizado para transferir la totalidad de los ficheros de instalación de Windows debido a su falta de mecanismos avanzados y la gran cantidad de datos que se envían, el papel de TFTP en este proceso es la transferencia inicial de los ficheros de arranque de red necesarios para que la máquina cliente pueda iniciar el proceso de instalación. Este protocolo se emplea en combinación con PXE y DHCP.

## 2.4. Directorio Activo de Windows

*Active Directory* o Directorio Activo (AD) es una tecnología desarrollada por Microsoft con el objetivo de proporcionar un servicio de compartición de ficheros mediante un directorio centralizado. Además, permite la administración centralizada de recursos y usuarios de una red privada [12]. El uso de *Active Directory* facilita la gestión de usuarios, políticas de seguridad y servicios.

El componente principal es el servicio de dominio, el cual reside en uno o varios servidores especializados, llamado controladores de dominio (*Domain Controllers*, DCs). Este tipo de servidores almacenan una base de datos con todos los objetos existentes en la red: usuarios, grupos, máquinas de trabajo, etc. Esta base de datos se estructura siguiendo el protocolo LDAP.

La arquitectura lógica de un AD se compone de dominios, árboles y bosques. Un dominio es la unidad básica de administración y seguridad, donde todos los objetos comparten una base de datos común. Varios dominios pueden formar un árbol, y múltiples árboles jerárquicamente relacionados pueden agruparse en un bosque.

Gracias al uso de AD es posible dividir lógicamente los objetos de una misma red, permitiendo al mismo tiempo aplicar políticas de grupo en función de las necesidades específicas de la organización.

AD está formado, a su vez, por tres tecnologías: SMB, LDAP y Kerberos. Estas tecnolo-

gías son pilares fundamentales en el funcionamiento de AD y se describirán en las secciones siguientes.

### 2.4.1. Samba/SMB

Samba constituye una implementación de código abierto de los protocolos de compartición de ficheros de red SMB/CIFS *(Server Message Block/Common Internet File System*), los cuales facilitan la compartición de recursos a través de una red y son ampliamente utilizados en entornos Windows. Su función principal es intervenir en el proceso de acceso a ficheros e impresoras entre sistemas operativos UNIX/Linux y Windows, permitiendo que las máquinas Windows accedan a recursos compartidos ofrecidos por servidores Samba Linux como si fueran servidores nativos de Windows [13].

La arquitectura de Samba se caracteriza por la ejecución de varios procesos en el servidor. Principalmente, se encuentran los demonios `smbd` y `nmbd`. El demonio `smbd` es el encargado de proporcionar los servicios de compartición de ficheros e impresoras a las máquinas clientes SMB/CIFS. Escucha las peticiones de los clientes y gestiona el acceso a los recursos compartidos configurados en el servidor. Por otro lado, el demonio `nmbd` proporciona servicios de resolución de nombres NetBIOS, permitiendo que los clientes localicen los servidores Samba en la red, de manera similar a como se realiza en redes Windows.

El proceso de acceso a un recurso compartido de Samba por parte de un cliente Windows involucra la negociación del protocolo SMB/CIFS, la autenticación del usuario, que puede realizarse ante un servidor de dominio, un servidor LDAP o utilizando las propias cuentas del servidor Samba; y, por último, el establecimiento de una conexión para acceder a los recursos compartidos. Samba soporta diversos niveles de seguridad y métodos de autenticación para garantizar el acceso controlado a todos los recursos.

Más allá de la simple compartición de ficheros e impresoras, Samba ofrece funcionalidades avanzadas como la integración con dominios de Windows *Active Directory*, actuando como controlador de dominio o como miembro del dominio. Esta característica permite una gestión centralizada de usuarios y políticas de seguridad en entornos mixtos.

La configuración de Samba se realiza principalmente a través de un fichero de configuración `smb.conf`, donde se definen los nombres y ubicaciones de los recursos compartidos, los parámetros de seguridad, la autenticación y otros aspectos. La flexibilidad de su configuración permite a los administradores adaptar Samba a una amplia variedad de necesidades, desde grupos con pocos usuarios hasta redes de grandes organizaciones.

Por todas estas características, Samba es un software robusto y flexible. Su capacidad para

integrar sistemas UNIX/Linux con entornos Windows lo convierte en una herramienta fundamental para la distribución de servicios en redes particulares, la implementación de servidores de ficheros centralizados y la exploración de la interoperabilidad de sistemas operativos a nivel de red [14].

En resumen, Samba es un componente esencial para la interoperabilidad de redes que combinan sistemas operativos Windows y UNIX/Linux. Su implementación del protocolo SMB/CIFS facilita la compartición transparente de recursos, la gestión de la autenticación y la integración con dominios Windows.

Para este proyecto en específico, las funciones de Samba son fundamentales en la compartición de ficheros de gran tamaño, en especial a la hora de compartir los ficheros que conforman la imagen de Windows 11. Adicionalmente, son clave a la hora de gestionar el acceso tanto a las máquinas de Windows, como a los recursos compartidos.

### 2.4.2. LDAP (*Lightweight Directory Access Protocol*)

LDAP (*Lightweight Directory Access Protocol*) es un protocolo estándar de red diseñado para acceder y modificar servicios de directorio distribuidos sobre una red IP. Surgió como una alternativa más ligera y eficiente al protocolo X.500, ya que este protocolo era demasiado complejo y necesitaba una gran cantidad de recursos para muchas aplicaciones [15].

LDAP presenta una estructura ligera y eficiente, por lo que se ha consolidado como un pilar fundamentar en la creación de infraestructuras de redes de organizaciones. Su uso principal es la gestión centralizada de administración y autenticación de usuarios, de políticas de grupo y recursos accesibles.

El protocolo LDAP presenta una arquitectura cliente-servidor, en donde los clientes pueden realizar consultas a los servidores de directorios LDAP sobre información sobre todos los objetos de la red. Esta información es almacenada y mantenida en una base de datos jerárquica en uno o varios servidores de directorios LDAP optimizada para la lectura. Esta base de datos almacena información sobre usuarios, grupos, dispositivos de red, certificados o políticas, entre otras cosas. Toda esta información se organiza en forma de árbol denominada DIT (*Directory Information Tree*), similar a un sistema de ficheros [16].

Cada entrada al directorio representa un objeto LDAP (como un usuario o un grupo), junto a sus atributos, y se identifica de forma única mediante un *Distinguished Name* (DN). Estas entradas están compuestas por atributos de pares clave-valor que describen todas las propiedades del objeto. Algunos de los atributos comunes incluyen un *Common Name* (cn), un *User ID* (uid) o un *objectClass*. Esta estandarización permite que múltiples aplicaciones y servicios interoperen

fácilmente con un servidor LDAP. En la figura 2 puede observarase un ejemplo.

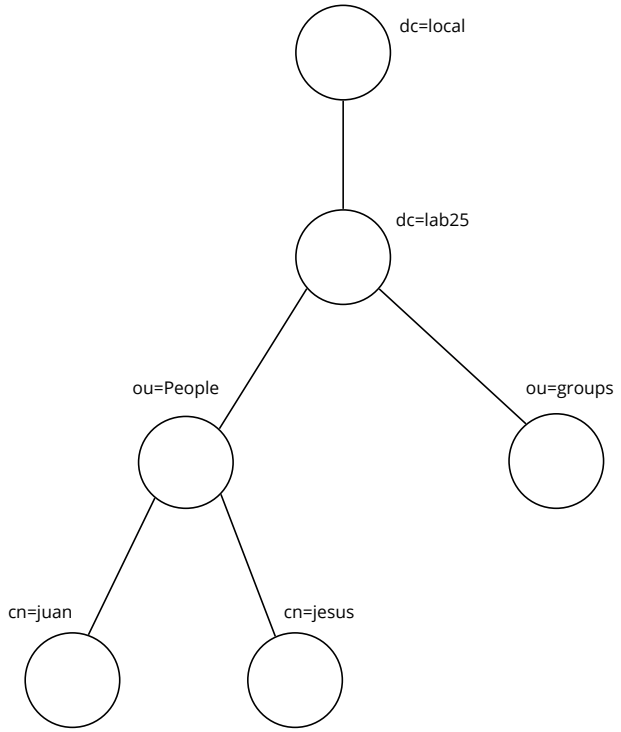


Figura 2: Ejemplo de un esquema arbol LDAP.

Una de las características más destacadas de este protocolo es su flexibilidad a la hora de realizar consultas sobre el directorio, siendo posible utilizar filtros a la hora de localizar objetos con mayor facilidad. Las operaciones más comunes y básicas sobre un servidor LDAP son: *bind* (autenticación), *search* (búsqueda), *compare* (comparación de atributos), *add*, *modify* y *delete*.

Otro de los aspectos importantes que presenta LDAP es su modelo de datos extensible, es decir, que es completamente personalizable, haciendo que las organizaciones puedan definir sus propios árboles de almacenamiento con información específica. Esto hace posible que el servidor LDAP pueda adaptarse a diversas necesidades, desde el almacenamiento de credenciales de usuario hasta la gestión de inventario hardware.

En cuanto a la seguridad que proporciona el protocolo, LDAP soporta varios mecanismos de autenticación y cifrado de conexiones. La autenticación puede realizarse de diversas maneras, desde la autenticación simple, con el uso de credenciales de usuario y contraseña, hasta la autenticación SASL [17] o el uso de certificados digitales mediante LDAP sobre TLS/SSL (LDAPS). Además, permite la integración de mecanismos de autenticación más avanzados, como Kerberos 5, cuyo funcionamiento será explicado más adelante.

Uno de las implementaciones más comunes es Microsoft Active Directory, el cual utiliza LDAP como protocolo base para la implementación de directorios destinados a la compartición

de ficheros y gestión de dominios de Windows. Esta característica va a ser fundamental en este proyecto, ya que aporta un grado de seguridad adicional al aplicativo, restringiendo el uso de los dispositivos de laboratorio a personas ajenas al equipo de laboratorio. Además, es compatible con el instalador de Windows.

### 2.4.3. Kerberos 5

Kerberos 5 es un protocolo de autenticación de red que permite que dos máquinas (por ejemplo un cliente y un servidor) se autentiquen mutuamente en una red insegura sin necesidad de trasmitir contraseñas en texto plano. Este protocolo fue desarrollado por el Massachusetts Institute of Technology y permite a los usuarios acceder a recursos y servicios en una red distribuida de forma segura [18].

El objetivo principal que tiene Kerberos 5 es proporcionar una autenticación robusta y un control de acceso basado en la criptografía simétrica. Se basa en un modelo de tercero confiable, en el que solo necesitan autenticarse una vez con el KDC (*Key Distribution Center*) para obtener tickets con los cuales acceden a los distintos servicios de la red. Este centro consta de dos componentes: el Servidor de Autenticación (AS) y el Servidor de Concesión de Tickets (TGS).

El proceso de autenticación de Kerberos 5 se divide en tres fases principales:

- Autenticación Inicial (*AS Request/Reply*): Cuando un cliente quiere hacer uso de un servicio y acceder a él, se autentica ante el KDC. Para ello, envía su nombre de usuario al AS y solicita un *Ticket Granting Ticket* (TGT). El AS verifica las credenciales y, si son válidas, envía a cliente un TGT cifrado junto con una clave de sesión para el TGT. Este TGT entregado al cliente tiene una validez limitada.
- Solicitud de Ticket de Servicio (*TGS Request/Reply*): En la segunda fase, el cliente puede solicitar tickets para servicios específicos al TGS. Para ello, el cliente envía el TGT obtenido junto a una solicitud para un servicio particular. Si el TGT es válido y el cliente está autorizado para acceder al servicio, se le concede al cliente un ticket con la clave secreta del servidor y una clave de sesión.
- Acceso al Servicio (*AP Request/Reply*): Finalmente, el cliente utiliza el ticket de servicio para acceder al servidor y hacer uso del servicio deseado. Tras enviar el ticket, el servidor lo descifra y establece un canal de comunicación autenticado y cifrado con el cliente, utilizando para ello la clave de sesión compartida.

Una de las mayores fortalezas clave de Kerberos 5 es el uso de criptografía simétrica y su capacidad para proporcionar autenticación mutua entre cliente y servidor. De esta manera

previene ataques suplantación de identidad (*phishing*), ataques de tipo MITM (*Man-In-The-Middle*) y ataques de copia y reproducción de tickets falsos (los tickets son válidos por un tiempo limitado).

Este protocolo es usado en diferentes aplicaciones como método de autenticación. Uno de las implementaciones más importantes es en *Active Directory*, el cual va a ser utilizado en el aplicativo. En esta implementación, Kerberos 5 permite a los usuarios autenticarse una sola vez para acceder a múltiples servicios.

## 2.5. El servidor HTTP Apache 2.0

El Protocolo de Transferencia de Hipertexto (HTTP) constituye la base de la comunicación de datos en la *World Wide Web*. Su función principal es establecer un marco para la transmisión de información entre clientes *web* y servidores *web*. La evolución de HTTP ha sido fundamental para el desarrollo de la Internet moderna, existiendo varias versiones de este protocolo, como HTTP/1.0, HTTP/1.1, HTTP/2 y HTTP/3 [19] [20].

HTTP opera bajo un modelo de petición-respuesta y cliente-servidor. Un cliente envía una petición a un servidor, especificando la acción que desea realizar y el recurso al que desea acceder. El servidor, tras procesar la petición, devuelve una respuesta que contiene la información solicitada o una indicación del resultado de la petición.

Existen cuatro tipos principales de métodos HTTP: el método GET se emplea para obtener un recurso del servidor, el método POST se utiliza para enviar datos o ficheros al servidor, el método PUT se utiliza para actualizar un recurso almacenado en el servidor y, por último, el método DELETE se emplea para eliminar un recurso existente. Estas peticiones deben ser utilizadas a través de una URI (*Uniform Resource Identifier*) que identifica el recurso, e incluir datos adicionales como la versión del protocolo HTTP utilizada, entre otros [21].

Uno de los servidores HTTP que implementan este protocolo es Apache 2.0 y es uno de los más extendidos. Este servidor implementa HTTP/1.1 e introdujo mejoras significativas en su arquitectura con respecto a versiones anteriores, incluyendo un sistema de módulos más flexible y una mejor gestión de la concurrencia a través de la introducción de la arquitectura *Multi-Processing Modules* (MPMs) [22].

La configuración del servidor Apache 2.0 se realiza mediante ficheros de configuración, siendo el fichero más importante httpd.conf, donde se definen directivas que controlan el comportamiento del servidor, como los puertos de escucha, los *hosts* virtuales (característica que permite alojar múltiples sitios *web* en un único servidor), las reglas de acceso y la configuración

de seguridad.

Para este proyecto en específico, el uso del servidor Apache es fundamental para la transmisión de ficheros de tamaño no trivial y que sean compatibles con el arranque mediante red, es decir, compatibles con máquinas que empleen el protocolo PXE. La preferencia por usar este servidor se debe a que Apache 2.0 es un servidor *web* robusto y flexible, necesario para la implementación del servicio HTTP, y la pasada experiencia del autor de este proyecto, relacionada con la configuración de servicios web, que ha tenido a lo largo de sus estudios.

## 2.6. Windows PE

*Windows Preinstallation Environment* (WinPE) es un entorno ligero de Windows, diseñado principalmente para el despliegue, implementación y reparación de ediciones de escritorio de Windows, incluyendo Windows, Windows Server y otros sistemas operativos de Windows [23]. Su arquitectura se basa en el núcleo de Windows, lo que hace que sea compatible con los controladores y las herramientas del ecosistema Windows, pero opera de forma autónoma, sin depender de una instalación completa del sistema operativo en el disco duro local. Esta característica, junto a su naturaleza portable, permite su ejecución desde una diversidad de medios de arranque, incluyendo unidades de estado sólido (SSD) o discos duros externos, unidades *flash* USB, CD-ROM/DVD, o a través de la red mediante el protocolo PXE, lo que subraya su versatilidad en entornos de despliegue.

La función principal de Windows PE radica en proporcionar un entorno operativo mínimo, pero que al mismo tiempo sea robusto, lo que permite a los técnicos y administradores de sistemas llevar a cabo una gran variedad de operaciones críticas antes de la instalación o durante el mantenimiento del sistema operativo completo. Entre las opciones que brinda, se incluyen la partición y el formato de discos duros, la aplicación de imágenes de sistema operativo, la configuración de la infraestructura de red, la ejecución de herramientas de diagnóstico para la resolución de problemas y la recuperación de sistemas ante fallos críticos. Por otro lado, su diseño optimizado minimiza el consumo de recursos, siendo idóneo para entornos donde la memoria RAM es limitada o donde es necesaria una inicialización rápida del sistema.

En el caso de una instalación desatendida de Windows 11, Windows PE desempeña un papel fundamental, ya que actúa como la plataforma de arranque inicial para el proceso de despliegue. Cuando un equipo se inicia desde un medio que contiene una imagen de Windows PE, el entorno operativo reducido se carga en la memoria RAM del sistema. Una vez está operativo, se convierte en el punto de partida para la ejecución de *scripts* y herramientas diseñadas para automatizar íntegramente el proceso de instalación de Windows 11. Esta automatización es fundamental para la eficiencia en despliegues a gran escala, donde se reduce a cero la intervención humana.

El uso de Windows PE en instalaciones en red es una de sus características más relevantes, lo cual optimiza la implementación de sistemas en entornos empresariales y/o educacionales. Algunas de las características que lo hacen clave son:

- **Compatibilidad con redes:** Windows PE incorpora un conjunto básico de controladores de red que permiten la conexión a servidores mediante protocolos TCP/IP y NetBIOS sobre TCP/IP, facilitando la instalación remota del sistema operativo. Aunque estas funcionalidades de red sean básicas, son suficientes para las tareas de despliegue, permitiendo que el entorno acceda a recursos compartidos en la red, donde pueden residir los ficheros de instalación de Windows 11y los ficheros de respuesta. Esta funcionalidad es fundamental, ya que permite despliegues almacenando los recursos en un solo punto accesible por red, eliminando la necesidad de utilizar un medio físico para cada equipo.
- **Automatización mediante *scripts*:** la compatibilidad con *scripts* y herramientas especializadas permite optimizar el proceso de instalación de Windows en múltiples dispositivos de manera simultánea. Al tener la capacidad de ejecutar *scripts*, permite a los administradores del sistema personalizar los *scripts* en Powershell o CMD para automatizar las inslaciones. La automatización reduce drásticamente el tiempo de implementación por equipo y minimiza los errores humanos, asegurando consistencia y uniformidad en todas las instalaciones.
- **Arranque a través de PXE:** Windows PE puede ser utilizado en combinación con PXE para iniciar sistemas desde una red sin necesidad de medios de almacenamiento físicos, lo que resulta ideal para despliegues a gran escala.
- **Entorno operativo de preinstalación:** Este entorno es indispensable para la manipulación de los dispositivos de almacenamiento y el sistema de ficheros, lo cual permite la creación de particiones, el formato de volúmenes y la preparación del disco para recibir la imagen de Windows 11.
- **Ejecución de herramientas:** Windows PE facilita la ejecución de utilidades de línea de comandos y herramientas. Entre ellas se encuentra DISM (*Deployment Image Servicing and Management*), una herramienta versátil que permite inyectar controladores adicionales necesarios para el *hardware* específico del equipo o incluso integrar actualizaciones y paquetes de idioma en la imagen antes de su despliegue. Además, Windows PE es el entorno desde el cual se ejecuta la herramienta de instalación `setup.exe` de Windows.

## 2.7. Windows ADK

En la sección anterior se ha explicado qué es Windows PE y para qué sirve dentro del proceso de instalación mediante red. Sin embargo, no se ha indicado cómo se genera y las herramientas

que se utilizan para modificarlo. Esto es posible hacerlo mediante el uso de Windows ADK.

Windows ADK (*Windows Assessment and Deployment Kit*) [24] consiste en una colección de herramientas y documentación producida por Microsoft, diseñada para facilitar la personalización, evaluación e implementación del sistema operativo Windows. Este conjunto de herramientas permite la creación de entornos de preinstalación personalizados y automatización e implementación a gran escala de sistemas operativos. La función primordial del Windows ADK es proporcionar las herramientas necesarias para adaptar la instalación de Windows a necesidades específicas de cada organización, automatizar el proceso de despliegue para reducir la intervención manual y el tiempo de implementación, y asegurar la compatibilidad y el rendimiento de los sistemas implementados.

Para un proyecto como este, en el que se lleva a cabo una instalación desatendida de Windows 11, Windows ADK es indispensable, ya que proporciona las herramientas esenciales para la creación de Windows PE y el fichero de respuestas que automatiza la configuración de la instalación (`Autounattend.xml`).

Para el primero de los dos casos, dentro del kit se encuentra la herramienta "`Windows Preinstallation Environment`" (ver figura 3, la cual permite generar imágenes de arranque personalizadas. Esta herramienta consiste en una consola de comandos que implementa otras herramientas más básicas, como `copype` o `dism`.


