Proyecto Fin de Carrera

Prototipo de Laboratorio Informático Virtual con Thinclients basados en Raspberry Pi

Para acceder al Título de

INGENIERO EN INFORMÁTICA

Autor: Mario Blanco Casado

Director: Pablo Abad Fidalgo

Septiembre - 2021

Resumen

La meta de este proyecto ha sido explorar la posibilidad de implementar un prototipo de laboratorio virtual, usable en docencia y fundamentada en herramientas OpenSource. La arquitectura de dicho prototipo estaba basada en la utilización de un servidor centralizado, que diera acceso a los puestos de laboratorio, implementados en base a diferentes máquinas virtuales y atendiendo a los requisitos de cada asignatura. El acceso a dichas máquinas se establece a través de una red privada, que conecta a múltiples equipos clientes, que a su vez son el puesto físico donde trabajan los alumnos.

Con el propósito de optimizar costes en la implementación del laboratorio, se han hecho pruebas de cliente tanto con PCs convencionales, como con *Thinclients* basados en Raspberry Pi. La toma de decisiones sobre las tecnologías usadas ha sido algo constante a lo largo del proyecto.

Con el objetivo de ocultar al alumnado todo este proceso remoto, se ha diseñado una pequeña aplicación para el cliente, que se despliega cuando inicia y proporciona acceso directo a las máquinas virtuales de las asignaturas.

Finalmente, se ha conseguido cerrar este proyecto con un prototipo operativo y consistente, en un equipo servidor con múltiples máquinas virtuales con diferentes sistemas operativos (tanto Windows como Linux) al que se conectan dos equipos cliente, siendo uno de ellos un *Thinclient* de tipo Raspberri Pi. Esta implementación demuestra la viabilidad de desarrollar un laboratorio virtual sencillo, con herramientas OpenSource y abre el camino a futuras implementaciones más reales, con un servidor mejor dimensionado y múltiples clientes.

Abstract

The goal of this project has been exploring the possibilities of implementing a prototype of a virtual laboratory usable in education based in OpenSouce tools. The architecture of this prototype was based in the use of a centralized server that provide access to each laboratory spot, implemented around different virtual machines attending the needs of each course. The access to these virtual machines is implemented through a private network that connects to multiple clients, that are the laboratory spots that the students work on.

With the objective of minimizing costs in the implementation of the laboratory, there has been client tests with normal PCs and with Thinclients based on Raspberry Pi. The decision making about the technologies that are being used in this project has been constant.

Finally, with the objective of hiding to the student that uses this system all the remote implementation, it has been design one application to the client that opens when the client boots and provides a direct access to the virtual machines of each subject (one icon for each one).

It has been accomplished closing this project with a prototype that is operative and consistent, in a server with multiple virtual machines with different operative systems (Windows and Linux) that the clients connect to, being one of them a Thinclient of a Raspberry Pi. This implementation shows the viability of implementing a simple virtual laboratory with OpenSource tools and opens the path of future and more real implementations, with a server that has the capacity of holding multiple virtual machines and clients.

ÍNDICE

1.- Introducción

La virtualización es un mecanismo de compartición de recursos cuyo objetivo principal es optimizar la utilización de los mismos. Gran parte de las tecnologías de virtualización son relativamente recientes, pero esta búsqueda de eficiencia se remonta décadas atrás, cuando los computadores eran *mainframes* solamente al alcance de grandes compañías. En este tipo de computadores, de un coste muy elevado, la gestión de la máquina como un recurso compartido era fundamental, con el objetivo de maximizar la utilización y optimizar los costes. Gracias a esta filosofía, los costes asociados a la disponibilidad de recursos de computación cayeron progresivamente [1], haciendo posible para organizaciones más pequeñas o incluso a personas de manera individual, el acceso a un computador sin la necesidad de adquirirlo.

Este proceso guarda muchas semejanzas con la forma en la que se saca partido a la virtualización en la actualidad. Las capacidades de cálculo de un servidor han alcanzado una cota extremadamente elevada, haciendo imposible para casi cualquier carga de trabajo (aplicación, programa) hacer un uso efectivo de todos los recursos hardware disponibles. La mejor forma de optimizar la utilización de recursos y al mismo tiempo, simplificar las tareas de administración de un servidor, es claramente la virtualización. Cualquier Centro de Procesado de Dados (CPD) actual hace uso te diferentes técnicas de virtualización, capaces de agrupar diferentes recursos hardware (CPU, memoria, disco, red, etc.) y ofrecerlos a cada usuario de manera exclusiva, en la forma de máquinas virtuales.

Todas las tecnologías de *Cloud Computing* se basan en herramientas de virtualización. Las herramientas de virtualización se encargan de proporcionar, a través de la red, los servicios apropiados al usuario final. Dependiendo del tipo de recurso que se ofrece, se puede hablar de diferentes tipos de virtualización. En la virtualización del tipo Infraestructura como Servicio (IaaS por sus siglas en inglés), se virtualizan los recursos hardware, de manera similar a la virtualización de servidor mencionada en el párrafo anterior. Es posible ir un paso más allá, con la virtualización de tipo Plataforma como Servicio (PaaS), donde además del IaaS se proporciona un entorno de desarrollo software completo. De manera alternativa, es también posible limitar la virtualización a una sola aplicación, ofreciendo solo una instancia de la misma y los datos asociados (Software como servicio o SaaS).

Es evidente que la virtualización, bien en entornos locales o en entornos Cloud, es una tecnología muy relevante en la actualidad. En este TFG hemos trabajado en los aspectos básicos de virtualización, como un primer paso hacia la consecución de un laboratorio docente totalmente virtualizado. El modelo empleado, conocido como Virtualización de escritorio (VDI) parece el más apropiado para esta tarea. En el resto del capítulo se justificará de manera adecuada la utilización de una infraestructura VDI, se describirá en detalle el prototipo de laboratorio a emplear y se enumerarán los pasos para su implementación.

1.1.- Motivación

Una de las vertientes de la virtualización es el VDI (Virtual Desktop Interface), que podría ser traducido como virtualización de escritorio. El VDI, una tecnología con menos de 20 años de vida [2], consiste en la virtualización y ejecución de múltiples entornos de escritorio (Sistema Operativo + Gestor Ventanas) en un servidor centralizado, accediendo a ellos a través de clientes que utilizan protocolos de display remotos (RDP [3] o PCoIP [4] son dos ejemplos).

Las ventajas asociadas a la virtualización de escritorio son múltiples, siendo algunas de las más relevantes las siguientes:

- Permite al usuario trabajar como si lo hiciera en su puesto de trabajo físico, con acceso a la red local de la compañía desde cualquier lugar y en cualquier momento, con el único requisito de tener una conexión a internet.
- Hace posible un mantenimiento centralizado de los puestos de trabajo. Las labores de actualización (tanto de sistema operativo como de aplicaciones) se pueden llevar a cabo con facilidad, sin necesidad de desplazarse. La implementación instantánea de nuevos escritorios y uso de aplicaciones es trivial, porque al solo necesitar conectarse remotamente al servidor donde se aloje el escritorio al que se quiera acceder, no se necesita nada más y esto hace de la implementación algo muy sencillo y fácil de mantener.
- Con un dimensionado adecuado se maximiza la utilización de recursos, minimizando los costes adicionales por mantener puestos de trabajo físicos inactivos.
- La delegación de los clientes en equipos de tipo *ThinClient* supone un ahorro energético importante, puesto que este tipo de equipos presentan un consumo mucho más reducido que un PC convencional. Este sistema es más ecológico y respetuoso con el medio ambiente, ya que consumen alrededor de un tercio de la energía que los ordenadores con un sistema tradicional [5].
- El escalado en recursos es más económico, pues únicamente el backend requerirá actualizaciones de hardware más costosas. También suma el hecho de que prácticamente hay tiempo cero de inactividad en caso de fallos hardware del cliente, pues su reemplazo es inmediato y el servidor mantiene la información relevante.
- Se trata de un entorno seguro, en el que el administrador tiene control total sobre lo que cada usuario puede hacer en todo momento y previene personalizaciones del puesto de trabajo no autorizadas.

