
# PROJECTE FINAL DE CARRERA

## VDI low cost implementation with KVM and Linux Dockers

*Estudis: Enginyeria de Telecomunicació*

*Autor:* Eric Collazo Vallduriola

Ponent: Beatriz Otero Calviño

*Director/a:* Antonio Javier Valverde Amador

*Any: 2015*

# Índice general

# Colaboraciones

AJUNTAMENT DE
SANT JOAN DESPÍ

Este proyecto de final de carrera se ha desarrollado en el Ayuntamiento de Sant Joan Despí, bajo la supervisión del Cap de Sistemes d'Informació Antoni Xavier Valverde Amador.

Sant Joan Despí es un municipio que pertenece a la comarca del Baix Llobregat, situado a la izquierda del río. El municipio está entre Sant Feliu de Llobregat, Sant Just Desvern, Esplugues de Llobregat, Cornella de Llobregat, Sant Boi y Santa Coloma de Cervelló. El censo de población en el año 2014 era de 32.981 habitantes y la extensión del municipio es de 6,39 $km^2$

El ayuntamiento ha aportado al proyecto todos los recursos informáticos y electrónicos necesarios para que este proyecto se pudiera llevar a cabo.

La tutora del proyecto Beatriz Otero, junto con el autor de esta memoria Eric Collazo, agradecen toda la colaboración que el ayuntamiento ha prestado en este proyecto.

# Agradecimientos

A Javier Silva y a Miguel Ramírez, técnicos en el Ayuntamiento de Sant Joan Despí, por apoyarme cuando no conseguía avanzar y ayudarme cuando lo necesitaba. También, por todo lo que he aprendido trabajando con ellos.

A Antoni Xavier Valverde, Cap de Sistemes d'Informació del Ayuntamiento de Sant Joan Despí, por permitirme hacer este trabajo en el ayuntamiento y por su colaboración personal y apoyo constante en todo momento.

A Beatriz Otero, por sus directrices, indicaciones y consejos durante todo el proceso de realización del trabajo.

A mi familia y a mi pareja, por apoyarme incondicionalmente en todo lo que hago y estar siempre conmigo cuando los necesito.

# Resum del projecte

Aquest projecte realitza una implementació d'un entorn d'escriptoris virtuals, també conegut com VDI o *virtual desktop infrastructure*, perquè es plantegi com una alternativa econòmica al producte comercial de VMWare, anomenat Horizon View[1].

Per dur a terme aquesta implementació, s'ha modificat el codi font d'una plataforma de virtualització de servidors, de codi lliure, anomenada Proxmox VE[2]. A més, s'han afegit diversos scripts amb l'objectiu que la nova eina tingui un funcionament semblant al de l'Horizon View. Addicionalment, s'ha creat una interfície web personalitzada, anomenada Proxmox View, que permet gestionar tot l'entorn.

Per altra banda, s'ha creat un software client-servidor, perquè diversos tipus de dispositius, entre els quals es troba la Raspberry Pi[3] i els ordinadors amb pocs recursos, puguin connectar-se a aquests escriptoris virtuals, sense la necessitat de configurar cada un d'ells de forma independent.

Finalment, als dispositius Raspberry Pi, se'ls ha afegit un doble mecanisme d'autentificació. Aquest mecanisme està format per una combinació d'usuari i contrasenya, a més de la validació d'una etiqueta NFC[4] programada mitjançant un algorisme original de seguretat, que també s'ha dissenyat en aquest projecte.

L'objectiu principal del projecte és disposar d'un producte integrat, que permeti la seva administració de forma centralitzada i redueixi els costes de l'Ajuntament, sense sacrificar en gran mesura l'experiència dels usuaris en els seus escriptoris virtuals.

[1] https://www.vmware.com/products/horizon-view
[2] https://www.proxmox.com/en/proxmox-ve
[3] https://www.raspberrypi.org/help/what-is-a-raspberry-pi/
[4] Near field communication: tecnologia de comunicació sense cables, de curta distancia i alta freqüència que permet l'intercanvi de dades entre dispositius.

# Resumen del proyecto

Este proyecto realiza una implementación de un entorno de escritorios virtuales, también conocido como VDI o *virtual desktop infrastructure*, para que se plantee como una alternativa económica al producto comercial de VMWare, llamada Horizon View[5].

Para llevar a cabo esta implementación, se ha modificado el código fuente de una plataforma de virtualización de servidores, de código libre, llamada Proxmox VE[6]. Además, se le han añadido varios scripts con el objetivo de que la nueva herramienta tenga un funcionamiento parecido al de Horizon View. Adicionalmente, se ha creado una interfaz web personalizada, llamada Proxmox View, que permite gestionar todo el entorno.

Por otra parte, se ha creado un software cliente-servidor, para que varios tipos de dispositivos, entre los que se encuentran la Raspberry Pi[7] y los ordenadores con bajos recursos, puedan conectarse a estos escritorios virtuales, sin la necesidad de configurar cada uno de ellos de forma independiente.

Finalmente, a los dispositivos Raspberry Pi, se les ha añadido un doble mecanismo de autentificación. Este mecanismo está formado por una combinación de usuario y contraseña, además de la validación de una etiqueta NFC[8] programada mediante un algoritmo original de seguridad, que también se ha diseñado en este proyecto.

El objetivo principal del proyecto es disponer de un producto integrado, que permita su administración de forma centralizada y que reduzca los costes del Ayuntamiento, sin sacrificar en gran medida la experiencia de los usuarios en sus escritorios virtuales.

[5] https://www.vmware.com/products/horizon-view
[6] https://www.proxmox.com/en/proxmox-ve
[7] https://www.raspberrypi.org/help/what-is-a-raspberry-pi/
[8] Near field communication: tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos.

# Abstract

This project develops a virtual desktop infrastructure, also known as VDI, to be an inexpensive alternative to the commercial product of VMWare, called Horizon View[9].

We have modified the open source code of a server virtualization environment called Proxmox VE[10]. In addition, we have designed some scripts in our environment to look like Horizon View. We also have developed a custom web interface to control the entire scenario and we named it Proxmox View.

On the other hand, we build up one server-client software, to connect all the devices to the virtual desktops, including the Raspberry Pi[11] and low profiled computers, without setting up the device configuration one by one.

To sum up, we have programmed a double authentication login in the Raspberry Pi devices. This mechanism is composed by a combination of user-password and the possession of a NFC[12] tag, designed with an original security algorithm written by us.

The main purpose of this project is to provide a complete product with centralized configuration, decreasing the investment of the Town Hall in this area and maintaining the virtual desktop user's experience.

[9] https://www.vmware.com/products/horizon-view
[10] https://www.proxmox.com/en/proxmox-ve
[11] https://www.raspberrypi.org/help/what-is-a-raspberry-pi/
[12] Near field communication: Wireless short-distance communication technology of high frequency, that allows sharing data between devices.

# 1. Introducción

Hoy en día, el mantenimiento de una infraestructura informática de más de 100 ordenadores, se ha convertido en una tarea que exige muchas horas a los técnicos informáticos que trabajan en una empresa. Por ese motivo, entre otros, cada vez más empresas optan por el uso de escritorios virtuales, en los que su principal característica, es que se distribuye a una serie de clientes, una plantilla de sistema operativo. En cuanto el usuario decide dejar de usar esta plantilla, esta se destruye, guardando los datos del cliente en algún tipo de almacenamiento en red. Esto permite la creación de plantillas que comparten varios usuarios y que por tanto, se pueden gestionar y mantener de forma centralizada, disminuyendo drásticamente, los tiempos que invierten los técnicos en su mantenimiento.

Otra opción, usada desde hace más años, es el clonaje en red de máquinas. Sin embargo, debido a la gran cantidad de datos que se transmiten por la red, así como problemas de compatibilidad con algunas placas bases, y los largos tiempos que conllevan la distribución de una nueva imagen a un gran grupo de ordenadores, hacen que esta opción sea inviable para una empresa que requiere tener sus sistemas disponibles la mayor parte del día.

## 1.1 Antecedentes

Este proyecto final de carrera se ha desarrollado en el Ayuntamiento de Sant Joan Despí, bajo la supervisión del Cap de Sistemes d'Informació Antoni Xavier Valverde Amador. La infraestructura de red de este ayuntamiento se divide en 2 redes internas: la primera se usa para la formación que se imparte en diferentes aulas, mientras que la segunda se usa como red local corporativa de los trabajadores del ayuntamiento.

Hace unos años, el ayuntamiento tuvo una experiencia con los escritorios virtuales de Virtualbox para la red de formación, las pruebas no funcionaron como ellos esperaban, principalmente por la poca adaptación que tenía el software a lo que ellos necesitaban. Finalmente, cambiaron el entorno de virtualización por VMWare que mediante Horizon View, es un sistema que entrega escritorios virtuales a todos los usuarios que necesitan. Este sistema lo contrataron a una empresa externa llamada Nexica que mediante sus

servidores que poseen en el Baix Llobregat entregaban los escritorios virtuales en la nube vía WAN[13] a estos clientes. Este sistema funciona bien pero, tiene una pérdida de rendimiento debido a la limitación de velocidad que suponía el hecho de recibir los datos vía WAN en lugar de LAN[14], lo que a veces provoca un mal funcionamiento de estos escritorios virtuales; además del alto coste que suponía alquilar estos servidores externos. En ese momento para satisfacer las necesidades de rendimiento de los trabajadores en la red local, el ayuntamiento decidió comprar cuatro nodos SimpliVity[15] y montar un VMWare Horizon View en LAN. Aunque la experiencia de los usuarios ha mejorado considerablemente, la compatibilidad de VMWare con estos servidores SimpliVity no ha sido la esperada, ya que a veces se han producido errores. Todo esto, junto con la gran cantidad de dinero que se necesita para las renovaciones anuales de las licencias de VMWare, ha propiciado la búsqueda de una solución alternativa, que reduzca los costes y se adapte al ayuntamiento.

Así pues, el objetivo principal del proyecto, es el de montar un sistema equivalente al que ofrece VMWare con su plataforma de administración Horizon View. Las funcionalidades que debe tener son:

a) Ser un software cliente-servidor. Que permita a un usuario autentificado en un cliente, solicitar un escritorio virtual para que este pueda usarlo (si tiene los permisos para hacerlo).
b) Gestionar las plantillas disponibles para poder añadir nuevas, modificar las existentes o borrar las que ya no se necesiten.
c) Crear y eliminar automáticamente máquinas virtuales, de manera que se pueda cumplir con las cantidades esperadas, según la configuración que se establezca.
d) Tener un control visual de los escritorios virtuales generados, ya sea de los que están siendo usados por un cliente, como los que están disponibles para ser reclamados.

Además de estas funcionalidades, se ha añadido un mecanismo de doble autentificación en algunos clientes (las Raspberry Pi). Este sistema permite que el usuario además de su identificador y contraseña, necesite de una etiqueta NFC para poder acceder al sistema.

[13] Red de computadoras que abarca varias ubicaciones físicas, proveyendo servicio a una zona, un país e incluso varios continentes.
[14] Es una red de computadoras que abarca un área reducida. Como una casa, un departamento o un edificio.
[15] https://www.simplivity.com/

## 1.2 Objetivos específicos

Para cumplir con todas las funcionalidades comentadas en el punto anterior, se deberán cumplir los siguientes objetivos específicos:

a) Comparar distintos protocolos de conexión a escritorios virtuales para encontrar alguno que utilice pocos recursos, sin que esto implique sacrificar la calidad de la experiencia del usuario.
b) Buscar un sistema de virtualización de servidores básico que esté preparado para tener máquinas virtuales en Windows y en Linux. Para este caso, se valorará que el sistema sea de código libre para poder adaptarlo al máximo a las necesidades del proyecto. Se valorará también el hecho de que el sistema sea compatible con la alta disponibilidad, para prever el caso en que el servidor se estropeara, pudiendo seguir funcionando todo el sistema en otro de los servidores operativos.
c) Modificar el sistema para que se puedan generar y eliminar escritorios virtuales automáticamente, según la configuración que se desee.
d) Diseñar algún sistema para que puedan actualizarse los clientes de forma centralizada, de manera que los técnicos, no tengan que desplazarse hasta el lugar donde están instalados. Además, este sistema de actualización centralizado permitiría realizar un cambio en la configuración de todos los clientes en tan solo unos pocos segundos.
e) Crear un software cliente-servidor. Esto permitirá que los clientes se puedan conectar a los escritorios virtuales del servidor, según los privilegios que tenga el usuario identificado en el cliente.
f) Crear una interfaz para gestionar y analizar toda la información de los escritorios virtuales de forma clara y concisa.
g) Añadir al software cliente-servidor un mecanismo de autentificación en dos pasos. Esta opción se diseñará para los clientes Raspberry Pi y permitirá la identificación de los usuarios mediante su identificador/contraseña y su etiqueta NFC.

## 1.3 Organización de la memoria

La memoria del trabajo se organiza en 5 capítulos. El capítulo 1 ya lo hemos visto y contiene una breve descripción de la motivación del trabajo a realizar, junto con los objetivos del mismo.

El capítulo 2 describe brevemente los trabajos anteriores relacionados con la temática del proyecto, explicando la tecnología de la virtualización, cómo esta se aplica a la virtualización de escritorios y cuál ha sido la solución elegida para este proyecto.

El capítulo 3 muestra todo el entorno que se ha configurado para implementar la solución planteada. Este capítulo es el más denso de toda la memoria, ya que describe todas las tecnologías usadas, así como los dispositivos utilizados y los algoritmos que se han diseñado.

El capítulo 4 compara la solución planteada en este proyecto con la solución que tenían antes en el Ayuntamiento de Sant Joan Despí. Esta comparación se fundamenta en un análisis económico y en la valoración del usuario.

Por último, el capítulo 5 contiene las conclusiones del trabajo y el 6, las posibilidades de trabajo futuro que genera este proyecto.

La memoria también incluye varios anexos para no desviar la atención del hilo principal, si se quiere tener una visión general. En ellos se detalla la información más técnica del trabajo, aunque recomendamos su lectura a medida que vayan apareciendo las referencias.

# 2. Estado del arte

## 2.1 La virtualización

En sentido general, cuando se habla de virtualización, a lo que se refiere es a la virtualización de servidores, lo que significa dividir un servidor físico en varios servidores virtuales.

Cada máquina virtual puede interactuar de forma independiente con otros dispositivos, aplicaciones, datos y usuarios, como si se tratara de un recurso físico independiente.

Diferentes máquinas virtuales pueden ejecutar diferentes sistemas operativos y múltiples aplicaciones al mismo tiempo, utilizando un solo equipo físico. Debido a que cada máquina virtual está aislada de otras máquinas virtuales, en caso en que ocurra el bloqueo de alguna de ellas, esto no afecta al resto de máquinas virtuales.

Esta tecnología se basa en el hipervisor (en inglés *hypervisor*) o monitor de máquina virtual (*virtual machine monitor*), que es una plataforma que permite aplicar diversas técnicas de control de virtualización para utilizar, al mismo tiempo, diferentes sistemas operativos en una misma computadora. Esta plataforma, se encuentra entre el hardware y el sistema operativo, separando el sistema operativo y las aplicaciones del hardware. El hipervisor asigna la cantidad de acceso que tendrá cada máquina virtual al procesador, memoria, disco duro y otros recursos.

La virtualización se introdujo por primera vez en la década de 1960 por IBM para impulsar la utilización de grandes sistemas (*mainframe*) dividiéndolos en máquinas virtuales separadas lógicas que podían ejecutar múltiples aplicaciones y procesos al mismo tiempo. En los años 1980 y 1990, este modelo de *mainframe* centralizado y compartido, dio paso a un modelo distribuido (la computación cliente-servidor) en el cual muchos servidores independientes x86 de bajo coste eran capaces de ejecutar aplicaciones específicas.

La virtualización se desvaneció como centro de atención durante un tiempo, aunque ahora es una de las últimas tendencias en la industria una vez más, ya que las organizaciones tienen por objeto aumentar la utilización, la flexibilidad y la rentabilidad de sus recursos

informáticos. VMware, Citrix, Microsoft, IBM, Red Hat y muchos otros proveedores ofrecen soluciones de virtualización.

## 2.2 Retos de la virtualización

### Índices de utilización más altos

Antes de la virtualización, los índices de utilización del servidor y almacenamiento en los centros de datos de la empresa eran inferiores al 50% (de hecho, del 10% al 15% de los índices de utilización fueron los más comunes). A través de la virtualización, las cargas de trabajo pueden ser encapsuladas y transferidas a los sistemas inactivos o sin uso, esto significa, que los sistemas existentes pueden ser consolidados y las compras de capacidad adicional del servidor pueden ser retrasadas o evitadas.

### Consolidación de Recursos

La virtualización permite la consolidación de múltiples recursos de TIC[16]. Más allá de la consolidación de almacenamiento, la virtualización proporciona una oportunidad para consolidar la arquitectura de sistemas, infraestructura de aplicación, datos y base de datos, interfaces, redes, escritorios, e incluso procesos de negocios, resultando en ahorros de coste y mayor eficiencia.

### Uso / coste menor de energía

La electricidad requerida para que funcionen los centros de datos de clase empresarial ya no está disponible en suministros ilimitados, y el coste está en una espiral ascendente. Por cada euro gastado en un servidor hardware, un euro adicional es gastado en energía (incluyendo el coste de los servidores en función y los sistemas de refrigeración). Utilizando la virtualización, se hace posible reducir el consumo total de energía y ahorrar dinero de una manera significativa.

### Ahorro de espacio

Las dimensiones de los servidores es un serio problema en la mayoría de los centros de datos empresariales, pero la expansión del centro de datos no es siempre una opción, con los costes de construcción promediando miles de euros por metro cuadrado. La virtualización

[16] Tecnologías de la información y la comunicación.

puede aliviar la tensión mediante la consolidación de muchos sistemas virtuales en menos sistemas físicos.

**Recuperación de desastre / continuidad del negocio**

La virtualización puede incrementar la disponibilidad de los índices del nivel de servicio en general y, proporcionar nuevas opciones de soluciones para la recuperación de desastres.

**Costes de operación reducidos**

La empresa de promedio gasta 8 euros en mantenimiento por cada euro invertido en una nueva infraestructura. La virtualización puede disminuir esta relación, además de reducir la carga total del trabajo administrativo, provocando un recorte del total de los costes de operación.

## 2.3 Ventajas de la virtualización

Existen muchos beneficios en la consolidación de servidores Linux o Windows mediante el aprovechamiento de los diferentes productos de virtualización de servidores existentes en el mercado. A continuación enumeramos algunos de los beneficios que aporta esta tecnología:

a) Reutilización de hardware existente (para utilizar software más moderno) y optimización de los recursos de hardware.[17]
b) Rápida incorporación de nuevos recursos para los servidores virtualizados.
c) Reducción proporcional de los costes de espacio y consumo respecto al índice de consolidación logrado (Estimación media 10:1).
d) Administración global centralizada y simplificada.
e) Nos permite gestionar nuestro CPD[18] como un conjunto de recursos o agrupación de toda la capacidad de procesamiento, memoria, red y almacenamiento disponible en nuestra infraestructura.
f) Mejora en los procesos de clonación y copia de sistemas. Mayor facilidad para la creación de entornos de test que permiten poner en marcha nuevas aplicaciones sin impactar a la producción, agilizando el proceso de las pruebas.

[17] https://es.wikipedia.org/wiki/Virtualización#cite_note-6
[18] Centro de procesado de datos o centro de datos, emplazamiento físico donde se instalan los sistemas informáticos y sus componentes asociados en una empresa.

g) Aislamiento: un fallo general de sistema de una máquina virtual no afecta al resto de máquinas virtuales.
h) Mejora del coste total de propiedad también llamado TCO[19] y mejora del retorno sobre la inversión definido como ROI[20].
i) Soporte del beneficio directo en la reducción del hardware necesario y de los costes asociados.
j) Reduce los tiempos de parada.
k) Migración en tiempo real de máquinas virtuales (sin pérdida de servicio) de un servidor físico a otro, eliminando la necesidad de paradas planificadas para el mantenimiento de los servidores físicos.
l) Balanceo dinámico de máquinas virtuales entre los servidores físicos que componen el conjunto de recursos, garantizando que cada máquina virtual se ejecute en el servidor físico más adecuado y, proporcionando un consumo de recursos homogéneo y óptimo en toda la infraestructura.
m) Contribución al medio ambiente por reducir el consumo de energía en los servidores físicos.[21]

En definitiva, hay que destacar que al beneficiarse de la virtualización de servidores, se puede facilitar la mejora de la eficiencia de un centro de datos, así como reducir el coste de su mantenimiento.

## 2.4 Tipos de virtualización

### 2.4.1 Virtualización completa

Tal y como se interpreta en la figura 1, esta virtualización emplea un hipervisor para compartir el hardware subyacente. Las máquinas virtuales son totalmente independientes y eso supone una gestión de los recursos mucho más sencilla pero una pérdida de la optimización de su uso. Las soluciones más conocidas son:

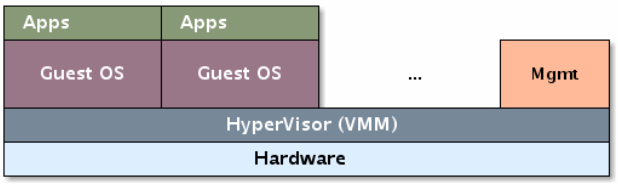


Figura 1 – Diagrama de virtualización completa

[19] Método de cálculo diseñado para ayudar a los usuarios y a los gestores empresariales a determinar los costes directos e indirectos, así como los beneficios, relacionados con la compra de equipos o programas informáticos.
[20] Razón financiera que compara el beneficio o la utilidad obtenida en relación a la inversión realizada
[21] https://es.wikipedia.org/wiki/Virtualización#cite_note-7

### KVM / QEMU[22]


Es una solución de virtualización completa para Linux en hardware x86 con extensiones de virtualización (Intel VT o AMD-V). Con KVM, podemos hacer funcionar múltiples máquinas virtuales sin modificar las imágenes de Linux o Windows que se estén ejecutando. KVM es parte de RedHat Emerging Technologies[23] (ET).

### Xen[24]


Es un monitor de máquina virtual que permite, a múltiples sistemas operativos, ejecutarse simultáneamente en el mismo hardware del ordenador. Xen ha sido la solución preferida para las distribuciones RedHat[25] desde 2005. El kernel[26], versión 2.6.18 ha abandonado el soporte para Xen, pero nuevamente se han agregado módulos/modificaciones al kernel en desarrollo, desde el 2.6.37 para DomU (huésped) y desde el 3.0 para Dom0. Por lo tanto, el soporte del anfitrión Dom0 de Xen, que fue abandonado después de Fedora 8, ha sido reintroducido, desde Fedora 16.

[22] http://www.linux-kvm.org/page/Main_Page
[23] https://www.redhat.com/en/about/blog/authors/emerging-technologies-team
[24] http://www.xenproject.org/
[25] http://www.redhat.com/en
[26] Es un software que constituye una parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado (conocido también como núcleo)

### VirtualBox[27]


Es una solución de virtualización completa para hardware x86 y AMD64/Intel64. Sun Microsystems comenzó este proyecto, que ahora está completamente soportado por Oracle. Existe un esquema de licenciamiento dual, entre los cuales está GPLv2[28]. VirtualBox es una de las soluciones de virtualización completa más extendida.

### VMware ESXi[29]


Es una plataforma de virtualización a nivel de centro de datos producido por VMware, Inc. El componente de su producto VMware Infraestructure se encuentra a un nivel inferior de la capa de virtualización, haciendo de hipervisor, aunque con herramientas y servicios de gestión autónomos e independientes.

Está compuesto de un sistema operativo autónomo que proporciona: el entorno de gestión, administración y ejecución al software hipervisor; y de los servicios y servidores que permiten la interacción con el software de gestión y administración de las máquinas virtuales.

[27] https://www.virtualbox.org/
[28] La licencia GPL puede ser usada por cualquiera, ya que su finalidad es proteger los derechos de los usuarios finales (usar, compartir, estudiar y modificar)
[29] https://www.vmware.com/products/vsphere-hypervisor

### 2.4.2 Paravirtualización

Tal y como se interpreta en la figura 2, consiste en ejecutar sistemas operativos (guests) sobre otro sistema operativo que actúa como hipervisor (host). Los guests tienen que comunicarse con el hipervisor para lograr la virtualización.

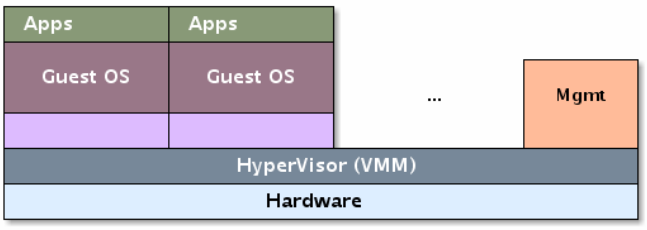



Figura 2 – Diagrama de paravirtualización

Entre las ventajas de este enfoque, se puede mencionar la mejora en el rendimiento respecto a la virtualización completa obteniendo además todas sus ventajas. Su desventaja es que los sistemas operativos instalados en las máquinas virtuales, deben ser modificados para funcionar en este esquema.

Las implementaciones que existen son las mismas que las de la virtualización completa, pero con la diferencia de que los sistemas operativos de las máquinas virtuales instaladas, deben ser modificados mediante la instalación de ciertos drivers que permiten la optimización del algún recurso. Entre estos recursos se encuentran, el disco duro, las interfaces de red y el acceso a la memoria RAM[30].

### 2.4.3 La virtualización a nivel de sistema operativo

También conocida como *Dockers*, tal y como se interpreta en la figura 3, en este esquema no se virtualiza el hardware y se ejecuta una única instancia del sistema operativo (un único kernel). Los distintos procesos pertenecientes a cada servidor virtual se ejecutan aislados del resto.

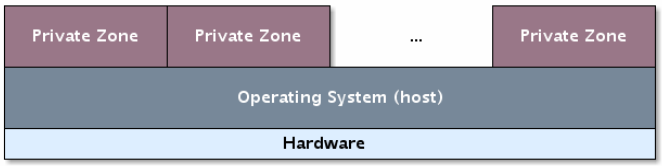



Figura 3 – Diagrama de virt. de sistema operativo

La ventaja de este enfoque es la separación de los procesos de usuario prácticamente sin pérdida en el rendimiento. Con lo que en cuanto a la optimización del rendimiento es la mejor de las virtualizaciones, por otro lado, al compartir el mismo kernel entre

[30] La memoria de acceso aleatorio (Random Access Memory, RAM) se utiliza como memoria de trabajo de computadoras para el sistema operativo, los programas y la mayor parte del software.

todas las máquinas virtuales, provoca que algunos cambios (como las activaciones de módulos del kernel) sean más complicados de hacer que en las otras virtualizaciones. Además eso impide por ejemplo ejecutar en el mismo host dos sistemas operativos que requieran de un kernel distinto. Algunas implementaciones son:

**OpenVZ**[31]


Es una tecnología de virtualización a nivel de sistema operativo. Permite múltiples instancias de sistemas operativos aislados, conocidos como VPS (*Virtual Private Server*).

A diferencia de otros sistemas de virtualización como VMWare, Xen, o KVM, OpenVZ es restrictivo a la elección del sistema operativo, ya que sólo admite GNU Linux. A pesar de esto, OpenVZ ofrece mayor rendimiento, escalabilidad, densidad, administración de recursos dinámicos y facilidad de manejo. Está basado en el software privativo Virtuozzo[32] pero OpenVZ en cambio es GPLv2.

**LXC (Linux Containers)**[33]


Es muy parecido a OpenVZ pero tiene la ventaja de que no requiere un micro-kernel o un parche completo al kernel original de Linux, ya que se basa exclusivamente en una tecnología de espacios y grupos (*cgroups*) que es nativa del kernel Linux desde la versión 2.6.30. Así pues, cualquier equipo con un kernel 2.6.30 (o superior), *cgroups* habilitado y los scripts de LXC, puede comenzar a crear contenedores.