Figura 3: Herramienta de creación de Windows PE.

Para el segundo de los casos, dentro del kit se encuentra la herramienta "`Windows System Image Manager`", la cual permite crear y modificar ficheros de respuestas mediante una interfaz gráfica, sin necesidad de introducir código de manera manual y teniendo una imagen completa de Windows 11 de referencia. Para lograr el objetivo de que se instale Windows 11 de manera desatendida, no basta simplemente con Windows PE, sino que también es necesario que exista un fichero de respuestas que, tal y como su nombre indica, responda a todas las preguntas que nos propone el sistema operativo durante el proceso de instalación normal. Dicho fichero se llama `Autounattend.xml` y permite a los administradores de redes y sistemas personalizar el entorno de instalación de Windows. En la figura 4 se observa la creación de un fichero de respuestas empleando esta herramienta.

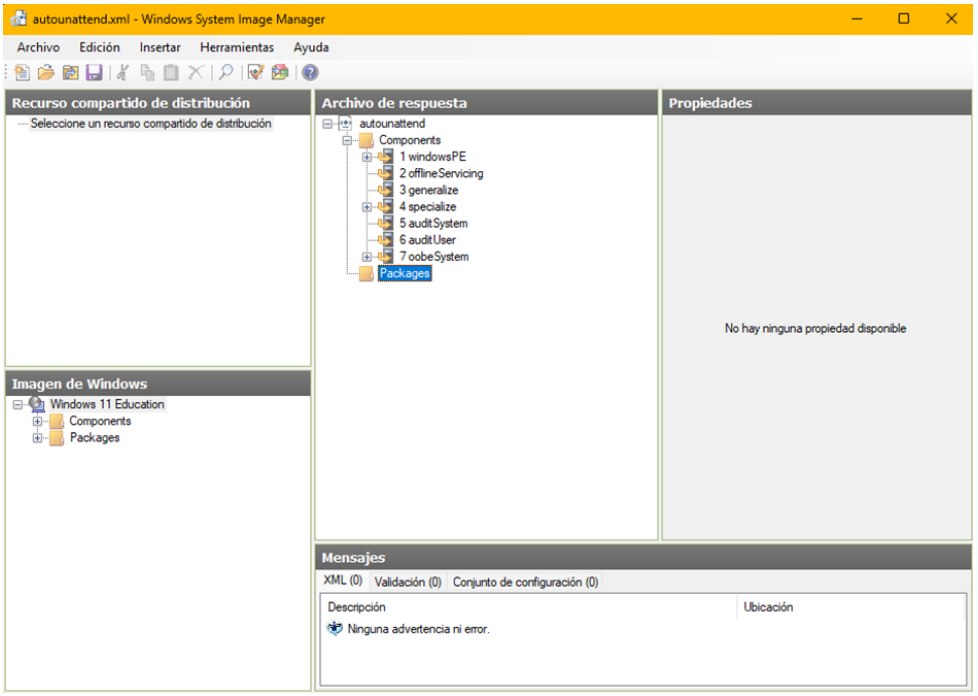


Figura 4: Herramienta “Windows System Image Manager”.

El fichero `Autounattend.xml` es utilizado en las instalaciones desatendidas de Windows y su propósito es proporcionar instrucciones predefinidas al programa de instalación de Windows para minimizar la intervención del usuario durante el proceso de instalación. Algunos ejemplos son: seleccionar el idioma, el navegador por defecto o las aplicaciones que todas las máquinas deben tener instaladas.

Para que Windows detecte el fichero y haga uso de él durante el proceso de instalación, debe ubicarse en la raíz de la unidad de instalación. Cuando comienza la instalación, el programa de instalación lee el fichero y aplica automáticamente la configuración especificada. Al finalizar, el sistema ya puede ser utilizado libremente por el usuario, haciendo un uso dentro del marco establecido por la organización.

## 2.8. Entorno de simulacuión VMware Workstation Pro

En el panorama tecnológico actual, la virtualización se ha convertido en una herramienta clave la hora del desarrollo de proyectos en el sector IT. Este *software* de virtualización permite ejecutar múltiples sistemas operativos como “huéspedes” dentro de un sistema operativo “anfitrión”, compartiendo los recursos de *hardware* subyacentes. Dentro del mercado existen varias soluciones de carácter gratuito que ofrecen a los usuarios una amplia gama de funcionalidades para la creación y gestión de máquinas virtuales. Este es el caso de VMware Workstation Pro

[25].

VMware Workstation Pro es un hipervisor de tipo 2, lo que significa que se ejecuta sobre un sistema operativo anfitrión, ya sea Windows o Linux. Este software permite crear y ejecutar múltiples máquinas virtuales (VMs) simultáneamente en una misma máquina física. Cada máquina virtual puede ejecutar su propio sistema operativo y aplicaciones, aisladas del sistema anfitrión y de otras VMs.

La necesidad de emplear un hipervisor para la realización de simulaciones durante el desarrollo del aplicativo es resuelta con esta solución. La decisión de emplear este software reside en sus ventajas:

- **Ejecución de múltiples sistemas operativos:** permite tener máquinas Linux y Windows ejecutándose al mismo tiempo, lo que es ideal para este proyecto.
- **Aislamiento y seguridad:** en caso de error en una máquina, no hay riesgo de que este se propague en el resto de máquinas.
- **Instantáneas:** posibilita guardar el estado actual de una máquina virtual para poder revertir cualquier cambio que se haya hecho. Esto es muy útil al hacer simulaciones de cualquier aplicativo.
- **Comunicación entre máquinas:** al igual que permite crear dispositivos de red, permite crear redes virtuales que interconectan estos dispositivos, permitiendo la comunicación directa entre ellos o incluso la salida a internet.

VMware Workstation Pro es una herramienta clave para el desarrollo de un proyecto como este, en el que se necesita simular un servidor, partiendo de una imagen del servidor actual desplegado, para posteriormente realizar cambios y comprobar el funcionamiento del aplicativo. Esto permite que cada vez que se produzca un cambio, no deje sin servicio al resto de usuarios de manera accidental.

En la figura 5 se muestra como se ve el servidor Debian 12 que va a ofrecer los servicios descritos en esta memoria corriendo en la herramienta VMware Workstation Pro.

## 2.9. Discusión

Para la necesidad planteada, en la que se necesita hacer una instalación a mediana escala del sistema operativo Windows 11 Education de manera desatendida, es fundamental la elección de las tecnologías útiles y de su funcionamiento.

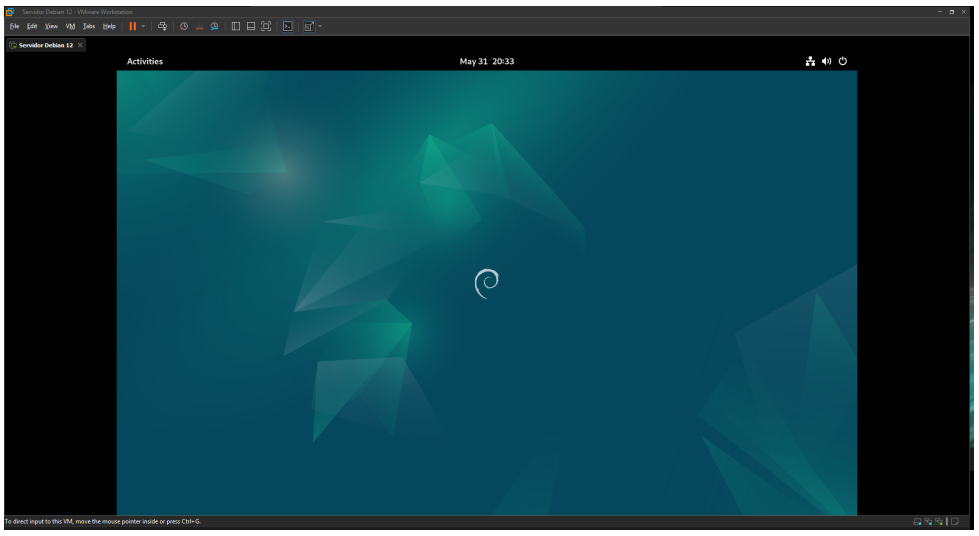


Figura 5: Herramienta VMware Workstation Pro.

El uso de cada tecnología es limitada para prestar el servicio necesario, jugando cada una un rol fundamental que hace que el conjunto de sus funciones consiga el objetivo final. Como resumen, a continuación se presenta una pequeña discusión sobre cuál es el rol de cada tecnología en el aplicativo descrito en la memoria, que posteriormente se describirá en mayor medida.

El cliente arranca en red haciendo uso de PXE y obtiene toda la información de red haciendo uso del protocolo DHCP. Con todos los parámetros de red, solicita el fichero de arranque al servidor TFTP y lo ejecuta. Una vez selecciona la instalación de Windows 11, comienza la descarga de los ficheros de Windows PE a través del protocolo HTTP, ya que el tamaño de estos ficheros no es despreciable y se necesita un protocolo orientado a grandes volúmenes de datos. Ejecutado Windows PE, se monta en la máquina cliente un directorio activo y se inicia sesión en él mediante el uso de Samba, para ejecutar el instalador de Windows 11 llamado `setup.exe`. La instalación de Windows 11 se hace a través de un fichero de respuesta, creado con las herramientas de Windows ADK. Durante la instalación, se registra la nueva máquina en el dominio, para que solamente el personal autorizado del laboratorio 25 de investigación pueda tener acceso. Cuando la instalación finaliza, los usuarios acceden mediante las credenciales del dominio de Samba.

Todo esto es primero simulado con la herramienta VMware, con el objetivo de hacer pruebas interactivas sin cortar el servicio actualmente desplegado.

# Capítulo 3

# Diseño de los servicios en red

En este capítulo se presenta la primera parte del desarrollo del aplicativo. En primer lugar, se presenta brevemente el objetivo final del proyecto. A continuación, se describe la configuración recomendada que debe tener el servidor del departamento previo al despliegue, para posteriormente presentar la configuración de todos los servicios que va a prestar el servidor a los clientes de la red.

A lo largo del desarrollo del proyecto y por motivos de seguridad, antes de la modificación de los ficheros de configuración de cada uno de los servicios se ha llevado a cabo una copia de seguridad. Debido a que se ha querido que la descripción de los pasos realizados fuese lo más clara posible, se han omitido estos pasos a lo largo del capítulo.

## 3.1. Proceso de instalación

Antes de comenzar con la el diseño y desarrollo de los servicios de red hay que primero definir la forma con la que se va a conseguir el objetivo final de este aplicatio.

El aplicativo consiste en una estructura cliente-servidor, en donde distintos servidores, alojados en la misma máquina física, van a prestar el servicio de instalación de Windows 11 de manera automatizada a una gran cantidad de clientes. Los servicios que se van a proporcionar son: PXE, DHCP, TFTP, HTTP y Samba. Los pueden hacer uso del aplicativo de manera simultánea, o de manera ordenada, como va a ser en la mayoría de los casos, ya que es está pensado para un solo uso por disposivo.

Un cliente, previamente configurado para iniciar por red, arranca y se conecta a la red local mediante el protocolo de red PXE. El servidor PXE, correctamente configurado, envía los ficheros

necesarios al cliente para mostrarle un menú de opciones entre las que se encuentra la opción "Windows 11'. El cliente, ejecuta el fichero y navega por el menú de opciones con las flechas del teclado y selecciona la opción deseada pulsando la tecla *Enter*. Al seleccionar la opción "Windows 11" del menú, comienza el proceso de descarga y ejecución de ficheros instalación y *scripts*, todo de manera automática, no siendo necesaria ninguna otra interacción por parte del cliente.

Los primeros ficheros que se van a descargar van a ser los siete necesarios para que pueda ejecutarse Windows PE. Windows PE va a ser fundamental ya que, a través de un *script*, se va a acceder al directorio compartido en Samba en donde se aloja la imagen de Windows 11.

Cuando Windows PE accede al directorio compartido en Samba, ejecuta automáticamente el instalador de Windows, comenzando así la instalación. La instalación se llevará a cabo sin intervención del administrador, ya que toda la configuración la leerá del fichero de respuestas `Autounattend.xml`.

El proceso de instalación de Windows consta de siete partes fundamentales [26]:

1. **`WindowsPE`:** Esta fase se ejecuta en el entorno de preinstalación de Windows y se configuran los aspectos relacionados con el programa de instalación. Se utiliza para realizar configuraciones de disco, creación de particiones, introducción de la clave del producto y ejección de comandos durante el programa de instalación.
2. **`OfflineServicing`:** Esta segunda se utiliza para instalar actualizaciones, paquetes y controladores en una imagen de Windows sin conexión antes de que se inicie.
3. **`Generalize`:** Este paso de la instalación prepara la imagen de Windows para ser utilizada en un gran cantidad diversa de equipos, eliminando información específica del hardware para evitar incompatibilidades. La fase `Generalize` se usa para crear una imagen de referencia de Windows que se puede usar en toda la organización.
4. **`Specialize`:** A partir de la imagen anterior, durante el paso de configuración `Specialize` del programa de instalación se aplican personalizaciones adicionales que se aplican a diferentes divisiones de una organización. Aquí se incluyen configuraciones específicas del sistema, como la configuración de red.
5. **`AuditSystem`:** Este paso se utiliza para pasar los procesos desatendidos de configuración de Windows en el contexto del sistema en modo auditoría. El modo auditoría permite instalar controladores de dispositivos adicionales y aplicaciones. Se utiliza para realizar pruebas y configuraciones adicionales antes de que el sistema esté listo para el usuario final. Solamente se ejecuta cuando se configura para arrancar en modo auditoría.
6. **`AuditUser`:** Similar a `AuditSystem`, pero se ejecuta cuando un usuario inicia sesión en modo auditoría y se utiliza principalmente para ejecutar *scripts*, aplicaciones u otros

ejecutables.

7. **`OobeSystem`:** Por último, en la fase `oobeSystem` se configuran las opciones que se aplican durante la experiencia del primer arranque de un dispositivo con Windows llamada OOBE (*Out-Of-Box Experience*) y se ejecuta durante la experiencia de configuración inicial. Permite configurar opciones como la creación de cuentas de usuario y la configuración regional.

Finalmente, cuando termina el proceso de instalación, se muestra el menú de inicio de sesión de usuarios de Windows. Los usuarios deben acceder a la máquina con el nuevo sistema operativo con las credenciales del dominio, facilitadas por el administrador del sistema.

## 3.2. Configuración del servidor

Tal y como se ha indicado anteriormente, el arranque por red mediante PXE permite a los dispositivos iniciar y cargar un sistema operativo sin necesidad de medios físicos como discos o USBs. Para poder llevar a cabo el despliegue es primordial tener un servidor configurado de manera correcta para ofrecer el servicio. Dicho servidor es uno ya existente desplegado en el laboratorio 25 de investigación de la Escuela de Ingenieros de Telecomunicaciones de la Universidad de Valladolid, con el sistema operativo Debian 12 Server ya instalado, y en el que previamente ya se haya configurado de manera correcta un servicio PXE para ofrecer un servicio de instalación en red de sistemas operativos Linux y Windows 7 hasta el momento.

Partiendo de la base proporcionada por el centro educativo, la primera parte es mucho más rápida de lo habitual, ya que no es necesaria la configuración de los servicios DHCP, TFTP y HTTP desde cero, sino que simplemente se han de modificar y/o añadir escasas líneas de código o instrucciones para adaptarse a la necesidad planteada. Por el contrario, se ha de configurar completamente la parte de los servicios de Samba y DNS para la resolución local de nombres.

Por el contrario, se ha de configurar completamente la parte de la instalación de Windows 11 de manera automática, así como controlar todo el proceso de envío de ficheros de instalación necesarios desde el servidor hasta el cliente, y llevar a cabo la personalización de la imagen ISO. Esta configuración debe ser prototípica y adaptada para todos los usuarios que vayan a conformar parte del equipo de laboratorio de investigación.

Antes de comenzar, es importante saber dónde se encuentran localizados los directorios base de cada servidor, para que de igual forma se puedan localizar los ficheros que se deben modificar para implementar esta funcionalidad en la red. El directorio base de los servidores de compartición de ficheros se ubica en `/srv/`, hallándose en ese directorio los subdirectorios

destinados a los servicios TFTP y Samba; y el directorio `/var/www/html/` para el caso del servidor HTTP.

Localizados los servicios, se deben actualizar todos los paquetes del servidor para que estén en la última versión posible a la hora de llevar a cabo las modificaciones:

```
sudo apt update
```

## 3.3. Modificación del servidor PXE

El primer paso de todos es llevar a cabo la revisión de la configuración actual del servidor PXE. Para que el sistema funcione de manera correcta, es necesario que tanto el servidor DHCP como el servidor TFTP estén debidamente configurados y activos en la red donde van a proporcionar sus servicios. En el caso de que no se disponga de uno de los dos, se debe instalar el *software* apropiado para dar el servicio. Sin embargo, en el caso que se acontece sí existen ambos *software* instalados en el mismo servidor, ya que el servidor es la máquina encargada de proporcionar todas las configuraciones de la red interna ya preexistentes a los usuarios.

Previo a la reconfiguración del servicio DHCP, se debe revisar y reconfigurar el servicio TFTP, para que el fichero con el menú de opciones del servicio PXE sea accesible para el usuario. Al fichero que se encontraba ya configurado y que servía para prestar el servicio, se ha añadido una opción llamada “Windows 11” que, si se selecciona pulsando la tecla *Enter*, comienza la descarga de los ficheros de Windows PE, y la posterior instalación automatizada del sistema operativo Windows.

Para el caso del servicio TFTP, el *software* elegido es el *software* `tftp-hpa` [27]. Como se ha comentado anteriormente, este *software* ya se encuentra previamente instalado en el servidor, por lo que es más sencilla su configuración, ya que no es necesario comenzar desde el principio. En primer lugar, se debe actualizar la versión de los paquetes del *software*:

```
sudo apt upgrade tftpd-hpa
```

Tras actualizar los paquetes, hay que revisar los principales ficheros de configuración del servicio para modificarlos y así añadir las actualizaciones descritas en esta memoria. El fichero a configurar se ubica en el directorio `/etc/default/` y se llama `tftpd-hpa`. Tras adaptar el servidor, el documento presenta la siguiente estructura:

Cabe destacar que se debe crear un directorio base del servidor TFTP, cuyo nombre será tftp con permisos de lectura y ejecución para todos los usuarios. Tal y como se muestra en el *script* 1,

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS="192.168.1.100:69"
TFTP_OPTIONS="--secure --verbose --ipv4"
```

*Script* 1: Fichero de configuración `tftpd-hpa`.

se ha escogido el directorio `/srv/tftp` como directorio base del servidor TFTP, previamente creado con el siguiente comando:

```
sudo mkdir -p /srv/tftp
```

Además, se han incluido las siguientes configuraciones:

- **`TFTP_USERNAME="tftp"`:** el demonio del servidor TFTP se ejecuta como usuario tftp y no como otro tipo de usuarios, como por ejemplo el usuario *root* (administrador). Esta medida es fundamental ya que este usuario debe tener los permisos mínimos y su función limitada al directorio TFTP.
- **`TFTP_ADDRESS="192.168.1.10:69"`:** se indica el puerto UDP en donde escuchará la llegada se solicitudes de descarga nuevas el servicio TFTP. En este caso se ha elegido el puerto estándar de TFTP, es decir, el puerto UDP 69.
- **`TFTP_OPTIONS="--secure --verbose --ipv4"`:** gracias a estas opciones, no se permite que ningún usuario pueda crear mediante una petición PUT un nuevo fichero dentro del directorio, se restringe el uso del servicio a solamente direcciones IPv4, y se aumenta la seguridad del servidor con la opción "secure", ya que el deminio realiza una operación *chroot*, aislándose dentro del directorio, por lo que no puede acceder a ningún fichero fuera de él.

Tras revisar y entender la configuración, es momento de avanzar al desarrollo de la configuración del servicio DHCP. El *software* instalado en el servidor para ofrecer el servicio DHCP es `isc-dhcp-server`, ya que las máquinas cliente en dicho laboratorio no disponen de una dirección IP estática, por lo que este servicio está ya activo. Para actualizar los paquetes a la versión más reciente disponible, se utiliza el siguiente comando:

```
sudo apt upgrade isc-dhcp-server
```

Teniendo esto en cuenta, simplemente basta con saber los ficheros de configuración que se deben modificar o revisar para que se preste el servicio objetivo de este trabajo. Estos ficheros se llaman dhcpd.conf e `isc-dhcp-server`, cuyas ubicaciones dentro del sistema de almacenamiento del servidor son `/etc/dhcp` y `/etc/default`, respectivamente.

Aunque el servidor DHCP esté ya configurado, debe existir una revisión del código de configuración del servicio, al igual que en el caso anterior. Para su correcto despliegue, las funcionalidades "*allow booting*" y "*allow bootp*" deben estar presentes en el fichero de configuración del servicio `dhcpd.conf`, en la red correspondiente:

- ***Allow booting*:** Permite que los clientes que intenten arrancar a través de la red puedan obtener una dirección IP y los ficheros necesarios para el proceso de arranque. Esta opción es fundamental, ya que es el mismo servidor que actúa como servidor DHCP y PXE, permitiendo así proporcionar imágenes de arranque a los clientes PXE.
- ***Allow bootp* (opcional):** Permite explícitamente a los clientes que utilizan BOOTP, precursor de PXE, recibir configuraciones para el arranque y configuraciones de red.