Es evidente que gran parte de las ventajas asociadas a los entornos VDI son extremadamente positivas en una implementación de laboratorio docente. El acceso remoto permite a los alumnos trabajar con las herramientas de las asignaturas en horarios distintos a los de clase, bien para completar las prácticas inacabadas o bien para labores de estudio. Con al menos un aula informática por facultad y un campus distribuido geográficamente (varias sedes en Santander y una en Torrelavega), la

administración centralizada de los laboratorios supone una gran ventaja. Finalmente, tanto las ventajas de tipo económico como de seguridad hacen este tipo de estructuras todavía más atractivas si cabe.

Todas las ventajas descritas dan sentido al proyecto realizado. La Universidad de Cantabria ha dado un primer paso en el acceso remoto a las aulas de laboratorio con el proyecto *Unican Labs* [6], pero en el momento en el que se plantea este proyecto carece de infraestructuras de laboratorio centralizadas y virtualizadas con una arquitectura cliente-servidor como la que se describe en el proyecto actual. En este documento se propone una estructura básica de laboratorio utilizando software OpenSource y basando la implementación de los clientes en equipos de bajo coste, con el objetivo de optimizar tanto el coste como la eficiencia energética.

1.2.- Objetivo

El objetivo marcado para este TFG es la puesta en marcha de manera experimental de un laboratorio de docencia con equipos informáticos basado en tecnología VDI (*Virtual Desktop Infrastructure*). En dicho prototipo se contempla una estructura como la de la Figura 1.

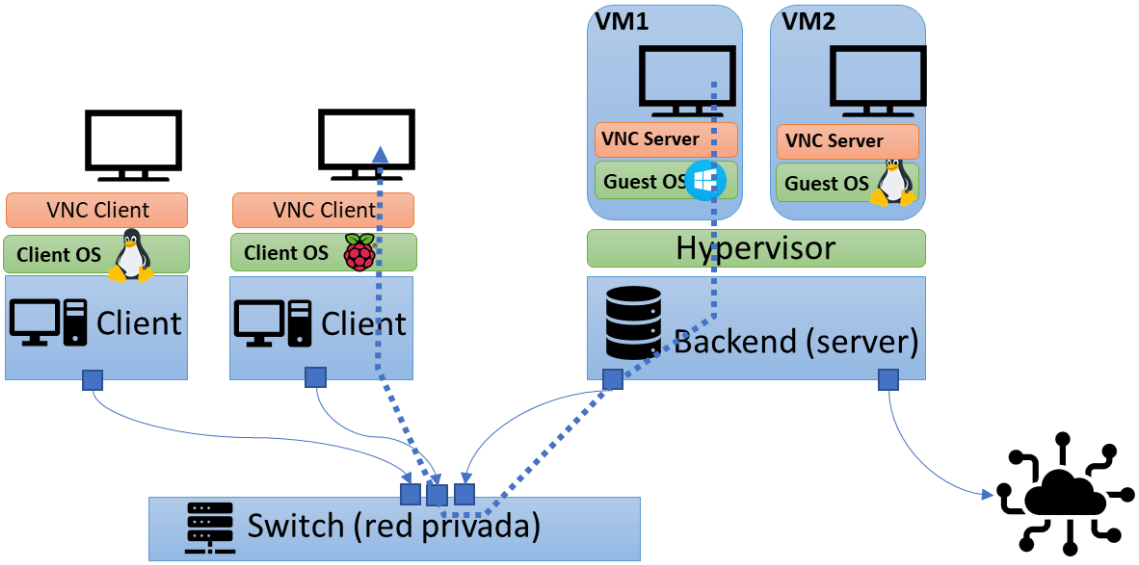

Figura 1. Esquema de implementación de infraestructura del proyecto.

La arquitectura de Laboratorio se compone de un Servidor que actuará de *backend*, sobre el que se instalará un hipervisor en el que se ejecutarán las máquinas virtuales, y múltiples clientes desde los que se configurará el acceso de escritorio remoto a las máquinas virtuales que se ejecutan en el servidor. Como clientes, probaremos dos equipos distintos, un PC "normal" y una Raspberry Pi. Desde el PC normal se hará la primera prueba intentando acceder a la máquina virtual y posteriormente, desde la Raspberry Pi se creará un programa que se ejecute al iniciar el propio ordenador. Este programa te dejara elegir entre las dos máquinas virtuales existentes para poder acceder a ellas.

Esta primera implementación busca ser escalable ya que solo se está probando con un par de clientes y máquinas virtuales. A futuro se intentará que esta implementación crezca para hacerla usable en docencia.

1.3.- Fases de Desarrollo

Los contenidos trabajados en este TFG han supuesto un complemento a la formación recibida durante el Grado. Con una orientación clara hacia la Administración de Sistemas, ha sido necesario un proceso continuo de aprendizaje de aspectos relacionados con la Virtualización.

Como se ha mencionado en el resumen el objetivo final de este proyecto ha sido implementar un prototipo de laboratorio usable. Esto ha llevado a distintas fases de desarrollo que se introducen en esta sección y se detallan a lo largo de los capítulos de este documento. Desde un principio se tenía bastante claro el objetivo final, aunque en el camino ha sido necesario adoptar algunos cambios para hacer frente a los problemas e incompatibilidades que han surgido.

Los primeros pasos del proyecto tuvieron como objetivo la búsqueda de la herramienta de virtualización más adecuada. Los condicionantes principales eran los siguientes:

- Hipervisor basado en Linux
- Software OpenSource
- Que fuera gratuito
- Que fuera fácilmente escalable
- Que fuera compatible con muchos sistemas operativos
- Que tuviera soporte para Thin Client

Con todos estos puntos de vista, se optó por la plataforma de virtualización XCP-ng, implementada a través del hipervisor Xen y una API sobre la que se implementan herramientas de gestión y *backup*. Estas herramientas se describen con detalle en el Capítulo 2.

Una vez escogida la herramienta de virtualización, el siguiente paso ha consistido en la instalación y configuración del servidor que actuará de *backend*, así como la creación de las máquinas virtuales que harán de entorno para diferentes asignaturas. El capítulo 3 describe este proceso con detalle.

El siguiente paso, correspondiente al Capítulo 4, se centra en la implementación de los equipos cliente. En este paso nos centraremos en describir la aplicación gráfica desarrollada y que hace de interfaz para acceder de manera transparente a los equipos de laboratorio (máquinas reales remotas).

El documento termina con un capítulo dedicado a enumerar las principales conclusiones del trabajo llevado a cabo y a marcar las posibles líneas de trabajo futuras para convertir el prototipo en una implementación funcional completa.