[31] https://openvz.org/Main_Page
[32] https://openvz.org/Virtuozzo
[33] https://linuxcontainers.org/

**Linux-VServer**[34]


Está obsoleto ya que las últimas noticias se remontan al 2009.

### 2.4.4 Tipo de virtualización elegida

En este proyecto, como se explicará en los puntos posteriores, todas las máquinas en que se desee usar un sistema operativo Linux (en este caso Debian) se virtualizarán mediante OpenVZ mientras, que las que deban contener un sistema operativo Windows 8.1, se virtualizarán mediante KVM/QEMU.

En el anexo A se detallan cada una de las dos virtualizaciones para los dos sistemas operativos y se comenta el problema que hubo con la versión 3.4 del entorno de virtualización elegido, Proxmox VE.

## 2.5 Virtualización de escritorios

Básicamente la virtualización de escritorios consiste en usar una de las máquinas virtuales de nuestro entorno (tanto en virtualización completa, como paravirtualizada o si esta virtualizada a nivel de sistema operativo) como un escritorio virtual para un usuario identificado en un cliente.

De esta forma se pueden usar dispositivos que actúen como clientes, con bajos recursos (también llamados clientes ligeros o ThinClients) para ejecutar programas que necesiten de muchos recursos, ya que el programa no usa los del cliente sino los del servidor que están asignados a esa máquina virtual. A continuación se explica la historia de los escritorios virtuales y se comentan sus mayores ventajas así como las soluciones más conocidas.

### 2.5.1 Descripción

La virtualización de escritorio es un término relativamente nuevo, introducido en la década de los 90, que describe el proceso de separación entre la máquina física y el escritorio, que engloba los datos y programas que utilizan los usuarios para trabajar. El escritorio virtualizado es almacenado remotamente en un servidor central en lugar del disco duro del ordenador personal. Esto significa

[34] http://linux-vserver.org/Welcome_to_Linux-VServer.org

que cuando los usuarios trabajan en su escritorio desde su portátil u ordenador personal, todos sus programas, aplicaciones, procesos y datos se almacenan y ejecutan centralmente, permitiendo a los usuarios acceder remotamente a sus escritorios desde cualquier dispositivo capaz de conectarse remotamente al escritorio, tales como un portátil, PC, smartphone o un cliente ligero.

La experiencia que tendrá el usuario, está orientada para que sea idéntica a la de un PC estándar, desde la misma oficina o remotamente.

#### 2.5.1.1 Ventajas

**Aumenta la seguridad de los escritorios y disminuye los costes de soporte**

Las organizaciones tienen una gran cantidad de problemas otorgando permisos a los usuarios de sus equipos informáticos. Resulta complicado otorgar los mínimos permisos posibles y ofrecer a los usuarios un entorno de trabajo flexible con todas las funcionalidades que requieren para realizar su trabajo, sobre todo a aquellos usuarios que viajan. La virtualización del escritorio posibilita centralizar los escritorios en servidores y gestionar dichos escritorios individualmente de manera remota. Esto provee a los usuarios una experiencia de escritorio completa y, permite al personal de TIC gestionar los escritorios virtualmente en lugar de físicamente. También disminuye radicalmente el coste de mantenimiento de los escritorios ya que, no se permite a los usuarios modificar nada del sistema operativo, principal causa de los problemas en las organizaciones.

**Reduce los costes generales de hardware**

La virtualización de escritorio implica que el usuario no tiene que tener equipos de última tecnología, porque todas las aplicaciones del escritorio son ejecutadas centralmente en un servidor remoto. Esto significa que los equipos que los usuarios están utilizando para conectarse al servidor tienen un periodo de vida mayor. Estos costes pueden reducirse aún más si para acceder a los escritorios virtuales utilizamos clientes ligeros, que son mucho menos costesos que los PC tradicionales.

**Alternativa ecológica**

Los escritorios virtuales almacenados en el servidor central son una alternativa ecológica a los PC tradicionales. Los PC consumen alrededor de tres veces más energía de lo que consumirían virtualizados en un CPD. Uno de los mayores beneficios de la virtualización de servidores es el ahorro de energía que se produce cuando varios servidores a baja utilización, ejecutando aplicaciones, se combinan en una única pieza física de hardware.

**Mejorar la seguridad de los datos**

La virtualización de escritorios hace que todos los datos de los usuarios de los escritorios, y por lo tanto de las organizaciones, se almacenen centralmente en sus servidores, absolutamente nada se almacena a nivel local. De esta forma, si el empleado pierde o le roban el portátil, no se pierden los datos y tampoco dichos datos pasan a manos peligrosas. Esta funcionalidad permite el cumplimiento de la LOPD (Ley Orgánica Protección de Datos), de obligado cumplimiento en todas las organizaciones y, especialmente en las Administraciones Públicas.

#### 2.5.1.2 Soluciones de virtualización de escritorios

##### 2.5.1.2.1 Libres

**LTSP**[35]

Son un conjunto de aplicaciones de servidor que proporcionan la capacidad de ejecutar Linux en computadores de pocas prestaciones de velocidad o de bajo coste, permitiendo reutilizar equipos que actualmente resultan obsoletos, debido a los altos requisitos que piden los sistemas operativos. LTSP se distribuye bajo licencia GPLv2 de software libre. La última versión estable es la 5.0

El sistema de funcionamiento del LTSP consiste en repartir por medio de la red, el núcleo Linux que es ejecutado por los clientes y que, posteriormente ejecutarán secuencias de scripts típicos de una mini

[35] http://www.ltsp.org/

distribución. Los clientes podrán acceder a las aplicaciones por medio de una consola textual o por un servidor gráfico que se comparte utilizando el protocolo XDMCP[36]. Actualmente uno de los campos donde se utiliza bastante LTSP es en la educación, debido al bajo coste de implantación. Actualmente, la compatibilidad de este servidor de terminales se ha extendido a todas las plataformas Linux de uso común, y su rendimiento y capacidad ha mejorado con la última versión. Otro uso, aunque con mayor complejidad de implantación, es para el manejo y gestión de estaciones de trabajo de ofimática para empresas u otras aplicaciones que no requieran alto rendimiento gráfico.

En el anexo B.1 Se enumeran las principales características de esta solución.

La gran limitación que tiene este software es que sólo puede distribuir escritorios de Linux. En la actualidad, esto supone un gran problema, sobre todo en las empresas, donde gran parte del software que se utiliza está diseñado para ser usado en Windows y muchas veces no existe una versión para Linux.

#### 2.5.1.2.2 Propietarias

**VMWare Horizon View**


Es la solución que plantea VMWare para la distribución de escritorios virtuales basándose en su virtualización de servidores VMWare ESXi. Está formada por un conjunto de herramientas que simplifican la gestión de escritorios y aplicaciones a la vez, que aumentan la seguridad y el control. Proporciona una experiencia personalizada de alta fidelidad a los usuarios finales en todo tipo de sesiones y dispositivos. Permite una mayor disponibilidad y agilidad de los servicios de escritorio que los ordenadores tradicionales no pueden igualar. Les permite acceder a los escritorios desde más dispositivos y

[36] Protocolo utilizado en redes para comunicar un ordenador servidor que ejecuta un sistema operativo con un gestor de ventanas basado en X, con el resto de clientes que se conectarán a éste con propósitos interactivos

ubicaciones, otorgando a los administradores de red un mayor control mediante políticas.

En el anexo B.2 Se enumeran las principales características de esta solución.

Esta solución es muy completa y de alta calidad pero los altos precios de sus licencias que se han de renovar anualmente hacen que solo se pueda usar en empresas con grandes recursos económicos.

**Citrix XenDesktop[37]**


Es la solución que plantea Citrix basada en la virtualización de servidores Xen. Puede entregar de forma segura aplicaciones individuales, ya sean de Windows, web o SaaS[38]. También puede entregar escritorios completos. No importa si el usuario está en la oficina, viajando o en casa. Con un PC, Mac, tablet, smartphone, portátil o un cliente ligero, el usuario podrá acceder a su escritorio en cualquier momento, ofreciéndole en todos ellos una experiencia de usuario excelente.

En el anexo B.3 Se enumeran las principales características de esta solución en su última versión.

Esta solución de la misma forma que la anterior también es muy completa pero las licencias, igualmente, son a precios muy elevados y obligan a una renovación anual. Por lo tanto tampoco se pueden implementar en sitios con pocos recursos económicos.

[37] https://www.citrix.com/products/xendesktop/
[38] Software as a Service, es un modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación (TIC), a los que se acceden remotamente.

## 2.6 Comparación del producto con el estado del arte

Las dos soluciones propietarias que se han comentado en el punto anterior, son las más usadas en la mayoría de negocios y otras entidades que optan por la implementación de escritorios virtuales para sus empleados.

Los costes de estos dos sistemas, por otro lado, son muy elevados, limitando el acceso a este tipo de soluciones a los clientes que no tengan un presupuesto elevado destinado a este fin. La idea de hacer este trabajo fue la de intentar ofrecer una solución completa de un servidor de escritorios virtuales, parecida a la de LTSP, en cuanto a basarse en software libre, pero que además pudiera ejecutar escritorios virtuales de Windows.

Hemos buscado algún tipo de solución ya implementada, que encaje con los requisitos anteriormente comentados, pero no la hemos encontrado. Las únicas soluciones que se le parecen, siempre requieren del pago, de cantidades elevadas, para obtener algún tipo de licencia, que además es por cliente y se renueva cada cierto tiempo.

Finalmente encontramos la solución del entorno de virtualización de servidores Proxmox VE, que permite paravirtualizar mediante KVM/QEMU y virtualizar a nivel de sistema operativo mediante OpenVZ. Aun así como este entrono no está previsto para la entrega de escritorios virtuales, sino simplemente para la virtualización de servidores, hemos tenido que modificar su código para que el entorno funcione como un servidor de escritorios virtuales. El hecho de que el host esté basado en Debian ha permitido que nuestra tarea en la creación de scripts para la generación de los escritorios virtuales fuera más sencilla al estar muy familiarizados con este sistema operativo.

Al no necesitar del pago de ninguna licencia, creemos que la solución ofrecida en este trabajo, puede ser realmente una solución alternativa para clientes que no puedan permitirse los entornos que requieren licencias.

La solución propuesta, además permite optimizar los escritorios virtuales que se ofrecen de Linux, al virtualizarse mediante OpenVZ,

que permite aprovechar casi al 100% los recursos dedicados del servidor a ese escritorio virtual en concreto.

Finalmente, se decidió añadir un mecanismo de autentificación en dos pasos para un tipo de clientes ligeros (Raspberry Pi), de tal forma que el sistema permitiera definir a un grupo de usuarios con un grado más elevado de seguridad para poder acceder a sus escritorios. Este punto será tratado al detalle en uno de los próximos apartados.

# 3. Entorno del proyecto

## 3.1 Descripción

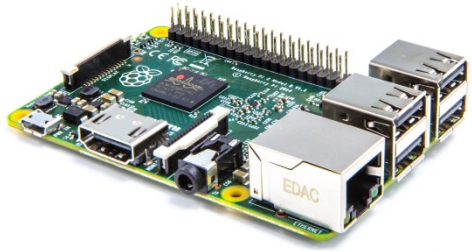
El ayuntamiento de Sant Joan Despí nos ha proporcionado un servidor con recursos suficientes como para poder probar la solución que este trabajo plantea. Se trata de un servidor con 2 interfaces de Red Intel Pro a 1000 Mbps, contiene dos procesadores Xeon CPU E5502[39], 48 GB de memoria RAM, y 5 discos duros de 1TB cada uno.

El switch utilizado para interconectar los ordenadores clientes con el servidor es un Dlink DGS-108[40] que soporta un ancho de banda de 2 Gbps por cada puerto, ya que son full dúplex.

El firewall del que separa la WAN del servidor es un Dlink DFL-860E[41] que el ayuntamiento no tenía en uso y se ha implementado para darle más seguridad en el sistema.

Los clientes que se usaran en nuestro entorno pueden ser de 3 tipos:

**Raspberry Pi 2 modelo B[42]**



Es un ordenador de placa reducida de bajo coste (aproximadamente 50€) desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de computación en las escuelas. Las características de este dispositivo son las siguientes:

a) Procesador: Quad-Core Cortex A7 a 900MHZ
b) RAM: 1GB

[39] http://ark.intel.com/products/37092/Intel-Xeon-Processor-E5502-4M-Cache-1_86-GHz-4_80-GTs-Intel-QPI
[40] http://www.dlink.com/uk/en/business-solutions/switching/unmanaged-switches/desktop/dgs-108-8-port-gigabit-ethernet-switch
[41] http://www.dlink.com/es/es/support/product/dfl-860-netdefend-utm-firewall-860
[42] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

c) Puertos
    a. 4 x USB 2.0
    b. 1 x 40 GPIO[43] pin
    c. 1 X HDMI
    d. 1 x Ethernet
    e. 1 x Combo audio/mic
    f. 1 x Interfaz de cámara (CSI)
    g. 1 X Interfaz de Pantalla (DSI)
    h. 1 x Micro SD
    i. 1 x Núcleo Gráfico 3D

La conexión de 40 GPIO nos permite conectar periféricos económicos que en ordenadores portátiles o en un PC no se podrían. Es por eso que el sistema de doble autenticación que se ha implementado en este proyecto sólo se ha diseñado para que funcione con la Raspberry Pi. De esta forma, este tipo de usuarios que pertenecen a este grupo de seguridad, solo podrán acceder al sistema desde una Raspberry Pi.

Otro motivo por el que se ha elegido este dispositivo como cliente ligero es el consumo que éste tiene. La Raspberry pi 2 modelo B consume sólo 900 mA a 5 V lo que sería una potencia de 4,5 W. Si lo comparamos con un ordenador medio que tiene un consumo de unos 180 W nos damos cuenta que el ahorro energético es abismal, concretamente la Raspberry consume un 2,5% del total que consume un ordenador estándar.

[43] General Purpose Input/Output, Entrada/Salida de Propósito General, es un pin genérico en un chip, cuyo comportamiento se puede controlar por el usuario en tiempo de ejecución

**Portátiles**

Se consideran candidatos para ser incorporados, todos aquellos portátiles que por sus bajos recursos hayan quedado apartados del entorno productivo. Los portátiles deben tener como mínimo 512 MB de RAM y se han de conectar a la red mediante una tarjeta de red por cable con una velocidad mínima de 100 Mbps.

**PC**

Se consideran candidatos para ser incorporados, todos aquellos ordenadores con muy bajos recursos que tengan al menos 512 MB de RAM. Se requiere también que tenga una tarjeta de red de una velocidad mínima de 100 Mbps.

## 3.2 Diagrama

A continuación la figura 4 muestra el entorno físico que describe todo el entorno en el que se ha implementado la propuesta de virtualización descrita en este proyecto.

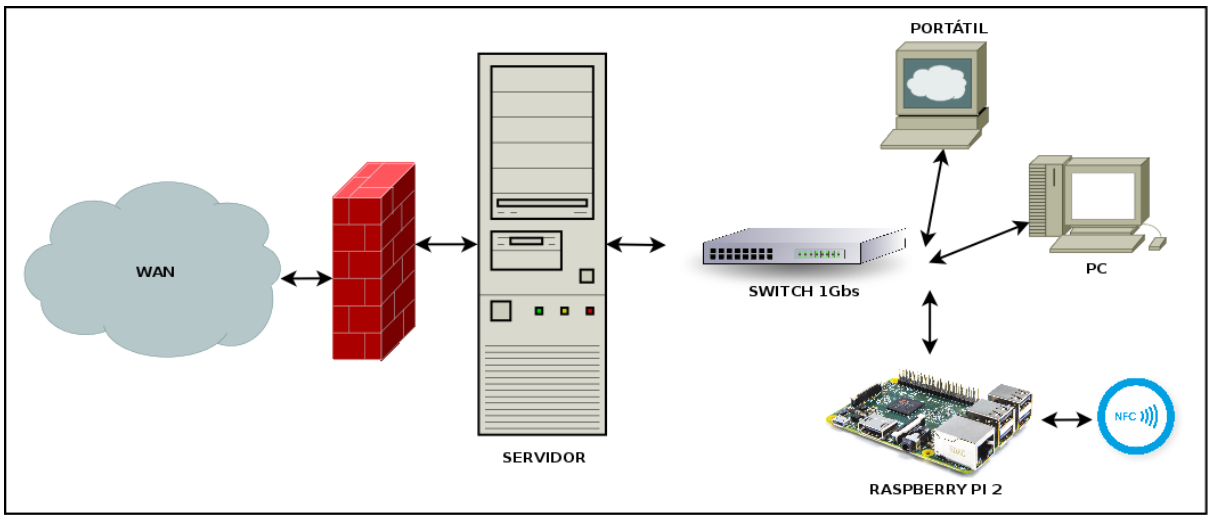



Figura 4 – Diagrama del entorno

Como se puede observar en este diagrama, el servidor se sitúa entre los clientes ligeros y la WAN. Entre la WAN y el servidor, como hemos comentado antes, colocamos un firewall para proteger el servidor de posibles ataques de piratas informáticos de la WAN. Además, en este firewall configuramos una VPN[44] para poder acceder al sistema desde fuera del ayuntamiento y así poder trabajar desde casa, aunque esta configuración no se describe al no ser una parte importante del proyecto.

En este servidor, como hemos comentado, está instalado el entorno Proxmox VE que consiste en un sistema operativo Linux, concretamente Debian Wheezy[45], que tiene un kernel adaptado para poder virtualizar máquinas mediante KVM y OpenVZ.

[44] Una red privada virtual o VPN, de las siglas en inglés de Virtual Private Network, es una tecnología de red que permite una extensión segura de la red local (LAN) sobre una red pública o no controlada como Internet

[45] https://www.debian.org/releases/wheezy/

Para entender un poco mejor el funcionamiento del servidor vamos a describir qué máquinas virtuales están funcionando en su interior. La figura 5 muestra en detalle cada una de éstas máquinas virtuales.

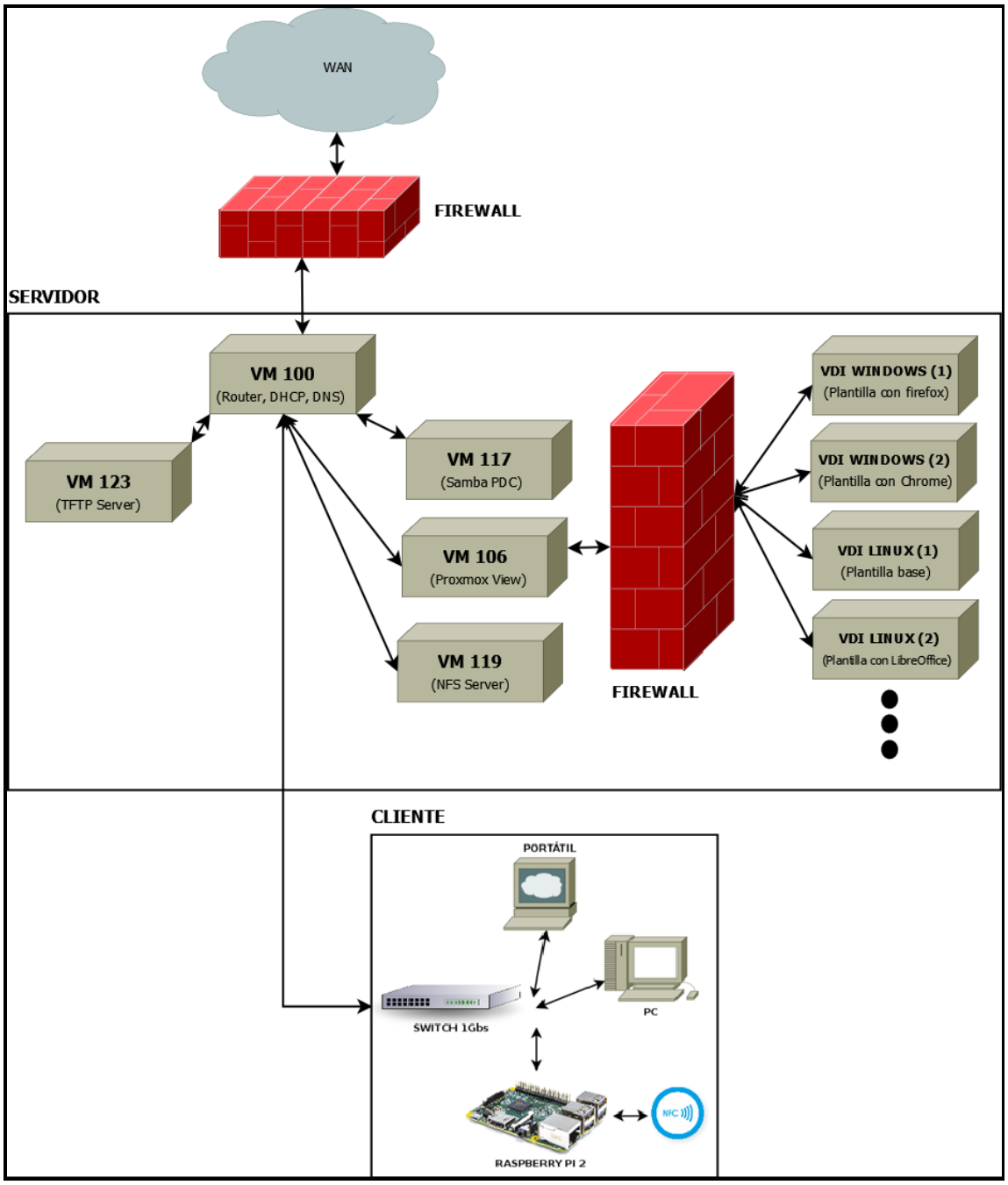



Figura 5 – Diagrama detallado del servidor

Tal y como se puede observar en el diagrama anterior, disponemos de una clara organización cliente-servidor. El servidor, está formado por unas máquinas que se usarán para la administración del entorno (VM 100, VM 123, VM 117, VM 106 y VM 119) y que se detallarán

más adelante. Sólo destacaremos en este punto la VM 100, que es la que se encarga de enrutar todos los paquetes, y decidir si los paquetes deben ir hacia el entorno de escritorios virtuales o hacia la WAN.

En el entorno que se ha diseñado se pueden crear distintos tipos de escritorios virtuales que llamaremos plantillas (que pueden ser o Linux o Windows con el software instalado que queramos y que se pueden adaptar a un grupo de usuarios determinado). De esta forma, cuando creemos una plantilla nueva se creará un grupo asociado a esta plantilla. A continuación deberemos asignar el grupo de usuarios que queramos que pueda utilizarlo, al grupo que contiene esta plantilla. Así pues, el usuario en el próximo acceso que haga podrá elegir usar esa nueva plantilla. Más adelante se detallará todo este procedimiento.

En la figura 5 hay 4 máquinas en la parte derecha, que podrían ser muchas más. Estas máquinas son las plantillas (o VDI[46]) que no son otra cosa que escritorios virtuales que se han generado en ese instante. El firewall que separa estos escritorios del resto del sistema, es un firewall implementado por software en la VM 106 que permite controlar qué usuario puede acceder a un escritorio virtual determinado.

Como podemos ver, en este caso hay 2 plantillas con Windows, una que contiene Firefox instalado y otra que contiene Chrome. Además, hay dos plantillas de Linux, una sin nada instalado y otra con Libre Office.

## 3.3 Máquinas virtuales del entorno

Una máquina virtual es un software que simula a un ordenador y puede ejecutar programas como si fuese un ordenador real. Este software en un principio fue definido como un duplicado eficiente y aislado de una máquina física. La acepción del término actualmente incluye a máquinas virtuales que no tienen ninguna equivalencia directa con ningún hardware real.

[46] Virtual Desktop Infrastructure

En este proyecto se usan un total de 5 máquinas virtuales, virtualizadas mediante OpenVZ (ejecutan la versión Wheezy de Debian), para la gestión y administración tal y como se ha comentado en el punto anterior. Un resumen de la información técnica de las máquinas es el siguiente:

| Número | Nombre | IP | MAC[47] |
|---|---|---|---|
| *100* | Router | Interna (LAN):<br>192.168.2.1<br>Externa (WAN):<br>192.168.1.37 | Interna:<br>16:7D:EA:E1:D9:A9<br>Externa:<br>8E:7A:D2:20:F9:B4 |
| *117* | Samba[48] PDC[49] | 192.168.2.76<br>(samba.ajuntament) | 06:2A:AC:B4:D9:C6 |
| *119* | NFS[50] Server | 192.168.2.75 | 5E:17:F3:0E:8A:8E |
| *123* | TFTP[51] Server | 192.168.2.7 | 22:1A:BA:3E:8E:8C |
| *106* | Proxmox View | Interna (VDI):<br>192.168.3.1<br>Externa (LAN):<br>192.168.2.82<br>(view.ajuntament) | Interna:<br>92:08:E6:C8:DE:7F<br>Externa:<br>86:A2:8D:FF:C9:F8 |

[47] Identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una tarjeta o dispositivo de red
[48] Es una implementación libre del protocolo de archivos compartidos de Microsoft Windows.
[49] Primary domain controller, controlador de dominio.
[50] Es un protocolo de nivel de aplicación, según el Modelo OSI. Es utilizado para sistemas de archivos distribuido en un entorno de red de computadoras de área local
[51] Es un protocolo de transferencia muy simple semejante a una versión básica de FTP.

Cada una de las máquinas virtuales expuestas tiene un papel imprescindible en el funcionamiento del entorno. A continuación explicaremos a grandes rasgos, cuál es el rol de cada una de ellas. Para una descripción más exhaustiva de estas máquinas, revisar el anexo C. Cada una de estas máquinas ha sido diseñada y configurada desde una instalación de Debian Wheezy base (sin ningún paquete instalado) para adaptarla a las necesidades del ayuntamiento optimizando los recursos disponibles.

### Máquina virtual 100 – Router, DNS[52] y DHCP[53]

Se trata de la máquina virtual más fundamental de todas. Hace la función de router encaminando todos los paquetes para que vayan hacia la LAN o hacia la WAN, también encamina los paquetes que tienen que ir hacia los escritorios virtuales. Además contiene el servidor DHCP y el DNS tal y como se explican extensamente en el anexo C.1.

### Máquina virtual 117 – Samba PDC

Esta máquina virtual realiza la función de controlador de dominio, es decir de controlar todos los permisos que tienen los usuarios registrados en el entorno, mediante la asignación de los mismos en determinados grupos. Contiene la base de datos LDAPS[54] y un servidor web para poder consultar los permisos que tienen cuando el administrador del sistema lo desee. El controlador de dominio es compatible tanto con los escritorios virtuales de Linux como los de Windows. De esta forma si configuramos a un usuario en un grupo determinado, este heredará los permisos tanto si ejecuta un escritorio Windows como uno de Linux. En el anexo C.2 se explica su funcionamiento detallado.

[52] Es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada. Asocia información variada a nombres de dominios para que sea más fácil de recordar.
[53] Dynamic Host Configuration Protocol, es un protocolo de red que permite a los clientes de una red IP obtener sus parámetros de configuración automáticamente.
[54] LDAP son las siglas de Lightweight Directory Access Protocol, que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. La variante LDAPS hace que el tráfico entre cliente-servidor viaje mediante un túnel encriptado.

### Máquina virtual 119 – NFS Server

En esta máquina virtual se encuentran todos los datos de los perfiles móviles de los usuarios del entorno, es decir, es donde se almacenan todos los archivos que los usuarios vayan creando durante el uso de los escritorios virtuales, para que cuando inicien uno nuevo después de cerrar el que estaban usando, puedan seguir teniendo esos datos. En el anexo C.3 se explica su funcionamiento detallado.

### Máquina virtual 123 – TFTP Server

Se trata de la máquina virtual que tiene el servidor TFTP que permite a los cliente (portátiles y PC) realizar el arranque por PXE[55] para disponer de una configuración centralizada. Además, esta máquina también contiene el sistema base, completamente diseñado por nosotros además de los scripts, que cargarán los clientes cuando realicen el arranque por red del sistema operativo antes de iniciar la conexión al escritorio remoto. En el anexo C.4 se explica su funcionamiento detallado.