```
authoritative;
subnet 192.168.1.0 netmask 255.255.255.0 {
    option routers 192.168.1.1;
    option domain-name "lab25.local";
    option broadcast-address 192.168.1.255;
    option domain-name-servers 192.168.1.100;

    range 192.168.1.110 192.168.1.160;
    next-server 192.168.1.100;

    default-lease-time 600;
    max-lease-time 7200;

    allow booting;
    allow bootp;

    if option arch = 00:07 or option arch = 00:09 {
        filename "ipxe.efi";
    } else {
        filename "ipxe.pxe";
    }
}
```

*Script* 2: Fichero de configuración `dhcpd.conf`.

Se debe añadir una condición indispensable para tratar de manera diferente a los clientes que presenten un *firmware* con UEFI de aquellos que no lo tengan. Debido a que, como hemos visto en la sección anterior, Windows 11 necesita instalarse sobre un *firmware* con UEFI, se debe enviar un fichero de extensión EFI, en donde se mostrará un menú interactivo al usuario. Este menú será la única interacción del usuario durante todo el proceso de instalación. Además del fichero, se debe indicar el directorio del servidor TFTP en donde se aloja este fichero, para que pueda ser encontrado. Por otro lado, solamente se enviará a los clientes que cumplan la condición del *firmware* UEFI, siendo mostrada una versión anterior del menú a aquellos usuarios

que no la cumplan, es decir, sin la opción de instalar Windows 11, ya que no cumple con uno de los requisitos mínimos para llevar a cabo la instalación. Con ambas modificaciones, el fichero resultante se muestra en el *script* 2.

Además del fichero `dhcp.conf`, se revisa el fichero de configuración `isc-dhcp-server`. En este fichero se muestran las interfaces físicas en donde va a estar desplegado el servicio. Al ser un servidor que va a prestar servicio a la red local del laboratorio 25, solamente es necesario que escuche en una interfaz, concretamente en la interfaz conectada a la red local. Junto a esto, solamente se prestará servicio DHCP para IPv4, por lo que el fichero de configuración isc-dhcp-server se observa en el *script* 3.

```
INTERFACESv4="enp0s3"
INTERFACESv6=""
```

*Script* 3: Fichero de configuración `isc-dhcp-server`.

Nada más conectarse una máquina cliente a la red haciendo uso del protocolo PXE, ésta solicita los datos necesarios para navegar y poder comunicarse con otras máquinas mediante un mensaje de difusión, esperando la respuesta de un servidor DHCP. El servidor, aparte de proporcionar todos los datos necesarios al cliente, le proporciona el nombre del fichero disponible en el servidor TFTP para que el cliente lo descargue. El el nombre del fichero es `ipxe.efi` en caso de tener el modo UEFI activo o `ipxe.pxe` en caso contrario, y la ruta del servidor TFTP en donde se encuentra el fichero. Tras recibir dicha información, el cliente solicita el fichero indicado por el servidor y lo ejecuta, presentando el menú principal con las opciones de arranque existentes.

## 3.4. Modificación del servidor HTTP

Una vez los anteriores servicios se encuentran preparados, se procede a realizar la configuración del tercer servicio, el servicio *web* o HTTP. El servicio HTTP es fundamental para la transmisión de ficheros de un tamaño no trivial al cliente, ya que es bastante más rápido y eficiente que el servicio TFTP.

Para prestar este servicio, el *software* elegido es Apache2, ya que es el que previamente se encontraba desplegado. Este *software* es uno de los más empleados y da respuesta a las peticiones de los navegadores. Al igual que en los casos anteriores, se deben actualizar todos los paquetes del servicio:

```
sudo apt upgrade apache2
```

La configuración del servicio se encuentra distribuida en varios ficheros. Para este servicio, se van a mantener todas las configuraciones por defecto, es decir, no se va a modificar el fichero de configuración `000-default.conf` ubicado en el directorio `/etc/apache2/sites-available/`, ni el fichero `apache2.conf` ubicado en el directorio `/etc/apache2/`. El directorio donde se van a almacenar todos los ficheros a compartir será el directorio `/var/www/html/`.

En este servidor se alojarán los ficheros fundamentales de la imagen Windows PE. Para poder ofrecerlos y que sean accesibles para el cliente, realizando las solicitudes indicadas en el fichero EFI enviado, se debe crear el directorio donde se van a almacenar. Dicho directorio se llama WinPE y se encuentra en el directorio base del servidor HTTP.

Estos ficheros de Windows PE contienen la información fundamental para que, de manera autónoma, se soliciten los ficheros de instalación de la imagen de Windows 11, haciendo uso de un directorio activo, y dé comienzo la instalación a través del protocolo SMB.

## 3.5. Configuración del servidor Samba

Tras configurar los tres servicios anteriormente descritos, es necesario activar un último servidor, que prestará a su vez dos servicios adicionales.

El primer objetivo que tiene este servidor es actuar como un controlador de dominio, para gestionar la seguridad y la autenticación de los usuarios dentro del dominio y, por consiguiente, de la red del laboratorio. Este servicio es necesario para permitir el acceso a los ordenadores en donde se instalará el nuevo sistema operativo solamente a los usuarios que pertenezcan al dominio existente, y restringir de igual forma el acceso a usuarios externos.

Por otro lado, el segundo objetivo que tiene es el de ofrecer el servicio de compartición de ficheros, para hacer accesibles todos los ficheros de la imagen ISO de Windows 11 que son necesarios para realizar la instalación remota.

El servidor que va a brindar este servicio es el servidor Samba, y el software que utiliza para el servicio es el software de código abierto llamado samba, el cual permite compartir ficheros e impresoras con Windows. Este servicio utiliza los protocolos SMB/CIFS, los mismos que usa Windows para los recursos compartidos en red. Para obtenerlo, hay que instalarlo a través de la siguiente instrucción:

```
sudo apt install samba
```

Este comando instala el servicio principal del servicio Samba, el cual consiste en un servidor

de ficheros y que actúa como un *Active Directory Domain Controller* (AD-DC), implementa los protocolos SMB/CIFS e incluye los demonios `smbd`, `nmbd`, y `samba`. Adicionalmente, se deben instalar las herramientas que permitan trabajar con las bases de datos de Samba, las cuales se usarán más adelante:

```
sudo apt install ldb-tools
```

### 3.5.1. Creación de un dominio y alta de usuarios

Tras la instalación del *software* en el servidor, así como de todos los ficheros de configuración y herramientas que se emplean para su despliegue, hay que realizar cambios en las reglas por defecto para adecuarlo a las necesidades planteadas.

Samba permite actuar a un servidor Debian como un Controlador de Dominio de Directorio Activo (AD-DC) compatible con clientes Windows. Esta funcionalidad es especialmente útil ya que permite centralizar la autenticación de usuarios, la gestión de políticas de seguridad y el acceso a recursos compartidos.

Antes de comenzar con la creación del dominio, hay que tener en cuenta que se debe configurar un servicio DNS para ese dominio, ya que es fundamental para traducir el nombre del dominio a una dirección IP con la cual comunicarse. Si no está correctamente configurado, el sistema operativo Windows no podrá unirse al dominio y, por lo tanto, el servicio fallará.

#### Configuración del *resolver*

Para configurar correctamente el *resolver* del propio servidor Samba, hay que modificar el fichero `/etc/hosts` (ver *script* 4), ya que en ese fichero se encuentran mapeados los nombres de dominio con respecto a las correspondientes direcciones IP. En este caso, solo se mapeará el propio servidor, por lo que se debe añadir la última línea del código presentado para que el nombre del servidor se resuelva con su dirección IP.

```
127.0.0.1       localhost
127.0.1.1       ServidorDebian12.lab25.local    ServidorDebian12
192.168.1.100   ServidorDebian12.lab25.local    ServidorDebian12
192.168.1.100   lab25.local                     ServidorDebian12
```

*Script* 4: Fichero de configuración `/etc/hosts`.

A continuación, se para el servicio `systemd-resolved`, es decir, el demonio encargado del *resolver*, para posteriormente eliminar el enlace simbólico del fichero `/etc/resolv.conf`,

ya que de esta manera es posible configurar manualmente el fichero de configuración. A continuación, se crea de nuevo el fichero:

```
sudo systemctl disable --now systemd-resolved
sudo unlink /etc/resolv.conf
sudo nano /etc/resolv.conf
```

Una vez creado, se modifica el contenido del fichero con el objetivo de que actúe como servidor DNS principal del servidor Samba, proporcionar un servidor DNS alternativo y, por último, se configura la búsqueda del dominio local. En el *script* 5 puede observarse como en este caso se ha elegido el DNS de Google por ser ampliamente conocido y por las reglas de seguridad que presenta [28], pero también puede ser otro, como por ejemplo el del proveedor de internet de la organización.

```
nameserver 192.168.1.100
nameserver 8.8.8.8
search lab25.local
```

*Script* 5: Fichero de configuración `/etc/resolv.conf`.

Para proteger el fichero de modificaciones no intencionadas, como una modificación por parte de NetworkManager [29] o una reescritura automática mediante DHCP, es recomendable hacerlo inmutable:

```
sudo chattr +i /etc/resolv.conf
```

Este comando aplica el atributo de inmutabilidad al fichero especificado, impidiendo su modificación, borrado o renombramiento. Esto quiere decir que ningún tipo de proceso puede hacer estas acciones sobre el fichero, incluso ningún usuario con privilegios de administrador. Para revertirlo, solamente el usuario administrador debe aplicar el mismo comando, pero cambiando el signo "+" por el signo "−".

### Configuración del servicio AD-DC

Configurado correctamente el *resolver* del servidor de dominio, se comienza en primer lugar con la instalación de paquetes adicionales que van a ser necesarios para prestar el servicio de dominio, junto a herramientas clave para la configuración del servicio:

```
sudo apt install -y acl attr samba-dsdb-modules samba-vfs-modules smbclient
    winbind libpam-winbind libnss-winbind libpam-krb5 krb5-config krb5-user
    dnsutils chrony
```

Para la implementación de un servidor Linux que actúe como Controlador de Dominio (*Domain Controller*, DC) compatible con *Active Directory* (AD), y que proporcione servicios de autenticación centralizada mediante Kerberos 5, así como la compartición de ficheros a través del protocolo SMB/CIFS, se ha procedido a la instalación de un conjunto de paquetes esenciales.

Se utiliza Kerberos 5 como sistema de autenticación en una arquitectura de controlador de dominio basada en Samba que se va a desplegar. El uso de Kerberos 5 no solo responde a una necesidad de compatibilidad con entornos Windows, sino que también responde a criterios de seguridad, interoperabilidad y eficiencia, ampliamente respaldados por la comunidad técnica y por los estándares industriales actuales. Kerberos 5 utiliza con fichero de configuración el llamado `krb5.conf` (*script* 6), ubicado en el directorio `/etc/` del servidor, y es absolutamente esencial. Sin él, tu sistema no sabrá cómo comunicarse con el AD DC para la autenticación Kerberos, lo que resultará en fallos de inicio de sesión y problemas de acceso a recursos.

```
[libdefaults]
    default_realm = LAB25.LOCAL
    dns_lookup_realm = false
    dns_lookup_kdc = true

[realms]
LAB25.LOCAL = {
    default_domain = lab25.local
}

[domain_realm]
    ServidorDebian12 = LAB25.LOCAL
```

*Script* 6: Fichero de configuración `krb5.conf`.

A continuación, se detallan los componentes seleccionados, agrupados según su funcionalidad.

- **`acl`:** habilita el uso de listas de control de acceso (ACLs) extendidas en el sistema de ficheros. Samba usa estas listas para replicar los permisos de UNIX con mayor fidelidad en el entorno Windows.
- **`attr`:** proporciona soporte para atributos extendidos de ficheros. Al igual que sucede con las `acl`, Samba usa estas listas para replicar los permisos de UNIX con mayor fidelidad en el entorno Windows
- **`samba-dsdb-modules`:** proporciona módulos adicionales para el backend de base de datos de Samba, fundamental para un controlador de dominio.
- **`samba-vfs-modules`:** incluyen módulos del sistema virtual de ficheros (VFS), los cuales permiten la implementación de características avanzadas sobre los ficheros compartidos, como *snapshots* o ACLs extendidas.

- **`smbclient`:** instala un cliente en línea de comandos para acceder a recursos compartidos SMB/CIFS y poder realizar pruebas en local.
- **`winbind`:** facilita la resolución de usuarios y grupos de Active Directory dentro del sistema Linux. Integra Samba con Directorio Activo.
- **`libpam-winbind`:** permite la autenticación de usuarios del dominio a través de PAM [30], habilitando el inicio de sesión con credenciales de Directorio Activo.
- **`libnss-winbind`:** proporciona soporte del servicio NSS para que Linux pueda reconocer usuarios y grupos del dominio. NSS es una interfaz que permite que las aplicaciones puedan acceder a diferentes fuentes de información de grupos y usuarios.
- **`krb5-config`** y **`krb5-user`:** instalan las herramientas y ficheros de configuración necesarios para habilitar la autenticación con Kerberos 5 en el sistema.
- **`dnsutils`:** incluye herramientas útiles para llevar a cabo la resolución de nombres.
- **`chrony`:** proporciona servicios de sincronización horaria, para que Kerberos 5 funcione correctamente.

Tras la instalación completa, se detienen y deshabilitan los servicios innecesarios para un servidor de *Active Directory* de Samba. El motivo principal por el que se emplea un AD y no un dominio Windows NT se debe a que un dominio Windows NT es un modelo más plano y limitado, con una estructura más sencilla pero con una gestión y configuración más compleja. El rol de Samba en un dominio AD son dos al mismo tiempo: el rol de controlador de dominio del *Active Directory* (gestionar un dominio) y la de servidor miembro de dominio de un AD (ser un miembro de otros dominio para permitir a los clientes acceder a esos dominio externos).

Al llevar la autenticación y el control de acceso al servidor mediante Kerberos 5, los servicios `smbd`, `nmbd` y `winbind` no son de utilidad para este proyecto (ver sección 2.4.1). Por el contrario, se habilita posteriormente el servicio correcto de Samba de Directorio Activo, es decir, el servicio `samba-ad-dc`, aunque no significa que está activo:

```
sudo systemctl disable --now smbd nmbd winbind
sudo systemctl unmask samba-ad-dc
sudo systemctl enable samba-ad-dc
```

Realizada la habilitación del servicio, es momento de configurar el dominio y el servidor Samba AD. En primer lugar, se crea el dominio que tendrá uso dentro del laboratorio haciendo uso de `samba-tool`:

```
sudo samba-tool domain provision
```

A las preguntas que se plantean al utilizar el comando, se indican los siguientes valores al asistente `samba-tool`:

```
Realm: LAB25.LOCAL
Domain: LAB25
Server Role: dc
DNS backend: SAMBA_INTERNAL
DNS forwarder IP address: 8.8.8.8
```

Tal y como puede observarse, el dominio que se va a provisionar para la red interna va a ser `lab25.local`, con nombre NetBIOS `lab25`. Además, se indica el rol del servidor Samba, en este caso el rol de controlador de dominio (dc), se indica también el DNS interno que va a utilizar Samba (se le da el valor `SAMBA_INTERNAL` para que Samba haga la consulta de resolución de nombres al *resolver* local), previamente configurado; y por último, un DNS al cual redirigir a los clientes cuando el DNS local no consiga resolver una petición dada. En esta última opción hay dos opciones posibles, la primera es dejarlo en blanco, y dejar que el DNS local lo redirija a otro cuando no pueda atender su petición; y como segunda opción puede indicarse un DNS alternativo. En este caso, se ha elegido como servidor alternativo el DNS de Google, para evitar cortes de servicio por este motivo.

A continuación, se copia el fichero de configuración de Kerberos 5 que se ha creado automáticamente con la generación del dominio en el directorio /etc/, para que sea accesible por el demonio:

```
sudo cp /var/lib/samba/private/krb5.conf /etc/krb5.conf
```

### Configuración de los usuarios

Con el servidor Samba ya en marcha, es momento de gestionar los usuarios que van a hacer uso de este servicio. Para que posteriormente los PCs puedan unirse al dominio de manera autónoma, es necesario crear un único usuario general con permisos restringidos dentro del servicio, pero que pueda añadir una gran cantidad de máquinas cliente. Este usuario se llamará `joinPC`, y se usará de ejemplo para esta sección.

Para gestionar los usuarios, lo primero es tener un registro de las personas físicas que van a hacer uso de las máquinas, para asociarlos a un usuario. Tras obtenerlo, se deben crear los usuarios y un grupo de usuarios con los permisos restringidos. Por ejemplo:

```
sudo samba-tool user create jesus 'Password456'
sudo samba-tool group add Lab25Users
sudo samba-tool group addmembers Lab25Users jesus
sudo samba-tool user create joinPC 'Password123'
samba-tool user setexpiry joinPC --noexpiry
```

```
sudo samba-tool group add JoinDomainUsers --description="Usuarios
   destinados a unir equipos"
sudo samba-tool group addmembers JoinDomainUsers joinPC
```

En el ejemplo anterior se muestra la creación de dos usuarios y dos grupos diferentes entre sí. El primer usuario llamado `jesus` pertenecerá a la cuenta personal de uno de los trabajadores del laboratorio, cuyos usuarios se añaden al grupo `Lab25Users`. Por otro lado, el usuario `joinPC` se va a añadir al grupo `JoinDomainUsers`, un grupo de carácter especial que va a ser empleado solamente para el proceso de instalación, no para uso individual o colectivo. Adicionalmente, se ha configurado que la contraseña del usuario `joinPC` no caduque, para así evitar tener que cambiar con cierta frecuencia la contraseña y, por tanto, las contraseñas de los ficheros de instalación. Cabe destacar que todos los usuarios pueden emplearse para unir un nuevo PC al dominio. En la figura 6 se observa el árbol LDAP del proyecto.

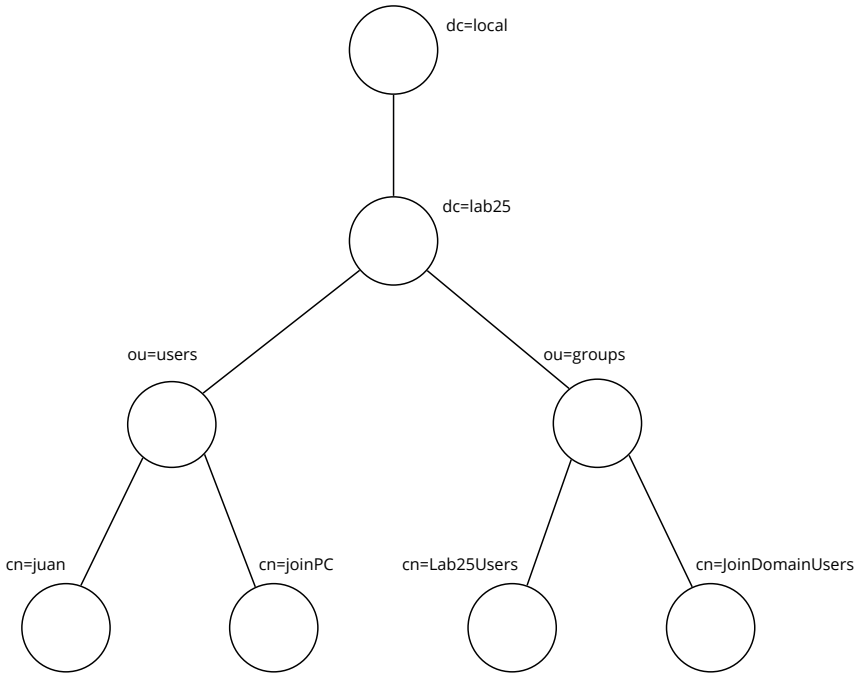


Figura 6: Esquema del arbol LDAP del proyecto.

Posterior a la creación de todos los usuarios y distintos grupos de trabajo, hay que modificar los permisos del grupo `JoinDomainUsers` para que los usuarios miembros estén habilitados para unir nuevas máquinas al dominio. Para poder dar permisos a este grupo de usuarios, se debe ejecutar el siguiente comando:

```
sudo samba-tool dsacl set --objectdn="CN=Computers,DC=lab25,DC=local" --
   permission="FULL" --principal="CN=JoinDomainUsers,CN=Users,DC=lab25,DC=
   local"
```

Este comando permite a los usuarios pertenecientes al grupo `JoinDomainUsers` unir nuevas máquinas (`CN=Computers`) al dominio. También se les autoriza a modificar las máquinas que pertenecen al dominio e incluso a eliminarlas. Una vez ya están autorizados, se debe modificar el número máximo de dispositivos que un único usuario puede unir al dominio. Debido a que por defecto en Windows cada usuario tiene limitadas las uniones a un total de diez máquinas, esto debe medificarse, ya que en la red donde se va a desplegar el aplicativo va a haber un mínimo de viente usuarios que van a utilizarlo. Esta modificación de lleva a cabo a través de un fichero creado específicamente para este objetivo llamado `cuota.ldif`, el cual va a modificar los registros necesarios de la base de datos para guardar los cambios.