2.- Conceptos y herramientas utilizadas

Con los principales objetivos del prototipo ya definidos, se dedica este capítulo a describir de manera breve tanto las tecnologías que ha sido necesario aprender durante el proceso de desarrollo y despliegue como las herramientas utilizadas en dicho proceso. Introduciremos el concepto de virtualización y sus aspectos básicos, dedicando un apartado a describir un tipo concreto, la virtualización de escritorio. Se explicará Xen como hipervisor y definiremos también qué son equipos de tipo *Thinclient* y *Zeroclient*, fundamentalmente las placas Raspberry Pi. Finalmente, hablaremos de manera breve sobre el IDE para desarrollo de aplicaciones gráficas para Linux utilizado.

2.1.- Virtualización de Servidor

Hay muchos tipos de virtualización. Desde la virtualización de software, pasando por la de hardware sin olvidar otras como de almacenamiento, datos o redes. Para el desarrollo de este TFG nos centraremos en la virtualización hardware, más concretamente en la virtualización de escritorio.

El objetivo principal de la virtualización de hardware es poner a distintos componentes del hardware a disposición de un software independientemente del soporte físico que estén utilizando. El componente básico en la implementación de esta tecnología son las máquinas virtuales. Una máquina virtual es una aplicación que simula un sistema de computación, pudiendo ejecutar su propio sistema operativo y programas como si fueran una computadora real, de manera transparente al usuario (los usuarios finales no distinguen entre la máquina virtual y el ordenador físico). Una propiedad importante de las máquinas virtuales es que los procesos que se ejecutan en ellas están limitados por recursos y abstracciones proporcionados por las propias máquinas virtuales. En esta tecnología, las maquinas se alojan en equipos llamados anfitriones (también denominados host en este documento) y la capa de abstracción que se crea entre host y cliente (máquina virtual) se llama hipervisor.

La virtualización de servidor proporciona importantes ventajas, entre las cuales merece la pena mencionar el aumento en la tasa de ocupación hardware del servidor, gracias a una distribución más efectiva de los recursos hardware, así como una mayor eficiencia energética tanto para su funcionamiento como para su refrigeración. La seguridad es otra ventaja importante, pues si alguien se infiltra en una de las máquinas virtuales que se alojan en ellas, el resto no se ven comprometidas, al ser unidades independientes. En este tipo de virtualización, se pueden destacar dos principales subclases desde el punto de vista técnico, que se describen en las siguientes secciones y cuya estructura básica se muestra en la Figura 2.

2.1.1.- Virtualización completa

En esta clase de virtualización cada máquina virtual simula un entorno completo de hardware, lo que le hace disponer de su propio conjunto de recursos hardware asignados por el hipervisor. Sobre este hipervisor, se ejecutan las aplicaciones, lo que hace que el sistema alojado no tenga acceso al hardware físico del sistema *host* (anfitrión). Dicho de otra forma, en este formato se tiene un servidor con un sistema

operativo instalado, y sobre este sistema operativo se ejecuta el hipervisor. Lo que se busca es que el hipervisor pase a ser un proceso del sistema operativo del *host*, lo que hace que las máquinas virtuales pasen a un nivel exterior. Los hipervisores de esta clase se definen de tipo 2 también llamados *hosted* (hospedados). Hay varias tecnologías que utilizan este sistema, entre ellas: Oracle VM VirtualBox [7], VMware Workstation [8] y Microsoft Virtual Server [9] entre otras.

2.1.2.- Paravirtualización

La principal diferencia entre la virtualización completa y la para virtualización, es que mientras la primera dispone del entorno completo, la segunda solo se ofrece una API con la que se pueda acceder al hardware físico del anfitrión. El requisito que existe para que se pueda hacer paravirtualización, es que la API integre el kernel del sistema operativo huésped. Lo que se persigue con esto, es que solo se puedan paravirtualizar los sistemas que son huésped modificado. En este tipo el hipervisor descansa sobre el hardware de la máquina, que tiene acceso a todos los recursos del sistema y poder dar la funcionalidad que quiera. Sobre este hipervisor se ejecutan las máquinas y los usuarios que quieran acceder a estas máquinas lo tienen que hacer también a través del hipervisor. Los hipervisores de esta clase se definen como tipo 1 como se puede ver en la figura 2 también llamados *bare-metal* (metal desnudo), nativos o *unhosted*, y serán los que se utilicen en el desarrollo de este TFG. Algunos de los hipervisores que permiten la paravirtualización son: Xen [10] y Oracle VM Server [11].

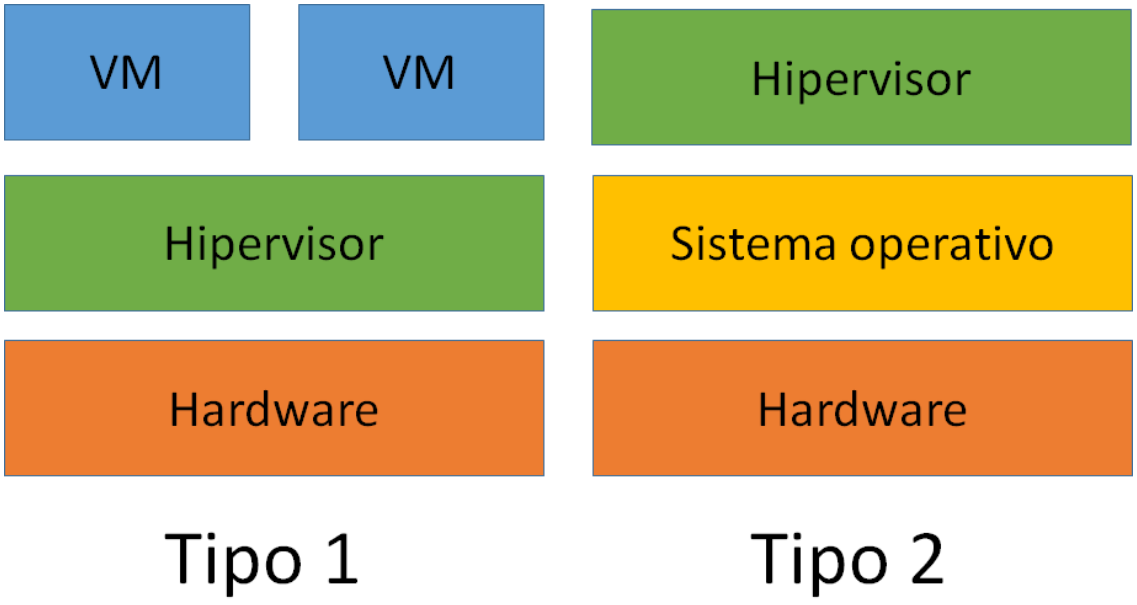

Figura 2. Esquema de tipos de hipervisores.

2.2.- Xen

Xen [10] es un hipervisor de código abierto de tipo 1. Fue creado en la Universidad de Cambridge y se centra en la virtualización avanzada para uso comercial. Tiene múltiples usos, tanto en el campo de la virtualización como servicio, como para seguridad de

aplicaciones, sistemas embebidos, etc. También sirve para manejar tanto software como hardware y soporta tanto paravirtualización como virtualización completa. Xen corre directamente sobre el hardware de la maquina en la que se hospeda, sin necesidad de un sistema operativo.

Hay muchos motivos por los que se ha elegido Xen sobre otros competidores, pero principalmente ha sido porque es una herramienta open source además de la eficiencia y la escalabilidad que tiene. Como este proyecto es un prototipo, la escalabilidad es clave de cara al futuro del proyecto, ya que se plantea ampliar.