### Máquina virtual 106 – Proxmox View

Esta máquina virtual tiene dos funciones principales: la primera es realizar la función de firewall para tener el control de en qué escritorios virtuales puede acceder un usuario determinado. Además, también es la máquina que se encarga de generar, preparar, borrar y mostrar mediante una plataforma web totalmente diseñada por nosotros, todos y cada uno de los escritorios virtuales de nuestro entorno. Toda esta plataforma de configuración la hemos desarrollado exclusivamente para este proyecto.

Es interesante ver su funcionamiento detallado en el anexo C.5.

[55] Es un entorno para arrancar e instalar el sistema operativo en computadoras a través de una red, de manera independiente de los dispositivos de almacenamiento de datos disponibles (como discos duros) o de los sistemas operativos instalados.

## 3.4 Diagrama de flujo

El funcionamiento, tanto para los clientes Raspberry como para los portátiles, aunque no se realiza de la misma forma, se basa en el mismo principio, por eso a continuación solo se explicará para uno de ellos, la Raspberry, y en los puntos posteriores se explicará en detalle las diferencias de arranque entre uno y otro dispositivo.

Para entenderlo mejor se puede observar en la figura 6, el diagrama de flujo del sistema.

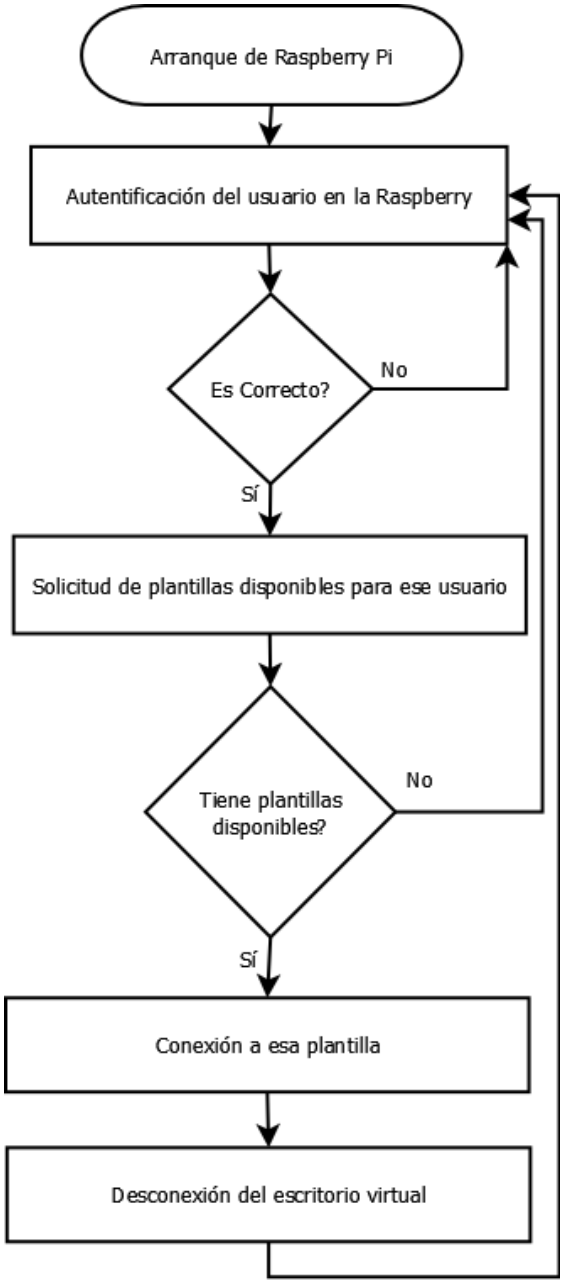



Figura 6 – Diagrama de flujo

A continuación se detalla este diagrama de forma exhaustiva.

### Arranque de Raspberry Pi

La Raspberry Pi solicita una IP al servidor DHCP y monta por NFS todo el sistema operativo que ejecutara el cliente ligero (Linux Debian Wheezy)

### Autentificación de usuario en la Raspberry

Una vez el sistema ha arrancado, aparece en la pantalla una ventana para introducir el usuario y la contraseña, tal y como se observa en la figura 7.

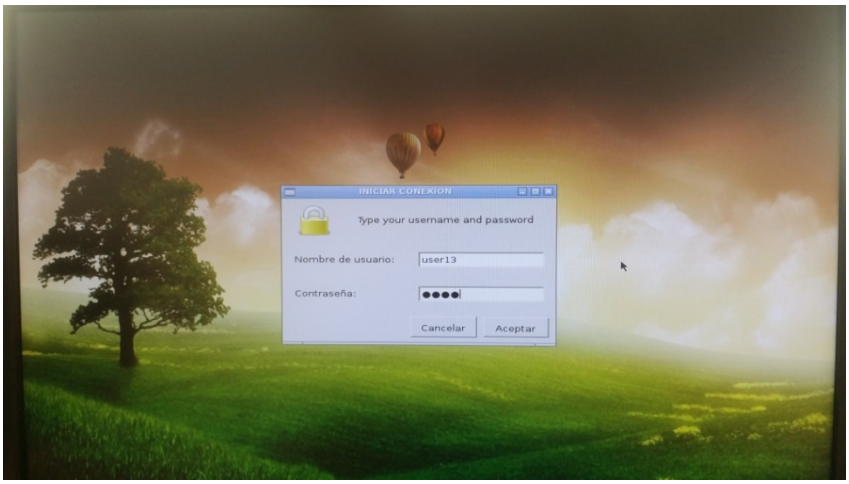



Figura 7 – Autentificación de usuario

En el caso de la Raspberry, además, si el usuario está en el grupo de doble autentificación, deberá colocar su etiqueta NFC encima de la Raspberry, para poder pasar a la siguiente fase. La figura 8 muestra la pantalla de espera que aparece mientras no se coloca la etiqueta.

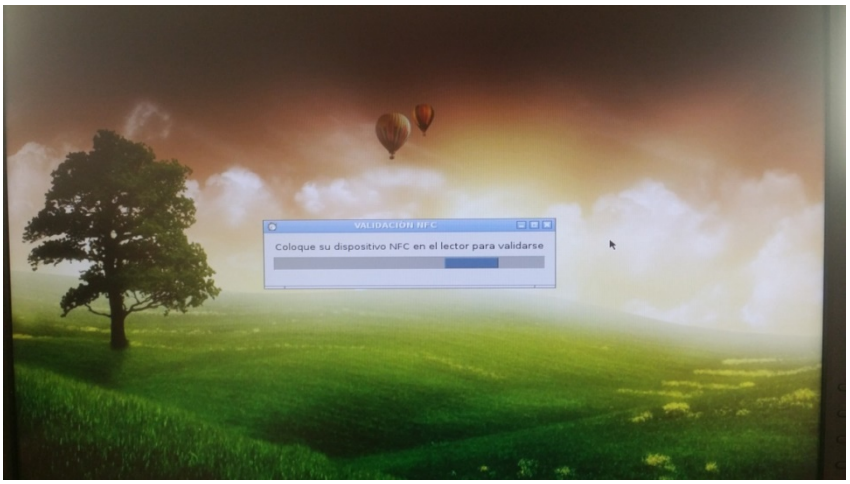



Figura 8 – Espera a que se coloque la etiqueta NFC encima de la Raspberry Pi

El usuario se validará mediante el protocolo LDAPS en el servidor de la máquina 117, si el usuario es correcto pasará a la siguiente fase, si no, volverá a pedir un usuario y una contraseña.

### Solicitud de plantillas disponibles para ese usuario

Una vez se ha autentificado el usuario, se consultará en la misma base de datos LDAP en qué grupos de plantillas está y se le dará a elegir en cuál de ellos quiere iniciar su escritorio virtual, tal y como se observa en la figura 9.

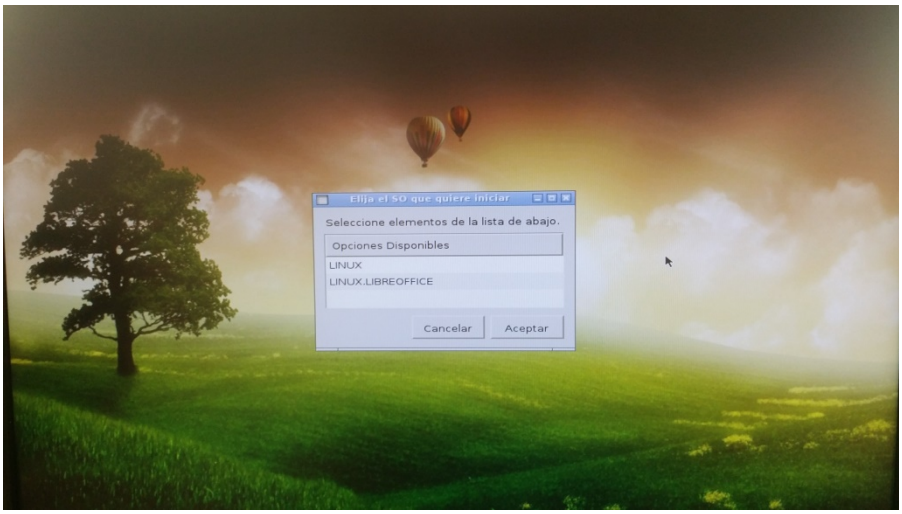



Figura 9 – Elección de plantilla

Una vez elegida la plantilla, empezará la asignación de un escritorio virtual de Proxmox View con esa plantilla, a ese usuario.

### Conexión a esa plantilla

Cuando ya se conoce cuál es la plantilla que el usuario quiere ejecutar, se inicia una conexión con la máquina virtual 106 (Proxmox View) en la que ésta le asignara una de las máquinas virtuales que hay preparadas con la plantilla que el usuario tiene. Toda la comunicación que se realiza entre el cliente ligero y la máquina 106, está encriptada mediante SSL[56], ya que todo el intercambio de datos se realiza mediante consultas https[57] para que el sistema sea más seguro y no se puedan interceptar paquetes con información confidencial.

El diagrama de flujo de esta conexión es el de la figura 10.

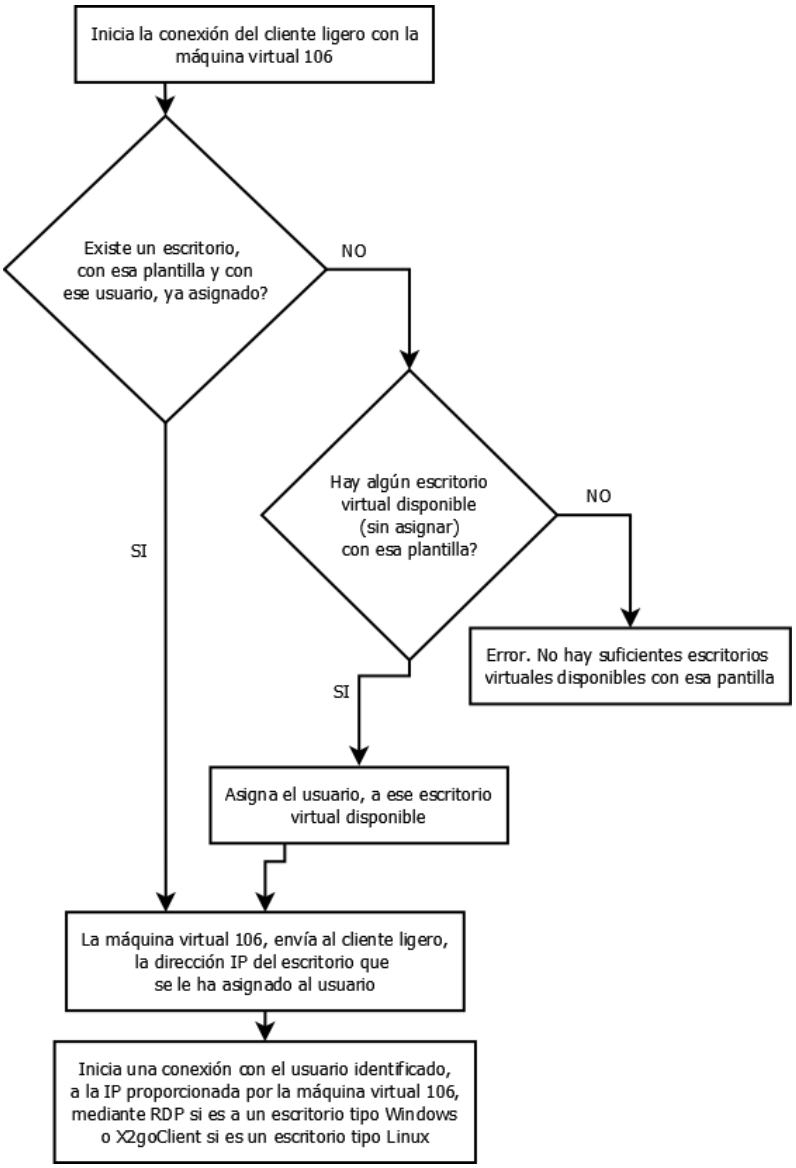



Figura 10 – Diagrama de flujo de la asignación de un escritorio a un usuario

[56] https://es.wikipedia.org/wiki/Transport_Layer_Security
[57] https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure

**Desconexión del escritorio virtual**

Cuando el usuario cierre sesión en su escritorio virtual volverá a la pantalla inicial para que introduzca sus datos de acceso, como si de un reinicio se tratara.

---

El script que realiza todas las acciones de este punto, y que ha sido diseñado completamente por nosotros para este proyecto, se encuentra en la documentación adjunta del trabajo.

## 3.5 Los clientes

En este apartado se detalla el funcionamiento de los clientes. Como se ha comentado anteriormente, en este entorno, se valoran 3 tipos de clientes: la Raspberry Pi, los ordenadores portátiles y los ordenadores fijos (PC) para el aprovechamiento de recursos. El funcionamiento de los ordenadores tanto portátiles como fijos es exactamente igual. Por ese motivo solo haremos la explicación para ordenadores de bajos recursos y se aplicará tanto a los ordenadores portátiles como a los fijos. Empezaremos describiendo este tipo de clientes y después explicaremos el funcionamiento en la Raspberry Pi.

### 3.5.1 Ordenadores con bajos recursos

Uno de los puntos clave en la implementación de este proyecto es el aprovechamiento de los ordenadores con bajos recursos, ya sean portátiles u ordenadores fijos (PC). Lo que sí que se requiere es que al menos tengan 512 MB de RAM y una tarjeta de red que funcione a una velocidad de 100 Mbps. En concreto, en el entorno que se ha simulado se usan tres ordenadores portátiles: uno de la marca Toshiba con 4GB de RAM y un procesador *Intel core Duo a 2,1 GHz*, otro de la marca HP con 2GB de RAM y un procesador *Intel Centrino Duo a 1,6 Ghz* y finalmente un Dell con 1GB de RAM y un procesador *Intel core Duo a 1,66 Ghz*. Todos estos clientes disponen de una tarjeta de red que funciona a una velocidad de 100 Mbps. Una ventaja añadida en este tipo de clientes es que no es necesario que dispongan de disco duro, ya que el arranque se realiza sin usar en ningún momento el disco. Por otro lado sí que hay que

configurar la BIOS del ordenador para que realice el arranque por *PXE* de manera predefinida.

#### 3.5.1.1 Arranque por red

En este tipo de clientes al disponer de un arranque por PXE, el arranque por red es más sencillo que en el caso de los clientes Raspberry Pi. Lo único que se ha de tener en cuenta es que en el archivo de configuración del servidor de DHCP, que se encuentra en la máquina virtual 100, aparezca la dirección MAC de ese ordenador con bajos recursos, asignado a la lista de dispositivos con arranque por PXE. Este tipo de arranque consiste en que el ordenador al iniciarse, solicita al servidor de DHCP una IP y este además de asignarle una le pasa la ubicación de un *network bootstrap program* (en nuestro entorno es la máquina virtual 123, el servidor TFTP) que permitirá al cliente arrancar un sistema operativo básico, totalmente diseñado por nosotros además de los scripts, para poder realizar la conexión al escritorio remoto. A continuación, monta el sistema de archivos base mediante NFS, por ese motivo, para poder mantener un mínimo grado de seguridad, se ha configurado para que este sistema de archivos sea montado en solo lectura y que el conjunto de carpetas que requieren permisos de escritura para que los programas se puedan ejecutar correctamente se monten en la memoria RAM y así se conviertan en carpetas volátiles tal y como se explica en detalle en el siguiente punto.

### 3.5.2 Raspberry Pi 2

Aunque en un principio también se valoró la opción de usar la Raspberry Pi 1 como cliente en el entorno, después de realizar varias pruebas, la fluidez en que el video se reproducía, así como el audio y el movimiento de ventanas, eran notablemente superiores en la versión 2. Como la diferencia de precio era muy baja entre la versión 1 y la 2 se decidió optar por usar esta última versión.

De la Raspberry Pi 2 modelo B hay que destacar dos elementos importantes que se han diseñado en este proyecto, el primero es su arranque en red para cargar el sistema de archivos base desde el servidor NFS y el segundo aspecto es la incorporación de la autentificación en dos pasos mediante NFC. A continuación explicaremos cada uno de éstos elementos:

#### 3.5.2.1 Arranque por red

El arranque de la Raspberry Pi 2 es un tanto particular comparado con el de un ordenador normal. La Raspberry a diferencia de estos ordenadores, no tiene BIOS y el arranque se ejecuta de la siguiente forma:

Cuando la Raspberry arranca, el núcleo ARM[58] está desconectado pero el núcleo de la GPU[59] está activo. En este punto la SDRAM[60] está desactivada. La GPU se inicia ejecutando la primera fase del arranque que está almacenada mediante ROM[61] en la SoC[62]. En esta primera fase, lee la tarjeta SD (el único almacenamiento que tiene) y carga la segunda fase del arranque (*bootcode.bin*) en la caché L2[63] y ejecuta el archivo. Este archivo activa la SDRAM y carga la tercera fase del arranque (*loader.bin*) en la RAM, a continuación lo ejecuta. El archivo loader.bin lee el firmware de la GPU (*start.elf*) el cual lee *config.txt, cmdline.txt* y el archivo *kernel.img*. El archivo *loader.bin* es el que se encarga de cargar el *start.elf* en los primeros bytes de memoria SDRAM y, a continuación se leen y cargan los archivos *cmdline.txt* y *config.txt*. El archivo *config.txt* contiene configuraciones como la resolución de pantalla, la salida de audio predeterminada, la velocidad a la que se quiere hacer funcionar el procesador (overclocking), entre otras opciones. El archivo *cmdline.txt* es el que nos permite decirle a la Raspberry donde está su sistema de archivos raíz, es aquí donde le especificamos que lo que queremos que cargue. No es una partición de la tarjeta (como viene configurada de fábrica) sino que queremos que monte un directorio de un servidor NFS como sistema de archivos raíz.

En la fase de investigación de este tipo de arranque se descubrió que había un método alternativo de arranque en red mediante la Raspberry. Este consistía en usar un programa llamado Das Uboot[64].

---

[58] https://es.wikipedia.org/wiki/Arquitectura_ARM
[59] Es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas.
[60] Es una familia de memorias dinámicas de acceso aleatorio (DRAM) que tienen una interfaz síncrona.
[61] Es la memoria de solo lectura y es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite solo la lectura de la información y no su escritura, independientemente de la presencia o no de una fuente de energía.
[62] System-on-chip, describe la tendencia cada vez más frecuente de usar tecnologías de fabricación que integran todos o gran parte de los módulos componentes de un computador o cualquier otro sistema informático o electrónico en un único circuito integrado o chip.
[63] Se encarga de almacenar datos de uso frecuente. Es más lenta que la caché L1, pero más rápida que la memoria principal (RAM). Se encuentra en el procesador.
[64] http://www.denx.de/wiki/U-Boot

*Das Uboot* es un *bootloader* para varios tipos de arquitecturas de ordenadores, incluyendo ARM, que permite realizar un arranque PXE, incluso copiando toda la imagen en la RAM, para que el sistema se pueda usar en el caso de perder la conexión con el servidor.

En nuestro caso en particular, nos pareció mucho más adecuado montar el sistema por NFS en vez de realizar una copia del sistema en RAM usando *Das Uboot* ya que la ventaja de no perder la conexión cuando se desconecta de la red, no resultaba útil, ya que la conexión al escritorio remoto de todas formas se perdería, y en contrapartida, el uso extra de la memoria RAM que se requería para tener el sistema copiado íntegramente, afectaba al rendimiento del sistema. Por ese motivo, se decidió montar el sistema de archivos raíz por NFS mediante el archivo *cmdline.txt*.

Uno de los problemas que surgió al tomar esa decisión fue que la conexión al servidor NFS, para que funcionase correctamente, se tenía que montar con permisos de escritura y lectura. Ya que algunos programas exigen generar logs[65] y, otros usan archivos temporales que de no generarse afectarían el correcto funcionamiento del sistema. Por otro lado, la solución no era tan fácil como permitir ese montaje con permisos de escritura y lectura (recordamos que tal y como se ha dicho en el punto anterior, el servidor está configurado para que se pueda montar en sólo lectura), ya que al ofrecer este tipo de permisos, se ponía en riesgo la seguridad de la imagen, al ser fácilmente manipulada por cualquier persona que tuviera permisos de administrador en un ordenador con acceso al servidor NFS. La primera solución que se planteó fue la de montar el sistema de archivos con permisos de solo lectura, pero como se ha comentado, muchas aplicaciones de inicio del sistema no arrancaban por no poder generar sus logs, con lo cual se debía buscar una alternativa. Finalmente, se decidió buscar cuales eran las carpetas que necesitaban tener esos permisos de escritura y lectura para montar, únicamente esas carpetas, con esos permisos, en la memoria RAM de cada cliente. De esta forma, estos archivos eran temporales e independientes entre todos los dispositivos del entorno. La carga de estas carpetas en RAM se realizó mediante el comando *tmpfs*. *Tmpfs* es un sistema de almacenamiento disponible en los sistemas operativos de tipo Unix. Aparece como un sistema de archivos normal aunque es volátil. Es similar a los discos RAM, que aparecen como

[65] Conjunto de archivos que contienen información para la depuración de programas.

discos virtuales, y pueden contener sistemas de archivos. Al usar memoria volátil, los datos en *tmpfs* no persisten después de reiniciar el sistema, lo que en este entorno no suponía un problema. Las carpetas que se montaron con este tipo de sistema de archivos fueron: `/etc`, `/var`, `/home` y `/tmp`.

Aplicando estos cambios se consigue que el sistema arranque montándose por NFS en solo lectura y, solo usando la memoria RAM necesaria para los archivos que deben tener permisos de escritura. Esta última opción no pone en peligro la imagen que se encuentra en el servidor, ya que en cada reinicio los cambios que se hayan hecho en cada cliente se descartan.

#### 3.5.2.2 Acceso en 2 pasos mediante NFC

Como se ha comentado anteriormente, la Raspberry Pi tiene un puerto de conexiones bastante completo para poder comunicarse con dispositivos periféricos (además de los USB). Así pues, como se quería incorporar algún tipo de seguridad adicional en el mecanismo de autentificación de los usuarios, se empezaron a buscar opciones.

La solución más usada es la de las tarjetas inteligentes con chip para conectarse a sistemas operativos, pero los lectores de tarjetas no son muy económicos y además, las tarjetas con certificado también son demasiado costosas. Por otra parte, se quiso pensar en alguna solución que no obligara al usuario a llevar consigo una tarjeta.

Adicionalmente, pensando en que cada vez es más frecuente que se haga uso del teléfono móvil, se quiso pensar en alguna opción que incluyese el teléfono móvil físico como segundo paso de identificación.

La primera opción que se pensó fue en el uso de un mecanismo de doble autentificación comúnmente usado, como es el caso del OAuth[66], que es usado por ejemplo, para poder acceder a una cuenta de Gmail cuando se activa la autentificación en dos pasos. Un proceso de autentificación mediante OAuth sería el siguiente:

a) Se introduce el usuario y la contraseña.
b) Si son correctos se solicita un código adicional.
c) Se debe acceder al teléfono móvil autorizado, abrir una aplicación y copiar el código que aparece.

[66] http://oauth.net/

d) Se debe introducir ese código en el formulario y a continuación se puede acceder al sistema.

Este proceso tiene el problema de que para el usuario es una molestia tener que desbloquear el móvil, abrir la aplicación y copiar ese código al ordenador. Además, si no se dispone de red de datos en el teléfono en ese momento, no es posible obtener este código y por tanto, no es posible acceder al sistema.

Al descartar esa opción se pensó en usar en la Raspberry, a través del puerto GPIO, el bus SPI[67]. Esta solución, supone un ahorro considerable ya que el lector de NFC mediante bus SPI (como se puede observar en la figura 11) sale por unos 3€.


Figura 11 – Lector NFC

Además esta opción ofrecía otra vez dos alternativas:

a) Usar el lector NFC del móvil que mediante una aplicación para móvil, identificase el dispositivo. Esta posibilidad tenía el problema de que el usuario debía de desbloquear el móvil, de la misma forma que ocurría con la autentificación mediante OAuth. Además, si al usuario se le acaba la batería no podría acceder al sistema y finalmente, si el dispositivo móvil del usuario no tuviera integrado un dispositivo NFC no podría conectarse.
b) Usar una etiqueta de NFC: Las etiquetas NFC son pequeñas etiquetas/chapas/pulseras, que incluyen un pequeño chip NFC en modo pasivo, que cuestan entre 15 céntimos y 2€ cada una dependiendo de la cantidad que se compre. No necesitan batería y se puede grabar en ellas cualquier cosa: una dirección web, unos datos de contacto, un número de teléfono, un programa, una serie de órdenes…

[67] Es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

Para poder usarlas tan solo tenemos que acercar la etiqueta al lector NFC para que se produzca una acción. Por ejemplo, podemos definir que al pasar la etiqueta por el lector, se active el WiFi, se suba al máximo el volumen del cliente o incluso que se apague. Estas etiquetas NFC tienen dos modos posibles: escritura-lectura o únicamente lectura. De esta manera, podemos ir cambiando su contenido a nuestro gusto o bien predefinir un contenido que no se pueda cambiar.

Figura 12 – Etiqueta NFC

Así pues, hemos elegido poner una etiqueta NFC como la que se aprecia en la figura 12, en la parte posterior del teléfono móvil de cada usuario al que se quiera dotar de esta doble autentificación para que este lo lleve siempre encima. Además, en el caso de que se perdiese, el usuario se daría cuenta mucho más rápido que si fuera una tarjeta inteligente ya que habría perdido su teléfono móvil. Finalmente, si el usuario se queda sin batería, puede seguir accediendo al sistema sin ningún tipo de problema.

En el anexo D se detalla la seguridad de este tipo de etiquetas, se recomienda su lectura si se desconoce cómo funciona.

En las figuras 13, 14 y 15 se puede observar cómo queda integrado el lector de NFC dentro de la misma caja de la Raspberry Pi.

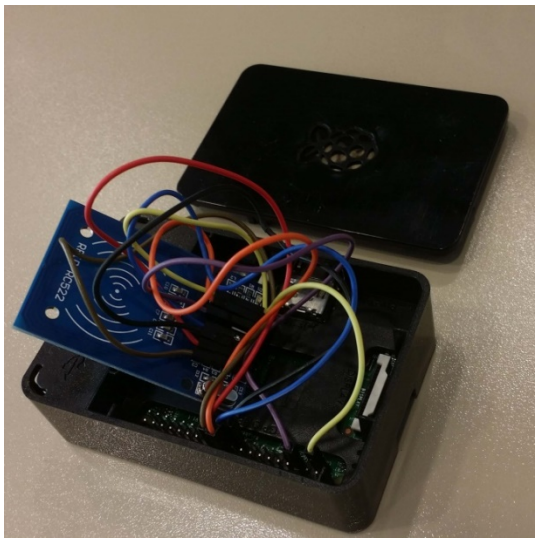


Figura 13 – Lector NFC y Raspberry Pi

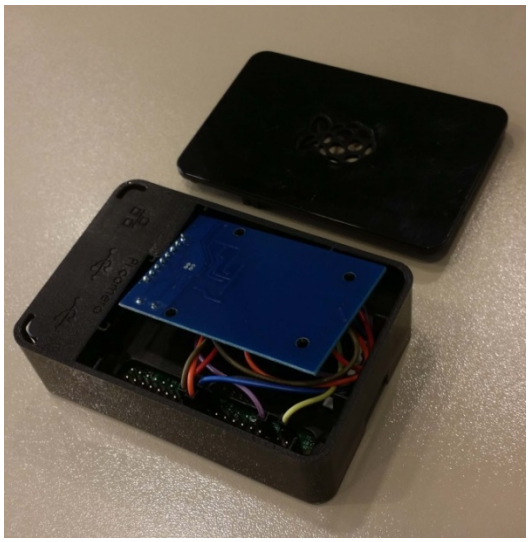


Figura 14 – Lector NFC y Raspberry Pi

Figura 15 – Lector NFC y Raspberry Pi

#### 3.5.2.3 Aumento de la seguridad de NFC

En este proyecto, se ha decidido diseñar un protocolo de seguridad exclusivo y particular, para el uso de las etiquetas NFC de 1 K.