```
dn: DC=lab25,DC=es
changetype: modify
replace: ms-DS-MachineAccountQuota
ms-DS-MachineAccountQuota: 50
```

*Script* 7: Fichero `cuota.ldif`.

En el *script* 7 se muestran las intrucciones a realizar en la base de datos. En este caso, se modifica el valor de la variable global de Samba "`ms-DS-MachineAccountQuota`" a 50 para el límite de uniones por usuario. Por último, se aplican las operaciones del fichero y comprobamos el valor nuevo de la variable:

```
sudo ldbmodify -H /var/lib/samba/private/sam.ldb cuota.ldif
sudo ldbsearch -H /var/lib/samba/private/sam.ldb "(&(objectClass=domainDNS)
   )" ms-DS-MachineAccountQuota
```

### 3.5.2. Configuración del directorio compartido para la imagen de Windows

Para hacer que la imagen ISO de Windows 11 sea accesible por cualquier usuario, debe hacerse modificando el fichero `smb.conf`, ubicado en el directorio `/etc/samba/`. Previo a la modificación del fichero, se debe crear el directorio base del servidor Samba y el subdirectorio que se desea compartir públicamente. Similar al caso del servicio TFTP, se crea un directorio llamado `samba` como directorio base, y un subdirectorio llamado `win11` con permisos de lectura y ejecución para todos los usuarios. Además, se debe cambiar el dueño y el grupo a los usuarios `nobody` y `nogroup` respectvamente, ya que son usuarios especiales con privilegios mínimos empleados para asignarlos a ficheros que no necesitan ser propiedad de un usuario real. Todos estos cambios se realizan con los siguientes comandos:

```
sudo mkdir -p /srv/samba
sudo mkdir -p /srv/samba/win11
sudo chown nobody:nogroup /srv/samba/win11
```

```
sudo chmod 0775 /srv/samba/win11
```

Una vez creado el directorio, se procede a modificar el fichero `smb.conf` para configurar tanto la forma de acceder al directorio, como para indicar al demonio que controla el servidor, que dicho directorio debe hacerlo accesible. Este fichero se encuentra en el directorio `/etc/samba/`. Las líneas que deben añadirse al final del fichero para este caso particular son las indicadas en el *script* 8.

```
[win11]
path = /srv/samba/win11
browsable = no
read only = yes
guest ok = no
```

*Script* 8: Fragmento del fichero `smb.conf`.

Tal y como muestra el fichero de configuración, el recurso compartido se llama win11. Este es el nombre lógico del recurso compartido, aunque coincide con el nombre del directorio dentro del servidor Samba que se ha creado anteriormente, para reducir la complejidad del sistema. Además, se indica la ruta completa del servidor, en este caso es /srv/samba/win11. Por último, se activa la opción llamada *read only* para que los usuarios que accedan no puedan modificar ningún fichero alojado allí, pero en cambio se desactivan las opciones *browsable* y *guest ok*. El objetivo principal de la primera opción desactivada es de hacer invisible el directorio cuando se navega por la red desde otro equipo, es decir, que no aparecerá listado al explorar el servidor Samba y que, por lo tanto, es necesario el conocimiento por parte del cliente del nombre del recurso y la ubicación exacta. Por otro lado, la segunda opción desactivada sirve para que el directorio no sea accesible sin que el usuario se autentique previamente.

Estas medidas de seguridad son necesarias, ya que evitan que posibles intrusos u otro tipo de usuarios accedan a los recursos compartidos en el servidor sin que realmente estén realizando el proceso de instalación desatendida de Windows 11.

Tras la configuración del directorio compartido, en el directorio `win11` se deben incluir todos los ficheros de la imagen de Windows 11 a compartir. Para ello, se debe montar en el servidor la imagen ISO, extraer los ficheros y copiarlos en la carpeta del servidor Samba. Junto a los ficheros extraídos se debe incluir además el fichero `Autounattend.xml` en el mismo directorio en donde se encuentre el instalador `setup.exe` de Windows, para que la instalación se realice de manera automática y desatendida.

Finalmente, una vez se han configurado todos los servidores y los servicios que van a prestar cada uno de ellos, se deben arrancar y poner en funcionamiento. Para las labores de arranque, reinicio y parada de los servicios del servidor se han empleado *scripts*, de manera que estas labores son mucho más rápidas a la hora de reiniciar el servidor o llevar a cabo labores de

mantenimiento. Estos ficheros cobran especial importancia a la hora de desarrollar el proyecto, ya que agilizan el proceso de mantenimiento del servidor. Además, a la hora de implementar una mejora o actualización de un servicio, es mucho más ágil y rápido con el apoyo de estos *scripts*. Los *scripts* utilizados son los *scripts* 9, 10 y 11.

```
systemctl start isc-dhcp-server
systemctl start apache2
systemctl start tftpd-hpa
sudo systemctl enable samba-ad-dc
sudo systemctl start samba-ad-dc
sudo systemctl stop smbd nmbd winbind
sudo systemctl disable smbd nmbd winbind
```

*Script* 9: Fichero auxiliar de arranque de todos los servicios llamado `startServices`.

En este primer *script* (*script* 9) se arrancan los cuatro servicios principales que se han configurado en este capítulo: `isc-dhcp-server`, `apache2`, `tftpd-hpa` y `samba-ad-dc`. Por otro lado, se deshabilitan y paran los servicios de Samba que no hacen falta y que podrían iniciarse al arrancar el servidor: `smbd`, `nmbd` y `winbind`.

```
systemctl restart isc-dhcp-server
systemctl restart samba-ad-dc
systemctl restart apache2
systemctl restart tftpd-hpa
```

*Script* 10: Fichero auxiliar de reinicio de todos los servicios llamado `resetServices`.

En este segundo *script* (*script* 10) se reinician todos los servicios anteriormente configurados. Este fichero es de utilidad cuando los servicios prestados no funcionan correctamente y se necesita una solución urgente.

```
systemctl stop isc-dhcp-server
systemctl stop samba-ad-dc
systemctl stop apache2
systemctl stop tftpd-hpa
```

*Script* 11: Fichero auxiliar de parada de todos los servicios llamado `stopServices`.

En este tercer y último *script* (*script* 11) se paran los cuatro servicios anteriormente configurados. Esto es de gran utilidad si se van a llevar a cabo grandes cambios en los servicios debido a labores de mantenimiento.

## 3.6. Discusión

La instalación y personalización de los diferentes servicios que va a prestar el servidor Debian del laboratorio es clave para el correcto funcionamiento del aplicativo.

Para todos los casos, exceptuando el servicio HTTP, se han personalizado con el objetivo de adecuarlos a las necesidades de la red. Si bien no se han modificado todos los ficheros de configuración de los servicios, si se han personalizado aquellos que son clave.

El servicio DHCP se ha adaptado para que otorgue toda la información de red adaptada a la arquitectura de red de la escuela. El servicio TFTP se ha configurado de manera que solamente escuche en una interfaz de red y no en todas las que tenga el servidor (configuración por defecto). El servicio HTTP no ha necesitado ninguna modificación a los ajustes por defecto, ya que Apache2 está pensado para llevar a cabo un despliegue de este servicio de manera sencilla y adaptada a la mayoría de las necesidades de los usuarios. Por otro lado, el servicio que más ha requerido de adaptación ha sido el servicio Samba AD-DC. Este servicio se ha modificado para separar de manera muy clara y evidente los tipos de usuarios que pueden acceder a las máquinas del laboratorio y a los recursos compartidos a los que pueden acceder dentro del directorio compartido.

# Capítulo 4

# Diseño de ficheros de automatización y personalización

En este capítulo se encuentra la segunda parte del desarrollo del aplicativo que tiene como objetivo esta memoria. En primer lugar, se presenta el fichero PXE que servirá para que el cliente pueda seleccionar y empezar con la instalación, para posteriormente describir el fichero de respuestas de Windows que permite la automatización de la instalación. En último lugar, se mencionan ciertos aspectos que se han hecho para personalizar la imagen de Windows a las necesidades del proyecto.

## 4.1. Ficheros de prearranque PXE

Cuando un cliente arranca por red, el cliente utiliza el protocolo PXE, el cual a su vez lo conforman los protocolos DHCP y protocolos de envío de ficheros como TFTP o HTTP. Para que el servidor pueda atender a las peticiones que lleva a cabo el cliente, ya se han configurado los servidores DHCP y TFTP. Sin embargo, para que el cliente pueda ejecutar ficheros y se pueda mostrar el menú interactivo por pantalla, se necesitan ficheros con un *software* de arranque de red.

Al recibir el servidor la petición del cliente de arranque por PXE, recibe además el tipo de *firmware* que tiene, es decir, si tiene un *firmware* UEFI o no. En caso de que el cliente tenga activado el modo UEFI, el servidor enviará el fichero binario ipxe.efi y, en caso contrario, el fichero ejecutable `ipxe.pxe`, ambos con las instrucciones a realizar.

El *software* específico para el arranque en red que se utiliza en este proyecto es el *software* de código abierto iPXE [31], debido a que es el *software* que ya se encuentra desplegado y cuyas

funcionalidades ya son conocidas por el administrador de la red. Cabe destacar que no se ha partido de cero, por lo que a lo largo de este capítulo se distinguirá aquellos ficheros que se han modificado de aquellos cuya configuración se ha mantenido igual.

iPXE es una versión avanzada y de código abierto del *firmware* PXE, que proporciona mayor flexibilidad y soporte para una amplia variedad de protocolos de red, incluidos HTTP, iSCSI [32], AoE [33] y FCoE [34], además de mejorar la funcionalidad del entorno. Gracias a estas capacidades, iPXE permite no solo la descarga eficiente de ficheros de arranque, sino también el acceso a discos remotos y la ejecución de *scripts* personalizados que permiten automatizar todo el proceso de arranque.

En primer lugar, se debe actualizar el *software* a la última versión, ya que así se evitan posibles errores, *bugs* y vulnerabilidades de versiones pasadas, al mismo tiempo que mejoran las prestaciones. El *software* se actualiza desde el directorio base a través de una petición al repositorio original en GitHub, haciendo uso de la herramienta git [35]:

```
git pull git://git.ipxe.org/ipxe.git
```

Actualizado el *software*, se descomentan las líneas necesarias en las cabeceras del fichero `console.h` (del subdirectorio `src/config/`) para habilitar las consolas convenientes para este proyecto. Las líneas a que hay que modificar en este caso son las que se muestran en el *script* 12.

```
[...]

#define CONSOLE_SERIAL      /* Serial port console */
#define CONSOLE_FRAMEBUFFER   /* Graphical framebuffer console */
#define CONSOLE_SYSLOG      /* Syslog console */
//#define CONSOLE_SYSLOGS   /* Encrypted syslog console */
//#define CONSOLE_VMWARE      /* VMware logfile console */
#define CONSOLE_DEBUGCON        /* Bochs/QEMU/KVM debug port console */
//#define CONSOLE_INT13   /* INT13 disk log console */
//#define CONSOLE_SBI     /* RISC-V SBI debug console */

[...]

#define KEYBOARD_MAP    es

[...]
```

*Script* 12: Fragmento del fichero `console.h`.

Tal y como se muestra en el *script* 12, se habilita el uso de distintos tipos de consolas disponibles para cada tipo de usuario. Además, se redefine el idioma del teclado del cliente para que no sea el teclado estadounidense, sino el teclado español.

A continuación, al igual que se ha hecho con el fichero anterior, se procede a modificar el fichero de cabecera de configuración `general.h` (del subdirectorio `src/config/` del proyecto IPXE). El objetivo es activar las mismas opciones de configuración que se encuntran en el fichero de la versión anterior del proyecto, descomentando para ello las líneas de código necesarias.

```
[...]

#define IMAGE_PNG     /* PNG image support */
#define IMAGE_DER     /* DER image support */
#define IMAGE_PEM     /* PEM image support */
#define IMAGE_EFISIG    /* EFI signature list image support */
#define IMAGE_ZLIB      /* ZLIB image support */
#define IMAGE_GZIP      /* GZIP image support */

[...]

#define AUTOBOOT_CMD        /* Automatic booting */
#define NVO_CMD           /* Non-volatile option storage commands */
#define CONFIG_CMD        /* Option configuration console */
#define IFMGMT_CMD        /* Interface management commands */
#define IWMGMT_CMD        /* Wireless interface management commands */
#define IBMGMT_CMD        /* Infiniband management commands */
#define FCMGMT_CMD        /* Fibre Channel management commands */
#define ROUTE_CMD       /* Routing table management commands */
#define IMAGE_CMD       /* Image management commands */
#define DHCP_CMD        /* DHCP management commands */
#define SANBOOT_CMD     /* SAN boot commands */
#define MENU_CMD          /* Menu commands */
#define LOGIN_CMD         /* Login command */
#define SYNC_CMD          /* Sync command */
#define SHELL_CMD         /* Shell command */
#define NSLOOKUP_CMD        /* DNS resolving command */
#define TIME_CMD          /* Time commands */
#define VLAN_CMD          /* VLAN commands */
#define REBOOT_CMD        /* Reboot command */
#define POWEROFF_CMD        /* Power off command */
#define PCI_CMD         /* PCI commands */
#define NEIGHBOUR_CMD       /* Neighbour management commands */
#define PING_CMD          /* Ping command */
#define CONSOLE_CMD       /* Console command */
#define IPSTAT_CMD        /* IP statistics commands */
#define NTP_CMD         /* NTP commands */
#define CERT_CMD          /* Certificate management commands */
#define IMAGE_ARCHIVE_CMD   /* Archive image management commands */

[...]
```

*Script* 13: Líneas descomentadas del fichero `general.h`.

Tal y como se puede observar en el *script* 13, se han habilitado una gran cantidad de comandos que pueden ser usados junto a iPXE. Sin embargo, no se han habilitado todos, ya que hay algunas

opciones o combinaciones no compilan al emplear UEFI. El resto de líneas de código se han mantenido iguales que el fichero original de iPXE. Algunos de los comandos que se ha decidido permitir son: comandos DHCP, comandos de login, comandos de configuración de la consola, comandos CMD, comandos de VLAN o comandos de reinicio y apagado.

Una vez ya estén configuradas ambas cabeceras del *software* que se va a usar en el *script*, se procede a la modificación del fichero fuente `ipxe.conf` para añadir al menú ya existente la opción que permitirá a los usuarios hacer uso de la funcionalidad planteada en este proyecto. Al fichero ya existente, se han modificado las siguientes líneas de código del *script* 14.

```
#!ipxe

[...]

item --gap -- --------------Instalaciones automaticas-------------------
item trixie Debian 13 Trixie
item bookworm Debian 12 Bookworm
item jammy Ubuntu jammy
item focal Ubuntu focal
item winpe Windows 7 (WinPE)
item win11 Windows 11                          # Opcion anadida al menu

[...]

:win11
kernel http://${next-server}/WinPE/wimboot
cpuid --ext 29 && set arch amd64
initrd http://${next-server}/WinPE/bootmgr          bootmgr
initrd http://${next-server}/WinPE/bootmgr.efi      bootmgr.efi
initrd http://${next-server}/WinPE/bootx64.efi      bootx64.efi
initrd http://${next-server}/WinPE/BCD            BCD
initrd http://${next-server}/WinPE/boot.sdi       boot.sdi
initrd http://${next-server}/WinPE/boot.wim       boot.wim
boot

[...]
```

*Script* 14: Fragmento del fichero `ipxe.conf`.

Tal y como se puede observar en el *script* 14, se ha añadido una única opción al menú de inicio y una nueva etiqueta con instrucciones a realizar por la máquina cliente al ser seleccionada.

Al administrador se le muestra un menú con diversas opciones de sistemas operativos. Entre ellas se encuentra la opción llamada "`Windows 11`" que, al ser seleccionada, el programa llama a la etiqueta llamada "`win11`". Esta etiqueta se encarga de elegir el tipo de *kernel* que va a utilizar, en este caso es el *kernel* de Windows especial llamado `wimboot` (ver sección 4.2.2) obtenido también del servidor. Este se encarga de seleccionar el tipo de arquitectura del sistema operativo y, por último, se encarga de realizar peticiones al servidor de los seis

ficheros fundamentales extraídos de la imagen Windows PE mediante el uso del protocolo HTTP. Los ficheros en cuestión son: `bootmgr`, `bootmgr.efi`, `bootx64.efi`, `BCD`, `boot.sdi`, `boot.wim`. Por último, con la instrucción `boot` se inicia el arranque del entorno WinPE como si fuera un sistema operativo local, utilizando `wimboot` y todos los ficheros anteriores.

Cabe destacar el uso del protocolo HTTP en esta segunda parte de la instalación, ya que el tamaño no trivial del fichero `boot.wim` crea la necesidad de emplear un protocolo de comunicaciones más veloz y eficiente que hasta el momento empleado TFTP.

El último paso en la creación de ficheros iPXE para el cliente consiste en la creación de un pequeño *script* llamado `embedded.ipxe` (ver *script* 15). Su función principal es descargar el *script* `ipxe.conf`, el cual contiene el menú de arranque con las opciones explicadas anteriormente, a través del protocolo TFTP. Este *script* va a ser embebido dentro del fichero ejecutable `ipxe.efi`, de forma que el cliente pueda automáticamente comenzar la descarga y ejecución del menú.