Xen funciona mediante la abstracción de los elementos hardware de una maquina física y su reproducción en una máquina virtual. Recordemos que una máquina virtual se puede definir como un contenedor donde se crea un espacio de virtualización en una ventana que se ejecuta en tu hipervisor, en este caso Xen. Esta VM se comporta como un dispositivo separado del propio ordenador y además permite incluso ejecutar una instalación de otro sistema operativo independiente con todas sus características.

Xen maneja dichas máquinas virtuales a través de una estructura denominada dominio. Hay un dominio principal o DOM0 que maneja el resto de los dominios o DOMUs y el sistema en general. Los DOMUs son dominios sin privilegios y sin acceso al hardware que manejan a la máquina virtual que contienen. Esto se refleja en la arquitectura que se puede observar en la figura 3. Esta arquitectura es muy clara y refleja perfectamente la estructura de Xen, un servidor físico con un hardware sobre el que se ejecuta el hipervisor y sobre él, todos los dominios que se ejecutan, tanto el DOM0 como los DOMUs.

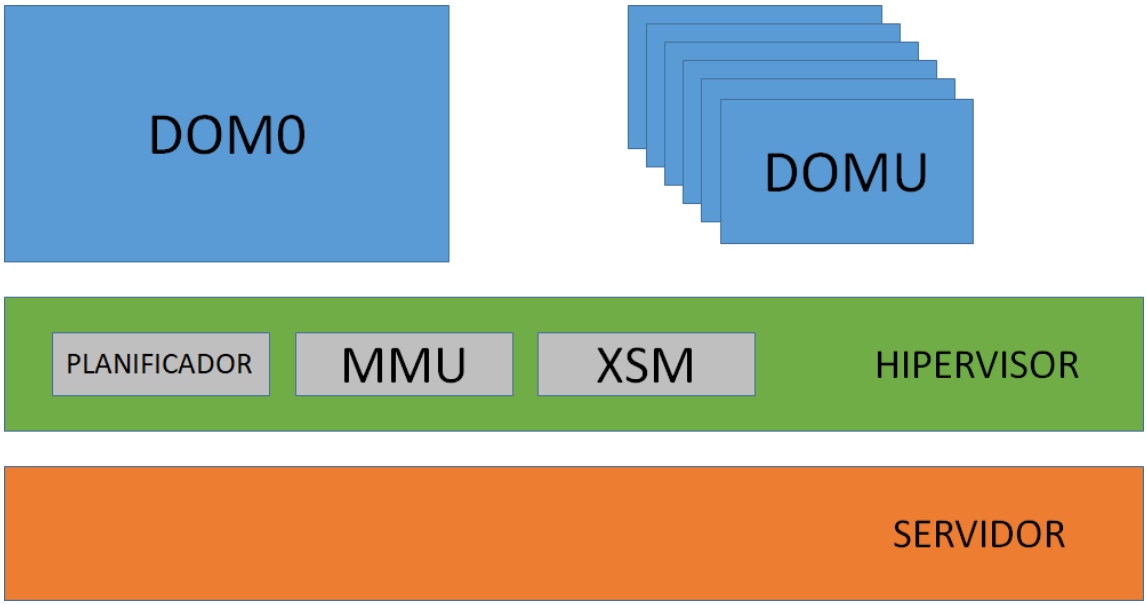

Figura 3. Esquema de arquitectura de Xen.

Como también se puede observar en la Figura 3, el hipervisor cuenta con varios módulos funcionales que administran las distintas partes de la gestión de máquinas virtuales. El funcionamiento básico de estos módulos se describe a continuación:

- MMU: Una de estas partes es la unidad de gestión de memoria implementada a través de LVM que es un sistema que permite crear, eliminar y modificar bloques de memoria sin afectar ni al sistema ni a otros bloques de memoria.
- XSM: o Xen Security Modules como dice la propia expresión se trata de un conjunto de módulos diferentes que proveen de seguridad al sistema. Este modelo permite controlar como interactúan los diferentes dominios entre ellos y proveerlos de reglas, esto da al sistema una capacidad muy extensa y minuciosa de manejar su seguridad. Entre las cosas más generales que se pueden hacer con este modelo están: crear y ejecutar una política de privacidad, etiquetar dominios con las diferentes políticas creadas y utilizar una aplicación simple que tenga permisos basados en estas políticas.
- Planificador: el trabajo que tiene un hipervisor en Xen, es decidir de las diferentes CPUs de cada máquina virtual, cual corre en que CPU del *hardware* físico de la maquina anfitriona. Xen tiene distintos planificadores como Credit[12] , Credit2[13], RTDS[14], etc.. que se pueden ejecutar a la vez en diferentes *pools* de máquinas virtuales para diferentes propósitos.

Una parte importante en la gestión de máquinas virtuales cuando su acceso será remoto es la gestión de las redes. Xen proporciona diferentes configuraciones de red, dependiendo de nuestros requisitos de trabajo. La conexión entre los interfaces de red virtual de nuestros dominios y los interfaces físicos de nuestro servidor host se gestionan mediante la creación de un puente o *bridge* que permite conectar estas VM a la red. Las configuraciones de red disponibles para las máquinas virtuales son las siguientes:

- *Single-server private network*: implementa una red privada que solamente se puede comunicar con las máquinas virtuales del mismo servidor.
- *Cross-server private network*: implementa una red privada al igual que *single-server private network*, pero esta si permite conectarse entre diferentes servidores. Esta funcionalidad permite que dos máquinas virtuales que no estén en un mismo servidor puedan comunicarse.
- *External network*: se utiliza para conectarse con una red exterior a nuestro sistema.
- *Bonded network*: se implementa conectando dos adaptadores de red en una misma red proveyendo redundancia y balanceo de carga.

2.2.1.- XCP-ng

Una vez elegido Xen como hipervisor que se va a utilizar y conociendo sus pros y sus contras, es momento de elegir la distribución de Xen con la que vamos a trabajar. La idea principal es continuar con el objetivo de utilizar exclusivamente software de código totalmente abierto. Con esta premisa, la distribución elegida fue XCP-ng.

En términos de capacidad, XCP-ng puede soportar 16 máquinas virtuales por cada *pool*, que es un grupo de máquinas virtuales o servidores que pueden ser de hasta un máximo de 64 unidades. Estos pools pueden ser bastante diferentes, pero tienen que cumplir una serie de requisitos de compatibilidad de hardware como tener el mismo proveedor de CPUs y tener compatibilidad con virtualización.

También tiene control dinámico de memoria que actúa automáticamente en las máquinas virtuales que se están ejecutando para garantizar el rendimiento y optimiza el sistema para una mayor capacidad. Otra de las ventajas de XCP-ng es que permite solventar problemas y actualizar el sistema sin necesidad de desconectar el sistema, esto permite que la gente pueda seguir utilizando el sistema, aunque este en mantenimiento.

XCP-ng tiene un balanceo de carga dinámico que es cuando se distribuye de manera más eficiente la carga computacional, para que unas CPUs no realicen mucha carga de trabajo mientras el resto hace poco o nada. Es más sencillo que otras implementaciones de Xen y sirve para que el manejo de las redes sea más eficiente permitiendo *multicast*. Esto hace que se generen menos problemas a la hora de escalar el sistema al dar más protección con alta disponibilidad.