Este protocolo de seguridad presenta dos fases: la escritura de etiqueta y la lectura. A continuación comentamos cómo se realiza cada una de estas fases. Como se ha comentado en el punto anterior, se recomienda la lectura del anexo D si se desconoce el funcionamiento interno de este tipo de etiquetas.

##### 3.5.2.3.1 Escritura de la etiqueta

La escritura de una nueva etiqueta en el sistema que se presenta, se realiza mediante la ejecución de un script, que se explica al detalle en el anexo E. En este script, básicamente, se genera un número aleatorio, que representa el bloque en el que se escribirá la clave que se validará con el servidor de usuarios LDAPS (este número solo puede estar en el rango *[4, 63]* excepto los números *7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59 y 63* ya que, son bloques de *tráiler* y no se pueden almacenar datos en ellos). A continuación, se recorren todos los bloques anteriormente comentados y sus correspondientes *tráileres,* generando una Key A y una Key B aleatoria para cada sector y rellenando sus correspondientes bloques con datos aleatorios. Los permisos que se establecen en cada sector son: los de que la Key B pueda leer, la Key A no pueda hacer nada y los que permitan que ni la Key A ni la Key B sean modificables ni legibles. Los permisos de los bits de acceso de cada sector son de lectura únicamente para la Key A y la Key B.

Una vez terminada la escritura, se une el número de bloque donde se tiene el código aleatorio que nos interesa, el bloque 0 del sector 0 de la etiqueta (que es único para cada etiqueta e indica el fabricante, modelo, etc...) y la Key B aleatoria que se ha generado anteriormente. Esta cadena se encripta mediante AES[68] de 128 bits usando como semilla[69] el nombre de usuario del propietario de la etiqueta. Esto produce una cadena que es la que se deberá almacenar en la base de datos LDAPS.

[68] También conocido como Rijndael, es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos.
[69] Valor, equivalente a una contraseña que se usa como referencia para encriptar o desencriptar una cadena.

De esta forma, conseguimos que toda la etiqueta esté escrita y ya no se pueda modificar ningún dato, además de que para cada usuario la información realmente importante de la etiqueta esté en un sector distinto.

Así pues, si alguien hiciera un clon de la etiqueta no podría acceder al sistema, ya que el bloque 0 del sector 0 de la nueva etiqueta sería distinto y por tanto la cadena del servidor LDAPS no coincidiría. Además con toda esta seguridad añadida, si alguien realizara un ataque de seguridad de fuerza bruta[70] o un ataque que buscara tendencias de repetición en los bits, tardaría mucho tiempo ya que debería leer toda la tarjeta (al tener cada usuario la información en un bloque distinto) y además todos los bits ya que son aleatorios y están escritos en todos los bloques.

#### 3.5.2.3.2 Lectura de la etiqueta

La lectura de etiqueta se realiza de la siguiente forma: en el momento en que se le solicita al usuario que coloque la etiqueta en el dispositivo lector de la Raspberry Pi, desde la base de datos LDAPS, se envía la cadena encriptada que se ha comentado en el punto anterior. Esa cadena es decodificada usando la semilla del nombre de usuario. Una vez decodificada, se obtiene la identificación única alojada en el bloque 0 del sector 0 de la etiqueta y se compara con el valor que corresponde a esa identificación en la cadena del servidor LDAPS. Si no coinciden, como este valor es único para cada etiqueta, significa que alguien ha intentado copiar los datos de la etiqueta original en otra distinta, así pues se deniega el acceso. Si el resultado coincide, se obtiene de la cadena que ha enviado antes el servidor LDAPS, el número de bloque que ha de leerse de la etiqueta así como, la Key B para poder leer todo su sector. Se realiza la lectura y se compara el valor leído de la etiqueta con el valor que se tenía en la base de datos. Si ambos valores coinciden, se considera que la validación es correcta, sino se deniega el acceso.

[70] Forma de recuperar una clave probando todas las combinaciones posibles hasta encontrar aquella que permite el acceso.

### 3.5.3 Proceso de arranque detallado

En los puntos anteriores, se ha explicado al máximo detalle como los clientes, tanto la Raspberry como los ordenadores con pocos recursos, realizan el arranque del sistema operativo que se encuentra en la máquina virtual con el servidor TFTP. Pero el arranque no termina aquí. En los dos casos, el arranque en red carga un sistema operativo que es un Linux, instalado desde la base, para optimizar su rendimiento. Concretamente es una Debian Wheezy (en el caso de la Raspberry una Raspbian Wheezy). En estos sistemas se ha instalado: el servidor de las X, es decir, de video (Openbox[71] y Xinit[72]), un servidor de Audio (Pulseaudio[73]) y dos clientes de escritorios remotos: para las conexiones a escritorios Windows DfreeRDP[74] y para los escritorios Unix X2goClient[75].

#### 3.5.3.1 DfreeRDP

RDP es una forma muy conveniente de realizar una conexión a un escritorio virtual de Windows. RDP es uno de los mejores protocolos, ya que transmite la información de forma muy eficiente al poder analizar la estructura de la pantalla y los elementos, evitando la transmisión de pixel por pixel. Además también permite la transmisión de audio en ambos sentidos e incluso la compartición de dispositivos usb entre el servidor y el cliente.

FreeRDP es una aplicación que permite la conexión hacia servidores RDP y que además incorpora una gran cantidad de opciones para ajustarse a nuestras necesidades. Pero freeRDP, en la práctica, requiere de unos recursos mínimos que la Raspberry Pi no cumple, en cuanto a procesador, con el programa Htop[76] se puede observar como los núcleos del procesador se colapsan produciendo bloqueos en todo el sistema ya que la mayoría del trabajo de descompresión y decodificación de las imágenes recibidas lo realiza este elemento.

Aun así se quiso realizar una prueba, ya que teóricamente parecía que debía de funcionar correctamente. Al realizarla, se pudo observar que el movimiento de ventanas era extremadamente lento y que la reproducción de audio del host en el cliente no funcionaba nada bien (cortes, errores y bloqueos del sistema). El problema, además del

[71] http://openbox.org/wiki/Main_Page
[72] https://wiki.archlinux.org/index.php/Xinitrc
[73] http://www.freedesktop.org/wiki/Software/PulseAudio/
[74] http://blog.pi3g.com/2013/07/high-performance-rdp-on-the-raspberry-pi/
[75] http://wiki.x2go.org/
[76] http://hisham.hm/htop/

alto nivel de decodificación al que se somete la Raspberry Pi, es que el servidor de las X añade una cabecera enorme a los frames que llegan del host. Eso produce que al tener que quitar esa cabecera para poder mostrarla en la pantalla, se requiere de un uso aun mayor de la CPU, que no puede dar al estar ocupada decodificando nuevos paquetes que entran con nuevos frames.

La solución consiste en compilar manualmente la aplicación de freeRDP para que dibuje los frames mediante *Direct frame Buffer*[77] y así se pueda eliminar la cabecera que le añade el servidor de las X. Al compilarlo de esta forma, la experiencia del escritorio remoto de Windows mediante RDP cambió abismalmente, pudiendo reproducir audio remoto en el dispositivo local incluso compartiendo a la vez un dispositivo usb conectado a dicho cliente. Al uso de esta forma distinta de compilación del programa, se le llama *DfreeRDP* haciendo referencia la *"D"* a *Direct frame Buffer*.

#### 3.5.3.2 X2goClient

Sobre la forma de conectarse a los escritorios remotos de Linux, en un principio, se eligió probar el DfreeRDP pero los resultados no eran tan satisfactorios como en Windows, eso es debido a que Windows lleva instalado el servidor RDP de fábrica y éste está muy optimizado para que la experiencia del usuario sea satisfactoria. En el caso de Linux, había que buscar otra alternativa. Fue entonces cuando se encontró una tecnología distinta a RDP era la tecnología NX[78].

En el anexo F se detallan las características de este protocolo.

Después de valorar distintas soluciones basadas en NX, se decidió usar X2go que mediante su paquete de cliente/servidor, en debian y raspbian, se desenvuelven fenomenalmente. Además, este cliente permite la configuración de muchísimos ajustes para adaptar el cliente a todas las necesidades.

[77] Es una biblioteca de software para el sistema operativo GNU/Linux que proporciona aceleración gráfica de hardware, manejo y abstracción de dispositivos de entrada, sistema integrado de ventanas, con soporte para ventanas translúcidas y capas múltiples de visualización sobre del dispositivo Linux Framebuffer.
[78] https://en.wikipedia.org/wiki/NX_technology

## 3.6 Entorno Proxmox VE

Proxmox VE es un entorno de virtualización de código libre basado en Debian Wheezy y con un kernel modificado que permite crear y gestionar máquinas virtualizadas mediante KVM y OpenVZ. Proxmox incluye un servidor web donde se pueden administrar estas máquinas y además incluye un conjunto de comandos para poder modificarlo y adaptarlo a las necesidades requeridas, tal y como se ha hecho en este trabajo. Proxmox se usa bajo licencia GPLv2. Las principales características de Proxmox son:

a) Modelo de Almacenamiento: Proxmox VE soporta almacenamiento local mediante grupos LVM[79] y ZFS[80], también acepta almacenamientos en red como ISCSI[81], fibra óptica, NFS, GlusterFS[82], CEPH[83] y DRBD[84]
b) Clúster de alta disponibilidad: Proxmox puede funcionar con clústeres a través de varios servidores. A partir de la versión 2.0, permite el uso de la alta disponibilidad, de manera que si uno de los servidores del clúster deja de funcionar, alguno de los otros servidores "adopta" esas máquinas reiniciándolas.
c) Migración en caliente: en un clúster de alta disponibilidad también existe la opción de mover las máquinas de un servidor a otro para poder, por ejemplo, realizar tareas de mantenimiento en uno de los servidores.
d) Clonaje de máquinas: En Proxmox se pueden clonar las máquinas KVM mediante *linked clones*, que son copias de una plantilla que se realizan muy rápidamente y, sin apenas suponer una carga en el servidor, de esta forma podremos generar nuevos escritorios virtuales de una forma rápida y efectiva. Para poder generar los escritorios virtuales que funcionan con OpenVZ se utiliza la técnica de las plantillas guardadas, esta técnica consiste en generar primero una plantilla con el escritorio que queremos distribuir. A continuación para generar los nuevos escritorios lo que se hace es generar nuevas máquinas virtuales con la plantilla que se ha creado anteriormente.

[79] https://es.wikipedia.org/wiki/Logical_Volume_Manager
[80] https://es.wikipedia.org/wiki/ZFS_%28sistema_de_archivos%29
[81] https://es.wikipedia.org/wiki/ISCSI
[82] https://es.wikipedia.org/wiki/Gluster_File_System
[83] https://es.wikipedia.org/wiki/Ceph_File_System
[84] https://pve.proxmox.com/wiki/DRBD

e) Paravirtualización: Si el sistema operativo que se quiere ejecutar es compatible, Proxmox ofrece una serie de *drivers* para beneficiarse de la paravirtualización y mejorar la comunicación de algunos dispositivos con la máquina virtual. En el caso de Windows 8.1 que será el sistema que distribuiremos para los escritorios virtuales de Windows, instalando los drivers correspondientes, se puede hacer uso de una interfaz de red y de un acceso al disco duro con unas prestaciones mucho mejores que si se utilizara una virtualización completa.

f) Administración web: El entorno, como se ha comentado anteriormente, incorpora un servidor web donde se puede gestionar prácticamente todo: crear máquinas, eliminarlas, configurarlas, interactuar con ellas, administrar el almacenamiento, entre otras muchas opciones. La figura 16 muestra una vista del entorno del servidor web, donde se puede apreciar en la parte izquierda el conjunto de máquinas virtuales, tanto las OpenVZ como las KVM, al final de la lista de la izquierda podemos ver las plantillas de Windows sobre las que hacemos los linked clones y a continuación los dispositivos de almacenamiento que tenemos en nuestro entorno. En la parte superior derecha, podemos ver las distintas pestañas desde donde configuramos todas las opciones para el servidor y cada una de las máquinas virtuales. En el anexo G se explica esto con más detalle al concretar el procedimiento de la creación de una nueva plantilla tanto en OpenVZ como en KVM

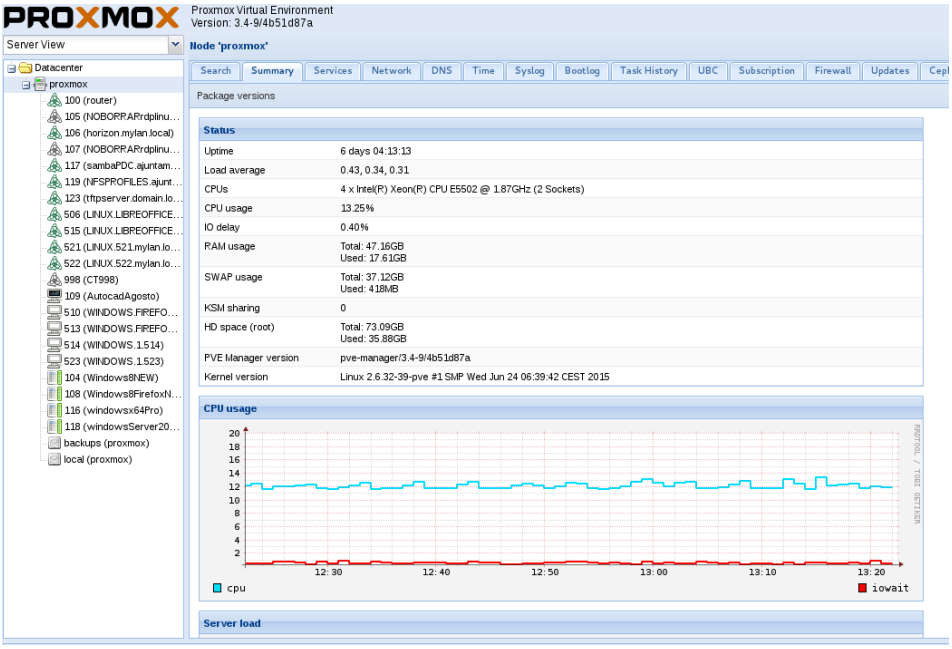



Figura 16 – Entorno web de Proxmox

## 3.7 Sistema de archivos ZFS

Una de las características destacables de Proxmox VE, como se ha comentado en el punto anterior, es que nativamente admite el uso del sistema de archivos ZFS.

El sistema de archivos ZFS es un sistema de archivos que aporta una forma totalmente distinta de administrar sistemas de archivos, con funciones y ventajas que no hay en ningún otro sistema de archivos actual. ZFS es sólido, escalable y fácil de administrar, las ventajas que aporta este sistema se pueden consultar en el anexo H.

Para este proyecto al disponer de 5 discos duros de 1 TB, en cada uno se ha montado todo el sistema de Proxmox sobre un RAID-Z2[85] que nos permite tener un almacenamiento útil de 2.7 TB.

## 3.8 Alta disponibilidad del sistema

Alta disponibilidad (*High availability*) es un protocolo de diseño de sistemas que asegura un cierto grado absoluto de continuidad operacional durante un período de medición dado. Disponibilidad se refiere a la habilidad de la comunidad de usuarios para acceder al sistema, someter nuevos trabajos, actualizar o alterar trabajos existentes o recoger los resultados de trabajos previos. Si un usuario no puede acceder al sistema se dice que está no disponible. El término tiempo de inactividad (*downtime*) es usado para definir cuándo el sistema no está disponible.

Existen dos tipos de tiempo de actividad:

a) Tiempo de inactividad planificado: Típicamente es un resultado del mantenimiento que es perjudicial para la operación del sistema y usualmente no puede ser evitado con la configuración del sistema actualmente instalada. Eventos que generan tiempos de inactividad planificados incluyen parches al software del sistema o cambios en la configuración del sistema que requieran un reinicio del servidor para aplicarse. En general el tiempo de inactividad planificado es usualmente el resultado de un evento lógico o de gestión iniciado.

[85] Similar al RAID 6, permite que dos discos puedan fallar sin comprometer los datos que guarda el grupo de discos y aumenta en 3 veces la velocidad de lectura.

b) Tiempo de inactividad no planificado: Los tiempos de inactividad no planificado surgen de algún evento físico tales como fallos en el hardware o anomalías ambientales. Ejemplos de eventos con tiempos de inactividad no planificados incluyen fallos de potencia, fallos en los componentes de CPU o RAM, una caída por recalentamiento, una ruptura lógica o física en las conexiones de red, rupturas de seguridad catastróficas o fallos en el sistema operativo o aplicaciones.

Muchos puestos computacionales excluyen el tiempo de inactividad planificado de los cálculos de disponibilidad, asumiendo, correcta o incorrectamente, que el tiempo de actividad no planificado tiene poco o ningún impacto sobre la comunidad de usuarios computacionales. Excluyendo el tiempo de inactividad planificado, muchos sistemas pueden reclamar tener alta disponibilidad, la cual da la sensación de disponibilidad continua. Sistemas que exhiben verdadera disponibilidad continua son comparativamente raros y caros, y tienen diseños cuidadosamente implementados que eliminan cualquier punto de fallo y permiten que el hardware, la red, el sistema operativo, parches y reemplazos se hagan en línea.

En este proyecto, aunque finalmente no se ha llevado a cabo por no disponer de más servidores ni dispositivos de *fencing*[86], se ha querido analizar la incorporación al entorno de un mecanismo de alta disponibilidad.

Usaríamos Proxmox VE HA Clúster para implementar esta alta disponibilidad en todas las máquinas del entorno. Esta plataforma busca dar con unos mínimos recursos, la máxima capacidad de crecimiento y disponibilidad.

Para poder aprovecharnos de dicho sistema, debemos de tener configurado el clúster con mínimo 3 nodos/servidores Proxmox y debemos de tener el almacenamiento compartido configurado en todos los nodos que disponga de algún mecanismo de alta disponibilidad integrado. Además los distintos nodos deberán estar interconectados mediante un switch que se pueda configurar como un dispositivo de *fencing*, ya que si no se corre el riesgo de que las máquinas virtuales se puedan corromper.

Proxmox VE HA Clúster, permite la administración y creación de servidores virtuales con alta disponibilidad, distribuidos entre un

[86] Es el proceso de aislar un dispositivo del entorno para proteger los recursos compartidos en el mismo.

conjunto de servidores que trabajan unidos mediante nodos. Si una máquina virtual se configura con la característica de alta disponibilidad, y por alguna razón fallara el host físico, la máquina virtual se reinicia automáticamente y se inicia desde otro nodo del clúster de los restantes de la implementación de Proxmox VE HA Clúster.

Proxmox VE HA Clúster está basado en la tecnología Linux HA, proporcionando servicios de Alta disponibilidad a través de una red de nodos y configuraciones de clúster que permiten que cuando un nodo físico falle, la máquina siempre esté disponible a través de otro nodo.

# 4. Proxmox View vs VMWare Horizon View

Finalmente, después del diseño de todo el sistema de escritorios virtuales, hemos querido realizar una comparación tanto económica como operacional de los resultados obtenidos con nuestro diseño, respecto a la solución comercial de VMWare Horizon View, que estaba instalada en el ayuntamiento.

## 4.1 Comparativa operacional

En cuanto a la calidad de la experiencia del usuario, tenemos que destacar que, aunque para el uso de programas de ofimática, navegadores web, reproducción de música e incluso Photoshop, la sensación al usar el escritorio es igual de buena; para la reproducción de videos a partir de la resolución de 480p[87], el rendimiento del entorno de VMWare, es superior al de nuestro entorno. Por otro lado, como en el ayuntamiento, tampoco les interesaba que los usuarios vieran videos en una calidad superior, por temas de saturación del tráfico de la red; no se planteó esta situación como un problema.

## 4.2 Comparativa económica

Si comparamos el coste que tiene el entorno de VMWare con el que tiene nuestra solución, observamos que la diferencia es muy grande.

Para el entorno de VMWare consideramos los clientes ligeros que les recomiendan la empresa que les instaló el sistema y que son concretamente *Praim C9050*[88]. Para nuestro entorno, consideramos como clientes ligeros, la *Raspberry Pi 2 modelo B*.

En la siguiente tabla se pueden apreciar las diferencias:

[87] https://es.wikipedia.org/wiki/480p

[88] http://praim.com/documenti/Datasheet%20ITA/Datasheet%20ENG/Datasheet_Compact_DC_ENG.pdf

| Concepto | Entorno Proxmox View | Entorno Horizon View | Ahorro económico |
|---|---|---|---|
| **Cliente ligero** | 73€ | 418€ | 345€ |
| **Licencia, por cliente, del entorno de administración del cliente ligero (vitalicia)** | 0€ | 50€ | 50€ |
| **Licencia para el entorno de virtualización (hipervisor), en un servidor con 2 procesadores (anual)** | 796€[89] | 950€[90] | 154€/anuales |
| **Licencia del entorno de escritorios virtuales (anual)** | 0€ | 250€[91] | 250€/anuales |
| **Licencia Windows Server + SQL Server[92] para la gestión del hipervisor ESXi (vitalicia)** | 0€[93] | 2.000€ | 2.000€ |
| **Licencia por cliente ligero que se conecte al entorno de virtualización de escritorios (cada 5 años)** | 0€ | 184€ | 37€/anuales |

[89] Aunque el software Proxmox VE es gratuito, para hacer la comparación más parecida, hemos considerado la opción del pago **opcional** de la suscripción standard, que da derecho a 10 tiquets de soporte al año así como al repositorio empresarial de los paquetes, además de muchos otros beneficios (https://www.proxmox.com/en/proxmox-ve/pricing)
[90] Referente a la licencia del vSphere (https://www.vmware.com/es/products/vsphere)
[91] Referente a la licencia de VMWare Horizon View
[92] http://www.microsoft.com/es-es/server-cloud/products/sql-server/
[93] No necesita, ya que el entorno se gestiona vía web y usa una base de datos de mysql (https://www.mysql.com/) que no requiere licencia.

Así pues, podemos observar que no solo hay una diferencia muy grande en el precio de los clientes ligeros (además de que la Raspberry tiene un coste energético inferior), sino que también, los clientes *Praim,* requieren del pago de una licencia para poder actualizarlos y configurarlos de forma centralizada, a diferencia de en nuestro entorno que es totalmente gratuito. También, el hecho de usar un entorno como VMWare Horizon View, obliga a comprar:

a) Una licencia para cada cliente ligero que le queramos conectar al entorno, con renovación cada 5 años.
b) Una licencia para tener el servidor de escritorios (VMWare Horizon View), con renovación anual.
c) Una licencia del gestor de hipervisores vSphere, con renovación anual.
d) El software vSphere ha de instalarse en un servidor con Windows Server y SQL Server, con lo que nos hace falta una licencia para cada uno, aunque estas licencias son vitalicias (excepto si se quiere cambiar de versión).

La diferencia de coste, en definitiva, es muy grande. Podemos concluir que, si consideramos, como ha hecho el ayuntamiento, que es un proyecto para 5 años, y tenemos en cuenta que el ayuntamiento dispone de 100 usuarios a los que quiere facilitar este entorno; la comparación de costes totales en estos 5 años sería la siguiente:

| | **Coste total cliente ligero**[94] | **Hipervisor**[95] | **Entorno de escritorios virtuales** | **TOTAL** |
|---|---|---|---|---|
| **Entorno Proxmox View** | 7.300€ | 3.980€ | 0€ | 11.280€ |
| **Entorno VMWare Horizon View** | 65.200€ | 6.750€ | 1.250€ | 73.200€ |

Como se puede observar, la diferencia entre la solución de VMWare y la nuestra, en estos cinco años, es de un total de 61.920€. Esto significa, que eligiendo nuestra solución, se benefician de un ahorro aproximado del 85%.

[94] Incluye coste del cliente ligero, más la licencia del entorno de administración (en VMWare Horizon View), más la licencia para poder conectar el cliente al entorno de escritorios virtuales (en Horizon View) para 100 clientes.
[95] Incluye la licencia del hipervisor y en el caso de Horizon View, la de Windows Server con la de SQL Server.

# 5. Conclusiones

Este proyecto ha implementado un entorno de escritorios virtuales, como una alternativa económica al producto comercial de VMWare. La solución integrada del entorno de escritorios virtuales que se ha creado para este proyecto cumple perfectamente las expectativas de su funcionamiento, al hacerlo prácticamente con la misma calidad de experiencia de usuario que mediante el entorno comercial de VMWare, Horizon View.

Además, la implementación desarrollada supuso una reducción del coste económico de hasta el 85%, lo que lo convierte en una alternativa mucho más económica que Horizon View.

Con la incorporación al entorno de los ordenadores con bajos recursos y de las Raspberry Pi, se ha conseguido disponer de clientes ligeros mucho más económicos, que los que el Ayuntamiento compraba a la empresa que les instaló el VMWare Horizon View para que tuvieran la máxima compatibilidad.

Estos clientes ligeros, disponían de una interfaz centralizada para poder controlarlos y actualizarlos de forma rápida y eficaz. Para el uso de esta interfaz se obligaba al cliente a pagar una licencia por dispositivo que quisiera incorporar al sistema de administración.

Con nuestro entorno, también disponemos de una configuración centralizada al tener los sistemas operativos que ejecutan en los clientes, para realizar la conexión, almacenados en una de nuestras máquinas virtuales. De esta forma, si realizamos un cambio en esta máquina, éste se realiza en todos los dispositivos del entorno cuando se reinician.

El añadido de la doble autentificación, mediante las etiquetas NFC, en los dispositivos Raspberry Pi, incrementa un nivel más la seguridad.

Este entorno de virtualización de escritorios ha sido implementado en una de las aulas de formación del centro cívico del Ayuntamiento de Sant Joan Despí. Así pues esta implementación se considera un éxito en la realización del proyecto.

# 6. Trabajo futuro

Como se ha comentado en el anexo A.3, la nueva versión de *Proxmox VE 4.0* apareció en verano de 2015 en su versión Beta, y en Octubre de 2015 se convirtió en versión estable. Es por eso que un trabajo futuro de este proyecto podría ser la migración de todo el entorno a este nuevo *Proxmox VE 4.0*, que entre otras mejoras, destaca por la utilización de la virtualización a nivel de sistema operativo LXC en vez de OpenVZ, la cual proporciona algunas mejoras como se comenta en el anexo A.3.

Por otro lado, se podría incrementar la seguridad de todo el entorno estableciendo VLANS para separar las subredes. Una VLAN[96] de administración, en la que los ordenadores que estuvieran asignados, pudieran acceder a todos los entornos web para la administración del sistema; y otra VLAN de usuarios en la que simplemente se permitiera a los dispositivos que se conectaran, acceder a los escritorios remotos, sin poder siquiera intentar identificarse en alguno de los elementos de la administración del entorno.

Otra posible mejora sería en el protocolo que se usa para la conexión de escritorios virtuales (DfreeRDP en Windows y X2go en Linux) los cuales en momentos de alta demanda de la red, no rinden de forma óptima en la reproducción de video y audio en tiempo real. Este problema también es uno de los problemas con los que trata VMWare en su entorno Horizon View, pero ellos lo mejoran usando su protocolo propietario PCoIP.