```
#!ipxe
dhcp
chain tftp://${next-server}/ipxe.conf
```

*Script* 15: Fichero de configuración `embedded.ipxe`.

Tras crear todos los ficheros de configuración, se deben compilar para así generar los binarios que van a ser ejecutados en la máquina del cliente, para así mostrar el menú de instalación. Para ello, en la línea de comandos del servidor desde el subdirectorio `src/` del proyecto iPXE se ejecuta la siguiente instrucción:

```
make bin-x86_64-efi/ipxe.efi EMBED=embedded.ipxe
```

Gracias a esta instrucción, se va a generar el fichero el cual es el que finalmente se va a enviar al cliente, llamado `ipxe.efi`. Dicho fichero se encuentra en el subdirectorio `src/bin-x86_64-efi/`, por lo que simplemente hay que copiarlo y moverlo al directorio compartido del servidor TFTP para que pueda ser encontrado. Al embeber el *script* `embedded.ipxe` dentro del fichero ipxe.efi, cada cambio que se produzca dentro del fichero de configuración `ipxe.conf` producirá que se actualice automáticamente, no siendo necesario recompilar.

Es importante recalcar que, debido a que el sistema operativo Windows no permite ser instalado en sistemas que tengan el *firmware* UEFI desactivado, solamente se procede a la compilación del nuevo programa para la generación del fichero `ipxe.efi`. El fichero adaptado para sistemas sin esta función no se compila con la versión de este proyecto, sino que se mantiene el menú de la versión anterior a este proyecto con las opciones preestablecidas, ya que es innecesario que se ofrezca una opción no disponible al usuario.

La máquina cliente, tras arrancar en con el modo UEFI activo, comienza la descarga del fichero `ipxe.efi` desde el servidor TFTP. Tras la descarga, la máquina lo ejecuta al instante, mostrando el menú de opciones. En la Figura 7 puede verse la opción dentro del menú de opciones que el cliente debe seleccionar para dar comienzo a la instalación desatendida.

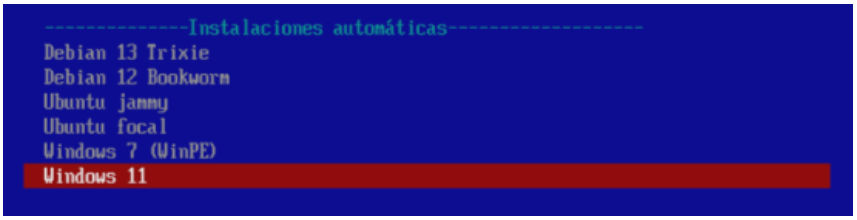


Figura 7: Opciones dentro del menú IPXE.

## 4.2. Ficheros de imagen Windows PE

Tras el arranque mediante PXE, se muestra al cliente el menú con diferentes opciones, estando entre ellas la opción “Windows 11”. Una vez seleccionada dicha opción por parte del administrador, se descargan mediante HTTP los ficheros de Windows PE para su posterior instalación.

Para que Windows 11 se instale en la máquina del cliente, debemos pasar en primer lugar un entorno ligero de Windows. Esta imagen ligera (WinPE) se descarga desde la página *web* oficial de Microsoft, al instalar las herramientas que proporciona Windows (Windows ADK) y la extensión “Windows Preinstallation Environment add-on” en una máquina Windows, ambas descritas en capítulos anteriores.

En las secciones siguientes se describen todo los pasos que se han seguido para actualizar la imagen Windows PE y personalizala con los drivers necesarios para las máquinas del laboratorio de investigación.

### 4.2.1. Actualización de la imagen

El primer paso para generar la imagen Windows PE es descargar las herramientas mencionadas en una máquina Windows de referencia, ya que no es posible descargarlas para un entorno Debian. Una vez descargadas, dentro de las que se proporcionan, se debe ejecutar como administrador la llamada “Deployment and Imaging Tools Environment”. La idea principal es crear una imagen Windows PE personalizada con los *drivers* de red necesarios, para que así sea compatible con los equipos de trabajo del laboratorio, ya que al ser una imagen ligera de Windows, no viene con todos los controladores compatibles instalados [36].

Esta aplicación tiene incorporadas una serie de herramientas en línea de comandos que van a ser de utilidad:

- **`Copype`:** *script* cuya función principal es crear un conjunto de ficheros de trabajo para construir una imagen personalizada de Windows PE.
- **`Dism` *(Deployment Image Servicing and Management)*:** se utiliza para preparar, modificar y reparar imágenes de Windows. Su función principal es montar y desmontar imágenes, y agregar y eliminar características, paquetes y controladores.
- **`Xcopy`:** se utiliza para copiar archivos y directorios de una ubicación a otra. Útil a la hora de personalizar la imagen al copiar elemementos externos.

Una vez ejecutada la herramienta como administrador (Figura 3), hay que actualizar la imagen Windows PE por defecto a la última versión, para que posteriormente la copia que se extraiga esté actualizada. Previamente, se debe crear un directorio de montaje, llamado `WinPE_amd64`, para seguidamente montar la imagen de referencia:

```
cd "..\Windows Preinstallation Environment\amd64"
md C:\WinPE_amd64\mount
Dism /Mount-Image /ImageFile:"en-us\winpe.wim" /index:1 /MountDir:"C:\
    WinPE_amd64\mount"
```

Tras montarla, se puede descargar la última versión desde la página *web* de Microsoft y posteriormente instalarla para actualizar la imagen `winpe.wim`. En este caso particular, está ya actualizada, por lo que no es necesario realizar este paso. Sin embargo, el motivo por el que se ha montado la imagen de referencia es debido a la necesidad de elegir el medio de arranque correcto para iniciar un dispositivo, ya que este debe ser compatible con el certificado Windows UEFI 2011 CA, contrarrestando así la vulnerabilidad publicada CVE-2023-24932 [37].

CVE-2023-24932 es una vulnerabilidad de seguridad en Windows que fue descubierta en 2023 y permite que un atacante desactive la función de seguridad llamada Secure Boot que, como se ha indicado anteriormente, es una función que ayuda a proteger la máquina durante el arranque. Para ello, se deben ejecutar los siguientes comandos:

```
Xcopy "C:\WinPE_amd64\mount\Windows\Boot\EFI\bootmgr.efi" "Media\bootmgr.
    efi" /Y
Xcopy "C:\WinPE_amd64\mount\Windows\Boot\EFI\bootmgfw.efi" "Media\EFI\Boot\
    bootx64.efi" /Y
```

Por último, tras comprobar de que se ha completado con éxito todo el proceso, se desmonta la imagen guardando los cambios con la opción `/Commit`:

```
Dism /Unmount-Image /MountDir:"C:\WinPE_amd64\mount" /commit
```

### 4.2.2. Personalización de la imagen

El siguiente paso a dar es llevar a cabo la personalización de la imagen de Windows PE, para así adaptarla a todas las necesidades del proyecto. Para esta tarea se utiliza la herramienta `Dism`. Para no modificar la original, se debe hacer una copia y montar dicha copia en un directorio a elegir:

```
Copype amd64 C:\WinPE
Dism /Mount-Image /ImageFile:C:\WinPE\media\sources\boot.wim /index:1 /
   MountDir:C:\WinPE\mount
```

La primera personalización que se realiza es la instalación de los controladores de red necesarios para que Windows pueda reconocer las tarjetas de red de las máquinas del laboratorio. Esto permitirá que el sistema almacene correctamente toda la información de red necesaria obtenida en la fase anterior, es decir, los datos proporcionados por el servidor DHCP, para que la máquina pueda comunicarase en red.

Esta instalación es de carácter obligatorio, ya que al ser WinPE un entorno ligero, es necesario agregar manualmente los controladores necesarios. Antes de comenzar, hay que averiguar cuáles son los modelos de las tarjetas de red de los ordenadores del laboratorio, ya que así es posible agregar sus controladores directamente.

Para tal propósito, ha de descargarse los *drivers* desde la página oficial del fabricante y guardarlos en una ubicación conocida una vez se saben los controladores que se necesitan. A continuación, ejecutar el siguiente comando para cada uno de los controladores:

```
Dism /Image:C:\WinPE\mount /Add-Driver /Driver:C:\Controladores\e1d.inf
```

En la línea anterior se muestra un ejemplo de cómo se ha añadido el controlador para el conector Ethernet de Intel I219-LM, pero se hace de igual manera con el resto que se necesita. Cabe destacar que es posible añadir todos los drivers de una sola instrucción con la opción `/Recursive`. Sin embargo, no es recomendable ya que tiende a fallar.

Posteriormente, se pueden añadir componentes opcionales a la imagen para, por ejemplo, cambiar el idioma. Para este caso particular carece de importancia cambiar el idioma, ya que es una imagen transitoria para llegar a obtener la imagen completa de Windows 11. En cambio, al llevar a cabo una instalación en red, Windows necesita dos componentes opcionales que deben ser instalados: `WinPE-WMI` y `WinPE-SecureStartup`:

```
Dism /Image:"C:\WinPE\mount" /Add-Package /PackagePath:"C:\Program Files (
   x86)\Windows Kits\10\Assessment and Deployment Kit\Windows
   Preinstallation Environment\amd64\WinPE_OCs\WinPE-WMI.cab"
Dism /Image:"C:\WinPE\mount" /Add-Package /PackagePath:"C:\Program Files (
   x86)\Windows Kits\10\Assessment and Deployment Kit\Windows
   Preinstallation Environment\amd64\WinPE_OCs\WinPE-SecureStartup.cab"
```

Ambos componentes son fundamentales y su ausencia puede derivar en un error en la instalación por red. El componente `WinPE-WMI` es el encargado de habilitar la automatización y la configuración avanzada, permitiendo la ejecución de *scripts* para configurar la red, acceder a recursos compartidos y preparar el sistema para la instalación de Windows 11. Por otro lado, el componente `WinPE-SecureStartup` agrega consideraciones de seguridad y compatibilidad, asegurando la integridad del entorno y preparando el sistema para las funcionalidades de seguridad del sistema operativo final [38].

Esta configuración asegura el correcto funcionamiento durante la instalación cuando se ha llevado a cabo a través de la red. A continuación, hay que automatizar el proceso de instalación de la imagen completa de Windows 11.

Cuando arranca Windows PE, se ejecuta el fichero `startnet.cmd`, el cual contiene por defecto una instrucción que inicializa el entorno de red y otras configuraciones básicas del sistema operativo. Aprovechando que este fichero se ejecuta automáticamente, antes de desmontar la imagen se ha de modificar con permisos de administrador dicho fichero. El objetivo principal es conectar la máquina cliente con el servidor Samba que contiene la imagen de Windows 11.

```
@echo off
wpeinit
net use Z: \\ServidorDebian12\win11 /user:joinPC 'Password123'
Z:
setup.exe
```

*Script* 16: Fichero `startnet.cmd` de Windows PE.

Tal y como puede observarse en el *script* 16, el sencillo programa en línea de comandos se ejecuta de manera silenciosa, ya que está la opción "@echo off" activa. A continuación, con la instrucción `wpeinit` se inicializa el entorno de Windows PE, la carga de controladores anteriormente agregados y la preparación del entorno. Finalmente, monta el directorio activo en red disponible del servidor Samba en el disco local, al cual le asigna la letra Z. El programa busca en red el directorio compartido `win11` de la máquina `ServidorDebian12`, cuyo nombre es sustituido por su dirección IP tras la resolución DNS. Para acceder al directorio activo del servidor, se introducen automáticamente las credenciales del usuario especial `joinPC`, el cual es el usuario creado para este propósito. Una vez montado, ejecuta el instalador de Windows, comenzando así la siguiente etapa del proceso de instalación desatendida.

Una vez ya se ha finalizado la instalación de todas las personalizaciones, tanto de los controladores como de los componentes `WinPE-WMI` y `WinPE-SecureStartup` y la automatización del sistema, el proceso de creación de la imagen Windows PE finaliza, por lo que se debe desmontar la imagen con la opción `/Commit` activa para guardar los cambios:

```
Dism /Unmount-Image /MountDir:C:\WinPE\mount /Commit
```

Desmontada la imagen y guardados todos los cambios, hay que mover los ficheros esenciales de la imagen para poder instalar Windows PE en el cliente, ya que son aquellos que va a solicitar. Los ficheros se encuentran en los siguientes directorios de la imagen Windows PE:

- **`/media/`:** Se encuentran los ficheros `bootgmr` y `bootgmr.efi`.
- **`/media/Boot/`:** Se encuentran los ficheros `BCD` y `boot.sdi`.
- **`/media/sources/`:** Se encuentra el fichero `boot.wim`.
- **`/media/EFI/Boot/`:** Se encuentra el fichero `bootx64.efi`.

Durante el arranque vía PXE, y tras la descarga de estos ficheros, se ejecuta el fichero de arranque `boot.wim`, posteriormente se monta el fichero compartido y, posteriormente, se ejecuta el instalador de Windows (`setup.exe`).

Junto a estos ficheros, es necesario añadir uno adicional que permita el arranque y la ejecución de ficheros con extensión "`.wim`" comprimidos sin necesidad de instalar todos los paquetes. Este fichero se llama `WimBoot`. `WimBoot` (*Windows Image File Boot*) es una característica introducida en Windows 8.1 cuya funcionalidad es que, en lugar de instalar los ficheros de Windows de la manera tradicional, `WimBoot` mantiene el sistema operativo dentro de un fichero WIM, lo que puede ahorrar espacio en disco. El sistema accede a los ficheros dentro del fichero WIM según sea necesario [39].

Este fichero es fundamental ya que permite minimizar la cantidad de ficheros que el servidor debe enviar al cliente para comenzar la ejecución de Windows PE.

## 4.3. Fichero de respuestas `Autounattend.xml`

Windows 11 puede ser instalado tanto de manera manual, respondiendo de manera interactiva a todas las ventanas emergentes que se van sucediendo a lo largo del proceso de instalación con la información solicitada, como de manera automática. Tal y como se ha comentado en secciones anteriores, para que la instalación se haga de manera automática, se debe configurar un fichero

XML de respuestas llamado `Autounattend.xml`. Este fichero contiene las respuestas a las preguntas que plantea el sistema operativo durante su instalación si esta fuese manual, por lo que evita que el administrador tenga que darles respuesta. Además, reduce la complejidad a la hora de configurar las máquinas, ya que se puede conseguir la misma configuración en una gran cantidad de dispositivos, siendo esta parte transparente para el usuario final.

Llevar a cabo la creación del fichero de respuestas es posible hacerlo de dos maneras distintas: la primera de ellas consiste en comprender la estructura y las funciones habituales de este tipo de ficheros XML para poder programarlo, y la segunda consiste en emplear una de las herramientas del set Windows ADK que proporciona de manera gratuita Windows para desarrolladores llamada "Windows System Image Manager". Gracias al uso de esta herramienta es posible crear el fichero desde cero, incluyendo los componentes uno a uno a partir de una imagen ISO de Windows 11 de referencia. Estos componentes dan respuesta a todas las necesidades del instalador.

Antes de comenzar, con la herramienta que proporciona Windows se puede crear el nuevo fichero al seleccionar *fichero >Nuevo fichero de respuesta...*, como se ve en la figura 8.

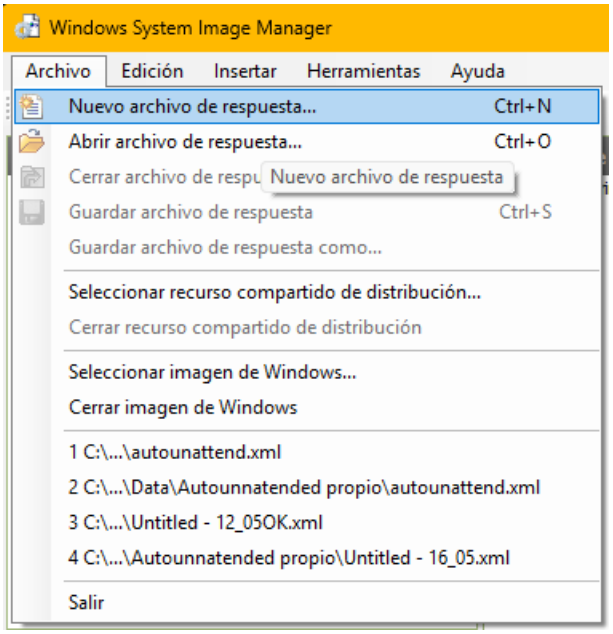


Figura 8: Menú para creación de un fichero de respuestas.

Una vez creado el fichero, se observan las siete fases de la instalación de Windows 11, explicadas en la sección 3.1. A la hora de plantear una correcta configuración del fichero de respuestas, es importante averiguar cuáles son las configuraciones obligatorias que el instalador de Windows necesita para completar todo el proceso sin necesidad de solicitar más datos. Por otro lado, existen configuraciones opcionales que pueden realizase a lo largo de la instalación.

Las necesidades obligatorias que necesita el instalador vienen definidas por Microsoft [40]. Tras una extensa consulta y documentación, para el proyecto que acontece, solamente es nece-

sario llevar a cabo la configuración de tres de las siete fases posibles: la fase `WindowsPE`, la fase `Specialize` y la fase `oobeSystem`. La configuración de cada fase se describe en las subsecciones siguientes.

### 4.3.1. Fase `WindowsPE`

La primera de las siete fases de las que forman el proceso de instalación es la fase llamada `windowsPE`. Durante esta fase se configuran aspectos relacionados con la configuración del *hardware*, como el formateo del disco duro y creación de particiones. También se gestiona la introducción de la clave del producto.

En primer lugar, se debe definir el idioma con el que Windows va a comunicarse con el usuario. Aunque el usuario no interaccione con el instalador, este va a ir mostrando mensajes a medida que va a ir avanzando con el proceso. Haciendo uso del componente de Windows `Microsoft-Windows-International-Core-WinPE`, es posible indicar que el idioma que debe emplear es el español de España, es decir, las variables del componente deben tomar los valores `es-ES` (ver *script* 17).

```
<component name="Microsoft-Windows-International-Core-WinPE"
   processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language
   ="neutral" versionScope="nonSxS" xmlns:wcm="http://schemas.microsoft.com
   /WMIConfig/2002/State" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance">
    <SetupUILanguage>
        <UILanguage>es-ES</UILanguage>
    </SetupUILanguage>
    <UserLocale>es-ES</UserLocale>
    <UILanguage>es-ES</UILanguage>
    <SystemLocale>es-ES</SystemLocale>
    <InputLocale>0c0a:0000040a</InputLocale>
</component>
```

*Script* 17: Configuración del idioma de la fase WindowsPE del fichero `Autounattend.xml`.

Cabe destacar que al incluir cada componente se indica tanto la arquitectura del procesador de la máquina en donde se va a instalar como el idioma que emplea. Esto permite una instalación precisa y adaptada al *hardware* de la máquina objetivo, así como la preferencia del idioma. En este caso, se incluyen los valores `amd64` (es el tipo de arquitectura que deben tener las máquinas que deseen instalar Windows 11) y `neutral` (para que sea independiente del idioma destino) respectivamente.

Tras seleccionar el idioma, el siguiente paso a dar es configurar el disco duro de la máquina, haciendo uso del componente `Microsoft-Windows-Setup`. Suponiendo que la máquina

cliente solamente va a constar de un disco duro con suficiente espacio de almacenamiento, es decir, de mínimo 64 GB, tal y como indican los requisitos de Windows 11, se procede a primero formatearlo para posteriormente particionar el disco en cuatro. Dichas particiones se observan en la tabla 1.

| ID | Letra | Nombre | Tamaño (MB) | Tipo | Formato |
|---|---|---|---|---|---|
| 1 | | Recovery | 1024 | Primary | NTFS |
| 2 | | System | 300 | EFI | FAT32 |
| 3 | C | Windows | resto | Primary | NTFS |
| 4 | | | 16 | MSR | |

Tabla 1: Particiones del disco duro de la máquina cliente de Windows 11.

Todos los tamaños de las particiones se han elegido basándose en los requisitos que exige Windows para cada una de las particiones. Para no dividir las particiones con el tamaño mínimo exigido, se ha decidido añadir un pequeño margen por si estos tamaños aumentan en futuras actualizaciones. Los tamaños indicados por Windows son:

- 300 MB mínimo para la partición de recuperación (*Recovery*).
- 200 MB mínimo para la partición EFI (*System*).
- 20 GB mínimo para la partición de Windows.
- 16 MB para la partición MSR (*Microsoft Reserved Partition*) [41].

Estas particiones son fundamentales para el correcto funcionamiento del sistema ya que cada una de ellas tienen un rol específico. La partición de recuperación (*Recovery*) contiene el entorno de recuperación de Windows utiliza para solucionar problemas, restaurar el sistema a un punto anterior o restablecer de fábrica el dispositivo. La partición EFI (*System*) contiene los ficheros necesarios para que la interfaz UEFI pueda arrancar el sistema operativo. La partición reservada por Windows (MSR) está vacía y reservada para futuras funcionalidades. Por último, la partición principal (Windows) es dónde se instala el sistema operativo Windows y sus aplicaciones.

Aunque existan etiquetas de configuración dentro del componente `Microsoft-Windows-Setup` específicas para llevar a cabo esta operación, como por ejemplo las etiquetas `DiskConfiguration`, `Disk`, `CreatePartition` o `ModifyPartition`; es preferible emplear una serie de *scripts* síncronos en la línea de comandos de Windows CMD. Esto es así debido a que estos *scripts* permiten lógica condicional, mejor depuración y mayor adaptabilidad a diversos escenarios de *hardware*, siendo menos probable que falle la selección de la partición durante la instalación.

A través del uso de las etiquetas `RunSynchronous` y `RunSynchronousCommand` se pueden incorporar *scripts* personalizados durante esta primera fase de la instalación. Estos *scripts*

van a utilizar `diskpart.exe`, una herramienta de línea de comandos de Windows preparada para realizar el particionado de discos. La configuración de los comandos para la creación de las particiones se observa en el *script* 18.

```
<RunSynchronous>
     <RunSynchronousCommand wcm:action="add">
        <Order>1</Order>
        <Path>cmd.exe /c ">>"X:\gestDisco.txt" (echo SELECT DISK=0&echo
            CLEAN&echo CONVERT GPT&echo CREATE PARTITION EFI SIZE=300&echo
            FORMAT QUICK FS=FAT32 LABEL="System"&echo CREATE PARTITION MSR
            SIZE=16)"</Path>
    </RunSynchronousCommand>
    <RunSynchronousCommand wcm:action="add">
        <Order>2</Order>
        <Path>cmd.exe /c ">>"X:\gestDisco.txt" (echo CREATE PARTITION
            PRIMARY&echo SHRINK MINIMUM=1024&echo FORMAT QUICK FS=NTFS LABEL
            ="Windows"&echo CREATE PARTITION PRIMARY&echo FORMAT QUICK FS=
            NTFS LABEL="Recovery")"</Path>
    </RunSynchronousCommand>
    <RunSynchronousCommand wcm:action="add">
        <Order>3</Order>
        <Path>cmd.exe /c ">>"X:\gestDisco.txt" (echo SET ID="de94bba4-06d1
            -4d40-a16a-bfd50179d6ac"&echo GPT ATTRIBUTES=0x8000000000000001)
            "</Path>
    </RunSynchronousCommand>
    <RunSynchronousCommand wcm:action="add">
        <Order>4</Order>
        <Path>cmd.exe /c "diskpart.exe /s "X:\gestDisco.txt" >>"X:\diskpart
            .log" || ( type "X:\diskpart.log" & echo diskpart encountered an
             error. & pause & exit /b 1 )"</Path>
    </RunSynchronousCommand>

    [...]