Una vez descritas las herramientas a través de las cuales se crearán y gestionarán las máquinas virtuales de laboratorio, la siguiente sección describe las peculiaridades de dichas máquinas, pues deben proporcionar entornos de escritorio completos.

2.3.- Desktop as a service, VDI

A través de las herramientas de virtualización conseguimos entornos de trabajo (Sistema Operativo + Aplicaciones) independientes sobre un hardware común. Es habitual que el acceso a dichos entornos se haga de manera remota, y aquí es donde entra la tecnología VDI. VDI o *virtual desktop infrastructure*, es una tecnología relativamente reciente que implementa una solución para la separación entre el escritorio y la maquina física. En esta solución tecnológica el entorno de escritorio virtualizado es distribuido a través de la red a diferentes dispositivos cliente, permitiendo la interacción con el Sistema Operativo y las aplicaciones como si se ejecutaran en el equipo local.

El origen de la tecnología VDI se sitúa a principios de la década de los 2000, cuando los usuarios del software de virtualización VMware ejecutaban sus entornos de escritorio sobre un servidor remoto (virtualizado con VMware y ESX) y accedían a dicho entornos a través de conexiones punto a punto (sobre el protocolo RDP). Fue de hecho VMware la primera empresa en formalizar el término VDI (*Virtual Desktop Interface*) para designar este tipo de tecnología, creando el "*VDI Alliance Program*" en 2006. En estos primeros años, dos compañías ofrecían soluciones VDI, VMware a través de su *Virtual Desktop Manager* y Citrix con XenDesktop. La popularización de los proveedores c*loud* amplió rápidamente el número de soluciones VDI, popularizando el término "*Desktop as a Service*" para definir un VDI en el que las máquinas virtuales se alojan en servidores de tipo *cloud*. Actualmente existen multitud de soluciones VDI disponibles, desde las versiones más recientes de los productos comerciales de Citrix o VMware, hasta

implementaciones basadas en su totalidad en software *opensource* como la que usaremos en este TFG.

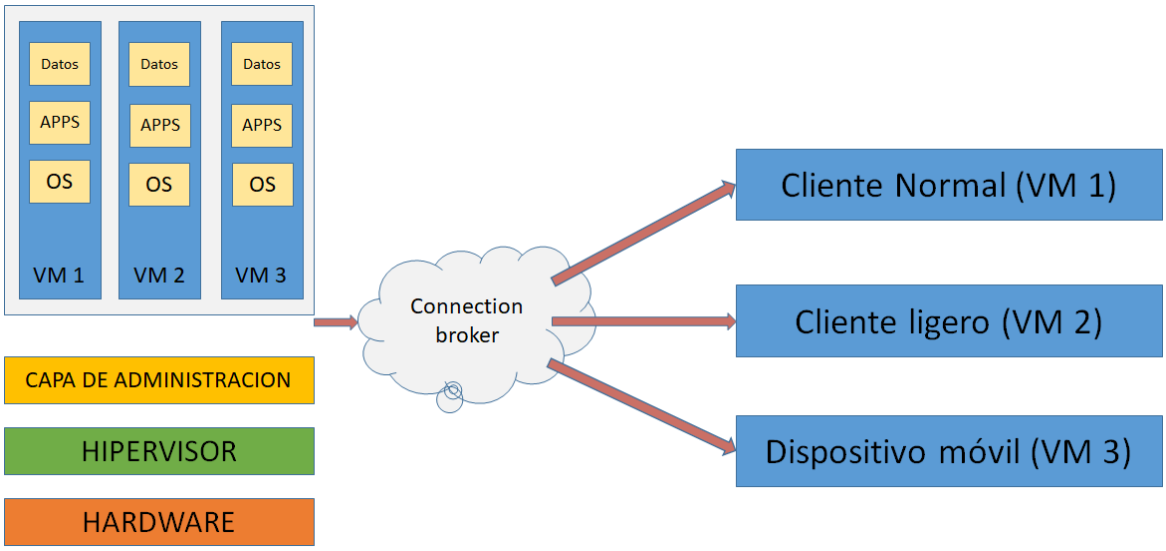

Figura 4. Esquema de implementación de infraestructura de laboratorio VDI.

Todos los entornos VDI se basan en una arquitectura similar, formada por tres componentes principales que se muestran en la figura 4: *backend* virtualizado, *connection broker* y protocolo de comunicación.

El *backend* virtualizado es el servidor que alojará las máquinas virtuales que van a poner los entornos de escritorio a disposición de los usuarios. La configuración del *backend* debe responder a decisiones de diseño relacionadas con la interacción entre el usuario final y su entorno de escritorio. En primer lugar, es posible asignar un entorno de escritorio exclusivo (máquina virtual) por cada cliente (configuración 1:1) o múltiples escritorios virtuales se pueden ejecutar sobre el mismo sistema operativo, con lo que varios usuarios compartirían la misma máquina virtual (configuración 1:N). De manera similar, las máquinas virtuales que se despliegan pueden ser persistentes o no. La diferencia principal entre ambas implementaciones reside en la capacidad de guardar datos locales (a la máquina virtual) o instalar aplicaciones de manera permanente. Para el prototipo inicial de *backend* de docencia trabajaremos con una configuración 1:1 persistente, pues parece el entorno más apropiado para el desarrollo de una asignatura.

El *connection broker* es un administrador de recursos que maneja las conexiones que existen entre el servidor que proporciona el servicio y los distintos clientes que acceden a él. En términos de administración, el *connection broker* está a cargo de aspectos como el aprovisionamiento de escritorios, asignación de roles, apagado y encendido programado de máquinas, etc. En nuestro prototipo se trabaja con un *broker* muy sencillo, pues existe una máquina virtual por asignatura y una configuración de *backend* de tipo 1:1, minimizando las labores de gestión del *broker*.

Por último, está el protocolo de comunicación, que es el conjunto de reglas que se utilizan para regir una conexión entre dos o más dispositivos a través de una red. Este tipo de protocolos utilizan técnicas de compresión de datos para optimizar la experiencia de usuario y reducir el consumo de ancho de banda. Cada plataforma un protocolo de *display* remoto propio. Citrix implementa un protocolo denominado *Independent Computing Architecture*. Por su parte, las sesiones remotas de VMware se pueden basar en múltiples protocolos, como *Blast Extreme*, PCoIP o RDP. En nuestro caso se utilizará RDP que es un protocolo desarrollado por Microsoft que permite la conexión entre un cliente y un servidor.

2.4.- ThinClients and ZeroClients

Existen distintos tipos de máquinas virtuales dependiendo de la cantidad de carga computacional que el servidor hace con respecto a los clientes. Esta diferencia hace que la parte importante de la carga computacional, al poderse ejecutar en diferentes partes, tenga una importancia mayúscula a la hora de decidir cosas tan importantes como el balanceo de carga. Aparte de que, a más carga computacional, se necesita un hardware más potente por lo que dependiendo de cómo sean las características de nuestro sistema se puede implementar de una manera u otra dando mucha flexibilidad al sistema.

Todo esto incluso se puede implementar de una manera hibrida dando más capacidad de cómputo a las unidades del sistema que se sepa que tengan un hardware más potente. Esto lleva a diferentes tipos de implementaciones que se explican a continuación.