Finalmente, también se podría llevar a cabo el diseño del sistema de alta disponibilidad *Proxmox VE HA Clúster*, que, mediante 3 servidores, permitiría que el sistema siguiera funcionando aunque uno de los servidores quedara inoperativo. En la versión 4.0 a diferencia de la 3.4, de Proxmox VE, ya no se necesitan dispositivos de *fencing* para que el sistema pueda funcionar. De esta forma se reducen los costes que tendría la implementación de la alta disponibilidad en este entorno.

[96] Acrónimo de *virtual LAN* (red de área local virtual), es un método para crear redes lógicas independientes dentro de una misma red física.

# Anexos

## Anexo A

### Anexo A.1

#### KVM para Windows 8.1

KVM es una virtualización completa acelerada por hardware que está incluida en el kernel de Linux a partir de la versión 2.6.20

De esta forma KVM permite crear e iniciar máquinas virtuales sobre Linux usando las herramientas de QEMU.

Para la creación de los escritorios virtuales de Windows, se ha elegido el uso de Windows 8.1 al ser una de las últimas versiones de Windows y disponer de las últimas versiones de todos los programas compatibles. La única forma de virtualizar este sistema operativo en el entorno que usaremos, tal y como se verá en los próximos puntos, es mediante KVM, aun así, hay un conjunto de opciones que se pueden editar en este tipo de virtualización para poder optimizarla y mejorar su rendimiento. A continuación se exponen los "Best Practices" o recomendaciones, a tener en cuenta al crear una máquina virtual KVM con Windows 8.1.

a) Como se ha comentado anteriormente, en la mayoría de virtualizaciones completas, hay algunos componentes de la máquina virtual que, mediante la instalación de ciertos drivers, pueden mejorar su rendimiento. En el caso de Windows 8.1 son 3: Red Hat VirtIO SCSI controller, Redhat VirtIO NIC and Red Hat Balloon driver. Estos drivers mejoran de forma notable, la velocidad de acceso al disco físico del servidor y la velocidad en que se comunica la interfaz de red de la máquina virtual con la del servidor.
b) Podemos elegir dos tipos de formato de archivos en que se guardará la imagen de la máquina virtual. El formato *raw* mejora un poco el rendimiento mientras que *qcow2* ofrece opciones avanzadas como son la copia o creación de instantáneas o "snapshots". Una instantánea permite conservar el estado de una o más máquinas virtuales para que puedas volver al mismo estado cuando sea necesario. La funcionalidad de poder crear instantáneas de las máquinas virtuales es muy

útil, sobretodo, cuando se necesita volver a un estado anterior de una máquina virtual sin necesidad de tener múltiples máquinas virtuales. La instantánea captura el estado completo de la máquina virtual en el preciso instante cuando la realizamos. Esta instantánea incluye el estado de la memoria en ese preciso instante, la configuración de la máquina virtual y el estado del disco o discos virtuales de dicha máquina virtual. En el caso que nos ocupa elegiremos el formato raw, ya que al ser escritorios virtuales que se crean y se destruyen, y se modifican como máximo una vez a la semana para realizar las actualizaciones, se da prioridad a mejorar, aunque sea poco, el rendimiento de la máquina a poder disponer de estas herramientas avanzadas.

c) Al usar, como se ha comentado en el punto anterior, los drivers de paravirtualización, tenemos que ser conscientes que tanto el disco duro como la tarjeta de red, deben ser añadidas a la máquina virtual en el modo *VirtIO* y que durante la instalación del sistema operativo, los drivers deben de ser instalados para que el sistema los reconozca y los pueda usar.
d) Hay que destacar que para que la máquina se pueda apagar mediante el comando “Shutdown” de QEMU, se ha de modificar la clave en el registro de Windows para que esté en 1.

```
HKLM\Software\Microsoft\WindowsNT\CurrentVersion\winlogon\
ShutdownWithoutLogon
```

# Anexo A.2

## OpenVZ para Debian 7 “Wheezy”

### Kernel

OpenVZ es una tecnología de virtualización a nivel de sistema operativo para Linux. OpenVZ permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales o Entornos Virtuales (EV).

El kernel de OpenVZ es un kernel de Linux modificado, que agrega una noción de Entorno Virtual. Este kernel proporciona virtualización y aislamiento, administración de recursos y punto de comprobación.

Virtualización y aislamiento

Cada EV es una entidad separada, y desde el punto de vista de su dueño se muestra como un servidor físico real. De manera que tiene sus propios:

a) Archivos: bibliotecas del sistema, aplicaciones, */proc* y */sys* virtualizado, etc...
b) Usuarios y grupos: cada EV tiene sus propios usuarios root, así como también otros usuarios y grupos.
c) Árbol de procesos: un EV solamente ve sus propios procesos (comenzando a partir de init). Los PID[97]s son virtualizados, de manera que el PID de init es 1, como debe ser.
d) Red: dispositivo de red virtual, que permite a un EV tener sus propias direcciones IP, así como también un conjunto de reglas de *iptables* y reglas de encaminamiento.
e) Dispositivos: si es necesario, solamente al EV puede otorgarse acceso a los dispositivos reales como interfaces de red, puertos seriales, particiones de disco, etc.

Administración de recursos

Como todos los EVs usan el mismo kernel, la administración de recursos es de suprema importancia. Realmente, cada EV debería permanecer dentro de sus límites y no afectar a otros EVs de ninguna manera, eso es de lo que se encarga la administración de recursos.

La administración de recursos de OpenVZ está formada por dos componentes: cuota de disco de dos niveles y monitor de usuarios.

[97] Identificador de un proceso que se ejecuta en el sistema

Hay que remarcar, que todos esos recursos se pueden cambiar durante el tiempo de ejecución de un EV y si se desea, no hay necesidad de reiniciar el sistema. Es decir, si se quiere dar a un EV menos memoria, sencillamente hay que cambiar los parámetros apropiados al vuelo. Ello es muy difícil de hacer o totalmente imposible con otros modos de virtualización tales como KVM.

*Cuota de disco de dos niveles*

El dueño (root) del sistema anfitrión (OpenVZ) puede configurar una cuota de disco por EV, en términos de bloques de disco e inodos[98] (aproximadamente igual al número de archivos) siendo el primer nivel de cuota de disco. Además de esto, un dueño de EV (root) puede usar las herramientas usuales de cuotas dentro de su propio EV para definir cuotas de disco estándar de UNIX por usuario y por grupo.

Si se desea dar a un EV más espacio, sencillamente hay que incrementar la cuota de disco. No se necesita redimensionar las particiones de disco.

*Monitor de usuarios*

El monitor de usuarios es un grupo de contadores por EV, límites, y garantías. Hay un conjunto de alrededor de 20 parámetros que se eligen cuidadosamente para cubrir todos los aspectos de la operación de EV, de manera que ningún EV por sí solo pueda abusar de cualquier recurso (el cual es limitado por el nodo) y hacer daño a otros EVs.

Los recursos contabilizados y controlados son principalmente memoria y objetos en el kernel tales como los segmentos de memoria compartidos IPC, buffers de red, etc... Cada recurso puede verse desde `/proc/user_beancounters` y tiene cinco valores asociados con este: uso actual, uso máximo (por el tiempo de vida de un EV), contador de barrera, de límite y de falla. El significado de los parámetros barrera y límite es equivalente a los parámetros relacionados con un límite soft y un límite hard, es decir un límite de aviso y un límite en que se deben tomar acciones. Si cualquier recurso alcanza el contador de límite o de falla, entonces el dueño del EV puede ver si algo malo está pasando analizando la salida de `/proc/user_beancounters` en su EV.

[98] Es una estructura de datos, propia de los sistemas de archivos tradicionalmente empleados en los sistemas operativos tipo UNIX, como es el caso de Linux.

Punto de comprobación y migración en vivo

La característica de la migración en vivo y punto de comprobación se liberó para OpenVZ a mediados de abril de 2006. Esta permite migrar un EV desde un servidor físico a otro sin necesidad de apagarlo o reiniciarlo. El proceso se conoce como punto de comprobación: un EV se congela y todo su estado se guarda al archivo en disco. Este archivo puede ser transferido a otra máquina y un EV puede descongelarse (es decir, restaurarse) allí. La demora es de unos pocos segundos.

Dado que cada pieza de estado de EV (incluyendo conexiones de red abiertas) se guarda, desde la perspectiva del usuario parece una demora en la respuesta: una transacción de base de datos que dura más tiempo de lo normal, pero cuando finaliza, el usuario no nota que su base de datos, ahora se está ejecutando en otra máquina. Esta característica hace posible escenarios tales como actualizar un servidor sin necesidad de reiniciarlo: si la base de datos necesita más memoria o recursos de CPU, sencillamente se debe comprar un servidor mejor y más nuevo y migrar en vivo el EV a éste, migrar todos los EVs a otro, apagarlo, agregar memoria, arrancarlo de nuevo y migrar todos los EVs de nuevo.

**Plantillas y vzpkg**

Las plantillas son imágenes precreadas que se usan para crear un nuevo EV. Básicamente, una plantilla es un conjunto de paquetes que contiene todo el sistema de ficheros de un sistema operativo, junto con sus aplicaciones instaladas, todo comprimido en un solo archivo tarball[99]. Durante la fase de creación de la máquina OpenVZ a partir de la plantilla, se desempaqueta el tarball. Usando una técnica de caché de plantillas, un nuevo EV se puede crear en segundos.

Las herramientas vzpkg constituyen un conjunto de herramientas para facilitar la creación de caché en plantillas.

Dado que estas plantillas cambian constantemente, existen enlaces donde puedes consultar directamente tanto las plantillas oficiales como las contribuidas por la comunidad, así como descargarlas directamente. Existen plantillas precreadas para casi todas las distribuciones más comúnmente utilizadas (Debian, Asianux, Centos,

[99] Conocido también como tar, es un software que permite agrupar un conjunto de paquetes en un solo archivo.

Fedora, Gentoo, Mandriva, OpenSuse, AltLinux, SlackWare, Suse y Ubuntu).

**Características distintivas**

Escalabilidad

Como OpenVZ emplea un modelo de kernel único, es tan escalable como el propio kernel de Linux, lo que significa que soporta hasta 64 CPUs y hasta 64 GB de RAM. Un entorno virtual único se puede escalar hasta el equipo físico entero, usando todos las CPUs y toda la RAM.

De hecho, algunos usuarios están usando una sola máquina OpenVZ como único Entorno Virtual. Esta situación puede sorprender a primera vista, pero dado que un EV único usa todos los recursos del hardware con un rendimiento nativo, y tiene agregados beneficios tales como la independencia del hardware, la administración de recursos y la migración en vivo, este tipo de uso se convierte en una opción más que justificada en muchos escenarios.

Densidad

OpenVZ es capaz de alojar cientos de Entornos Virtuales con un hardware decente (las principales limitaciones son RAM y CPU).

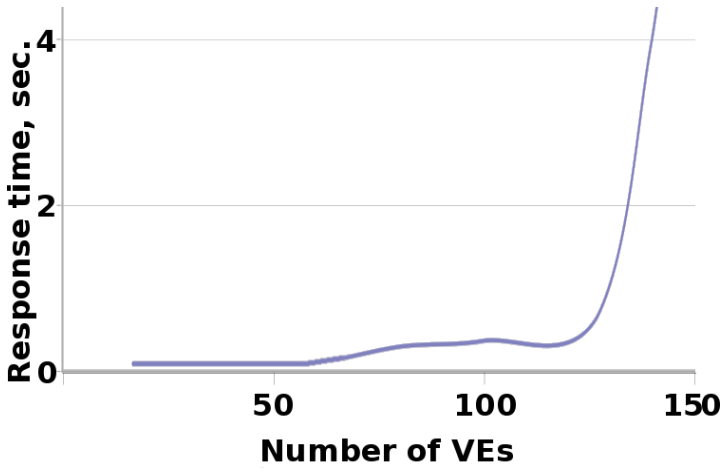



Figura 17 – Tiempo de respuesta respecto número de EV

La figura 17, muestra la relación del tiempo de respuesta de un servidor web Apache de un EV sobre el número de EVs. Las medidas fueron hechas en una máquina con 768 MB de RAM; cada EV estaba ejecutando el conjunto usual de procesos: init, syslogd, crond, sshd y Apache. El servidor de apache estaba sirviendo páginas estáticas, que fueron transmitidas por http, y se midió el primer tiempo de respuesta.

Como se puede ver, según crece el número de EV, el tiempo de respuesta se hace más alto a causa de la disminución de la memoria RAM.

En este escenario es posible ejecutar hasta 120 EVs en estos 768 MB de RAM. Esto se extrapola linealmente, de manera que es posible ejecutar hasta aproximadamente 320 EVs en un equipo con 2 GB de RAM.

Administración masiva

Un propietario (root) de un servidor físico OpenVZ (también conocido como nodo de hardware) puede ver todos los procesos y archivos del EV. Esto hace que la administración masiva de escenarios sea posible. Si consideramos que en VMware o Xen para aplicar una actualización de seguridad a unos 10 servidores virtuales se debe iniciar una sesión en cada uno y ejecutar el procedimiento de actualización nos damos cuenta que sería el mismo proceso que deberíamos hacer si tuviéramos diez servidores físicos independientes.

Por otro lado, en el caso de OpenVZ, se puede ejecutar un simple script de intérprete de comandos que actualice todos (o sólo algunos seleccionados) EVs a la vez.

# Anexo A.3

### Problema con LXC versión 4.0 beta de Proxmox VE

Cuando se empezó el proyecto, en enero de 2015, la última versión de Proxmox VE que había, era la estable, la 3.4. Esta versión, como he comentado antes, virtualiza los contenedores mediante OpenVZ, así pues, todo el proyecto se ha realizado con este tipo de contenedores. En verano de 2015, apareció la versión beta de Proxmox VE 4.0. Esta versión virtualiza los contenedores mediante LXC, eliminando por completo la virtualización mediante OpenVZ, ahora mismo, LXC es la opción que se aconseja y OpenVZ empieza a quedar obsoleto. De esta forma, como posible mejora de este trabajo, estaría la actualización a la versión 4.0 de Proxmox VE y la transformación de las máquinas OpenVZ a LXC, que simplemente consiste en la exportación de la máquina en OpenVZ y su posterior importación como máquina LXC.

LXC es el nuevo estandarte de la virtualización de contenedores en Linux. Aunque existe desde hace tiempo, LXC no ha sido la opción favorita hasta hace muy poco. Es por eso que aún hay mucha gente que sigue usando OpenVZ y es también el motivo por el que no se ha usado en este proyecto.

La tabla comparativa entre OpenVZ y LXC es la siguiente

| | **OpenVZ** | **LXC** |
|---|---|---|
| **Viene con el kernel de Linux** | No | Sí |
| **Limita el uso de memoria** | Sí | Sí |
| **Limita el uso de memoria del kernel** | Sí | No |
| **Limita el uso de la CPU** | Sí | Sí |
| **Limita el uso del disco** | Sí | Sí |
| **Limita las IO del disco** | No | Sí |
| **Snapshot** | Sí | Sí |
| **Migración Live** | Sí | Sí |
| **Seguridad del contenedor** | Sí | Sí |

A continuación se detallan estas diferencias:

Viene con el kernel de Linux

LXC está incorporado en el kernel de Linux mientras que OpenVZ no. En consecuencia, instalar LXC, es mucho más simple porque no necesita la instalación de un kernel personalizado. Además, las actualizaciones y los parches de OpenVZ sólo se realizan en los kernels antiguos, así que en los nuevos kernels no se podrán instalar, y por tanto, no podrán funcionar.

Limita el uso de memoria

Limitar el uso de la memoria, funciona tanto en OpenVZ como en LXC. En LXC se pueden establecer opciones *cgroup* en el archivo de configuración para ajustar estos límites de uso. En OpenVZ, dependiendo del kernel que se use, se pueden añadir más parámetros para realizar una configuración más precisa.

Limita el uso de memoria del kernel

La memoria del kernel puede ser usada para interactuar con el sistema. Cuando se usa OpenVZ, esta memoria puede ser limitada por cada EV. Actualmente esta opción no está en LXC aunque hay previsiones de que se implemente en un futuro.

Limita el uso de la CPU

Este límite funciona tanto en LXC como en OpenVZ. El ancho de banda de la CPU se limita asignando *"shares"*. Cuantos más *"shares"* tenga un contenedor, más fácil es que este contenedor obtenga tiempo de cálculo de la misma. También se pueden limitar el número de núcleos que se usaran en ese contenedor.

Limita el uso del disco

El límite del uso de disco funciona tanto en OpenVZ como en LXC. De esta forma se puede definir cuánto espacio, del disponible en el servidor, se quiere asignar a esa máquina virtual y poderlo modificar dinámicamente según convenga.

Limita las IO del disco

Limitar las entradas y salidas del disco es importante en algunos escenarios donde se tienen varias aplicaciones que realizan muchos accesos y escrituras al disco. OpenVZ tiene ese parámetro de configuración sólo en la aplicación comercial Virtuozzo mientras que en LXC se configura mediante las opciones *cgroup* comentadas anteriormente.

Snapshot

Es una función muy parecida a la hibernación. Está disponible tanto en OpenVZ como en LXC y permite guardar el estado de una máquina en un archivo del que más adelante puede ser recuperado.

Migración Live

Significa que una máquina virtual puede ser trasladada a otro host físico sin tener que apagarse ni reiniciarse. En otras palabras, el contenido de la memoria es trasladado al nuevo host. Esta opción funciona en OpenVZ desde hace tiempo pero es una nueva característica de las últimas versiones de LXC.

Seguridad del contenedor

La seguridad es un tema a tener muy en cuenta en la virtualización. El aislamiento entre contenedores es esencial ya que sino entre dos contenedores se podrían transmitir datos y generar brechas de seguridad graves. Esta seguridad existe tanto en LXC como en OpenVZ. Aunque hace poco tiempo, en LXC este tema fue un problema, llegando al extremo de que desde un contenedor se podía llegar a reiniciar el servidor de máquinas virtuales entero. De todas formas, esas brechas fueron solucionadas muy rápidamente.

# Anexo B

## Anexo B.1

Las principales características de que dispone LTSP son:

**Información centralizada**
Como la información se encuentra en un solo lugar, esto facilita la realización de backups y evita que se guarden archivos que no sean del negocio.

**Más fácil de asegurar**
Los clientes ligeros pueden ser diseñados de modo que ni siquiera los datos de aplicación residan en el cliente (apenas son exhibidos en la pantalla), centralizando la protección contra el *malware* y reduciendo los riesgos de hurto de los datos físicos.

**Seguridad de datos mejorada**
Si un dispositivo del cliente ligero sufre una avería, no se perderá ningún dato, puesto que residen en el servidor de terminales y no en el dispositivo de operación.

**Ejecución en local**
En ordenadores clientes que disponen de un mínimo de recursos, se pueden configurar para que algunas aplicaciones se puedan ejecutar en local permitiendo reducir la carga del servidor que entrega los escritorios.

## Anexo B.2

Entre las características más significativas de VMWare Horizon View destacan:

a) Proporcionar gráficos virtualizados de alto rendimiento y gran densidad comparables a los de un PC de altas prestaciones.
b) Reducir el coste total de propiedad del almacenamiento con VMware Virtual SAN[100]. Se puede crear un almacén de datos en clúster a partir del almacenamiento (las unidades de disco duro y SSD) presentes en los hosts ESXi existentes. Es asequible,

[100] Virtual SAN es una solución de almacenamiento virtual ampliable de alto rendimiento y que se configura fácilmente

eficiente y fácil de implementar, con costes fijos y previsibles de la capacidad de ampliación.

c) Usar el protocolo PCoIP[101] para la conexión de los clientes con los escritorios, compatibilidad con una gran cantidad de clientes ligeros del mercado.
d) Simplificar la implementación y el funcionamiento de cualquier entorno de PC virtuales.
e) Acelerar la consecución de resultados empresariales con soluciones integrales predefinidas.
f) Confiar en arquitecturas de referencia preconfiguradas que van más allá de la simple información sobre la infraestructura.
g) Ampliar las necesidades de la empresa con opciones de servicios administrados.
h) Reducir el riesgo de pérdida de datos mediante la centralización de los datos, las aplicaciones y los recursos.
i) Simplificar el mantenimiento gracias a una prestación de asistencia que cubre las soluciones.

Las principales ventajas de VMWare Horizon View en su versión 6, respecto a sus antiguas versiones son:

**Mejora del protocolo PCoIP**

Consumiendo ahora hasta un 30% menos de ancho de banda, sin realizar ninguna modificación.

**VMware Horizon 6 no es tan solo una evolución**

Se trata de un nuevo concepto en el que los usuarios tienen todo en un mismo portal web, Escritorio mediante VMware, aplicaciones de Office 365, aplicaciones Citrix, etc... sin importar donde se estén ejecutando todas ellas.

**Monitorización eficiente**

Del entorno de VMware Horizon 6, mucho más que en sus anteriores versiones.

**Nueva Arquitectura Cloud Pod[102]**

Esta tecnología permite escalar de manera más eficiente el almacenamiento y el entorno, a través de diferentes Data Center.

[101] http://www.teradici.com/pcoip-technology
[102] https://blogs.vmware.com/euc/2014/04/vmware-horizon-6-introducing-cloud-pod-architecture.html

# Anexo B.3

En su última versión, la 7.6, desde XenDesktop defienden que está diseñado para ayudar a los negocios a movilizar las aplicaciones de Windows, mientras simplifica la distribución y el acceso al escritorio. Además de mantener segura la propiedad intelectual y la privacidad de los datos.

El nuevo HDX Mobile[103] optimiza las aplicaciones de Windows con un aspecto y una sensación natural. El entorno de desarrollo que ofrece, otorga a los desarrolladores de grandes empresas el poder de optimizar sus aplicaciones Windows existentes para cualquier entorno móvil.

Las tecnologías de redireccionamiento y compresión incorporados en HDX están específicamente diseñadas para tratar con redes de banda ancha y dispositivos de consumo móviles. Los administradores también pueden dar acceso a videos completos en alta definición mediante redes 3G, lo que permite aprovechar la potencia de los smartphones y las tabletas para decodificar los gráficos de video y, a la vez, reducir la carga del servidor mediante la aceleración por unidad de procesamiento de gráficos compartida. Las principales ventajas que aporta son:

**Arquitectura de administración mejorada que permite simplicidad y automatización**

a) Simplifica la distribución de aplicaciones mediante la automatización y la unificación.
b) Ejecuta de forma simultánea múltiples versiones de Windows Server y de los sistemas operativos de escritorio, con el soporte técnico para Windows Server 2012 y Windows 8.
c) Elimina errores e incrementa la velocidad de producción con flujos de trabajo fáciles de usar.
d) Accede a la nueva consola optimizada para línea de soporte y resolución de problemas en tiempo real, con herramientas analíticas integradas para la administración de la capacidad a largo plazo, planeación de recursos y seguridad del nivel de servicio.
e) Asegura la propiedad intelectual y protege la privacidad de sus datos con el nuevo NetScaler Gateway[104] que proporciona

[103] https://www.citrix.com/go/mobile-sdk-for-windows-apps.html
[104] https://www.citrix.com/products/netscaler-gateway/overview.html

políticas de acceso individualizado, configuradas automáticamente con una simple dirección corporativa de correo electrónico.

f) Optimiza aplicaciones como Skype por medio de una colaboración de voz y video basada en la nube con procesamiento local.

**Virtualización de aplicaciones y escritorios para cualquier estilo de trabajo**

a) Distribuye aplicaciones o puestos de trabajo virtuales sin las restricciones de una plataforma tradicional, a una fracción del coste, a través de una plataforma de distribución única y preparada para la nube.
b) Dispone de aplicaciones y datos corporativos en cualquier entorno.
c) Distribuye puestos de trabajo virtuales completamente personalizados.
d) Es independiente del hipervisor, almacenamiento y redes.

# Anexo C

Las máquinas virtuales del entrono tienen la siguiente función:

## Anexo C.1

### Máquina 100 (Router)

Esta máquina es la que se encarga de simular un router de salida a la WAN. Se decidió incorporar esta máquina para poder crear un entorno que estuviera separado totalmente de la red corporativa del Ajuntament de Sant Joan Despí y, así poder realizar pruebas en la configuración del DHCP, de las DNS, etc… sin arriesgarse a que se desconfigurará la red corporativa.

Los servicios que se han instalado son:

Router

El servicio de router básicamente se usa para interconectar dos redes. Las dos redes que se usarán tienen el siguiente rango:

a) Interna: 192.168.2.0/24
b) Externa: 192.168.1.0/24 con la gateway[105] 192.168.1.1

Se ha integrado también en esta máquina un firewall por software mediante *iptables*. Las opciones básicas que se han configurado han sido: que los paquetes de la red interna dirigidos a la red externa, puedan volver con la respuesta solicitada al ordenador que ha hecho la consulta inicial (SNAT); también se han configurado las reglas para bloquear las conexiones entrantes de algunos puertos y, así simular la seguridad que podría haber en el firewall de una red corporativa.

DHCP

DHCP es un protocolo de red que permite a los clientes de una red IP obtener sus parámetros de configuración automáticamente. Se trata de un protocolo de tipo cliente-servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas que, las va asignando a los clientes conforme éstas van quedando libres, sabiendo en todo momento quién ha estado en posesión de esa IP,

[105] O puerta de enlace es el dispositivo que permite interconectar redes de computadoras con protocolo de comunicaciones y arquitecturas diferentes a todos los niveles de comunicación.

cuánto tiempo la ha tenido y a quién se la ha asignado después. Mediante este servicio, también se envía a los clientes la configuración que tienen que cargar para realizar un arranque por PXE que se explicará con más detalle en un punto posterior.

DNS

Es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada. Este sistema asocia información variada con nombres de dominios asignado a cada uno de los participantes. Su función más importante, es traducir (resolver) nombres inteligibles para las personas en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente.

El servidor DNS utiliza una base de datos distribuida y jerárquica que almacena la información asociada, a nombres de dominio en redes como Internet. Aunque como base de datos, el DNS es capaz de asociar diferentes tipos de información a cada nombre, los usos más comunes son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico de cada dominio.

La asignación de nombres a direcciones IP es ciertamente la función más conocida de los protocolos DNS. Por ejemplo, si la dirección IP del sitio web de Google es *200.64.128.4*, la mayoría de la gente llega a este equipo especificando *google.es* y no la dirección IP. Además de ser más fácil de recordar, el nombre es más fiable ya que la dirección numérica podría cambiar por muchas razones, sin que tenga que cambiar el nombre. En este proyecto, se usará el protocolo DNS principalmente para resolver los nombres del dominio en el directorio PDC, aunque también se usará para no tener que recordar las direcciones IP de todo el entorno de administración.

Esta máquina tiene las siguientes direcciones IP:

a) Interna: 192.168.2.1 (será la gateway de todo el entorno)
b) Externa: 192.168.1.37 (es la IP que está dentro del rango de una de las redes corporativas de Sant Joan Despí).

Configuraciones adicionales

Hay que destacar que para poder controlar las *iptables* dentro de los contenedores OpenVZ en Proxmox hay que cambiar la configuración del servidor manualmente, concretamente añadiendo en el archivo de configuración de OpenVZ la siguiente línea:

```
IPTABLES="ipt_REJECT ipt_recent ipt_owner ipt_REDIRECT
    ipt_tos ipt_TOS ipt_LOG ip_conntrack ipt_limit
ipt_multiport iptable_filter iptable_mangle ipt_TCPMSS
  ipt_tcpmss ipt_ttl ipt_length ipt_state iptable_nat
                    ip_nat_ftp"
```

# Anexo C.2

### Máquina 117 (Samba PDC)

Ésta máquina es la encargada de montar el controlador del dominio para todo el entorno.