<RunSynchronous>
```

*Script* 18: Primeros cuatro comandos de la fase WindowsPE del fichero `Autounattend.xml`.

El *script* 18 tiene como objetivo mostrar los cuatro primeros *scripts* síncronos que se ejecutan una vez inicializada la primera fase, ejecutados en secuencia según el orden. La lógica que siguen los *scripts* es utilizar la línea de comandos de Windows CMD para crear un fichero llamado `gestDisco.txt` en la ruta `X:`
(partición de instalación temporal), el cual contiene los comandos que `DiskPart` ejecutará posteriormente. Con la opción `/c`, el CMD se abre para posteriormente cerrarse de manera automática, tras terminar de ejecutar las intrucciones indicadas:

1. En el primer bloque de comandos se selecciona el disco cero, se eliminan todas las particiones existentes con la función CLEAN y se convierte a disco en formato GPT (*GUID Partition Table*). A continuación, se crea una partición EFI de 300 MB, se formatea

dicha partición en formato FAT32 y, por último, se crea la partición MSR (*Microsoft Reserved*) de 16 MB de tamaño. Los IDs de las particiones son 1 y 2 respectivamente.

2. En el segundo comando se definen dos particiones adicionales. Se crea la partición principal con el resto del tamaño del disco duro, de tipo *primary*, en donde se va a instalar Windows, se reduce en 1 GB el tamaño final, se formatea en formato NTFS y se le añade a esa partición la etiqueta "Windows". Con los 1024 MB libres restantes se crea la partición de recuperación, de tipo *primary*, se formatea también en formato NTFS y se le asigna la etiqueta "*Recovery*".
3. A la última partición se le agrega un ID establecido por Windows para que se detecte como partición de recuperación [41]. Además, se establecen atributos GPT especiales para marcarla como una partición protegida del sistema (por ejemplo, hace que la partición sea invisible para el usuario y que se considere como una partición crítica del sistema).
4. Se ejecuta el fichero generado `gestDisco.txt` utilizando `Diskpart` con la opción `/s` activa (sin esta opción `Diskpart` no puede ejecutarse desde un fichero de texto). El resultado se almacena en un *log* llamado `diskpart.log` y, si se detecta un error, se muestra el contenido del *log*, se pausa la instalación y se aborta el proceso, permitiendo una depuración segura.

```
<RunSynchronous>

    [...]

    <RunSynchronousCommand wcm:action="add">
        <Order>5</Order>
        <Path>reg.exe add &quot;HKLM\SYSTEM\Setup\LabConfig&quot; /v
            BypassTPMCheck /t REG_DWORD /d 1 /f</Path>
    </RunSynchronousCommand>
    <RunSynchronousCommand wcm:action="add">
        <Order>6</Order>
        <Path>reg.exe add &quot;HKLM\SYSTEM\Setup\LabConfig&quot; /v
            BypassSecureBootCheck /t REG_DWORD /d 1 /f</Path>
    </RunSynchronousCommand>
    <RunSynchronousCommand wcm:action="add">
        <Order>7</Order>
        <Path>reg.exe add &quot;HKLM\SYSTEM\Setup\LabConfig&quot; /
            BypassRAMCheck /t REG_DWORD /d 1 /f</Path>
    </RunSynchronousCommand>
</RunSynchronous>
```

*Script* 19: Últimos tres comandos de la fase WindowsPE del fichero `Autounattend.xml`.

Con la creación de las particiones llevada a cabo de manera exitosa, se avanza en el proceso de instalación. El siguiente paso que realiza Windows es llevar a cabo la comprobación de los tres requisitos fundamentales que debe tener el ordenador: TPM 2.0, *Secure Boot* activo y tamaño de RAM suficiente (mínimo 4 GB). Debido a que estas comprobaciones pueden dar

falsas incompatibilidades, se ha decidido hacer un *bypass* de estas reglas de seguridad editando los registros necesarios (ver *script* 19).

El primer comando de los tres que se ejecutan crea un valor de registro llamado `BypassTPMCheck` dentro de la clave `HKLM\SYSTEM\Setup\LabConfig`. Este valor hace que se omita la comprobación del TPM. Sucede lo mismo con la comprobación tanto del arranque seguro (segundo comando) y con la RAM (tercer comando).

### 4.3.2. Fase `Specialize`

La fase `Specialize` se ejecuta inmediatamente después de aplicar la imagen del sistema operativo sobre el *hardware* de la máquina cliente y antes de que el sistema complete su preparación para la fase `oobeSystem` (última fase de la instalación de Windows). Esta etapa se caracteriza por configurar elementos que deben ser únicos y no replicables entre diferentes instalaciones.

La fase `Specialize` tiene la capacidad de unir un equipo automáticamente a un dominio de Directorio Activo. Esta operación se realiza antes de que el sistema operativo se inicie por primera vez y es posible haciendo uso del componente `Microsoft-Windows-UnattendedJoin`, incluido en el fichero de respuestas.

```
<settings pass="specialize">
    <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="
       amd64" publicKeyToken="31bf3856ad364e35" language="neutral"
       versionScope="nonSxS" xmlns:wcm="http://schemas.microsoft.com/
       WMIConfig/2002/State" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance">
        <ComputerName>*</ComputerName>
    </component>
    <component name="Microsoft-Windows-UnattendedJoin"
       processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
       language="neutral" versionScope="nonSxS" xmlns:wcm="http://schemas.
       microsoft.com/WMIConfig/2002/State" xmlns:xsi="http://www.w3.org
       /2001/XMLSchema-instance">
        <Identification>
            <Credentials>
                <Password>Password123</Password>
                <Username>joinPC</Username>
                <Domain>lab25.local</Domain>
            </Credentials>
            <JoinDomain>lab25.local</JoinDomain>
        </Identification>
    </component>
</settings>
```

*Script* 20: Configuración de la fase `Specialize` del fichero `Autounattend.xml`.

Según se observa en el *script* 20, gracias al uso de las etiquetas `Identification`, Credentials y `Joindomain` es posible introducir todos los valores del dominio necesarios para establecer la unión de la máquina. En los valores de la etiqueta `Credentials`, se incluyen las credenciales del usuario `joinPC` que, tal y como se ha indicado anteriormente, es el usuario que se ha creado únicamente con este fin, junto al nombre del dominio completo objetivo. La etiqueta `JoinDomain` sirve para especificar de nuevo el nombre del dominio.

Aparte de incluir las credenciales, para que una máquina pueda unirse a un dominio, debe tener un nombre asignado para poder ser localizada y asociada a un usuario del dominio. Normalmente, Windows asigna el nombre de la máquina en la última fase (fase `oobeSystem`). Esto entra en conflicto con el objetivo de esta fase, ya que la unión debe hacerse en esta fase de la instalación. Para lograrlo, se fuerza a Windows a asignar en esta fase un nombre a la máquina con el componente `Microsoft-Windows-Shell-Setup` y cambiando el valor de la variable `ComputerName`, siendo en este caso elegido por Windows de manera aleatoria (*).

### 4.3.3. Fase `oobeSystem`

La fase `oobeSystem` es la última de las siete etapas de la instalación de Windows y se ejecuta inmediatamente después de la instalación principal del sistema, pero antes de la primera experiencia del usuario conocida como OOBE. En esta fase se lleva a cabo la preparación final del entorno antes de presentar al usuario su escritorio del sistema operativo por primera vez. Estas configuraciones serán visibles o efectivas una vez que el usuario inicie sesión por primera vez.

En esta fase de configuración, y al igual que se ha llevado a cabo en la fase windowsPE, se puede configurar el idioma tanto del sistema como del teclado a utilizar. En este caso, con el componente `Microsoft-Windows-International-Core` se puede incluir esta configuración dando el valor `es-ES` a todas las variables que lo conforman (ver *script* 21).

En esta fase de configuración, si la instalación fuese hecha de manera manual, Windows mostraría una gran cantidad de ventanas emergentes relacionadas con la configuración de políticas de privacidad, configuración de comunicaciones Wi-Fi, envío de datos voluntarios a Microsoft... Esto requiere interacción del usuario con el instalador, cosa que se debe evitar, ya que la instalación completa del sistema operativo debe ser automática. Gracias a la etiqueta `OOBE` del componente `Microsoft-Windows-Shell-Setup` es posible preconfigurar las respuestas a dichas ventanas emergentes, logrando así el objetivo propuesto. Adicionalmente, se evita la creación de cuentas locales, evitando así que acceda personal fuera del dominio.

Por otro lado, se configuran con etiquetas adicionales tanto el nombre de la organización (Universidad de Valladolid) y como la zona horaria (W. Europe Standard Time).

Una vez finaliza esta última fase, todo el proceso de instalación de Windows termina con éxito, logrando el objetivo propuesto. En ese momento, se solicita al cliente final las credenciales de usuario, es decir, el nombre de usuario del dominio asignado a esa persona física y su contraseña.

```
<settings pass="oobeSystem">
    <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="
       amd64" publicKeyToken="31bf3856ad364e35" language="neutral"
       versionScope="nonSxS" xmlns:wcm="http://schemas.microsoft.com/
       WMIConfig/2002/State" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance">
        <OOBE>
            <HideEULAPage>true</HideEULAPage>
            <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>
            <ProtectYourPC>3</ProtectYourPC>
            <HideOnlineAccountScreens>true</HideOnlineAccountScreens>
            <HideOEMRegistrationScreen>true</HideOEMRegistrationScreen>
            <HideLocalAccountScreen>true</HideLocalAccountScreen>
            <NetworkLocation>Work</NetworkLocation>
            <SkipMachineOOBE>true</SkipMachineOOBE>
            <SkipUserOOBE>true</SkipUserOOBE>
        </OOBE>
        <RegisteredOrganization>Universidad de Valladolid</
           RegisteredOrganization>
        <TimeZone>W. Europe Standard Time</TimeZone>
    </component>
    <component name="Microsoft-Windows-International-Core"
       processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
       language="neutral" versionScope="nonSxS" xmlns:wcm="http://schemas.
       microsoft.com/WMIConfig/2002/State" xmlns:xsi="http://www.w3.org
       /2001/XMLSchema-instance">
        <InputLocale>es-ES</InputLocale>
        <SystemLocale>es-ES</SystemLocale>
        <UILanguage>es-ES</UILanguage>
        <UserLocale>es-ES</UserLocale>
    </component>
</settings>
```

*Script* 21: Configuración de la fase oobeSystem del fichero `Autounattend.xml`.

## 4.4. Personalización de la imagen ISO de Windows 11

En esta sección se explica la creación de una imagen ISO personalizada y generalizada del sistema operativo Windows 11, que incluya aplicaciones de uso común entre el personal de investigación, como Matlab o Firefox, orientada a su posterior despliegue en los dispositivos del laboratorio. Esta personalización es el último paso del desarrollo del aplicativo.

Se identificaron un conjunto de herramientas comunes cuya instalación por defecto resulta beneficiosa para el conjunto de usuarios. De este modo, se optó por incorporar esas aplicacio-

nes en una imagen generalizada que pueda implementarse a la hora de realizar la instalación desatendida.

El primer paso que hay que dar para llevar a cabo esta labor es instalar en un entorno controlado el sistema operativo Windows 11 Education, en este caso, en una máquina virtual en el entorno de simulación anteriormente descrito (VMWare). Una vez completada la instalación del sistema operativo de manera exitosa, se instalan y configuran todas las aplicaciones previamente seleccionadas para formar parte de la imagen final. Además, es posible personalizar el entorno de Windows, como por ejemplo el fondo de pantalla.

Con el sistema completamente configurado, se procede a generar la imagen personalizada. Este proceso se denomina generalización de la instalación y se emplea la herramienta incluida en Windows llamada Sysprep.

La generalización consiste en eliminar toda la información específica del *hardware* del equipo en donde está instalado el sistema operativo, como identificadores únicos o controladores, que puedan provocar conflictos al momento del despliegue de la imagen en otras máquinas. Asimismo, se configura el sistema para que, al iniciarse por primera vez tras el despliegue, se ejecute la fase OOBE, permitiendo al sistema utilizar la configuración del fichero de respuestas para la configuración de los últimos parámetros.

Desde la máquina virtual con Windows 11 instalado, se debe ejecutar la consola de comandos CMD como administrador y acceder al directorio donde se encuentra el programa Sysprep.:

```
cd C:\Windows\System32\Sysprep
```

Una vez en el directorio, se debe ejecutar el programa con las siguientes opciones:

```
sysprep /generalize /oobe /shutdown
```

Este comando generaliza la instalación, la deja lista para su despliegue en otros equipos y apaga automáticamente la máquina una vez finalizado el proceso.

Una vez que la máquina se apaga tras la ejecución de Sysprep, se inicia la fase de captura de la imagen. Para ello, se arranca el sistema desde un medio externo USB/DVD con Windows PE. Desde Windows PE, se emplea la herramienta Diskpart para identificar las particiones y determinar la letra asignada a la unidad que contiene la instalación de Windows generalizada. A continuación, se utiliza la herramienta `Dism` para capturar la partición en un fichero de imagen con extensión "`.wim`" en un medio externo:

```
Dism /capture-image /imagefile:D:\install.wim /capturedir:C:\ /name:"
   Windows con MATLAB y Firefox" /compress:maximum
```

Finalmente, se procede a integrar el fichero `install.wim` generado en la imagen ISO que se encuentra disponible en el servidor, de manera que se sustituye el fichero `install.wim` de la carpeta `sources` por el fichero extraído.

## 4.5. Discusión

Más allá de llevar a cabo una simple implementación de una instalación desatendida de Windows 11 se ha personalizado todo el aplicativo. La propuesta de valor de este proyecto se centra en la optimización de los procesos de desarga de ficheros y la personalización adaptada a las necesidades de los usuarios. En este sentido, la modificación de los ficheros de configuración PXE permite generar las instrucciones de descarga de solamente los ficheros de imagen fundamentales para su ejecución.

Sin embargo, el fichero de respuestas `Autounattend.xml` es el elemento diferencial. Gracias al uso de este fichero se configuran todos los elementos obligatorios de una instalación de manera que el instalador haga su trabajo lo más rápido posible, al mismo tiempo que se elimina por completo la intervención del usuario en es proceso de instalación.

Finalmente, la personalización de la imagen ISO de Windows 11 permite integrar aplicaciones específicas para el uso del personal del laboratorio, ofreciendo un entorno de trabajo preparado para ser utilizado posteriormente a la instalación.

# Capítulo 5

# Simulación y despliegue

En este capítulo se encuentran los resultados del funcionamiento del aplicativo y las pruebas que se han llevado a cabo hasta la finalización del proyecto. Además, se muestran las fases del despliegue del proyecto y los requisitos específicos de cada caso.

## 5.1. Pruebas en entorno simulado VMWare

La primera parte del proyecto se ha desarrollado en un entorno simulado, haciendo uso de la herramienta VMWare Workstation Pro. En esta herramienta se ha replicado el servidor físico original, con el objetivo de hacer pruebas de despliegue sin afectar al funcionamiento de los servicios de la red activos hasya el momento.

Tras replicar el servidor en una máquina virtual, se han seguido los pasos de los capítulos 3 y 4 para conseguir el despliegue del aplicativo. Configurado el servidor, se ha creado otra máquina virtual con los requisitos mínimos de *hardware* que necesita Windows 11 para ser instalado y funcionar sin errores. Cabe destacar que se ha creado vacía, es decir, sin un SO instalado, y configurada para que arranque prioritariamente por red.

Tras configurar ambas máquinas, se define una red interna a la que se van a conectar tanto el servidor como la máquina cliente. Dicha red interna se trata de una red virtual dedicada exclusivamente a la comunicación entre ambas máquinas.

Preparada toda la red, comienza el proceso de simulación. El cliente, al arrancar mediante PXE con el modo UEFI activo, solicita al servidor el fichero EFI necesario para seleccionar dentro del menú presentado la opción “Windows 11”, visible en la Figura 9. Al seleccionar dicha opción pulsando la tecla *Enter*, finaliza toda interacción del usuario con el sistema y comienza la

instalación desatendida.

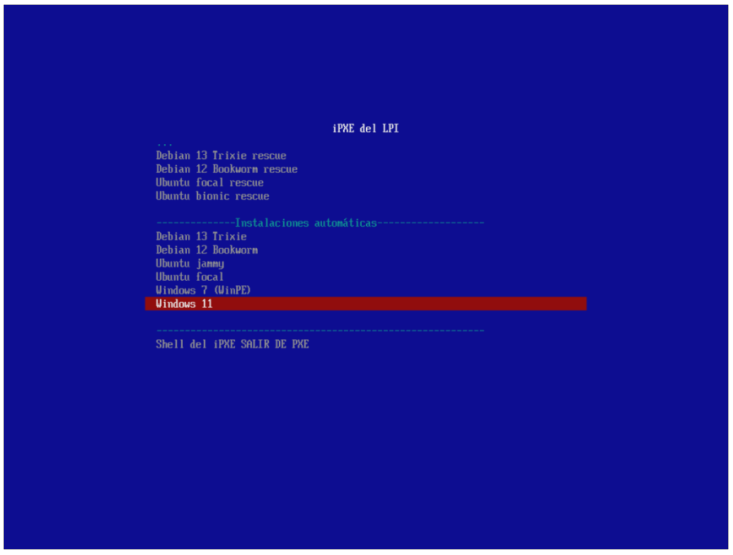


Figura 9: Menú iPXE con distintas opciones de instalación.

En ese mismo momento comienza la descarga de los seis ficheros necesarios para poder ejecutar Windows PE en el ordenador, tal y como puede observarse en la Figura 10. Tras la descarga, se ejecuta el fichero `boot.wim` y se muestra por pantalla una consola de comandos con el fondo de color azul. Es decir, el *software* Windows PE, se está ejecutando correctamente.


Figura 10: Descarga de los ficheros de Windows PE.

De manera transparente para el usuario, Windows PE intenta establecer conexión con el servidor Samba y acceder al directorio compartido en donde se encuentra la imagen ISO de Windows 11. Tras establecer conexión y montar el Directorio Activo con la letra Z:, Windows PE ejecuta el instalador de Windows 11 (`setup.exe`) para comenzar la instalación (figura 11).

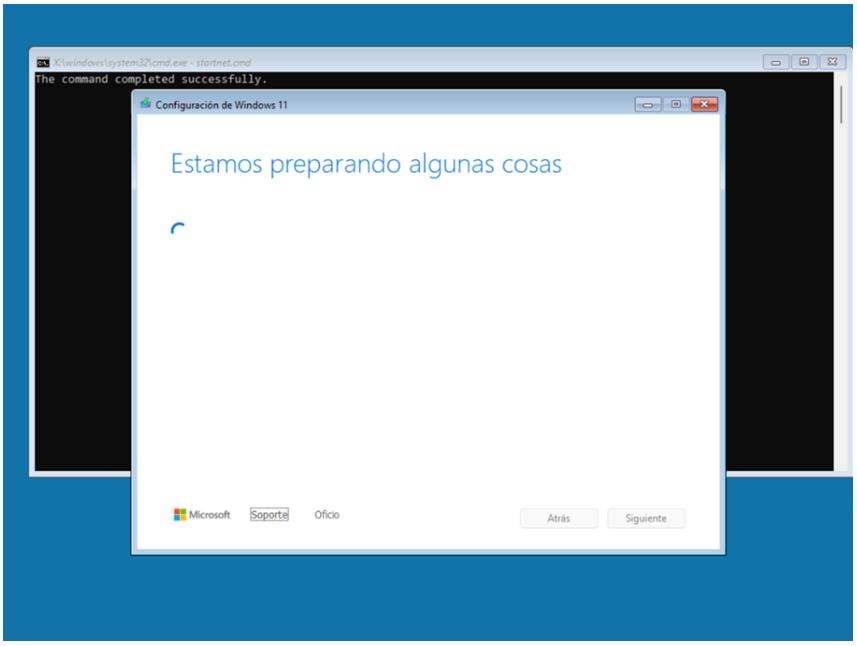


Figura 11: Primera fase de instalación de Windows 11.


Figura 12: Segunda fase de instalación de Windows 11.

Cuando el proceso de instalación comienza, se ejecutan los siete primeros *scripts* que se encargan de gestionar el disco duro y sus particiones, y de llevar a cabo los *bypass* de las comprobaciones de Windows. Además, se inyecta la licencia de uso, por lo que cuando el sistema comprueba la veracidad de lo anteriormente hecho, pasa a la siguiente fase.

La siguiente pantalla que se muestra es una pantalla azul (Figura 12), típica de un proceso de

instalación de Windows, en donde de manera totalmente transparente para el usuario registra el PC en el dominio de la red de laboratorio. Posteriormente, hace el primero de los reinicios.

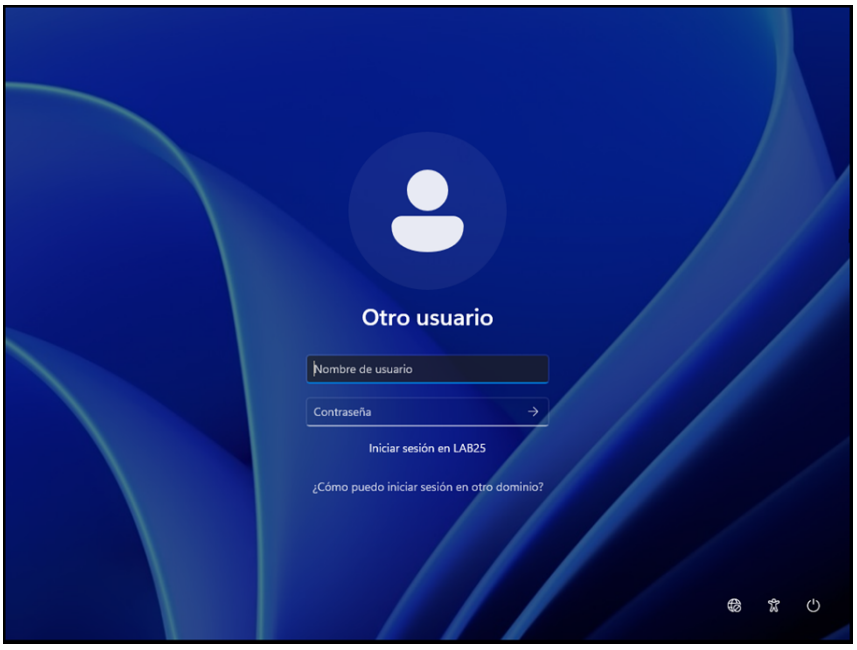


Figura 13: Menú de inicio de sesión en la máquina cliente.

Tras el primer reinicio se muestra la última ventana del proceso de instalación, donde en una situación normal el usuario configura su ordenador. Durante este proceso se configura el idioma tanto del teclado como del sistema operativo y la zona horaria en la que se encuentra.

Finalmente, si la instalación se ha completado con éxito y si el usuario tiene credenciales de acceso, se muestra la ventana de inicio de sesión, solamente estando disponible el acceso para los usuarios registrados en el dominio (figura 13).

## 5.2. Despliegue del servicio

Como en cualquier proyecto, antes de un despliegue general, deben llevarse a cabo pruebas y comprobaciones sobre el aplicativo de manera que se encuentren fallos y vulnerabilidades. Tras llevar a cabo dichas pruebas en un entorno controlado y simulado, la siguiente fase es desplegarlo.

Antes de replicar el servidor con las nuevas funcionalidades en la red real, se dispone de un PC en el laboratorio con el cual se han de hacer pruebas, al conectarlo directamente con un cable ethernet. Una vez conectado, se debe repetir el proceso de la sección 5.1, pero con un cambio fundamental antes del comienzo: al arrancar el equipo se debe acceder a la BIOS del sistema ya

que, como se ha mencionado anteriormente, se debe invertir el orden de preferencia del modo de arranque de la máquina, poniendo en primer lugar el arranque por red (figura 14). Tras guardar los cambios y reiniciar, comienza el funcionamiento del aplicativo.

Es de suma importancia que se cambie el orden de preferencia de arranque, no que se elimine de las opciones posibles el arranque por disco duro debido a que el instalador de Windows, durante la instalación, reinicia varias veces la máquina y, para que funcione, cambia de nuevo el orden. Si el usuario elimina esta opción, el instalador, al reiniciar la máquina, volverá a arrancar por red, dejando la instalación a medias y entrando en un bucle infinito.


Figura 14: Configuración de la BIOS de la máquina cliente. Nótese que el primer método de arranque es la red.

En la Figura 15 se observa el resultado del aplicativo. El ordenador de sobremesa de la parte izquierda es la máquina cliente, es decir, es que ha realizado la instalación desatendida. El ordenador ha sido conectado directamente al portátil gris de la derecha, el cuál es el servidor de la red, desconectádolo de la red general para realizar las primeras pruebas en un entrono real.

Siendo la instalación exitosa, se pasa al despliegue final del aplicativo. Para llevarlo a cabo, existen dos opciones: la primera de ellas es replicar lo descrito en esta memoria en los capítulos 3 y 4 paso a paso para que funcione, y la segunda es replicar la máquina virtual modificada. Por motivos de control y seguridad, se ha decidido replicar la máquina virtual una vez su comportamiento correcto está comprobado.

Cuando el servidor se encuentra de nuevo operativo, los usuarios ya pueden hacer uso de este servicio simplemente estando conectados a la misma red que el servidor. Adicionalmente, deben acceder al menú de configuración de su BIOS al igual que en el caso anterior para configurar el método de arranque. A partir de este momento, ya está operativo el aplicativo.


Figura 15: Instalación exitosa llevada a cabo en un PC del laboratorio.

## 5.3. Discusión

El aplicativo de instalación de Windows 11 se ha abordado de dos formas complementarias: la primera en un entorno simulado y la segunda en un entorno real. Esta forma de proceder ayuda a la validación del sistema y a ajustar los procedimientos del instalador de Windows.

Al crear un entorno de pruebas completamente simulado se han detectado grandes cantidades de errores sin poner en riesgo los servicios actualmente desplegados, como configuraciones incorrectas en el servicio PXE y TFTP.

Por otro lado, al llevar a cabo un despliegue posterior en un entorno real del aplicativo, se demuestra el carácter portable de la solución y la facilidad de implementación en una red similar a la planteada como objeto de este proyecto. Además, es posible observar errores que no son perceptibles en las simulaciones, como la falta de controladores de Windows PE o los fallos en las configuraciones de la BIOS/UEFI de las máquinas clientes (orden de preferencia de arranque incorrecto, versión TPM inferior a la 2.0, modo Arranque Seguro desactivado, etc).

En general, la combinación de ambos métodos ha permitido aumentar significativamente la robustez del aplicativo. La virtualización ha servido como entorno seguro de validación, mientras que el despliegue real ha consolidado la viabilidad del proyecto en condiciones reales.

# Capítulo 6

# Conclusiones y líneas futuras

En este último capítulo de la memoria, se presenta al lector una serie de conclusiones que el autor ha llegado tras la finalización del proyecto. Junto a las conclusiones, se incluyen unas líneas futuras para el aplicativo, con el objetivo de actualizarlo y mejorar su eficiencia, seguridad y compatibilidad.

## 6.1. Conclusiones

La necesidad para un equipo de investigación de actualizar sus sistemas operativos hace que se busquen nuevas formas de llevarlo a cabo de una manera escalable, de forma que sea solamente configurada una vez y replicada múltiples ocasiones. La implementación de un sistema en red con instalación desatendida de Windows ha demostrado ser una solución eficaz y eficiente para optimizar el despliegue de sistemas en el entorno del equipo de investigación.

Esto no solo ahorra tiempo a los administradores de redes, sino que mejora la seguridad y el control sobre qué o quién hace cambios en las máquinas de la red, lo que permite monitorizar amenazas y comportamientos no autorizados.

La realización de este proyecto ha permitido implementar la solución al problema planteado inicialmente con éxito para que los usuarios del laboratorio de investigación puedan usarlo. Esto se ha llevado a cabo con el uso de una gran cantidad de tecnologías, cada una con un rol específico, hacen que sea posible automatizar completamente el proceso de instalación, eliminando la intervención manual y garantizando una configuración uniforme en todos los equipos.

Este enfoque ha demostrado ser altamente beneficioso en varios aspectos. En primer lugar, ha

reducido significativamente el tiempo necesario para desplegar nuevos equipos, lo que se traduce en una mayor eficiencia operativa al reducir los errores humanos y gastos de gestión.

En segundo lugar, la estandarización de la instalación asegura que todos los equipos del laboratorio cuenten con la misma configuración base, incluyendo idioma, zona horaria, configuración de los parámetros de red y políticas de seguridad. Esto no solo facilita el mantenimiento y la resolución de problemas, sino que también mejora la seguridad y el cumplimiento de normativas internas.

Desde una perspectiva técnica, el proyecto ha requerido un conocimiento profundo de las tecnologías que necesitaban alojarse en el servidor (PXE, DHCP, TFTP, HTTP, AD-DC, etc), del proceso de instalación de Windows 11 y de *scripts* de automatización. Dentro del proceso de intalación de Windows 11, se ha profundizado en la investigación sobre la forma de conectar un dispositivo sin sistema operativo con el servidor que lo aloja, para concretar el envío de ficheros para su intalación y automatización. Esta automatización, llevada a cabo a través del fichero `Autounattend.xml`, ha obligado a adquirir conocimientos en profundidad sobre todas las fases del proceso. Por otro lado, ha sido necesario realizar pruebas iterativas para validar cada sección del fichero y asegurar su correcto funcionamiento en distintos escenarios de *hardware*.

En términos de impacto, esta solución representa una mejora sustancial en la gestión de la infraestructura tecnológica del equipo de investigación. No solo se ha optimizado el tiempo y los recursos dedicados a la instalación de sistemas, sino que también se ha establecido una base sólida para futuras automatizaciones.

En conclusión, la instalación desatendida de Windows no solo ha cumplido con los objetivos planteados al inicio del proyecto, sino que ha ayudado a mejorar la eficiencia, escalabilidad y adaptabilidad de la red. Este aplicativo se presenta como una herramienta estratégica para cualquier entorno técnico que requiera rapidez, consistencia y control en la implementación de sistemas operativos.

Adicionalmente, el aplicativo desarrollado puede ser fácilmente replicado o adaptado a otros laboratorios o entornos corporativos con estructuras de red y necesidades similares. Su modularidad permite integrar nuevas aplicaciones y políticas sin rediseñar completamente el sistema. Esto convierte a la solución en una base sólida sobre la que construir futuros entornos automatizados más complejos.

## 6.2. Líneas futuras

La versión del proyecto descrita en esta memoria corresponde a una implementación completa y funcional, diseñada para operar de manera eficiente desde el momento de su despliegue. Aún así, es necesario pensar en versiones futuras de este proyecto y en la suma de nuevas funcionalidades.

Con el objetivo de restringir el acceso no autorizado al proceso de instalación desatendida, es posible incorporar una capa de seguridad inicial que requiera al usuario introducir la contraseña habilitante configurada antes de que comience el proceso. Esta medida debería implementarse antes de instalar el entorno Windows PE, ya que es la primera fase del proceso de instalación. De este modo, se evita que cualquier persona con acceso físico a la red pueda iniciar el despliegue del sistema sin autorización. La contraseña actúa como un filtro de seguridad que garantiza que solo el personal técnico autorizado pueda ejecutar la instalación, reduciendo así el riesgo de instalaciones accidentales o maliciosas.

Con el objetivo de mejorar la seguridad de las comunicaciones, es posible sustituir en un futuro el uso del protocolo HTTP por el protocolo HTTPS para suplir sus funciones. Con el protocolo HTTPS se encripta toda la comunicación entre el cliente y el servidor, protegiendo los datos de ser leído y manipulado por posibles intrusos.

Finalmente, con el objetivo de mejorar el registro de nuevas máquinas al dominio, es posible crear un *script* que se ejecute durante la fase `Specialize` de instalación de Windows 11, para personalizar los nombres de las nuevas máquinas en donde se instale el nuevo sistema operativo.

# Anexos

### Fichero `ipxe.conf`

```
#!ipxe

#console --x 1024 --y 768
menu iPXE del LPI

item --gap ******************************************
item --gap * ATENCION: TODO ESTO SON COSAS PELIGROSAS *
item --gap ******************************************
item --gap
item --gap Si has llegado aqui sin saber como, mejor apaga la maquina antes
    de que se
item --gap te borre el disco. En caso contrario, sigue adelante por tu
   cuenta y riesgo.
item --gap
item --gap Fede
item --gap
item --gap -- ----------------------
item --gap next-server=${next-server}
item --gap platform=${platform}
item --gap root-path=${root-path}
item --gap -- ----------------------
item --gap

item --gap -- ----------------------Distros live------------------------
item gparted GParted
item memtest Memtest86+
item --gap

item --gap -- ----------------Arranques de rescate----------------------
item trixie_rescue Debian 13 Trixie rescue
item bookworm_rescue Debian 12 Bookworm rescue
item focal_rescue Ubuntu focal rescue
item bionic_rescue Ubuntu bionic rescue
item --gap
```

```
item --gap -- --------------Instalaciones automaticas-------------------
item trixie Debian 13 Trixie
item bookworm Debian 12 Bookworm
item jammy Ubuntu jammy
item focal Ubuntu focal
item winpe Windows 7 (WinPE)
item win11 Windows 11                        # Changed_Jesus
item --gap

item --gap -- ------------------------------------------------------------
item shellipxe Shell del iPXE               SALIR DE PXE

choose selection && goto ${selection}

:gparted
kernel gparted/vmlinuz initrd=initrd.img boot=live config components union=
   overlay username=user noswap noeject vga=788 fetch=http://${next-server
   }/gparted/filesystem.squashfs
initrd --name initrd.img gparted/initrd.img
boot

:memtest
kernel memtest/memtest
boot

:trixie_rescue
kernel trixie/linux initrd=initrd.gz rescue/enable=true
initrd --name initrd.gz trixie/initrd.gz
boot

:bookworm_rescue
kernel bookworm/linux initrd=initrd.gz rescue/enable=true
initrd --name initrd.gz bookworm/initrd.gz
boot

:focal_rescue
kernel focal/vmlinuz initrd=initrd.gz ip=dhcp recovery url=http://${next-
   server}/focal/focal-live-server-amd64.iso
initrd --name initrd.gz focal/initrd
boot

:bionic_rescue
kernel bionic/linux initrd=initrd.gz rescue/enable=true
#kernel bionic/linux initrd=initrd.gz rescue/enable=true ks=nfs
   :10.0.102.213:/srv/nfs/bionic.cfg
initrd --name initrd.gz bionic/initrd.gz
boot
```

```
:trixie
kernel trixie/linux initrd=initrd.gz auto=true url=http://${next-server}/
   trixie/preseed priority=critical DEBCONF_DEBUG=5
initrd --name initrd.gz trixie/initrd.gz
boot

:bookworm
kernel bookworm/linux initrd=initrd.gz auto=true url=http://${next-server}/
   bookworm/preseed priority=critical DEBCONF_DEBUG=5
initrd --name initrd.gz bookworm/initrd.gz
boot

:jammy
kernel jammy/vmlinuz initrd=initrd.gz ip=dhcp netboot=nfs nfsroot=${root-
   path}/ubuntu/jammy ds=nocloud-net;s=http://${next-server}/jammy/${
   platform}/ autoinstall
initrd --name initrd.gz jammy/initrd
boot

:focal
kernel focal/vmlinuz initrd=initrd.gz ip=dhcp url=http://${next-server}/
   focal/focal-live-server-amd64.iso ds=nocloud-net;s=http://${next-server
   }/focal/${platform}/ autoinstall
initrd --name initrd.gz focal/initrd
boot

:winpe
chain efi/microsoft/boot/cdboot.efi

# Changed_Jesus
################################################
:win11
kernel http://${next-server}/WinPE/wimboot
cpuid --ext 29 && set arch amd64
initrd http://${next-server}/WinPE/bootmgr bootmgr
initrd http://${next-server}/WinPE/bootmgr.efi bootmgr.efi
initrd http://${next-server}/WinPE/bootx64.efi bootx64.efi
initrd http://${next-server}/WinPE/BCD BCD
initrd http://${next-server}/WinPE/boot.sdi boot.sdi
initrd http://${next-server}/WinPE/boot.wim boot.wim
boot
################################################
# End_Changed_Jesus

:shellipxe
```

```
shell
```

*Script* 22: Fichero `ipxe.conf`.

### Fichero `Autounattend.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
    <settings pass="windowsPE">
        <component name="Microsoft-Windows-International-Core-WinPE"
           processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
           language="neutral" versionScope="nonSxS" xmlns:wcm="http://
           schemas.microsoft.com/WMIConfig/2002/State" xmlns:xsi="http://
           www.w3.org/2001/XMLSchema-instance">
            <SetupUILanguage>
                <UILanguage>es-ES</UILanguage>
            </SetupUILanguage>
            <UserLocale>es-ES</UserLocale>
            <UILanguage>es-ES</UILanguage>
            <SystemLocale>es-ES</SystemLocale>
            <InputLocale>0c0a:0000040a</InputLocale>
        </component>
        <component name="Microsoft-Windows-Setup" processorArchitecture="
           amd64" publicKeyToken="31bf3856ad364e35" language="neutral"
           versionScope="nonSxS" xmlns:wcm="http://schemas.microsoft.com/
           WMIConfig/2002/State" xmlns:xsi="http://www.w3.org/2001/
           XMLSchema-instance">
            <ImageInstall>
                <OSImage>
                    <InstallTo>
                        <DiskID>0</DiskID>
                        <PartitionID>3</PartitionID>
                    </InstallTo>
                </OSImage>
            </ImageInstall>
            <UserData>
                <ProductKey>
                    <Key>YNMGQ-8RYV3-4PGQ3-C8XTP-7CFBY</Key>
                    <WillShowUI>OnError</WillShowUI>
                </ProductKey>
                <AcceptEula>true</AcceptEula>
            </UserData>
            <UseConfigurationSet>false</UseConfigurationSet>
            <RunSynchronous>
                <RunSynchronousCommand wcm:action="add">
                    <Order>1</Order>
                    <Path>cmd.exe /c &amp;quot;&amp;gt;&amp;gt;&amp;quot;X:\gestDisco.txt&
                       amp;quot; (echo SELECT DISK=0&amp;amp;echo CLEAN&amp;amp;echo
                       CONVERT GPT&amp;amp;echo CREATE PARTITION EFI SIZE=300&
                       amp;amp;echo FORMAT QUICK FS=FAT32 LABEL=&amp;quot;System&
                       quot;&amp;amp;echo CREATE PARTITION MSR SIZE=16)&amp;quot;</
                       Path>
```

```
                </RunSynchronousCommand>
                <RunSynchronousCommand wcm:action="add">
                    <Order>2</Order>
                    <Path>cmd.exe /c &quot;&gt;&gt;&quot;X:\gestDisco.txt&
                       quot; (echo CREATE PARTITION PRIMARY&amp;echo SHRINK
                        MINIMUM=1024&amp;echo FORMAT QUICK FS=NTFS LABEL=&
                       quot;Windows&quot;&amp;echo CREATE PARTITION PRIMARY
                       &amp;echo FORMAT QUICK FS=NTFS LABEL=&quot;Recovery&
                       quot;)&quot;</Path>
                </RunSynchronousCommand>
                <RunSynchronousCommand wcm:action="add">
                    <Order>3</Order>
                    <Path>cmd.exe /c &quot;&gt;&gt;&quot;X:\gestDisco.txt&
                       quot; (echo SET ID=&quot;de94bba4-06d1-4d40-a16a-
                       bfd50179d6ac&quot;&amp;echo GPT ATTRIBUTES=0
                       x8000000000000001)&quot;</Path>
                </RunSynchronousCommand>
                <RunSynchronousCommand wcm:action="add">
                    <Order>4</Order>
                    <Path>cmd.exe /c &quot;diskpart.exe /s &quot;X:\
                       gestDisco.txt&quot; &gt;&gt;&quot;X:\diskpart.log&
                       quot; || ( type &quot;X:\diskpart.log&quot; &amp;
                       echo diskpart encountered an error. &amp; pause &amp
                       ; exit /b 1 )&quot;</Path>
                </RunSynchronousCommand>
                <RunSynchronousCommand wcm:action="add">
                    <Order>5</Order>
                    <Path>reg.exe add &quot;HKLM\SYSTEM\Setup\LabConfig&
                       quot; /v BypassTPMCheck /t REG_DWORD /d 1 /f</Path>
                </RunSynchronousCommand>
                <RunSynchronousCommand wcm:action="add">
                    <Order>6</Order>
                    <Path>reg.exe add &quot;HKLM\SYSTEM\Setup\LabConfig&
                       quot; /v BypassSecureBootCheck /t REG_DWORD /d 1 /f
                       </Path>
                </RunSynchronousCommand>
                <RunSynchronousCommand wcm:action="add">
                    <Order>7</Order>
                    <Path>reg.exe add &quot;HKLM\SYSTEM\Setup\LabConfig&
                       quot; /v BypassRAMCheck /t REG_DWORD /d 1 /f</Path>
                </RunSynchronousCommand>
            </RunSynchronous>
        </component>
    </settings>
    <settings pass="specialize">
        <component name="Microsoft-Windows-UnattendedJoin"
           processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
           language="neutral" versionScope="nonSxS" xmlns:wcm="http://
```

```
        schemas.microsoft.com/WMIConfig/2002/State" xmlns:xsi="http://
        www.w3.org/2001/XMLSchema-instance">
         <Identification>
             <Credentials>
                 <Password>Password123</Password>
                 <Username>joinPC</Username>
                 <Domain>lab25.local</Domain>
             </Credentials>
             <JoinDomain>lab25.local</JoinDomain>
         </Identification>
     </component>
     <component name="Microsoft-Windows-Shell-Setup"
        processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
        language="neutral" versionScope="nonSxS" xmlns:wcm="http://
        schemas.microsoft.com/WMIConfig/2002/State" xmlns:xsi="http://
        www.w3.org/2001/XMLSchema-instance">
         <ComputerName>*</ComputerName>
     </component>
 </settings>
 <settings pass="oobeSystem">
     <component name="Microsoft-Windows-Shell-Setup"
        processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
        language="neutral" versionScope="nonSxS" xmlns:wcm="http://
        schemas.microsoft.com/WMIConfig/2002/State" xmlns:xsi="http://
        www.w3.org/2001/XMLSchema-instance">
         <OOBE>
             <HideEULAPage>true</HideEULAPage>
             <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>
             <ProtectYourPC>3</ProtectYourPC>
             <HideOnlineAccountScreens>true</HideOnlineAccountScreens>
             <HideOEMRegistrationScreen>true</HideOEMRegistrationScreen>
             <HideLocalAccountScreen>true</HideLocalAccountScreen>
             <NetworkLocation>Work</NetworkLocation>
             <SkipMachineOOBE>true</SkipMachineOOBE>
             <SkipUserOOBE>true</SkipUserOOBE>
         </OOBE>
         <RegisteredOrganization>Universidad de Valladolid</
            RegisteredOrganization>
         <TimeZone>W. Europe Standard Time</TimeZone>
     </component>
     <component name="Microsoft-Windows-International-Core"
        processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
        language="neutral" versionScope="nonSxS" xmlns:wcm="http://
        schemas.microsoft.com/WMIConfig/2002/State" xmlns:xsi="http://
        www.w3.org/2001/XMLSchema-instance">
         <InputLocale>es-ES</InputLocale>
         <SystemLocale>es-ES</SystemLocale>
         <UILanguage>es-ES</UILanguage>
```

```
            <UserLocale>es-ES</UserLocale>
        </component>
    </settings>
</unattend>
```

*Script* 23: Fichero `Autounattend.xml`.