- *Thick clients*: en este tipo de clientes también llamados pesados, la mayor carga computacional se localiza en el cliente. A este cliente se le provee de una imagen del sistema operativo al que va a acceder y todo el computo se ejecuta en el propio cliente dejando solo al servidor con la tarea de proveer, guardar y dar seguridad a los datos.
- *Thin clients*: también llamados clientes ligeros, en este tipo de clientes la mayor carga computacional se ejecuta en el servidor. Aunque el servidor es quien procesa principalmente todos los datos, no son todos ya que el cliente también tiene la capacidad de hacerlo, pero de manera reducida. El objetivo de estos clientes es reducir la carga computacional generalmente porque si ejecutaran en el propio cliente toda la carga, la virtualización sería muy lenta. Estos clientes se enfocan en crear un canal de comunicación donde comunicarse con el servidor y donde envían datos para procesar y los recogen ya procesados.
- *Zero clients*: en este tipo de clientes, el servidor ejecuta toda la computación de datos, sin dejar nada al cliente, y siendo este un mero espejo de lo que se computa lejos de él. Esta tecnología se utiliza cuando los clientes tienen un hardware muy pobre y tener que ejecutar cualquier tipo de cómputo generaría un cuello de botella, al igual que con los *thin clients*. La diferencia es que estos clientes utilizan toda su capacidad para crear el canal de comunicación con el

servidor donde enviar y recibir datos. Este cliente es simplemente un espejo de lo que se ejecuta en el servidor.

Los clientes principales que vamos a utilizar son los clientes ligeros, porque el ordenador que queremos que acceda al sistema para virtualizar es la Raspberry Pi. Estos ordenadores se crearon para que personas con poco acceso a la informática, pudieran acceder a un sistema de manera barata. Este sistema se creó teniendo como objetivo primero la enseñanza de informática en las escuelas, lo cual fue un punto importante para añadirlo a este proyecto.

Este sistema tiene poca capacidad de cómputo con cuatro procesadores de poco más de 1GHz y 1GB de memoria, por lo que crear un cliente pesado podría causar una virtualización lenta. Además, al querer hacer el sistema más grande, lo que se busca es crecer desde la parte del servidor, más concretamente, del hardware que lo soporta, porque es la manera más eficiente de hacerlo.

El modelo utilizado para la virtualización ha sido la Raspberry PI 3 B v1.2 que es un sistema con Quad Core 1.2GHz Broadcom BCM2837 de 64 bits, 1GB de RAM, 4 puertos USB [15].

2.5.- Herramientas de cliente

Desde el principio se buscó tener la implementación más ligera posible en el cliente dado que la idea era ampliarlo a más máquinas y este proceso se tendría que hacer en todas las maquinas que quieran acceder al sistema. Por lo tanto, en cada cliente solamente se tienen que instalar dos herramientas.

La primera es la más notoria que es Remmina. Este es un cliente para protocolos de escritorio remoto que te permite establecer una conexión remota con un servidor o con cualquier otro tipo de sistema que esté conectado a una red. Este programa permite acceder por medio de diferentes protocolos. El que utiliza Remmina es RDP (*Remote Desktop Protocol*) que permite la conexión entre un usuario con interfaz gráfica y otro ordenador. El segundo es SFTP (*Secure File Transfer Protocol*) que permite transferir de manera cifrada datos desde tu ordenador con conexión a la dirección que desees a través de SSH (*Secure Shell*).

La segunda herramienta es Java. El programa que se ha hecho para este proyecto utiliza Java, por lo tanto, es necesario instalar la última versión de este lenguaje para poder ejecutar el programa.

2.6.- IDE programa

Como ya se explicó el programa que se ejecuta al iniciar nuestra Raspberry Pi es una parte fundamental del proyecto. En esta parte se utiliza una interfaz gráfica que crea un par de botones, esta interfaz es la de *Windows Builder*.

Se busco una interfaz gráfica compatible con Java ya que era el lenguaje de programación que íbamos a utilizar. *Windows Builder* proporciona una sencilla manera de implementar un programa simple como el que queríamos hacer.

3.- Implementación del Backend.

En este apartado se describirá el proceso de implementación del *backend* con todas las decisiones de diseño tomadas, los procesos seguidos y los problemas que se han ido encontrando a lo largo de este proceso.

Se empezará explicando la instalación de Xen, seguidamente se explicará la configuración que se ha implementado, después se continuará con la creación de las máquinas virtuales y su configuración interna y finalmente se terminará con la configuración de la red.

3.1.- Instalación y configuración de Xen

Una vez explicado en el capítulo anterior el funcionamiento de Xen y justificado la elección de la distribución de XCP-ng es momento de instalar todo el sistema.

La instalación de XCP-ng está bien documentada. Se descarga a través de una imagen y se instala en el sistema. La forma que se ha optado ha sido a través de un USB. Se utilizo RUFUS para añadir esta imagen al USB. La instalación sigue el mismo proceso que un sistema operativo normal, con un programa de arranque. En el momento que se reinicia el ordenador, se entra en la BIOS para cambiar la partición de entrada y hacer que se ejecute en el USB. Se vuelve a reiniciar y se empieza a configurar.

El proceso de instalación es convencional, en él se gestionan los aspectos básicos de la configuración del sistema como el particionado del disco para administrar el espacio disponible para las máquinas virtuales.

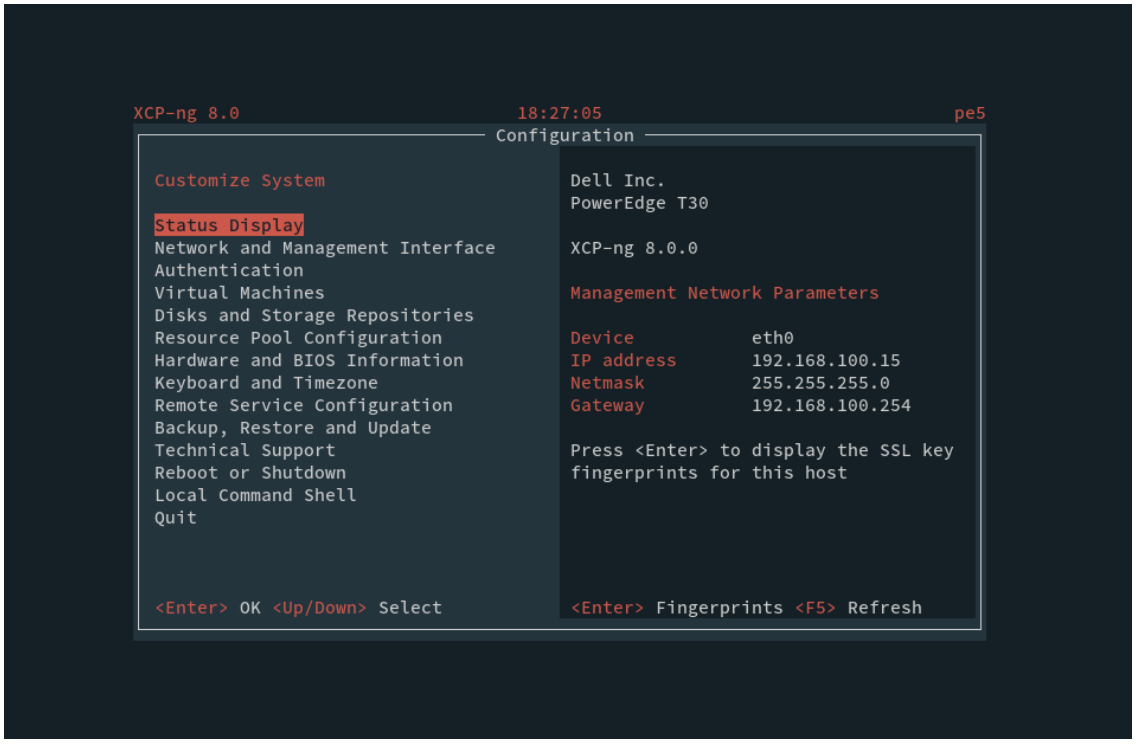

Figura 5. Interfaz de servidor de XCP-ng

En el momento que se inicia el sistema se accede a la interfaz que es la que se muestra en la figura 5. Desde aquí se pueden ver las distintas opciones que tiene el sistema.