El controlador de dominio es el centro neurálgico de un dominio Windows, tal como un servidor NIS lo es del servicio de información de una red Unix. En este caso en concreto, como tenemos clientes Unix y clientes Windows, la mejor opción es la de elegir el controlador de dominio ya que también es compatible con Unix. Los controladores de dominio tienen una serie de responsabilidades siendo la más importante la de la autentificación. La autentificación es el proceso de garantizar o denegar a un usuario el acceso a recursos compartidos o, a otra máquina de la red, normalmente a través del uso de una contraseña. Cada controlador de dominio usa un *security account manager* (SAM) para mantener una lista de pares de nombre de usuario y contraseña. De esta forma, puede crear un repositorio centralizado con estos pares, lo cual es más eficiente que mantener en cada máquina cliente centenares de contraseñas para cada recurso de red disponible.

En un dominio Windows, cuando un cliente no autorizado solicita un acceso a los recursos compartidos de un servidor, el servidor actúa y pregunta al controlador de dominio si ese usuario está autentificado. Si lo está, el servidor establecerá una conexión de sesión con los derechos de acceso correspondientes para ese servicio y usuario. Si no lo está, la conexión es denegada.

Una vez que un usuario es autenticado por el controlador de dominio, una ficha especial (o token) de autentificación será retornada al cliente, de manera que el usuario no necesitará volver a identificarse para acceder a otros recursos en dicho dominio.

Para este proyecto, como se le ha dado mucha importancia al objetivo *lowcost* se ha decidido montar el dominio PDC mediante Linux y así, poder prescindir de la cuota que se tendría que pagar para poder usar un Windows Server que actuase de controlador. En contrapartida, montar un controlador con Linux, tiene algunas limitaciones cuando se tienen que establecer directivas de usuario, ya que, no se tienen tantas opciones como tendría un Windows Server. De todas formas, para los objetivos de este proyecto, estas limitaciones no suponen un problema más allá de la dificultad de su configuración inicial.

La compatibilidad del controlador con Linux, permite que con este solo servidor tanto los escritorios virtuales de Windows como los de Linux, puedan iniciar la sesión con la base de datos central y heredar los permisos de cada usuario.

La base de datos que se ha configurado para que funcione el sistema es una LDAP (Lightweight Directory Access Protocol). Este es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también se considera una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas. En este caso hemos configurado una base de datos LDAPS que es una LDAP pero en que los datos entre el cliente y el servidor viajan encriptados.

El programa principal de esta máquina virtual es Samba, que es una implementación libre del protocolo de archivos compartidos de Microsoft Windows (antiguamente llamado SMB i renombrado recientemente a CIFS) para sistemas de tipo UNIX. De esta forma, es posible que ordenadores con Linux se vean como servidores o actúen como clientes en redes de Windows. Samba también permite validar usuarios haciendo de Controlador Principal de Dominio (PDC) y actuar como miembro del dominio.

Además, es capaz de servir colas de impresión, directorios compartidos y autentificar con su propio archivo de usuarios, se encarga de generar el dominio y de permitir que Windows pueda ver las carpetas que se le comparten mediante la directiva de usuario.

En esta máquina también se ejecuta un servidor web, concretamente un apache2[106], que se encarga de realizar cambios en la base de datos LDAPS, añadiendo nuevos usuarios al dominio, creando las carpetas personales necesarias para cada usuario, administrando los grupos del controlador, etc... El anexo I describe los pasos para crear un usuario en este sistema.

Esta máquina tiene la dirección IP *192.168.2.76* con el nombre de dominio *samba.ajuntament*.

# Anexo C.3

### Máquina 119 (NFS Server)

Esta máquina se encarga de contener los archivos de todos los usuarios del entorno. Contiene las carpetas personales que el controlador de dominio distribuye en los inicios de sesión de los clientes Windows y Linux. Hay que destacar que este proceso se realiza de forma distinta según el cliente. Si el cliente es:

Linux

La carpeta se monta directamente mediante NFS de la máquina cliente a esta máquina virtual. NFS es un protocolo de nivel de aplicación, según el Modelo OSI. Es utilizado para sistemas de

[106] https://httpd.apache.org/

archivos distribuidos en un entorno de red de ordenadores de área local. Permite que distintos sistemas conectados a una misma red accedan a ficheros remotos como si se tratara de locales.

Windows

La carpeta personal del usuario se monta como carpeta compartida mediante NFS en la máquina virtual anterior (Samba PDC), a partir de ahí, el controlador se encarga de pasar al cliente la ruta del tipo Windows (*[Nombre del servidor]**[Recurso]*) para que este lo monte como disco compartido del usuario.

Esta máquina tiene exclusivamente, en ejecución, el servidor de NFS. La razón para separarla de las otras máquinas, es que realmente es la máquina que más cambios sufrirá ya que cada vez que un usuario modifique, cree o elimine un archivo, este cambio se verá reflejado en esta máquina únicamente. De esta forma, al tener la máquina separada, podemos hacer que se realice una copia de seguridad más a menudo que la de las demás, permitiendo optimizar el espacio de estas copias al máximo.

Para poder ejecutar un servidor de NFS, hay que destacar que en OpenVZ se tiene que añadir en el archivo de configuración de la máquina la siguiente línea:

```
Features="nfs:on"
```

Esta máquina tiene la dirección IP *192.168.2.75*

# Anexo C.4

### Máquina 123 (TFTP Server)

Esta máquina tiene dos funciones principales:

Servidor de TFTP

Es un protocolo de transferencia muy simple semejante a una versión básica de FTP. TFTP a menudo se utiliza para transferir pequeños archivos entre ordenadores en una red.

Las características principales del TFTP son:

a) Utiliza UDP (en el puerto 69) como protocolo de transporte
b) No puede listar el contenido de los directorios.
c) No existen mecanismos de autenticación o cifrado.
d) Se utiliza para leer o escribir archivos de un servidor remoto.
e) Soporta tres modos diferentes de transferencia, “netascii”, “octet” y “mail”.

En este proyecto, este servidor se usa para conseguir que los clientes puedan lograr un arranque por PXE (Preboot execution environment). Este es un entorno para arrancar e instalar el sistema operativo en ordenadores a través de una red, de manera independiente a los dispositivos de almacenamiento de datos disponibles (como discos duros) o de los sistemas operativos instalados. Su funcionamiento es el siguiente:

1. El ordenador cliente debe estar configurado para que arranque a través de la red. Casi todos los ordenadores del mercado tienen esa opción en la BIOS[107], pero por ejemplo, los clientes que se usarán en este proyecto, las Raspberry Pi, no tienen esa opción. Más adelante se explicará cómo se consigue que estos dispositivos arranquen de este modo.
2. Al arrancar, el cliente el firmware del cliente solicita al servidor de DHCP (máquina 100) que le otorgue una IP, como he comentado anteriormente. El servidor le da la IP junto con los servidores de arranque PXE disponibles.

[107] Es el primer programa que se ejecuta cuando se enciende la computadora.

3. Tras analizar la respuesta, el firmware solicitará al servidor de arranque apropiado la ruta de un *network bootstrap program* (NBP), lo descargara en la memoria RAM del computador mediante TFTP, lo verificará, y finalmente lo ejecutará. Más adelante, se explicará más detalladamente cómo realiza a partir de aquí el arranque de los dos tipos de clientes que se consideran en este entorno: las Raspberry Pi y los ordenadores fijos o portátiles.

Servidor NFS con las imágenes de arranque

Esta máquina virtual además de contener el servidor TFTP, tiene un servidor de NFS el cual se usa para que los clientes puedan cargar el sistema operativo base montándoselo en el sistema de ficheros raíz y, permitiendo que sea el mismo en todos los clientes, facilitando así, la administración de todos los dispositivos. Por la variación en la arquitectura del procesador, se ha tenido que distinguir entre dos carpetas (o dos exports), la que contiene el sistema de archivos de los clientes tipo Raspberry Pi (ARMv7l) y, la que contiene el sistema de los clientes (en este caso con una arquitectura x86 para que sea compatible tanto para AMD64 como para x86). Estos dos directorios contienen el sistema de archivos completo que el cliente tendrá que cargar. Adicionalmente, se ha configurado la máquina virtual de tal forma que se puede modificar el sistema operativo realizando un CHROOT sobre la carpeta. De esta forma, para añadir alguna opción, por ejemplo, a los clientes tipo Raspberry que realizan la conexión RDP[108], solamente habrá que hacer un CHROOT a la carpeta que contiene el sistema de archivos para Raspberry y, a continuación realizar los cambios. A partir de ese momento cualquier Raspberry que inicie tendrá esa nueva configuración.

Esta máquina tiene la dirección IP 192.168.2.7

[108] Es un protocolo propietario desarrollado por Microsoft que permite la comunicación en la ejecución de una aplicación entre un terminal (mostrando la información procesada que recibe del servidor) y un servidor Windows (recibiendo la información dada por el usuario en el terminal mediante el ratón o el teclado).

# Anexo C.5

## Máquina 106 (Proxmox View)

Esta máquina es la que realiza la función del Horizon View en el paquete que vende VMWare, es decir, se encarga de gestionar las distintas plantillas y sistemas operativos que hay en el sistema, tanto de crearlas como de eliminarlas u obtener información de ellas. La máquina consta básicamente de 4 servicios

Proxy

Todos los clientes de este entorno, después de que el usuario haya elegido el sistema operativo que quiere utilizar, preguntan a esta máquina cuál es la IP de la máquina que está ejecutando la plantilla y el sistema operativo que el cliente quiere acceder. Este proxy les asigna una de las que están disponibles y la reserva para que ningún otro usuario pueda acceder a ella. Todos estos datos son almacenados en una base de datos para que sea más fácil tratar con ellos. De esta forma, una vez el cliente recibe esta IP, iniciará la conexión de escritorio remoto con la máquina virtual que corresponda.

Firewall

Como mecanismo adicional de seguridad en el entorno, esta máquina tiene configurado mediante *iptables* un firewall que solo permite acceder al conjunto de máquinas virtuales con sistemas operativos, a los clientes cuya *MAC* esté autorizada en la base de datos que esta misma máquina virtual contenga. Es decir, si queremos añadir un dispositivo cliente nuevo a nuestro entorno, debemos antes, añadirla en la base de datos para que pueda acceder a los escritorios.

Router

Para separar la subred que contiene los escritorios virtuales con la local, donde están el resto de máquinas virtuales que se han comentado anteriormente, se usa la aplicación de router básico que existe en Linux utilizando la configuración adecuada. Como por ejemplo tal y como se ha comentado anteriormente en la máquina virtual que hace de router, para que los paquetes puedan llegar de su origen a su destino (SNAT). Así pues las dos redes son:

a) Interna (escritorios): 192.168.3.0/24
b) Externa (LAN): 192.168.2.0/24

Servidor web

Para poder controlar el estado de los escritorios virtuales, forzar su eliminación, añadir o eliminar las *MAC* que están permitidas en el entorno y, elegir los parámetros de creación de escritorios virtuales, se ha instalado este servidor web donde se puede tener control de todo ello. En la figura 18 se puede observar como es la interfaz web que hemos programado.

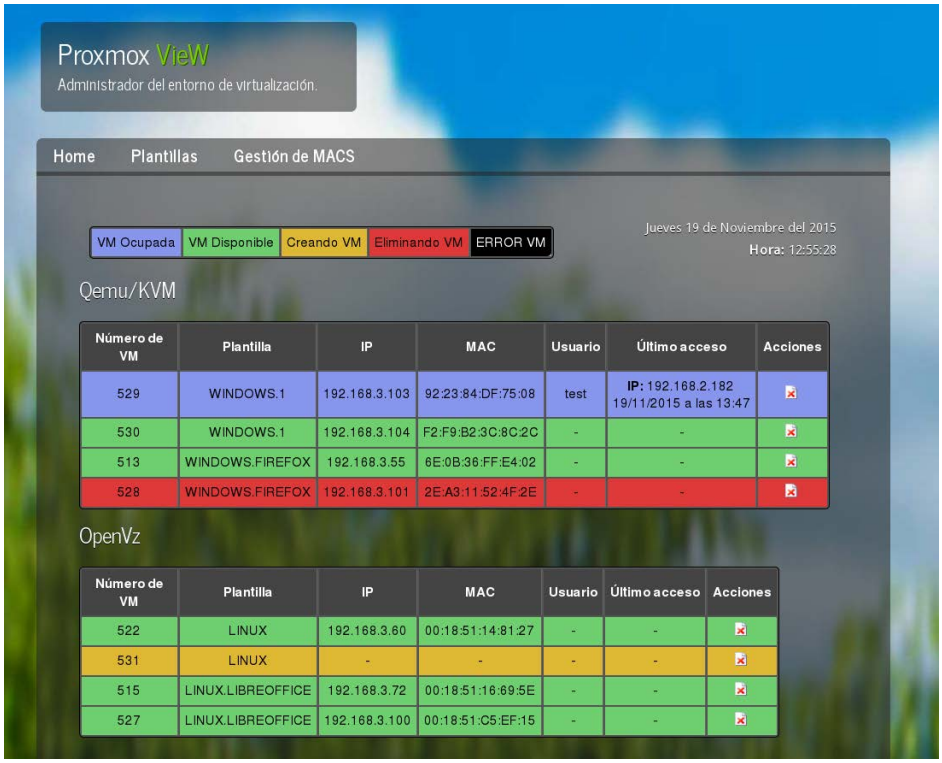



Figura 18 – Menú principal Proxmox View

A continuación se detalla la información que se ofrece, así como los parámetros que se pueden cambiar

En la sección *"Home"* de la página web se puede obtener un resumen de los escritorios virtuales que se están ejecutando en ese mismo momento. Como se puede observar en la figura 18, tenemos una leyenda en la parte superior que asocia los colores a los estados de los escritorios.

a) VM Ocupada (color azul): el escritorio virtual está reservado por un usuario, de tal forma que no es accesible por ningún otro usuario y, que solo podrá eliminarlo él y el administrador del sistema.
b) VM Disponible (color verde): el escritorio virtual está listo para ser entregado a cualquier cliente que solicite uno similar con esa plantilla.
c) Creando VM (color amarillo): el escritorio virtual está generándose e iniciándose para obtener la *MAC* y la *IP* del escritorio y poder pasar al estado *"VM Disponible".*
d) Eliminando VM (color rojo): cuando el usuario que tiene el escritorio ocupado, o el administrador del sistema, decide eliminar el escritorio virtual que estaba utilizando, éste pasa a estado de *"Eliminando VM"*, de tal forma que, cuando el proceso de eliminación finalice, el escritorio desaparecerá de la lista, siendo irrecuperable.
e) ERROR VM (color negro): si por cualquier tipo de situación el escritorio estuviera en un estado diferente a los anteriores, sería considerado un error y por tanto, se pondría en este estado. La actuación más lógica sería que el administrador eliminará el escritorio para que se volviese a generar.

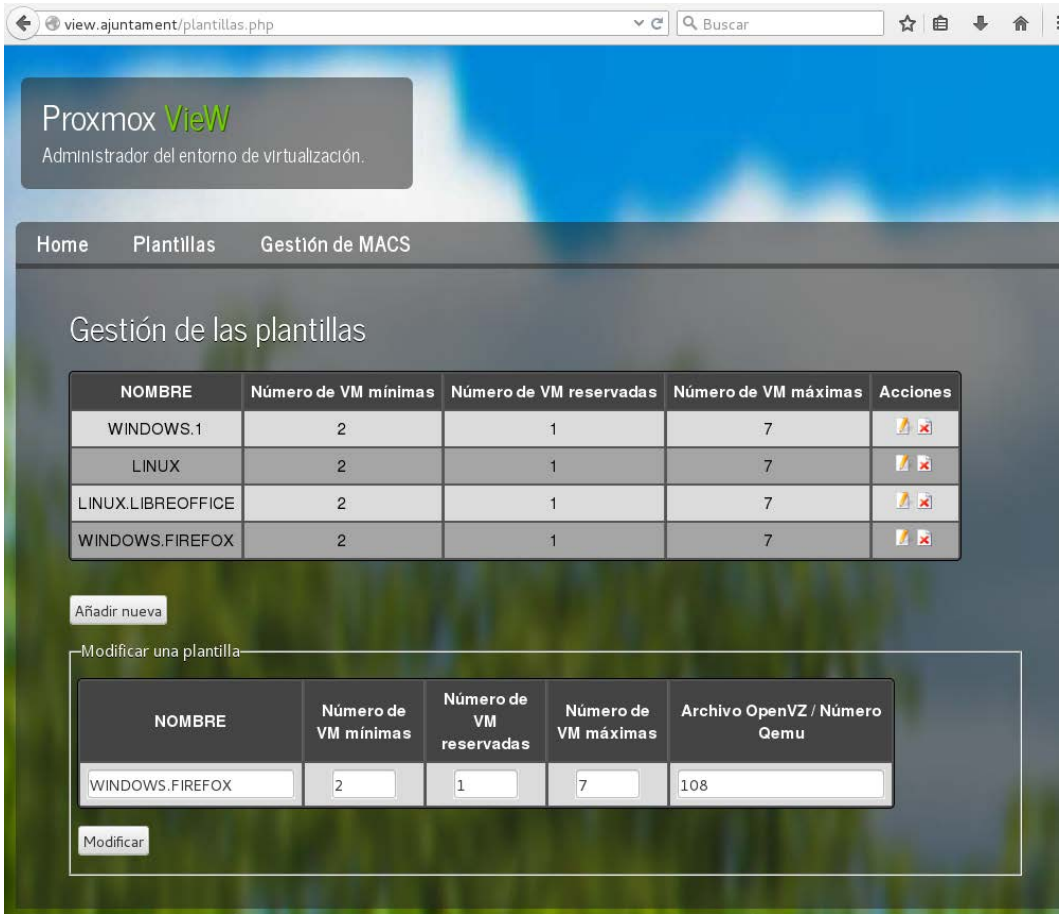



Figura 19 – Menú plantillas Proxmox View

Por otra parte, en la sección “Plantillas” (ver figura 19) se pueden configurar todas las opciones relacionadas con la generación automática de los escritorios virtuales.

Como podemos observar en esta sección, se puede elegir el número mínimo de escritorios virtuales de esa plantilla que se deben generar y el número de escritorios que deben estar disponibles para que un nuevo usuario pueda acceder a ellos. Finalmente, se puede indicar el número máximo de escritorios virtuales de esa plantilla, que pueden estar generados. También debemos especificar el número o ruta que tiene la plantilla para que el programa sepa a qué plantilla del entorno Proxmox se refiere esa configuración. En el anexo G se explica detalladamente cómo añadir un nuevo escritorio virtual al entorno.

En la sección “Gestión de MACS” (ver figura 20) es donde se añaden las direcciones MAC de los clientes (Raspberry Pi, portátiles y ordenadores

fijos) que quieran permitirse que accedan a los escritorios virtuales. También se puede añadir una breve descripción de cada cliente para que se puedan reconocer más fácilmente. Si la MAC del dispositivo no está en esta lista, éste no podrá cargar ni tan solo el sistema operativo básico, para poder iniciar la conexión al escritorio remoto.


Figura 20 – Gestión de Macs de Proxmox View

Esta máquina tiene las siguientes direcciones IP:

a) Interna: *192.168.3.1* (será la gateway de todos los escritorios virtuales)
b) Externa: *192.168.2.82* (es la IP que está dentro del rango de la LAN de nuestro entorno). Responde al dominio *view.ajuntament*.

# Anexo D

## Seguridad en las etiquetas NFC

Las etiquetas NFC usadas en este proyecto, por su bajo precio y por su compatibilidad con el lector que usaremos en la Raspberry Pi, son las etiquetas *Mifare*, concretamente las de *Mifare S50 de 1 KB*[109].

La seguridad de este tipo de tecnologías presenta deficiencias que pueden permitir a usuarios malintencionados realizar acciones ilícitas como fraude en sistemas de pago, bypass del sistema de encendido de automóviles, suplantar la identidad de personas o acceder a áreas de acceso restringido, entre otras cosas.

Antes de entrar en los detalles técnicos de las debilidades de seguridad, describiremos su funcionamiento y estructura. En una arquitectura básica *Mifare* encontramos un lector y una etiqueta, ejerciendo una comunicación dentro del radio de acción inalámbrico (unos 5 cm). *Mifare Classic* está basado en el estándar ISO 14443[110] e incorpora un protocolo anti colisiones, es decir, permite trabajar con varias tarjetas que estén dentro del radio de acción del lector al mismo tiempo. La principal mejora respecto a otros tipos de etiquetas, es que la comunicación se produce de forma cifrada tras un proceso de autenticación mutuo lector-tarjeta. Cada 8 bits de transmisión, uno se utiliza para calcular la paridad, pero este es cifrado también, dejando la comprobación de integridad a la capa de aplicación.

Por lo que a su estructura lógica se refiere, la tarjeta está representada como un mapa de memoria con bloques de datos de 16 bytes. Estos bloques de datos se agrupan en sectores. Aquí hay que diferenciar las tarjetas de 1K de las de 4K.

Las de 1K tienen 16 sectores con 4 bloques de datos cada uno.

Las de 4K tienen los 32 primeros sectores compuestos por 4 bloques de datos y los 8 últimos sectores por 16 bloques de datos.

Para ambas, el último bloque de datos de cada sector se denomina *tráiler* y, es donde se guardarán las claves privadas de acceso y los permisos para acceder a cada sector. Por último, hay que destacar que existe un bloque especial que está localizado en el bloque de

[109] https://es.wikipedia.org/wiki/Mifare
[110] https://es.wikipedia.org/wiki/ISO_14443

datos 0 del sector 0 (ver figura 21), que alberga la siguiente información en modo lectura:

a) UID – Identificador único de la tarjeta.
b) BCC – bit count check, calculado por las sucesivas operaciones de los bytes del UID.
c) 11 bytes de datos que identifican al fabricante de la etiqueta.

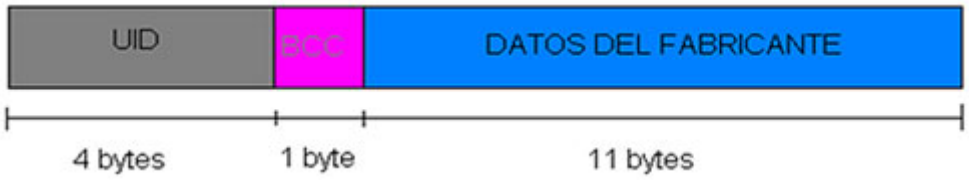



Figura 21 – Memoria del bloque 0 del sector 0

Para que el lector pueda leer y escribir el contenido de un sector, es decir, cualquiera de los 4 bloques de datos que tiene, necesita en primera instancia autenticarse contra él. Este proceso se realiza si y sólo si la clave utilizada por el lector coincide con la del sector tráiler. Mifare permite la utilización de dos claves por sector, la Key A (6 bytes) y la Key B (6 bytes), donde las condiciones o permisos de acceso al sector los marca la máscara *Access Conditions* (3 bytes). El byte restante U no tiene propósito específico aunque a veces se usa para almacenar algún dato adicional. La siguiente figura (figura 22) muestra la distribución en bytes de las *keys* y de la máscara.

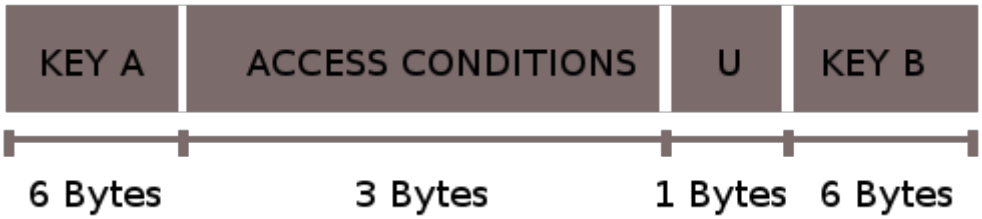



Figura 22 – Memoria del bloque 3 de cada sector

La máscara de 3 bytes que define las condiciones de acceso está compuesta por 3 bits de acceso por cada bloque, definidos como C1, C2 y C3. Como hay 4 bloques por sector, hay 4 C1, 4 C2 y 4 C3. El acceso al bloque 0 se define mediante los bits $C1_0$, $C2_0$ y $C3_0$; el bloque 1 mediante $C1_1$, $C2_1$ y $C3_1$; el bloque 2 mediante $C1_2$, $C2_2$ y $C3_2$ y finalmente el bloque 3 (el tráiler) se define mediante $C1_3$, $C2_3$ y $C3_3$.

Para los bloques del 0 al 2, los permisos se definen como lo muestra la siguiente tabla:

| **Access Bits** | | | **Access Condition** | |
|---|---|---|---|---|
| $C1_0$ $C1_1$ $C1_2$ | $C2_0$ $C2_1$ $C2_2$ | $C3_0$ $C3_1$ $C3_2$ | Lectura | Escritura |
| 0 | 0 | 0 | Key A\|B | Key A\|B |
| 0 | 1 | 0 | Key A\|B | nunca |
| 1 | 0 | 0 | Key A\|B | Key B |
| 1 | 1 | 0 | Key A\|B | Key B |
| 0 | 0 | 1 | Key A\|B | nunca |
| 0 | 1 | 1 | Key B | Key B |
| 1 | 0 | 1 | Key B | nunca |
| 1 | 1 | 1 | nunca | nunca |

Para los permisos del bloque *tráiler* ($C1_3$, $C2_3$ y $C3_3$), los permisos se establecen como sigue:

| Access Bits | | | Access Condition | | | | | | Notas |
|---|---|---|---|---|---|---|---|---|---|
| | | | Key A | | Access Bits | | Key B | | |
| $C1_3$ | $C2_3$ | $C3_3$ | Read | Write | Read | Write | Read | Write | |
| 0 | 0 | 0 | nunca | Key A | Key A | nunca | Key A | Key A | Key A puede leer Key B |
| 0 | 1 | 0 | nunca | nunca | Key A | nunca | Key A | nunca | Key A puede leer Key B |
| 1 | 0 | 0 | nunca | Key B | Key A\|B | nunca | nunca | Key B | |
| 1 | 1 | 0 | nunca | nunca | Key A\|B | nunca | nunca | nunca | |
| 0 | 0 | 1 | nunca | Key A | Key A | Key A | Key A | Key A | key A puede leer Key B |
| 0 | 1 | 1 | nunca | Key B | Key A\|B | Key B | nunca | Key B | |
| 1 | 0 | 1 | nunca | nunca | Key A\|B | Key B | nunca | nunca | |
| 1 | 1 | 1 | nunca | nunca | Key A\|B | nunca | nunca | nunca | |

Una vez definidos todos los bits pertenecientes a los 3 bytes de las condiciones de acceso ($C1_0$, $C2_0$, $C3_0$, $C1_1$, $C2_1$, $C3_1$, $C1_2$, $C2_2$, $C3_2$, $C1_3$, $C2_3$ y $C3_3$) procedemos a generar los 3 bytes poniendo dichos bits en el siguiente orden:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte 6 | $C2_3$ | $C2_2$ | $C2_1$ | $C2_0$ | $C1_3$ | $C1_2$ | $C1_1$ | $C1_0$ |
| Byte 7 | $C1_3$ | $C1_2$ | $C1_1$ | $C1_0$ | $C3_3$ | $C3_2$ | $C3_1$ | $C3_0$ |
| Byte 8 | $C3_3$ | $C3_2$ | $C3_1$ | $C3_0$ | $C2_3$ | $C2_2$ | $C2_1$ | $C2_0$ |

Los bits en rojo corresponden con la negación de los bits mostrados en color azul.

Así pues en resumen, si por ejemplo quisiéramos:

Un sector que tuviera permisos de escritura y lectura mediante la Key B (0xABCDF1234567) pero sólo de lectura mediante la Key A (0x1234567ABCDF) en los bloques 0, 1 y 2. Además, que no se pudiera leer la Key A, ni la Key B y que una vez programado, ninguno de ellos, ni siquiera los bits de acceso, pudieran ser modificados, entonces tendríamos que generar el siguiente bloque 3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte 6 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Byte 7 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Byte 8 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Por lo que finalmente el bloque 3 de ese sector quedaría como se muestra a continuación

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Byte 0** | 0xAB | **Byte 1** | 0xCD | **Byte 2** | 0xF1 | **Byte 3** | 0x23 |
| **Byte 4** | 0x45 | **Byte 5** | 0x67 | | | | |
| **Byte 6** | 0x70 | | | | | | |
| **Byte 7** | 0xF7 | | | | | | |
| **Byte 8** | 0x88 | | | | | | |
| **Byte 9** | Byte U (no se usa) | | | | | | |
| **Byte 10** | 0x12 | **Byte 11** | 0x34 | **Byte 12** | 0x56 | **Byte 13** | 0x7A |
| **Byte 14** | 0xBC | **Byte 15** | 0xDF | | | | |

Y de esta forma los bloques 0, 1 y 2 de ese sector tendrían los permisos que se han planteado en el ejemplo.