### Fichero `console.h`

```
#ifndef CONFIG_CONSOLE_H
#define CONFIG_CONSOLE_H

/** @file
 *
 * Console configuration
 *
 * These options specify the console types that iPXE will use for
 * interaction with the user.
 *
 */

FILE_LICENCE ( GPL2_OR_LATER_OR_UBDL );

#include <config/defaults.h>

/*
 * Default console types
 *
 * These are all enabled by default for the appropriate platforms.
 * You may disable them if needed.
 *
 */

//#undef  CONSOLE_PCBIOS    /* Default BIOS console */
//#undef  CONSOLE_EFI   /* Default EFI console */
//#undef  CONSOLE_LINUX   /* Default Linux console */

/*
 * Additional console types
 *
 * These are not enabled by default, but may be useful in your
 * environment.
 *
 */

#define CONSOLE_SERIAL    /* Serial port console */
#define CONSOLE_FRAMEBUFFER /* Graphical framebuffer console */
#define CONSOLE_SYSLOG    /* Syslog console */
//#define CONSOLE_SYSLOGS   /* Encrypted syslog console */
//#define CONSOLE_VMWARE    /* VMware logfile console */
#define CONSOLE_DEBUGCON  /* Bochs/QEMU/KVM debug port console */
//#define CONSOLE_INT13   /* INT13 disk log console */

/*
 * Very obscure console types
```

```
 *
 * You almost certainly do not need to enable these.
 *
 */

//#define CONSOLE_DIRECT_VGA  /* Direct access to VGA card */
//#define CONSOLE_PC_KBD    /* Direct access to PC keyboard */

/* Keyboard map (available maps in hci/keymap/) */
#define KEYBOARD_MAP  es

/* Control which syslog() messages are generated.
 *
 * Note that this is not related in any way to CONSOLE_SYSLOG.
 */
#define LOG_LEVEL LOG_NONE

#include <config/named.h>
#include NAMED_CONFIG(console.h)
#include <config/local/console.h>
#include LOCAL_NAMED_CONFIG(console.h)

#endif /* CONFIG_CONSOLE_H */
```

*Script* 24: Fichero `console.h`.

### Fichero `general.h`

```
#ifndef CONFIG_GENERAL_H
#define CONFIG_GENERAL_H

/** @file
 *
 * General configuration
 *
 */

FILE_LICENCE ( GPL2_OR_LATER_OR_UBDL );

#include <config/defaults.h>

/*
 * Banner timeout configuration
 *
 * This controls the timeout for the "Press Ctrl-B for the iPXE
 * command line" banner displayed when iPXE starts up.  The value is
 * specified in tenths of a second for which the banner should appear.
 * A value of 0 disables the banner.
 *
 * ROM_BANNER_TIMEOUT controls the "Press Ctrl-B to configure iPXE"
 * banner displayed only by ROM builds of iPXE during POST.  This
 * defaults to being twice the length of BANNER_TIMEOUT, to allow for
 * BIOSes that switch video modes immediately before calling the
 * initialisation vector, thus rendering the banner almost invisible
 * to the user.
 */
#define BANNER_TIMEOUT    20
#define ROM_BANNER_TIMEOUT  ( 2 * BANNER_TIMEOUT )

/*
 * Network protocols
 *
 */

#define NET_PROTO_IPV4    /* IPv4 protocol */
//#define NET_PROTO_IPV6  /* IPv6 protocol */
#undef  NET_PROTO_FCOE    /* Fibre Channel over Ethernet protocol */
#define NET_PROTO_STP   /* Spanning Tree protocol */
#define NET_PROTO_LACP    /* Link Aggregation control protocol */
#define NET_PROTO_EAPOL   /* EAP over LAN protocol */

/*
 * PXE support
 *
```

```
 */
//#undef  PXE_STACK   /* PXE stack in iPXE - you want this! */
//#undef  PXE_MENU    /* PXE menu booting */

/*
 * Download protocols
 *
 */

#define DOWNLOAD_PROTO_TFTP /* Trivial File Transfer Protocol */
#define DOWNLOAD_PROTO_HTTP /* Hypertext Transfer Protocol */
#undef  DOWNLOAD_PROTO_HTTPS  /* Secure Hypertext Transfer Protocol */
#undef  DOWNLOAD_PROTO_FTP  /* File Transfer Protocol */
#undef  DOWNLOAD_PROTO_SLAM /* Scalable Local Area Multicast */
#undef  DOWNLOAD_PROTO_NFS  /* Network File System Protocol */
//#undef DOWNLOAD_PROTO_FILE  /* Local filesystem access */

/*
 * SAN boot protocols
 *
 */

//#undef  SANBOOT_PROTO_ISCSI /* iSCSI protocol */
//#undef  SANBOOT_PROTO_AOE /* AoE protocol */
//#undef  SANBOOT_PROTO_IB_SRP  /* Infiniband SCSI RDMA protocol */
//#undef  SANBOOT_PROTO_FCP /* Fibre Channel protocol */
//#undef  SANBOOT_PROTO_HTTP  /* HTTP SAN protocol */

/*
 * HTTP extensions
 *
 */
#define HTTP_AUTH_BASIC   /* Basic authentication */
#define HTTP_AUTH_DIGEST  /* Digest authentication */
//#define HTTP_AUTH_NTLM  /* NTLM authentication */
//#define HTTP_ENC_PEERDIST /* PeerDist content encoding */
//#define HTTP_HACK_GCE   /* Google Compute Engine hacks */

/*
 * 802.11 cryptosystems and handshaking protocols
 *
 */
#define CRYPTO_80211_WEP  /* WEP encryption (deprecated and insecure!) */
#define CRYPTO_80211_WPA  /* WPA Personal, authenticating with passphrase
   */
#define CRYPTO_80211_WPA2 /* Add support for stronger WPA cryptography */

/*
```

```
 * Name resolution modules
 *
 */

#define DNS_RESOLVER    /* DNS resolver */

/*
 * Image types
 *
 * Etherboot supports various image formats.  Select whichever ones
 * you want to use.
 *
 */
//#define IMAGE_NBI   /* NBI image support */
//#define IMAGE_ELF   /* ELF image support */
//#define IMAGE_MULTIBOOT   /* MultiBoot image support */
//#define IMAGE_PXE   /* PXE image support */
//#define IMAGE_SCRIPT    /* iPXE script image support */
//#define IMAGE_BZIMAGE   /* Linux bzImage image support */
//#define IMAGE_COMBOOT   /* SYSLINUX COMBOOT image support */
//#define IMAGE_EFI   /* EFI image support */
//#define IMAGE_SDI   /* SDI image support */
#define IMAGE_PNM   /* PNM image support */
#define IMAGE_PNG   /* PNG image support */
#define IMAGE_DER   /* DER image support */
#define IMAGE_PEM   /* PEM image support */
#define IMAGE_ZLIB    /* ZLIB image support */
#define IMAGE_GZIP    /* GZIP image support */

/*
 * Command-line commands to include
 *
 */
#define AUTOBOOT_CMD    /* Automatic booting */
#define NVO_CMD     /* Non-volatile option storage commands */
#define CONFIG_CMD    /* Option configuration console */
#define IFMGMT_CMD    /* Interface management commands */
#define IWMGMT_CMD    /* Wireless interface management commands */
#define IBMGMT_CMD    /* Infiniband management commands */
#define FCMGMT_CMD    /* Fibre Channel management commands */
#define ROUTE_CMD   /* Routing table management commands */
#define IMAGE_CMD   /* Image management commands */
#define DHCP_CMD    /* DHCP management commands */
#define SANBOOT_CMD   /* SAN boot commands */
#define MENU_CMD    /* Menu commands */
#define LOGIN_CMD   /* Login command */
#define SYNC_CMD    /* Sync command */
#define SHELL_CMD   /* Shell command */
```

```
#define NSLOOKUP_CMD    /* DNS resolving command */
#define TIME_CMD    /* Time commands */
//#define DIGEST_CMD    /* Image crypto digest commands */
//#define LOTEST_CMD    /* Loopback testing commands */
#define VLAN_CMD    /* VLAN commands */
//#define PXE_CMD   /* PXE commands */    // no compila en EFI
#define REBOOT_CMD    /* Reboot command */
#define POWEROFF_CMD    /* Power off command */
//#define IMAGE_TRUST_CMD /* Image trust management commands */
#define PCI_CMD   /* PCI commands */
//#define PARAM_CMD   /* Form parameter commands */
#define NEIGHBOUR_CMD   /* Neighbour management commands */
#define PING_CMD    /* Ping command */
#define CONSOLE_CMD   /* Console command */
#define IPSTAT_CMD    /* IP statistics commands */
//#define PROFSTAT_CMD    /* Profiling commands */
#define NTP_CMD   /* NTP commands */
#define CERT_CMD    /* Certificate management commands */
//#define IMAGE_MEM_CMD   /* Read memory command */
#define IMAGE_ARCHIVE_CMD /* Archive image management commands */

/*
 * ROM-specific options
 *
 */
#undef  NONPNP_HOOK_INT19 /* Hook INT19 on non-PnP BIOSes */
#define AUTOBOOT_ROM_FILTER /* Autoboot only devices matching our ROM */

/*
 * Virtual network devices
 *
 */
#define VNIC_IPOIB    /* Infiniband IPoIB virtual NICs */
//#define VNIC_XSIGO    /* Infiniband Xsigo virtual NICs */

/*
 * Error message tables to include
 *
 */
#undef  ERRMSG_80211    /* All 802.11 error descriptions (~3.3kb) */

/*
 * Obscure configuration options
 *
 * You probably don't need to touch these.
 *
 */
```

```
#undef  BUILD_SERIAL    /* Include an automatic build serial
         * number.  Add "bs" to the list of
         * make targets.  For example:
         * "make bin/rtl8139.dsk bs" */
#undef  BUILD_ID    /* Include a custom build ID string,
         * e.g "test-foo" */
#undef  NULL_TRAP   /* Attempt to catch NULL function calls */
#undef  GDBSERIAL   /* Remote GDB debugging over serial */
#undef  GDBUDP      /* Remote GDB debugging over UDP
         * (both may be set) */
//#define EFI_DOWNGRADE_UX  /* Downgrade UEFI user experience */
#define TIVOLI_VMM_WORKAROUND /* Work around the Tivoli VMM's garbling of
   SSE
         * registers when iPXE traps to it due to
         * privileged instructions */

#include <config/named.h>
#include NAMED_CONFIG(general.h)
#include <config/local/general.h>
#include LOCAL_NAMED_CONFIG(general.h)

#endif /* CONFIG_GENERAL_H */
```

*Script* 25: Fichero `general.h`.

# Bibliografía


[1] Windows-Driver-Content, *Descripción de las herramientas de Windows ADK*, jun. de 2023. dirección: https://learn.microsoft.com/es-es/windows-hardware/test/weg/understanding-the-windows-adk-tools.

[2] Mestew, *Requisitos de Windows 11*, mar. de 2024. dirección: https://learn.microsoft.com/es-es/windows/whats-new/windows-11-requirements.

[3] M. Corporation, *¿Qué es un Módulo de plataforma segura (TPM)? - Soporte técnico de Microsoft*, 2025. dirección: https://support.microsoft.com/es-es/topic/-qu%C3%A9-es-un-m%C3%B3dulo-de-plataforma-segura-tpm-705f241d-025d-4470-80c5-4feeb24fa1ee.

[4] I. Corporation, "Preboot Execution Environment (PXE) Specification Version 2.1," inf. téc., sep. de 1999.

[5] I. Corporation, "Extensible Firmware Interface Specification Version 1.10," inf. téc., dic. de 2002.

[6] S. Alexander y R. Droms, "DHCP options and BOOTP vendor extensions," RFC Series, inf. téc., mar. de 1997. DOI: 10.17487/rfc2132. dirección: https://www.rfc-editor.org/rfc/rfc2132.html.

[7] M. Johnston, "Dynamic Host Configuration Protocol (DHCP) options for the Intel Preboot eXecution Environment (PXE)," RFC Series, inf. téc., nov. de 2006. DOI: 10.17487/rfc4578. dirección: https://www.rfc-editor.org/rfc/rfc4578.html.

[8] R. Droms, "Dynamic Host Configuration Protocol," RFC Series, inf. téc., mar. de 1997. DOI: 10.17487/rfc2131. dirección: https://www.rfc-editor.org/rfc/rfc2131.html.

[9] B. Croft, *RFC 951: Bootstrap Protocol*, sep. de 1985. dirección: https://datatracker.ietf.org/doc/html/rfc951.

[10] J. Postel, *RFC 959: File Transfer Protocol*, oct. de 1985. dirección: https://datatracker.ietf.org/doc/html/rfc959.

[11] K. Sollins, "The TFTP Protocol (Revision 2)," RFC Series, inf. téc., jul. de 1992. DOI: 10.17487/rfc1350. dirección: https://www.rfc-editor.org/rfc/rfc1350.html.

[12] Meaghanlewis, *Active Directory Domain Services overview*, nov. de 2024. dirección: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.

[13] J. Lentini, C. Everhart, D. Ellard, R. Tewari y M. Naik, "Requirements for federated file systems," RFC Series, inf. téc., ene. de 2010. DOI: 10.17487/rfc5716. dirección: https://www.rfc-editor.org/rfc/rfc5716.html.

[14] Samba, *Samba Documentation*, 2025. dirección: https://www.samba.org/samba/docs/.

[15] T. Howes, M. Smith y G. S. Good, *Understanding and deploying LDAP directory Services*. Addison-Wesley Professional, ene. de 2003. dirección: https://archive.org/details/understandingdep0000timo/page/n5/mode/2up.

[16] J. Sermersheim, "Lightweight Directory Access Protocol (LDAP): the protocol," inf. téc., jun. de 2006. DOI: 10.17487/rfc4511. dirección: https://www.rfc-editor.org/rfc/rfc4511.html.

[17] A. Melnikov, *RFC 4422: Simple Authentication and Security Layer (SASL)*, jun. de 2006. dirección: https://datatracker.ietf.org/doc/html/rfc4422.

[18] C. Neuman, *RFC 4120: The Kerberos Network Authentication Service (V5)*, jul. de 2005. dirección: https://datatracker.ietf.org/doc/html/rfc4120.

[19] M. Thomson y C. Benfield, "RFC 9113: HTTP/2," RFC Series, inf. téc., jun. de 2022. dirección: https://www.rfc-editor.org/rfc/rfc9113.html.

[20] M. Bishop, "RFC 9114: HTTP/3," RFC Series, inf. téc., jun. de 2022. dirección: https://www.rfc-editor.org/rfc/rfc9114.html.

[21] R. T. Fielding, M. Nottingham y J. Reschke, *RFC 9110: HTTP Semantics*, jun. de 2022. dirección: https://www.rfc-editor.org/rfc/rfc9110.html.

[22] T. A. S. Foundation, *Apache HTTP Server Version 2.4 Documentation - Apache HTTP Server Version 2.4*, 2025. dirección: https://httpd.apache.org/docs/2.4/.

[23] Windows-Driver-Content, *Windows PE (WinPE)*, mar. de 2023. dirección: https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/winpe-intro?view=windows-11.

[24] Windows-Driver-Content, *Download and install the Windows ADK*, ene. de 2025. dirección: https://learn.microsoft.com/en-us/windows-hardware/get-started/adk-install.

[25] C. Inc, *Using VMware Workstation Pro*, mayo de 2025. dirección: https://techdocs.broadcom.com/us/en/vmware-cis/desktop-hypervisors/workstation-pro/17-0/using-vmware-workstation-pro.html.

[26] Windows-Driver-Content, *Archivos de respuesta (unattend.xml)*, mayo de 2023. dirección: https://learn.microsoft.com/es-es/windows-hardware/manufacture/desktop/update-windows-settings-and-scripts-create-your-own-answer-file-sxs?view=windows-11.

[27] D. Project, *tftpd(8) — tftpd-hpa — Debian testing — Debian Manpages*, jun. de 2014. dirección: https://manpages.debian.org/testing/tftpd-hpa/tftpd.8.en.html.

[28] G. LLC, *Introducción al DNS público de Google*, sep. de 2024. dirección: https://developers.google.com/speed/public-dns/docs/intro?hl=es-419.

[29] N. blog, *NetworkManager for administrators*, 2025. dirección: https://networkmanager.dev/docs/admins/.

[30] Billmath, *Gestión de Acceso Privilegiado para Active Directory Domain Services*, abr. de 2025. dirección: https://learn.microsoft.com/es-es/microsoft-identity-manager/pam/privileged-identity-management-for-active-directory-domain-services.

[31] iPXE Org., *iPXE - Open source boot firmware - Documentation*, nov. de 2024. dirección: https://ipxe.org/docs.

[32] M. Chadalapaka, *RFC 7143: Internet Small Computer System Interface (ISCSI) Protocol (Consolidated)*, abr. de 2014. dirección: https://datatracker.ietf.org/doc/html/rfc7143.

[33] C. Zhou y C. He, "A Performance Analysis of the AoE Protocol," en *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, ene. de 2009. DOI: 10.1109/WICOM.2009.5302932. dirección: https://ieeexplore.ieee.org/document/5302932.

[34] R. H. E. Linux, *9.2. Configuración de un dispositivo FCoE por software | Gestión de dispositivos de almacenamiento*, ene. de 2025. dirección: https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/managing_storage_devices/setting-up-a-software-fcoe-device_configuring-fibre-channel-over-ethernet.

[35] G. Inc, *About GitHub and Git - GitHub Docs*, ene. de 2025. dirección: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git.

[36] Windows-Driver-Content, *WinPE: Mount and Customize*, mayo de 2023. dirección: https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/winpe-mount-and-customize?view=windows-11#add-updates-to-winpe-if-needed.

[37] M. Corporation, *Cómo administrar las revocaciones de Windows Boot Manager para cambios de arranque seguro asociados con CVE-2023-24932 - Soporte técnico de Microsoft*, mayo de 2023. dirección: https://support.microsoft.com/es-es/topic/c%C3%B3mo-administrar-las-revocaciones-de-windows-boot-manager-para-cambios-de-arranque-seguro-asociados-con-cve-2023-24932-41a975df-beb2-40c1-99a3-b3ff139f832d.

[38] Windows-Driver-Content, *WinPE Optional Components (OC) Reference*, mar. de 2023. dirección: https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/winpe-add-packages--optional-components-reference?view=windows-11#winpe-optional-components.

[39] iPXE Org., *iPXE - open source boot firmware [wimboot]*, abr. de 2021. dirección: https://ipxe.org/wimboot.

[40] Windows-Driver-Content, *Automatizar programa de instalación de Windows*, jun. de 2023. dirección: https://learn.microsoft.com/es-es/windows-hardware/manufacture/desktop/automate-windows-setup?view=windows-11#automating-windowssetup-pages.

[41] Windows-Driver-Content, *UEFI/GPT-based hard drive partitions*, feb. de 2023. dirección: https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/configure-uefigpt-based-hard-drive-partitions?view=windows-11.