Desde esta interfaz, se puede manejar todo el sistema. Desde aquí se pueden ver las distintas máquinas virtuales que tienes, manejar las redes y en general ver toda la información que quieras del sistema. Pero no todo se puede manejar desde esta interfaz ya que tiene opciones limitadas. Para poder administrar al completo este servidor, hay que acceder a la *Local Command Shell* (Linea de comandos) ya que es donde se tiene acceso al todo el sistema. A lo largo de este proyecto me ha sido útil sobre todo el apartado de *Virtual Machines* porque en él se podían administrar las máquinas virtuales del sistema y ver su estado, lo cual ha sido de gran ayuda para solucionar problemas y *Network and Manager interfaces* para crear y administrar de manera más sencilla todas las redes ya que han sido uno de los principales escollos de este proyecto.

Además de poder configurar el sistema desde esta interfaz, también hay programas que te ayudan a manejar XCP-ng desde otro ordenador con una interfaz más fácil que la del propio servidor. Principalmente se han utilizado 2, openxenmanager para Linux y XCP-ng center para Windows. La instalación de XCP-ng center es bastante sencilla, simplemente te descargas el ejecutable y se instala. Con respecto a la instalación de openxenmanager, esta es más complicada.

En esta instalación he tenido unos cuantos problemas ya que se requiere de ejecutar una serie de comandos, los cuales me han producido bastantes errores. Uno de los más notables ha sido que al instalar dependencias, no se encontraban los paquetes necesarios, por lo que ha sido necesario actualizar el fichero de repositorios añadiéndole uno en el que se encontraban estos paquetes. Para ello, busqué un repositorio en el que estuvieran estos paquetes y lo añadí a /etc/apt/sources.list. También cabe mencionar que la instalación se ha hecho a través de git, clonando e instalando el programa con los comandos necesarios.

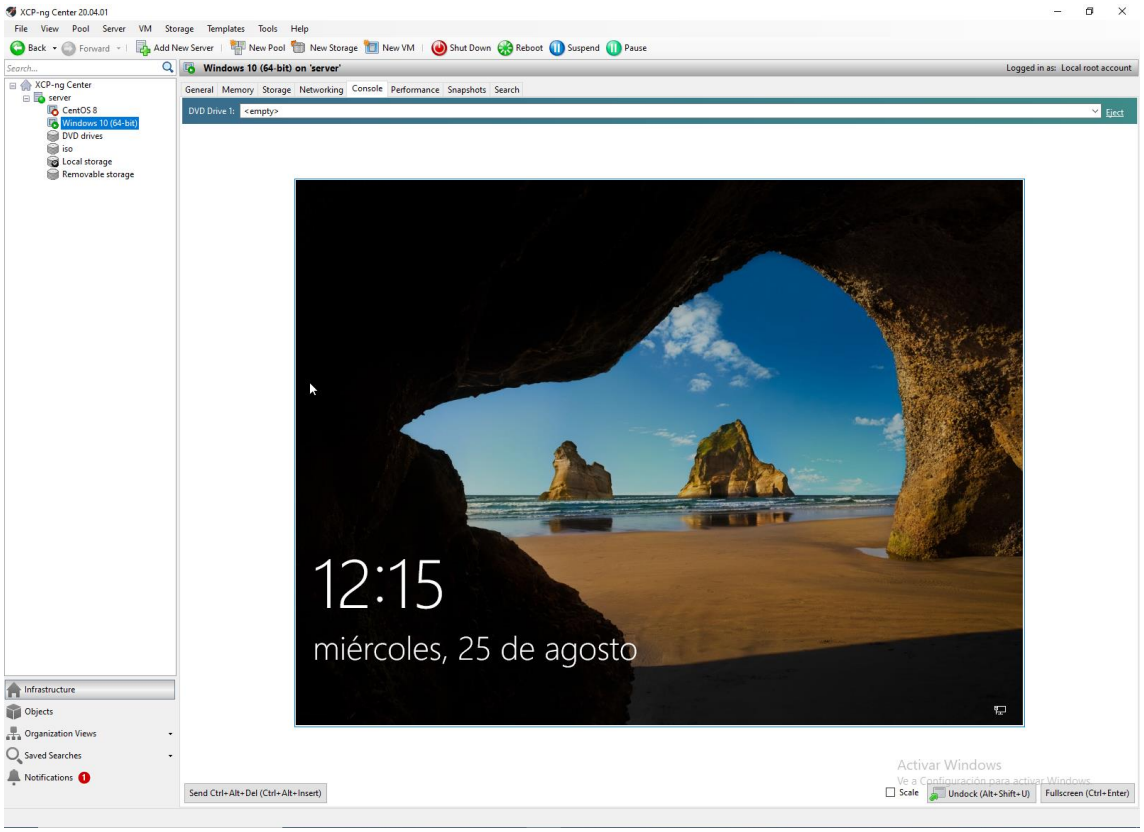

Figura 6. Interfaz de cliente de XCP-ng center

En este punto ya tenemos las dos instalaciones completadas. Sus interfaces y funcionalidades son bastante parecidas, por lo que me voy a centrar en explicar sus características desde la versión de XCP-ng Center para Windows que es la que aparece en la figura 6.

En el momento que se entra en el programa, lo primero que te va a pedir es acceder a nuestro servidor. Una vez accedido lo que se ve es lo que se muestra en la figura 6. Desde aquí se puede administrar bastantes cosas del sistema. Principalmente la creación de máquinas virtuales. Para ello se selecciona la opción de New VM la cual se explica con más detalle en el siguiente capitulo.

También me ha sido bastante útil la parte de gestión de redes de este programa porque permite crear, asignar, administrar y eliminar redes de manera muy sencilla e intuitiva. Todo esto me ha permitido hacer muchas pruebas diferentes para la configuración de la red hasta hallar la mejor. Algunas de estas pruebas consistían en crear redes conjuntas para varias máquinas virtuales agrupándolas en pools o crear redes con distintas características para ver su funcionamiento dentro del sistema.

Desde este programa, no solo se puede administrar todas las máquinas virtuales, sino que también se puede encenderlas y apagarlas, aparte de poder acceder a ellas. Como se puede observar en la figura 6 se está accediendo a una VM. Cuando se accede a las máquinas virtuales, se virtualiza todo el sistema al completo, pudiendo acceder a todas sus funcionalidades. El encendido y apagado se hace dando clic en un solo botón lo que facilita mucho su manejo.

3.2.- Creación y configuración de VM

En este apartado se explicarán los pasos necesarios para crear las diferentes máquinas y las configuraciones de las mismas, aparte de las cosas aprendidas y los problemas encontrados.