Además de los permisos de lectura y de escritura, también existen permisos para poder aumentar y disminuir los valores almacenados en la etiqueta, pero en este proyecto no se tienen en cuenta.

En resumen, el aspecto del mapa de memoria de una tarjeta *Mifare 1K* sería el que se muestra en la figura 23.

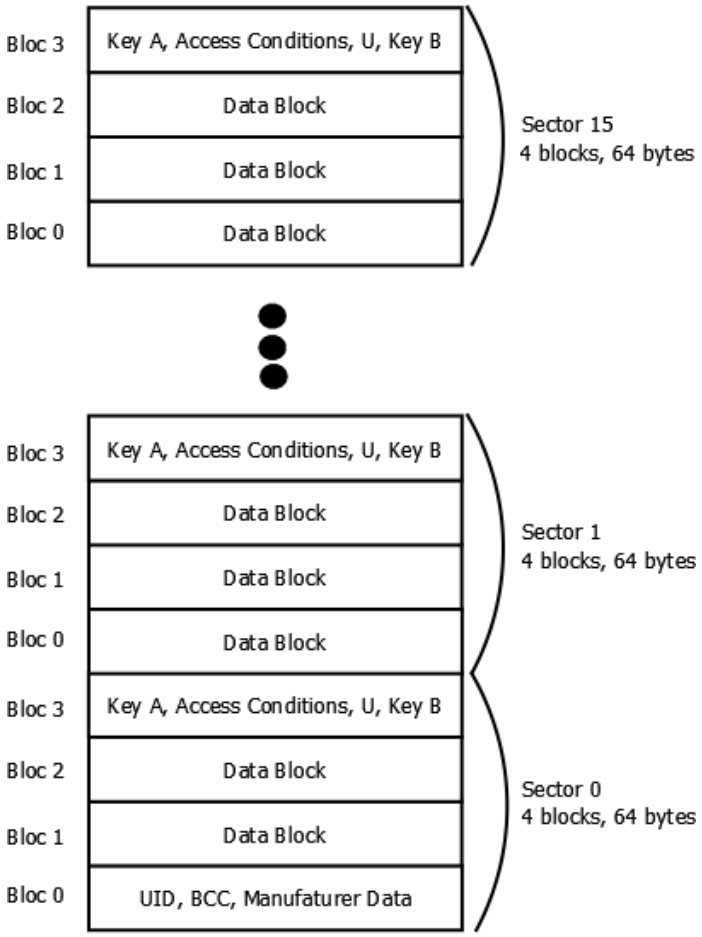



Figura 23 – Mapa de memoria tarjeta Mifare 1K

# Anexo E

Para la escritura de una nueva etiqueta de NFC, podemos usar cualquiera de las Raspberry Pi de nuestro entorno con el lector NFC conectado, solo debemos abrir una terminal, identificarnos como root y ejecutar los scripts que se encuentran en */home/temp/MFRC522-python/*. Todos los scripts de esta esta sección así como los demás del entorno han sido totalmente programados por nosotros, aunque en este caso en particular, se usan librerías de *Python*[111], preprogramadas para la gestión de dispositivos NFC por el puerto SPI.

Antes de programarla debemos ejecutar el script que comprueba que la etiqueta es compatible con nuestro sistema y que se puede programar. Este script es el siguiente:

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
#importamos las librerías necesarias
import RPi.GPIO as GPIO
import random
import os
import MFRC522
import signal
import Read
import Write
from random import randint

#definimos una función para escribir en el terminal con color verde y rojo
def hilite(string, status, bold):
    attr = []
    if status:
        # verde
        attr.append('32')
    else:
        # rojo
        attr.append('31')
    if bold:
        attr.append('1')
    return '\x1b[%sm%s\x1b[0m' % (';'.join(attr), string)

#desactivamos los warnings del puerto de comunicaciones de la Raspberry
GPIO.setwarnings(False)
#leemos la id de la etiqueta a comprobar
cardid=Read.id()

#especificamos todos los sectores para comprobar los permisos
trailers=[7,11,15,19,23,27,31,35,39,43,47,51,55,59,63]
range=[4, 5, 6, 8, 9, 10, 12, 13, 14, 16, 17, 18, 20, 21, 22, 24, 25, 26, 28,
29, 30, 32, 33, 34, 36, 37, 38, 40, 41, 42, 44, 45, 46, 48, 49, 50, 52, 53,
54, 56, 57, 58, 60, 61, 62]
#comprobamos que con la Key 0xFFFFFFFFFFFF podemos acceder a todos ellos
for cadauno in trailers:
        print "-------------------------------->"+str(cadauno)
        Read.read([255,255,255,255,255,255],cadauno)
for cadauno in range:
```

[111] https://www.python.org/

```
        print "------------------------------->"+str(cadauno)
        Read.read([255,255,255,255,255,255],cadauno)
os.system("clear")
#si no recibimos ningún error al leer los sectores quiere decir que la
etiqueta está lista para ser grabada
print ""
print hilite("Si no ha dado ningún error la etiqueta está lista para ser
programada!",1,1)
print ""
```

Después de ejecutar el script anterior y recibir el mensaje de que la etiqueta es programable, hay que ejecutar el script de escritura incorporándole como único argumento el nombre del usuario al que queremos programarle la nueva etiqueta.

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
#cargamos las librerías que usaremos
import sys
import RPi.GPIO as GPIO
import os
import MFRC522
import random
import signal
import Read
import Write
from random import randint
#si no hay el argumento del usuario, comunica que debe especificarse
if len(sys.argv) != 2:
        print "La forma de ejecutar el comando es:"
        print " "
        print "   python MFRC522-python/write.py $USUARIO"
        print " "
        sys.exit(1)

#asignamos el usuario a la variable USER_SERVER
USER_SERVER=str(sys.argv[1])

#desactivamos los warnings del puerto de comunicaciones de la Raspberry
GPIO.setwarnings(False)

#función que realiza las preguntas de Si y No en la línea de comandos
def confirm(prompt=None, resp=False):
    if prompt is None:
        prompt = 'Confirm'
    if resp:
        prompt = '%s [%s]|%s: ' % (prompt, 'y', 'n')
    else:
        prompt = '%s [%s]|%s: ' % (prompt, 'n', 'y')

    while True:
        ans = raw_input(prompt)
        if not ans:
            return resp
        if ans not in ['y', 'Y', 'n', 'N']:
            print 'please enter y or n.'
            continue
        if ans == 'y' or ans == 'Y':
            return True
        if ans == 'n' or ans == 'N':
            return False
```

```
#función que realiza la escritura de la etiqueta
def setwrite():
	#leemos la id de la etiqueta
	cardid=Read.id()
	#definimos el rango de sectores donde podremos escribir nuestra clave
	range=[4, 5, 6, 8, 9, 10, 12, 13, 14, 16, 17, 18, 20, 21, 22, 24, 25,
26, 28, 29, 30, 32, 33, 34, 36, 37, 38, 40, 41, 42, 44, 45, 46, 48, 49, 50,
52, 53, 54, 56, 57, 58, 60, 61, 62]
	#y separamos los sectores de tráiler de la lista anterior
	trailer=[7,11,15,19,23,27,31,35,39,43,47,51,55,59,63]
	#elegimos un sector al azar junto con su sector de tráiler
	random=random.choice(range)
	index2=(random-4)//4
	index=trailer[index2]
	#lo eliminamos de la lista anterior
	trailer.remove(trailer[index2])
	#para todos los demás bloques escribimos datos aleatorios
	#primero recorremos los tráileres correspondientes
	for cadauno in trailer:
		print cadauno
		#escribimos la key A y key B generadas aleatoriamente para cada
		#bloque. Los permisos de todos los bloques serán: key A y key B
		#pueden leer y escribir en ese sector, key A nunca puede leer la
		#key A aunque sí que puede cambiarla. Y finalmente que solo la
		#key A puede leer y escribir los bits de acceso del tráiler y
		#puede leer y escribir la key B.
		nulltrailer = [randint(0,255), randint(0,255), randint(0,255),
randint(0,255), randint(0,255), randint(0,255),255,7,128,0, randint(0,255),
randint(0,255), randint(0,255),randint(0,255),randint(0,255),randint(0,255)]
		print "password: "+str(nulltrailer[0:6])
		Write.escribe([255,255,255,255,255,255],cadauno,nulltrailer)
		print nulltrailer
		#para cada uno de los tres sectores de ese bloque escribimos
		#datos aleatorios y los escribimos a la etiqueta con nuestra
		#key A generada
		for i in [1,2,3]:
			data=[randint(0,255), randint(0,255), randint(0,255),
randint(0,255), randint(0,255), randint(0,255), randint(0,255),
randint(0,255),randint(0,255),randint(0,255),randint(0,255),randint(0,255),ran
dint(0,255),randint(0,255),randint(0,255),randint(0,255)]
			print cadauno-i
			print nulltrailer[0:6]
			Write.escribe(nulltrailer[0:6],cadauno-i,data)
			print data
		print ""
		print ""
	#creamos el tráiler del bloque en que escribiremos la clave con los
	#mismos permisos que antes y una key A y B aleatorias
	newtrailer=[randint(0,255),randint(0,255),randint(0,255),randint(0,255),
randint(0,255),randint(0,255),255,7,128,0, randint(0,255), randint(0,255),
randint(0,255),randint(0,255),randint(0,255),randint(0,255)]
	#generamos la clave y la grabamos en ese sector
	data=[randint(0,255),randint(0,255),randint(0,255),randint(0,255),randin
t(0,255),randint(0,255),randint(0,255),randint(0,255),randint(0,255),randint(0
,255),randint(0,255),randint(0,255),randint(0,255),randint(0,255),randint(0,25
5),randint(0,255)]
	Write.escribe([255,255,255,255,255,255],index,newtrailer)
	Write.escribe(newtrailer[0:6],random,data)
	#en index tenemos el tráiler del sector donde tenemos la clave
	#en newtrailer tenemos los datos del trailer
	#en random tenemos el número de sector donde tenemos la clave y en data
	#tenemos la clave
	print index
	print newtrailer
	print random
	print data
	print ""
	print ""
```

```
        print ""
        #generamos la cadena con los datos importantes separados por "|"
        #con la id de la tarjeta en primera posición
        shell=cardid+"|"+str(random)+"|"+str(newtrailer[0:6]).strip('[]').replac
e(" ", "")+"|"+str(data).strip('[]')
        print shell
        #la encriptamos con aes 128 bits usando como semilla el nombre de
        #usuario
        out=os.popen("echo '"+shell+"'| openssl enc -aes-128-cbc -a -salt -pass
pass:"+USER_SERVER).read()
        out = out[:-1]
        #imprimimos por pantalla la cadena encriptada
        print out.replace("\n","&&&")
confirm=confirm(prompt='Has ejecutado el comprobador de etiquetas primero?',
resp=False)

#comprobación inicial preguntando si se ha ejecutado el script de test de
#etiqueta que hemos comentado antes y si no se ha realizado lo realiza.
if confirm:
         setwrite()
else:
         import checkblank
```

A grandes rasgos, el script realiza una escritura total de todos los bloques de la etiqueta con una key A y una key B, aleatorias para cada bloque y con unos permisos muy restrictivos. De todos esos bloques, uno se elige al azar y en él, también se generan una key A y una key B aleatoria. En 2 de los 3 sectores de ese bloque, también se escriben datos aleatorios sin importancia, pero en uno de esos sectores se coloca una clave de 16 bytes. Finalmente el script genera una cadena con los datos:

```
[id tarjeta]|[sector clave]|[key A]|[clave 16 bytes]
```

Esta cadena se encripta mediante una encriptación AES de 128 bits, usando como semilla el nombre de usuario que había en el argumento. Así pues, el resultado de este script es una cadena encriptada del tipo

```
U2FsdGVkX1+Fww5yiCMkNgktk1aSnlu5yCdE0VYdiZ07JE1n8UhecQN+Ofk
577tM&&&zJ0JEJZyztEs4PocByKEP+nZeLqZ+4TzuyIZxLpK3XNKA6tIQkr
4Exz7f0u1DGI4&&&Gm/fNVMxJoALr5CfKqPexOc0tfakKSkpBeB/sqMK1Mw
=
```

Esta cadena deberá ser copiada y almacenada, por parte del administrador del sistema en el servidor LDAPS, en el campo *"Departamento"* del usuario al que pertenece esa etiqueta.

# Anexo F

NX es un programa informático que realiza conexiones remotas X en su versión 11 muy rápidas, lo que permite a los usuarios acceder a escritorios remotos de Linux o Unix incluso bajo conexiones lentas como las realizadas con módem. NX realiza una compresión directa del protocolo X, lo que permite una mayor eficiencia respecto a otros programas. La información se envía mediante SSH[112], por lo que toda la información que se intercambian entre el servidor y el cliente está cifrada.

NX está desarrollado por la empresa italiana NoMachine[113], que ha liberado el código.

En otros protocolos de conexión remota, cuando un cliente necesita dibujar algo en pantalla realiza una cantidad de peticiones al servidor, muchas de las cuales necesitan una respuesta. Cada par petición-respuesta se conoce como *roundtrip*. Estos *roundtrips* son los que hacen al sistema más lento debido principalmente al tiempo que se necesita para completar la ida y vuelta. Cuando el cliente y el Servidor X son ejecutados en el mismo host la comunicación se realiza de forma muy rápida, sin embargo cuando cliente y servidor ejecutan en diferentes hosts la comunicación debe realizarse a través de TCP/IP por lo que en este caso la comunicación es mucho más lenta. Otro factor que puede afectar el desempeño es la velocidad de la conexión y latencia de la red. La tecnología NX ofrece mayor eficiencia que otras debido principalmente a las características que se listan a continuación:

### Realiza una eficiente compresión del tráfico X

La compresión del tráfico de forma eficiente es necesaria para lograr ejecutar aplicaciones sobre medios de poco ancho de banda y también para permitir ejecutar múltiples sesiones de usuario en redes LAN.

[112] Es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red.
[113] https://www.nomachine.com/

**Mecanismos de caché para almacenar y reutilizar la información transferida entre cliente y servidor**

NX utiliza un método de caché innovador que divide el mensaje X en dos partes, uno de identificación y otro de datos. La tecnología mantiene en caché únicamente los datos de los últimos mensajes enviados, clasificados por protocolo. A este caché se le conoce como *MessageStore* y hace que el número de peticiones para mostrar los elementos de pantalla disminuya notoriamente.

**El tiempo consumido en realizar *roundtrips* es prácticamente nulo**

La reducción de *roundtrips* es fundamental, al igual que el estricto control del flujo de datos que viaja por la red.

**Utiliza un algoritmo de codificación perezoso para realizar actualizaciones de pantalla**

NX posee mecanismos de adaptación para ajustarse a las propiedades de la red (latencia y velocidad de conexión), lo que permite pasar de métodos estrictos de codificación a métodos perezosos que retrasan la actualización de pantalla cuando la red está congestionada.

# Anexo G

En este anexo, se ejemplifica la forma en que se crea una nueva plantilla KVM y una OpenVZ.

## KVM

En primer lugar, abriremos un navegador y nos dirigiremos a la web de gestión del entrono Proxmox VE, que en nuestro caso es *https://192.168.2.2:8006*. Una vez allí, nos identificaremos con un usuario administrador, y seguidamente haremos un *full clone*, o copiado completo, de la máquina virtual básica de Windows 8.1 (*Windows8NEW*), tal y como muestra la figura 24.

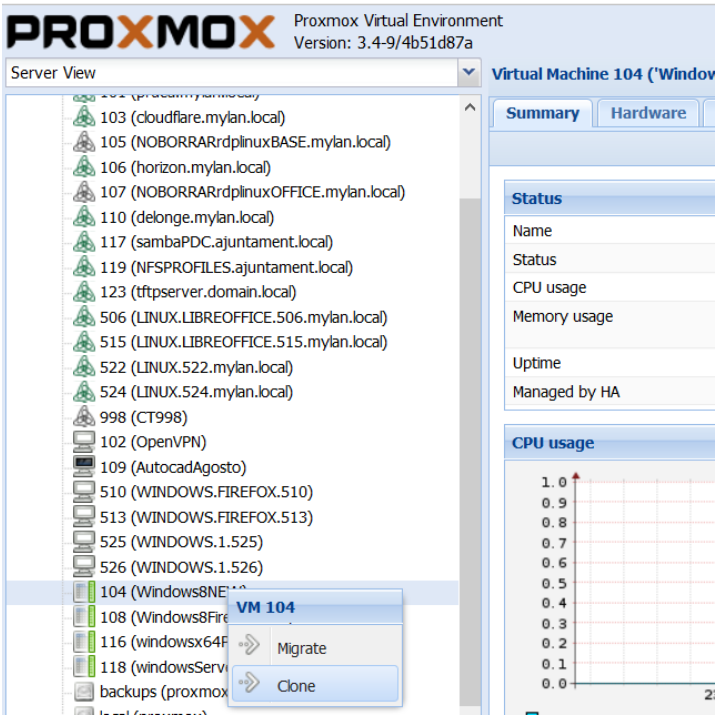



Figura 24 – Clonamos la máquina virtual básica de Windows 8.1

Una vez la hayamos clonado, la iniciamos, y seguidamente la podemos editar desde este mismo entorno, instalando los programas que deseemos. Una vez tengamos la plantilla que deseamos, debemos apagar la máquina y convertirla en *template*, tal y como muestra la figura 25, para que más adelante, la máquina virtual 106 (*Proxmox View*), pueda realizar *linked clones* de ella, con el objetivo de generar los escritorios virtuales.

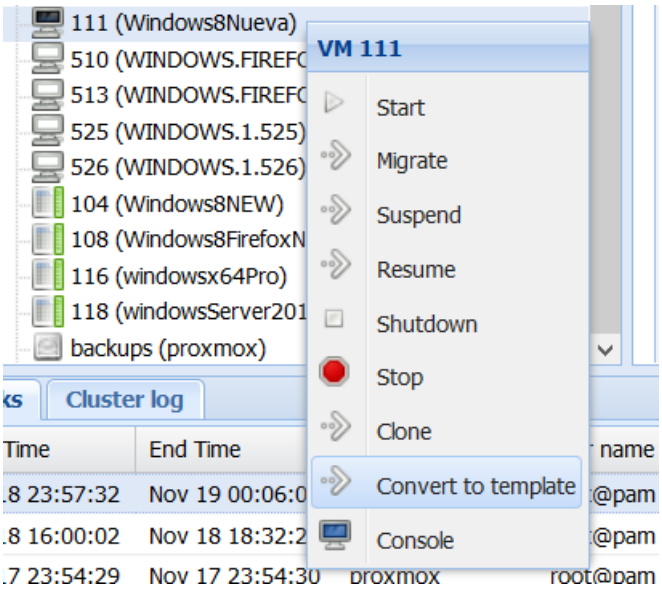



Figura 25 – Convertimos la máquina virtual en *template*

A continuación, accedemos al entorno de Proxmox View, es decir, a *http://view.ajuntament*, y nos identificamos como un usuario administrador. Allí nos dirigimos a la pestaña *Plantillas* y clicamos en *Añadir nueva*, tal y como muestra la figura 26.


Figura 26 – Añadimos una nueva plantilla

En las opciones que aparecen para añadir una nueva plantilla, que vemos en la figura 27, debemos asignar el número de máquinas virtuales mínimas, el de máquinas virtuales reservadas y el máximo número de máquinas que queremos que haya en el entorno. Es muy importante poner en *Archivo OpenVZ/Número Qemu*, el número de la

máquina virtual que hemos convertido en *template* en el paso anterior, en este caso, 111. También es importante que guardemos el nombre que le ponemos, porque nos será útil a continuación. Para continuar, pulsamos en *Crear.* En este momento, los escritorios virtuales de esa plantilla, empezaran a generarse; y en unos minutos, estarán todos listos para ser asignados a los usuarios que lo soliciten.

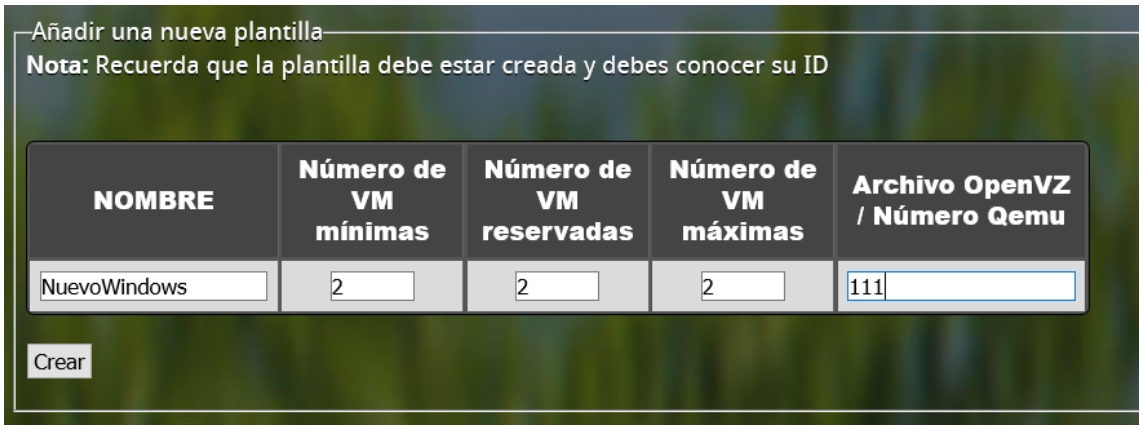



Figura 27 – Configuramos las opciones de la nueva plantilla

Para finalizar el proceso de creación de esta nueva plantilla, lo único que nos queda, es asignar qué usuarios podrán usar esta plantilla. Para realizar esta configuración, nos dirigimos a la plataforma web LAM[114], que se explica al detalle, en el anexo I.

[114] https://www.ldap-account-manager.org/lamcms/

En ella nos identificamos como administrador y accedemos a la pestaña *Groups*. En esta pestaña, pulsamos sobre *New group*, tal y como muestra la figura 28.

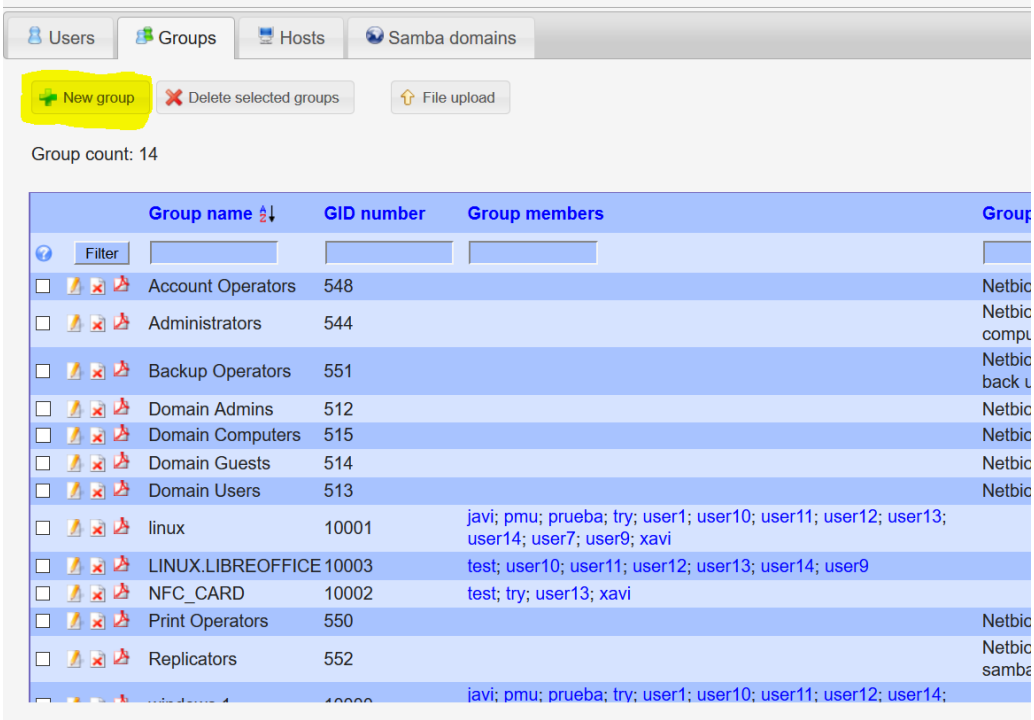



Figura 28 – Creamos un nuevo grupo

Lo único que debemos hacer, es poner como *Group name* el nombre que hemos comentado en el paso anterior que era importante guardar, y a continuación pulsamos *Save*, tal y como muestra la figura 29.

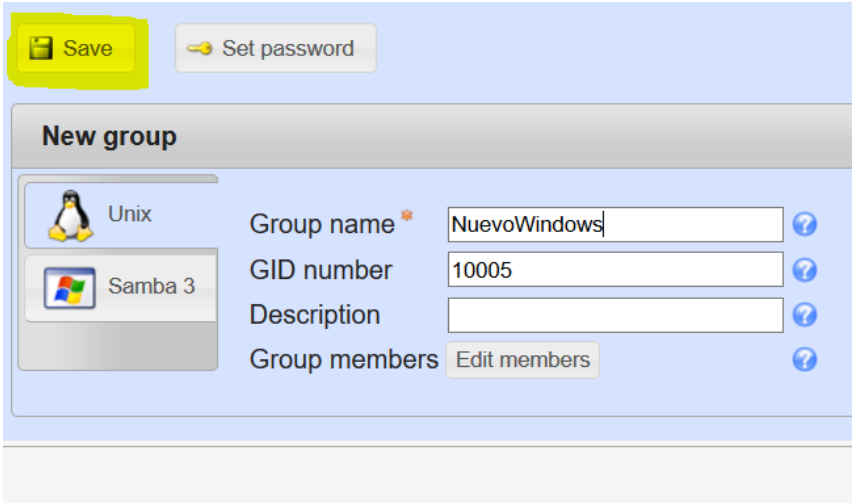



Figura 29 – Configuramos el grupo

Una vez creado el grupo, lo único que hay que hacer es asignar ese grupo a los usuarios que queramos que puedan usar esa plantilla de escritorio virtual. También se puede realizar esa asignación en *Edit members* del paso anterior, tal y como se observa en la figura 29.

### OpenVZ

El proceso de creación de una nueva plantilla OpenVZ es ligeramente distinto al de la KVM, aunque hay algunos puntos que son idénticos.

Para empezar, accederemos al entorno de Proxmox VE e iniciaremos la máquina base de Linux, tal y como se observa en la figura 30.

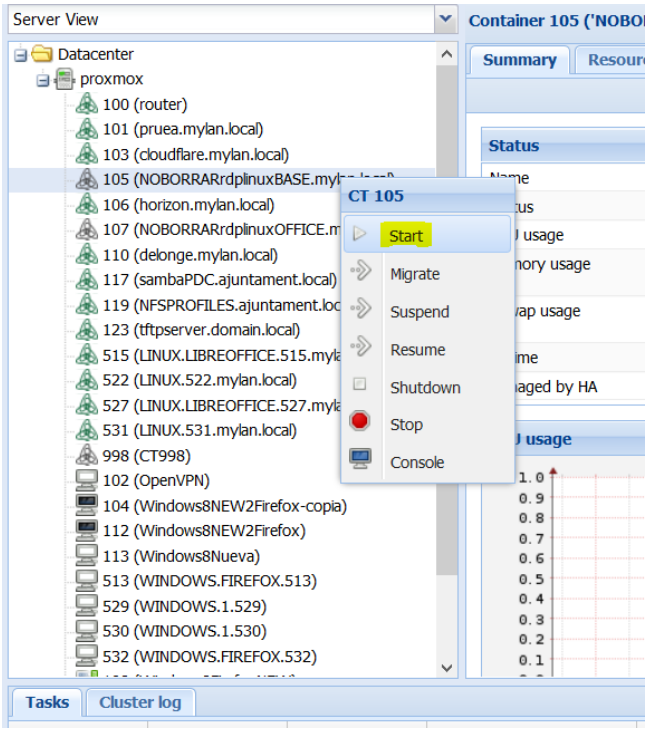



Figura 30 – Iniciamos la máquina base de Linux

A continuación, realizamos los cambios que queremos e instalamos las aplicaciones que necesitamos que tenga esa plantilla. Cuando estos cambios han sido realizados, apagamos la máquina, nos situamos en la pestaña *Backup* de esta máquina virtual e iniciamos una copia de seguridad, tal y como se muestra en la figura 31.

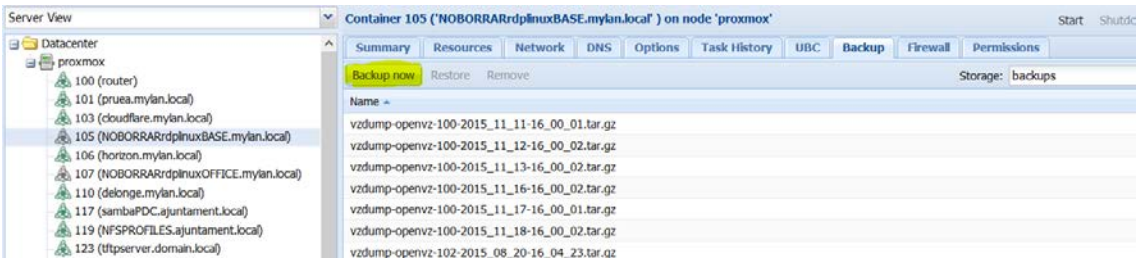



Figura 31 – Realizamos un *Backup* de la máquina

Seguidamente, en la pantalla que nos aparece que se está realizando el *Backup*, copiamos la ruta que aparece en la figura 32 y la guardamos ya que la necesitaremos en el siguiente paso.

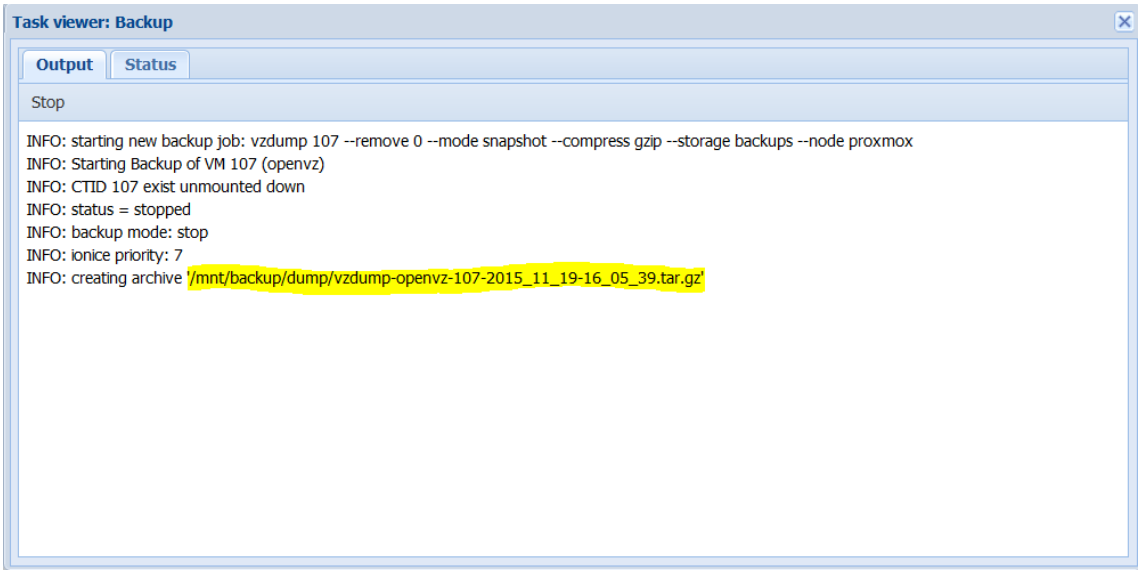



Figura 32 – Ruta a copiar después de realizar el *Backup*

Ahora debemos acceder al entorno Proxmox View, en el que crearemos la nueva plantilla. De la misma forma que con KVM, nos identificamos y accedemos a la pestaña *Plantillas,* a continuación, clicamos en *Añadir nueva.*

En esta sección configuramos el número de máquinas virtuales mínimas, el de máquinas virtuales reservadas y el máximo número de máquinas que queremos que haya en el entorno. De la misma forma que en las plantillas KVM, los dos campos más importantes son

*Archivo OpenVZ/Número Qemu* y el nombre de la plantilla. En *Archivo OpenVZ/Número Qemu* debemos introducir la ruta que hemos copiado en el punto anterior y en nombre, el nombre del grupo que crearemos para asignar los usuarios, que queremos que puedan acceder a este escritorio. Esta configuración se puede observar en la figura 33.

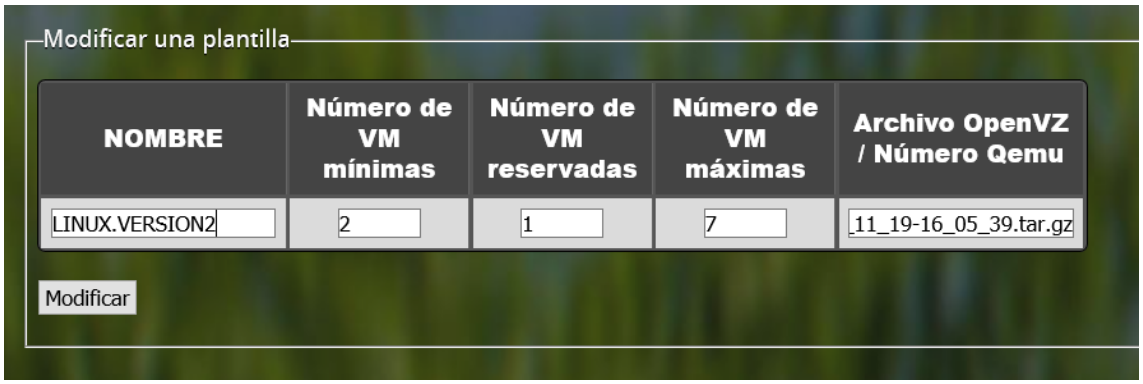



Figura 33 – Configuración de la nueva plantilla

Finalmente, debemos realizar el mismo último paso que en el caso de las plantillas KVM. Debemos acceder a la plataforma web LAM y crear un grupo, con el mismo nombre que hemos escrito en el paso anterior. Para terminar, tenemos que asignar, a este grupo, los usuarios que queramos que puedan usar esta plantilla.

# Anexo H

Ventajas que aporta el sistema de archivos ZFS:

### Almacenamiento en grupo

ZFS se basa en el concepto de grupos de almacenamiento para administrar el almacenamiento físico.

Desde siempre, los sistemas de archivos se estructuraban a partir de un único dispositivo físico. Para poder ocuparse de varios dispositivos y ofrecer redundancia de datos, incorporaban el concepto del administrador de volúmenes, con el fin de ofrecer una representación de un único dispositivo y evitar que los sistemas de archivos tuvieran que modificarse para aprovechar las ventajas de varios dispositivos. Este diseño significaba otro nivel de complejidad y obstaculizaba determinados avances en los sistemas de archivos, al carecer de control sobre la ubicación física de los datos en los volúmenes virtualizados.

ZFS elimina del todo la administración de volúmenes. En vez de tener que crear volúmenes virtualizados, ZFS agrega dispositivos a una agrupación de almacenamiento. La agrupación de almacenamiento describe las características físicas del almacenamiento (organización del dispositivo, redundancia de datos, etc…) y actúa como almacén de datos arbitrario en el que se pueden crear sistemas de archivos. Los sistemas de archivos se limitan a dispositivos individuales y eso les permite compartir espacio en el disco con todos los sistemas de archivos de la agrupación. Ya no es necesario predeterminar el tamaño de un sistema de archivos, ya que el tamaño de los sistemas de archivos crece automáticamente en el espacio asignado a la agrupación de almacenamiento. Al incorporar un nuevo almacenamiento, todos los sistemas de archivos de la agrupación pueden usar de inmediato el espacio en el disco adicional sin procesos complementarios.

### Semántica transaccional

ZFS es un sistema de archivos transaccional. Esto significa que el estado del sistema de archivos siempre es coherente en el disco. Los sistemas de archivos tradicionales sobrescriben datos *in situ*. Esto significa que, si el equipo se queda sin alimentación (por ejemplo, entre el momento en que un bloque de datos se asigna y cuando se vincula a un directorio), el sistema de archivos se queda en un estado

incoherente. En el pasado, este problema se solucionaba mediante el comando *fsck*. Este comando verificaba el estado del sistema de archivos e intentaba reparar cualquier incoherencia durante el proceso. Este problema de sistemas de archivos incoherentes daba muchos quebraderos de cabeza a los administradores, y el comando *fsck*, nunca garantizaba la solución a todos los problemas. Posteriormente, los sistemas de archivos han incorporado el concepto de registro de diario. El registro de diario guarda las acciones en un diario aparte, el cual se puede volver a reproducir con seguridad si el sistema se bloquea. Este proceso supone cargas innecesarias, porque los datos se deben escribir dos veces y a menudo provoca una nueva fuente de problemas. Con un sistema de archivos transaccional, los datos nunca se sobrescriben y ninguna secuencia de operaciones se compromete ni se ignora por completo. Este mecanismo hace que el sistema de archivos nunca pueda dañarse, por una interrupción imprevista de la alimentación o un bloqueo del sistema. Aunque pueden perderse fragmentos de datos escritos más recientemente, el propio sistema de archivos siempre será coherente.

**Datos de reparación automática y sumas de comprobación**

En ZFS se verifican todos los datos y metadatos mediante un algoritmo de suma de comprobación seleccionable por el usuario. Los sistemas de archivos tradicionales con suma de comprobación la efectúan por bloques, debido a la capa de administración de volúmenes y a la disposición del sistema de archivos tradicional. El diseño tradicional significa que algunos errores, como la escritura de un bloque completo en una ubicación incorrecta, pueden hacer que los datos no sean correctos, pero no producen errores de suma de comprobación. Las sumas de comprobación de ZFS se almacenan de forma que estos errores se detecten y haya una recuperación eficaz. La suma de comprobación y la recuperación de datos se efectúan en la capa del sistema de archivos y resultan transparentes para las aplicaciones. Así mismo, ZFS ofrece soluciones para la reparación automática de datos admitiendo agrupaciones de almacenamiento con diversos niveles de redundancia de datos. Si se detecta un bloque de datos incorrecto, ZFS recupera los datos correctos de otra copia redundante y repara los datos incorrectos al sustituirlos por una copia correcta.

### Escalabilidad incomparable

Un elemento de diseño clave en el sistema de archivos ZFS es la escalabilidad. El sistema de archivos es de 128 bits y permite 256 trillones de Zettabytes[115] (ZB) de almacenamiento. Todos los metadatos se asignan de forma dinámica, con lo que no hace falta asignar previamente nodos ni limitar la escalabilidad del sistema de archivos cuando se crea. Todos los algoritmos se han escrito teniendo en cuenta la escalabilidad. Los directorios pueden tener hasta $2^{48}$ (256 billones) de entradas; no existe un límite para el número de sistemas de archivos o de archivos que puede haber en un sistema de archivos.

### Instantáneas de ZFS

Una instantánea es una copia de sólo lectura de un sistema de archivos o volumen. Las instantáneas se crean rápida y fácilmente. Inicialmente, las instantáneas no consumen espacio adicional en el disco dentro de la agrupación. Como los datos de un conjunto de datos activo cambian, la instantánea consume espacio en el disco al seguir haciendo referencia a los datos antiguos. Como resultado, la instantánea impide que los datos pasen al grupo.

### Administración simplificada

Uno de los aspectos más destacados de ZFS es su modelo de administración muy simplificado. Mediante un sistema de archivos con distribución jerárquica, herencia de propiedades y administración automática de puntos de montaje y, semántica share de NFS, el ZFS facilita la creación y gestión de sistemas de archivos sin tener que usar varios comandos ni editar archivos de configuración. Con un solo comando puede establecer fácilmente cuotas o reservas, activar o desactivar la compresión, o administrar puntos de montaje para diversos sistemas de archivos. Puede examinar o sustituir dispositivos sin aprender un conjunto independiente de comandos de administrador de volúmenes. Puede enviar y recibir flujos de instantáneas del sistema de archivos. Además, administra los sistemas de archivos a través de una jerarquía que permite la administración simplificada de propiedades como cuotas, reservas, compresión y puntos de montaje.

[115] 1 Zettabyte equivale a 1000 TB

# Anexo I

A continuación se explica detalladamente el proceso que hay que realizar para añadir un nuevo usuario al entorno. Para hacerlo se usará la plataforma web LAM, también conocida como *LDAP Account Manager*, y que es una plataforma que facilita la gestión de los usuarios de un directorio LDAP.

En primer lugar, abrimos el navegador web desde uno de los ordenadores del sistema, nos dirigimos a la dirección *http://samba.ajuntament/lam* y nos identificamos con un usuario administrador.

A continuación hacemos clic en *New User*, tal y como aparece en la figura 34.

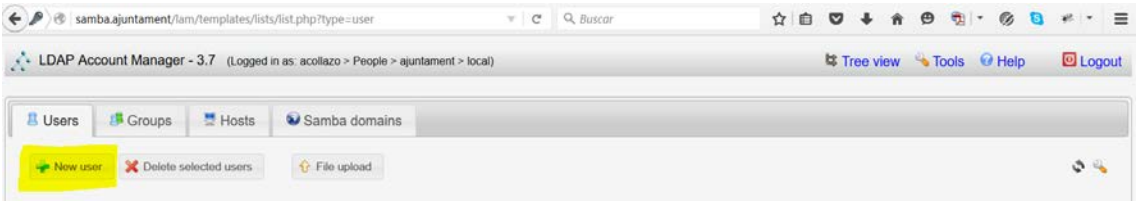



Figura 34 – Botón para crear nuevo usuario

A continuación, tal y como se observa en la figura 35, rellenamos todos los campos que queramos, los únicos que son obligatorios son: *Last name* y en el caso de que queramos que el usuario requiera de la doble autentificación mediante NFC, el campo *Department* donde pondremos la cadena que nos ha proporcionado el script de grabación de la tarjeta NFC.

Es muy importante pulsar en *Set password*, y establecer una contraseña para ese usuario, antes de proseguir.

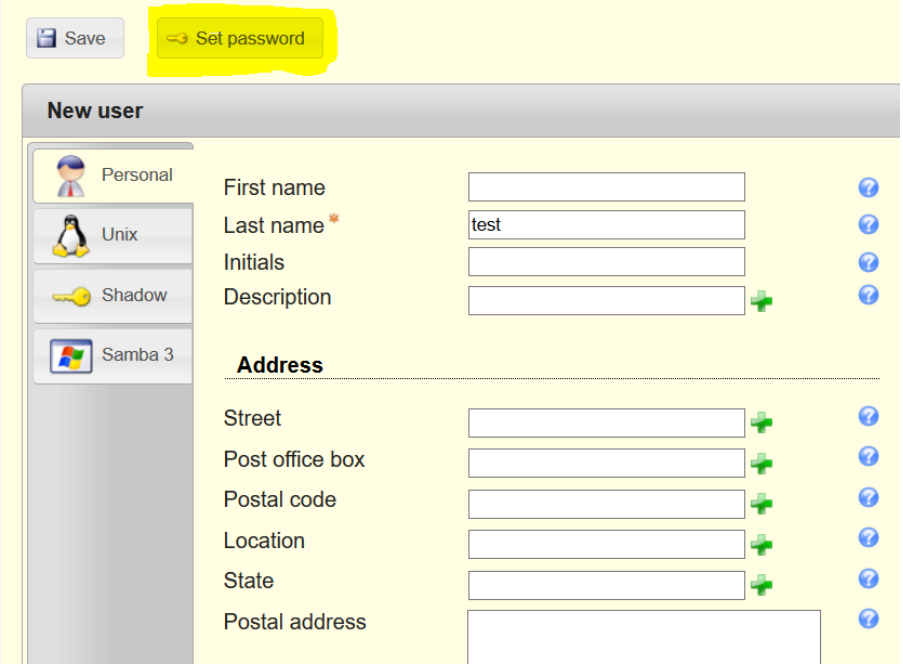



Figura 35 – Campos obligatorios y establecer contraseña

Seguidamente, haremos clic en el apartado *Unix* de la columna de la izquierda. En este apartado, tal y como se observa en la figura 36, no introduciremos nada ya que se nos autocompletará. Lo único que deberemos hacer, es asignar los grupos, y lo haremos haciendo clic en *Edit groups.*

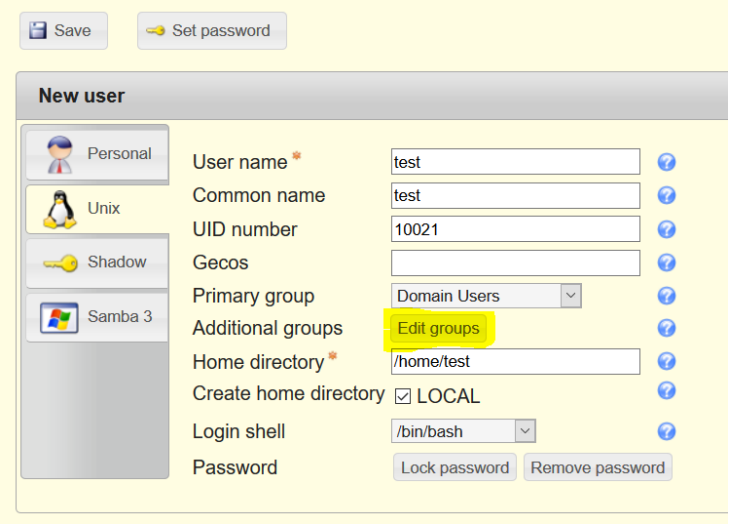



Figura 36 – Menú *Unix* y botón *Edit groups*

En este menú podemos elegir los escritorios a los que podrá acceder el usuario, así como asignarlo al grupo *NFC_CARD* en el que se le obligará a hacer la autentificación en dos pasos. En la figura 37 se puede observar el menú de un usuario que tendrá la autentificación en dos pasos y además tendrá disponibles las plantillas: Linux con Libre Office y Windows con Firefox.

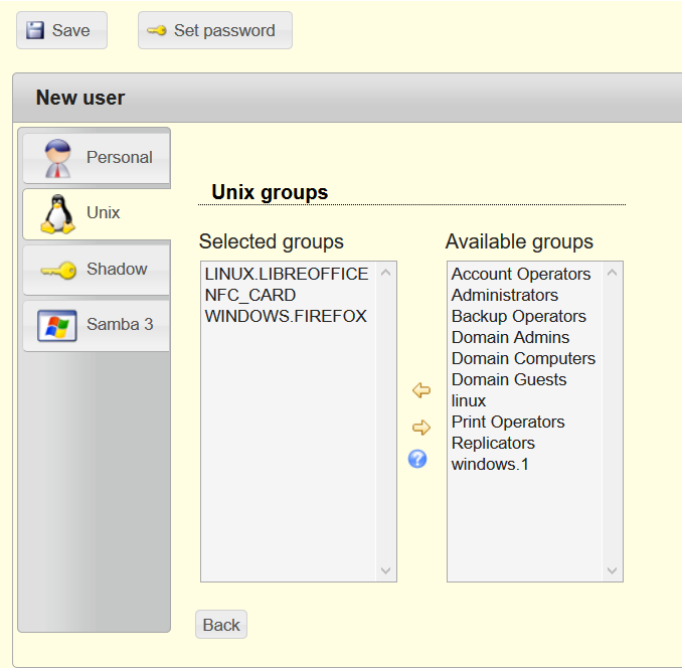



Figura 37 – Selección de grupos del usuario

Finalmente, hacemos clic en el apartado *Samba 3*, en el que tampoco se tiene que completar nada, ya que lo hemos configurado de forma que los campos se autocompleten y que así sea más fácil la creación del nuevo usuario. Para terminar, hacemos clic en *Save,* tal y como se observa en la figura 38. De esta forma, nuestro usuario ya estará disponible en todo nuestro entorno.

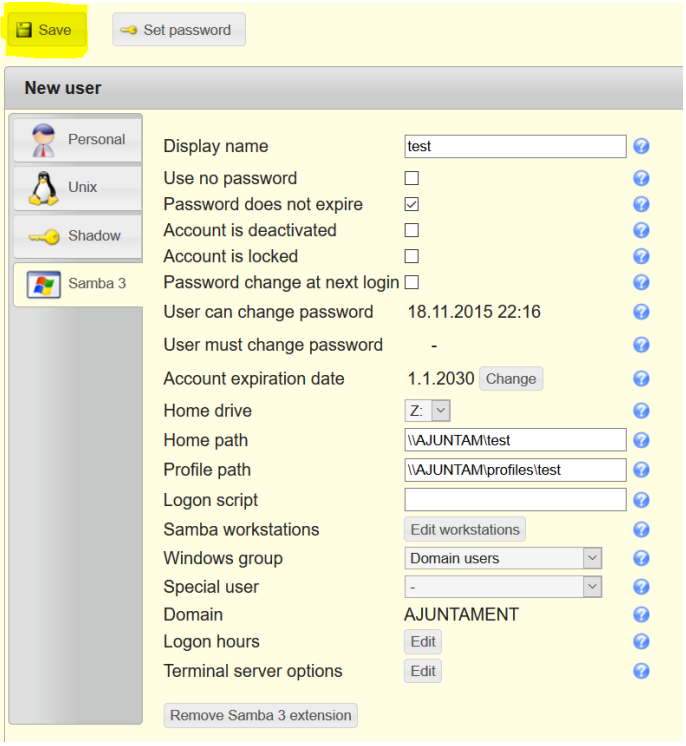



Figura 38 – Sección *Samba 3* de LAM

# Bibliografía

http://blog.capacityacademy.com/2012/08/07/que-es-la-virtualizacion-y-cuales-son-sus-beneficios/
https://es.wikipedia.org/wiki/Virtualizaci%C3%B3n
http://wiki.x2go.org/doku.php/doc:usage:x2goclient
http://www.molinum.com/proxmox-cluster-ha.html
https://fedoraproject.org/wiki/Virtualization/es
https://en.wikipedia.org/wiki/Comparison_of_remote_desktop_software
https://es.answers.yahoo.com/question/index?qid=20110622124838AAADEC8
https://phenobarbital.wordpress.com/2012/03/04/1621/
https://openvz.org/Setting_up_an_iptables_firewall
https://pve.proxmox.com/wiki/Windows_8_guest_best_practices
http://miguelcarmona.com/notas-y-recortes/habilitar-iptables-para-los-containers-openvz-en-proxmox
http://raspberrypi.stackexchange.com/questions/28090/how-to-u-boot-for-arch-linux-over-nas
http://elinux.org/RPi_U-Boot
http://elinux.org/RPi_Advanced_Setup
http://www.raspberrypi.org/forums/viewtopic.php?f=66&t=56914
https://github.com/raspberrypi/firmware/tree/master/boot
http://lynxline.com/lab-3-r-pi-booting-process/
http://elinux.org/RPi_Advanced_Setup#Formatting_the_SD_card_via_the_mkcard.txt_script
http://openthinclient.org/de/download-openthinclient/
https://www.howtoforge.com/pxe_booting_debian
http://wiki.centos.org/es/HowTos/PXE/PXE_Setup
http://wiki.centos.org/es/HowTos/PXE/PXE_Setup/Menus
http://www.cyberciti.biz/faq/install-configure-tftp-server-ubuntu-debian-howto/
http://www.slicesofcomputer.com/2014/10/systemd-hate-console-autologin-on-jessie.html
http://www.kaibader.de/homemade-minimal-raspberry-pi-raspbian-image/
http://www.richardsramblings.com/2013/02/minimalist-raspberry-pi-server-image/
http://www.opentechguides.com/how-to/article/raspberry-pi/5/raspberry-pi-auto-start.html
http://blogs.wcode.org/2013/09/howto-boot-your-raspberry-pi-into-a-fullscreen-browser-kiosk/
http://www.micronetinternational.com/clients/knowledgebase.php?action=displayarticle&id=12
https://www.youtube.com/watch?v=TsM6fAYmSTk
https://github.com/FreeRDP/FreeRDP
https://github.com/FreeRDP/FreeRDP/wiki/Compilation
https://github.com/FreeRDP/FreeRDP/wiki/RemoteApp
https://jeffskinnerbox.wordpress.com/2012/11/15/getting-audio-out-working-on-the-raspberry-pi/
https://github.com/FreeRDP/FreeRDP/wiki/Compilation
https://github.com/neutrinolabs/xrdp/wiki/xrdp-hardware-encoder-project
https://github.com/neutrinolabs/xrdp/wiki/Compiling-rdp-driver
http://javierin.com/wp-content/uploads/2011/03/UniversalTermsrvPatch_20090425.zip
https://wiki.debian.org/es/Xfce
https://github.com/FreeRDP/FreeRDP/issues/1540
http://myatus.com/p/x-server-with-sound-inside-an-openvz-proxmox-container/
https://openvz.org/X_inside_VE
http://stackoverflow.com/questions/27637465/record-local-audio-in-a-docker-container
http://askubuntu.com/questions/71863/how-to-change-pulseaudio-sink-with-pacmd-set-default-sink-during-playback
http://askubuntu.com/questions/28176/how-do-i-run-pulseaudio-in-a-headless-server-installation
http://xmodulo.com/x2go-remote-desktop-linux.html
https://vpsboard.com/topic/4241-running-a-lightweight-gui-on-your-vps-via-x2go/
https://billing.nodeserv.com/knowledgebase.php?action=displayarticle&id=1
https://github.com/neutrinolabs/xrdp/wiki/Building-on-Debian-6
http://blog.pi3g.com/2013/10/freerdp-for-directfb-on-the-raspberry-pi-package-make-your-rdp-session-fly/
https://wiki.debian.org/LDAP/OpenLDAPSetup

http://backports.debian.org/Instructions/
https://packages.debian.org/search?keywords=samba&searchon=names&suite=all&section=all
http://wiki.x2go.org/doku.php/wiki:development:deb-buildguide#build_client_components_on_raspberry_pi_raspbian_wheezy
http://www.slicesofcomputer.com/2014/10/systemd-hate-console-autologin-on-jessie.html
http://wealsodocookies.com/posts/openbox-a-windows-environment-for-hackers/
http://mindref.blogspot.com.es/2010/12/debian-openldap-ssl-tls-encryption.html
http://how-to.linuxcareer.com/using-openssl-to-encrypt-messages-and-files
http://www.cnx-software.com/2012/07/31/84-mb-minimal-raspbian-armhf-image-for-raspberry-pi/
http://blog.qruizelabs.com/2014/04/23/stripping-down-a-standard-raspbian-installation/
http://blog.pi3g.com/2013/07/high-performance-rdp-on-the-raspberry-pi/
http://fuenteabierta.teubi.co/2013/07/utilizando-el-lector-nfc-rc522-en-la.html
https://dangerousthings.com/wp-content/uploads/2012/08/NFC-Access-Control-for-Mifare-S50.pdf
http://www.securityartwork.es/2010/01/29/hacking-rfid-rompiendo-la-seguridad-de-mifare-i/
http://www.nxp.com/documents/data_sheet/MF1S50YYX.pdf
https://geekytheory.com/tutorial-raspberry-pi-graficar-la-temperatura-de-la-cpu-con-node-js-y-highcharts/
https://www.samba.org/samba/docs/man/Samba-Guide/happy.html
http://blog.pi3g.com/2014/04/make-raspbian-system-read-only/
http://www.doc.ic.ac.uk/~mgv98/MIFARE_files/report.pdf
https://www.firefart.at/how-to-crack-mifare-classic-cards/