Para entender las configuraciones utilizadas es necesario saber las características del sistema con el que estamos trabajando. Se trata de un ordenador con un procesador Intel Xeon E3-1500 v5 con 3GHz. X de memoria RAM y 25GB de almacenamiento. 7 puertos USB (no sé qué más poner). Como se puede comprobar, este es un ordenador medio, lejos de los que se puede esperar de un servidor con capacidad para ejecutar decenas de máquinas virtuales a la vez que es el objetivo final, pero con capacidad suficiente para hacer el prototipo que se pretende para este proyecto.

En el momento que se termina la instalación de Xen y sus clientes es momento de empezar la creación de las VM. Para esto antes de poder crear la maquina se tiene que obtener sus imágenes. En este paso simplemente descargamos las que queramos de internet. Estas imágenes que se han descargado hay que guardarlas en el sistema para que las detecte y se puedan crear máquinas virtuales con ellas. Para ello se hace un repositorio de ISOS. Este repositorio es un fichero dentro del servidor especializado en guardar estas ISOS. Una de las principales ventajas es poder añadir las ISOS que se quieran para posteriormente poder crear máquinas virtuales que generen los sistemas operativos deseados. Esto es necesario para poder crear las máquinas virtuales que se requerían para el proyecto.

Lo primero que hay que hacer para crear un repositorio de ISOS es analizar nuestro sistema y seleccionar el lugar donde se quieren almacenar estas imágenes. Hay que tener en cuenta que generalmente las imágenes de cualquier sistema operativo son bastante pesadas por lo que van a ocupar bastante espacio de disco y nuestro sistema no tiene una capacidad de disco limitada.

En el momento que se selecciona el lugar, se crea una carpeta. En nuestro caso con mkdir /isos. Posteriormente se ejecuta un comando que añade el fichero que hemos creado como un repositorio de tipo ISO en XCP-ng.

En el momento que se ejecuta este comando, el repositorio ya está listo para usarse. Lo único que falta es meter las imágenes dentro, esto se puede hacer por numerosos métodos de transferencia de datos, en mi caso al haber descargado las imágenes en un ordenador diferente al servidor, he tenido que transferir esas imágenes y luego añadirlas. Para ello he transferido esas imágenes por la línea de comandos a través de una conexión ssh.

Cuando ya tenemos nuestro repositorio creado y con sus imágenes dentro, ya se puede crear una máquina virtual. Para esto hay numerosos métodos, tanto si estas en el servidor como si estas en un cliente como openxenmanager o XCP-ng center. En este caso me voy a centrar en explicarlo como se haría a través de un cliente como openxenmanger o XCP-ng center.

Lo primero es seleccionar New VM para empezar la creación de una nueva máquina virtual. En el momento que se selecciona esta opción se abre una pestaña en la que hay que seleccionar las diferentes opciones que queremos para nuestra máquina virtual. Lo primero es elegir la plantilla del sistema operativo que queramos virtualizar. Hay plantillas para muchas versiones de sistemas operativos, pero obviamente no están todos los que existen. Por lo que, si no tenemos exactamente la misma versión, se puede seleccionar una versión cercana. Por ejemplo, si tenemos ISO de Ubuntu 20.10 y el template es de Ubuntu 20.04 se puede seleccionar.

Después se introduce el nombre de la máquina virtual que se quiera dar y opcionalmente una breve descripción de cuál es su propósito. Una vez introducido estos datos, se continua.

El siguiente paso es seleccionar el método de arranque de la ISO, al principio tuve problemas con este paso ya que la maquina no se iniciaba correctamente porque estaba seleccionando *UEFI boot* que era un método de arranque incorrecto. El método de arranque que acabe seleccionando fue *BIOS boot*. Este modo permite que las ISOS se arranquen correctamente y que se pueda ejecutar la instalación del sistema.

Una vez elegido el método de arranque hay que seleccionar en que servidor vamos a almacenar nuestra nueva máquina virtual. En nuestro caso, no tiene perdida porque solamente contamos con un servidor. El siguiente paso es seleccionar la capacidad de nuestra nueva VM. Aquí se selecciona tanto la capacidad de almacenamiento como la capacidad de cómputo de nuestro sistema. En este punto se probaron diferentes capacidades para saber cómo responde el sistema. El resultado de estas pruebas fue que, dada la limitada capacidad de nuestro servidor, las maquinas iban casi siempre lentas. Solamente iban relativamente fluidas cuando se les daba mucha capacidad. Mas adelante se explica los parámetros seleccionados para cada una de las máquinas y el porqué de cada uno.

Una vez seleccionados estos parámetros, te pide seleccionar el lugar donde se van a almacenar estos datos, siempre teniendo en cuenta la capacidad seleccionada. Una vez seleccionada la deseada se sigue a delante.

Por último, te pide seleccionar la red que se va a utilizar. Como se va a explicar más adelante la configuración de redes, en este paso simplemente seleccionamos la red deseada y continuamos. Al dar continuar nos saldrá un resumen de todo lo que hemos seleccionado por última vez preguntándonos si queremos cambiar algo. Aquí simplemente seleccionamos Create Now para que nos cree nuestra nueva máquina virtual.

Siguiendo estos pasos se han creado 2 máquinas virtuales, una Windows y otra CentOS con capacidad de 3GB y 2GB respectivamente. El servidor tenía un tope de 5GB a la hora de virtualizar por lo que se repartieron entre las dos dándole 1GB extra a la máquina de Windows porque es más costoso de virtualizar. Además, la máquina de Windows cuenta con 2 CPU con 1 socket y CentOS cuenta con 1 CPU y 1 socket estas diferencias se explican por el mismo motivo que él anterior. Se ha dado estas capacidades

relativamente limitadas a las dos máquinas dada la capacidad del servidor. La idea siempre ha sido poder acceder a ambas máquinas de manera simultánea y no era posible si se le asignaban más recursos. El motivo por el cual la máquina de Windows tiene más recursos asignados, es que el sistema operativo es más costoso de virtualizar, por lo que necesita más que la VM de CentOS.

En el momento que se crean estas máquinas virtuales, lo primero que se tiene que hacer es instalarlas, por ello la primera vez que se inician es como instalar un sistema operativo de cero seleccionando todas las características necesarias. Al no haber ninguna característica especial, simplemente se instalan con valores por defecto.

Llegados a este punto, todas las maquinas ya están creadas, pero aún no están configuradas. Lo primero que se hace es dotarlas de acceso a internet, esto se explica en capítulo de configuración de red. Una vez con acceso a internet y visibles para el resto de los sistemas que quieran acceder a ella, se tiene que configurar la máquina virtual para que se pueda acceder a ellas de manera remota.

Un LightDM [16] es un programa que permite controlar de manera remota desde un sistema cliente. Este programa es de software libre y gratuito de que implementa un protocolo RDP con sistema de ventanas de *X Window System*. Este sistema es completamente compatible con el sistema operativo donde lo queremos instalar por lo que es perfecto para nuestro proyecto.

En la VM de CentOS se ha instalado LightDM [16], este programa se instala con solamente un comando por lo que es extremadamente sencillo de introducir en el sistema. La configuración del mismo requiere de bastante poco porque la mayoría de las opciones se seleccionan en la instalación y no hay ninguna reseñable.

Para la VM de Windows, el proceso es más sencillo. Simplemente hay que entrar en la configuración de Windows, entrar en la parte de sistema y seleccionar escritorio remoto como se observa en la figura 7. Una vez dentro hay que habilitar el escritorio remoto como se muestra en la imagen. La única condición para que esto funcione es que el Windows tiene que ser Pro, no puede ser Home.

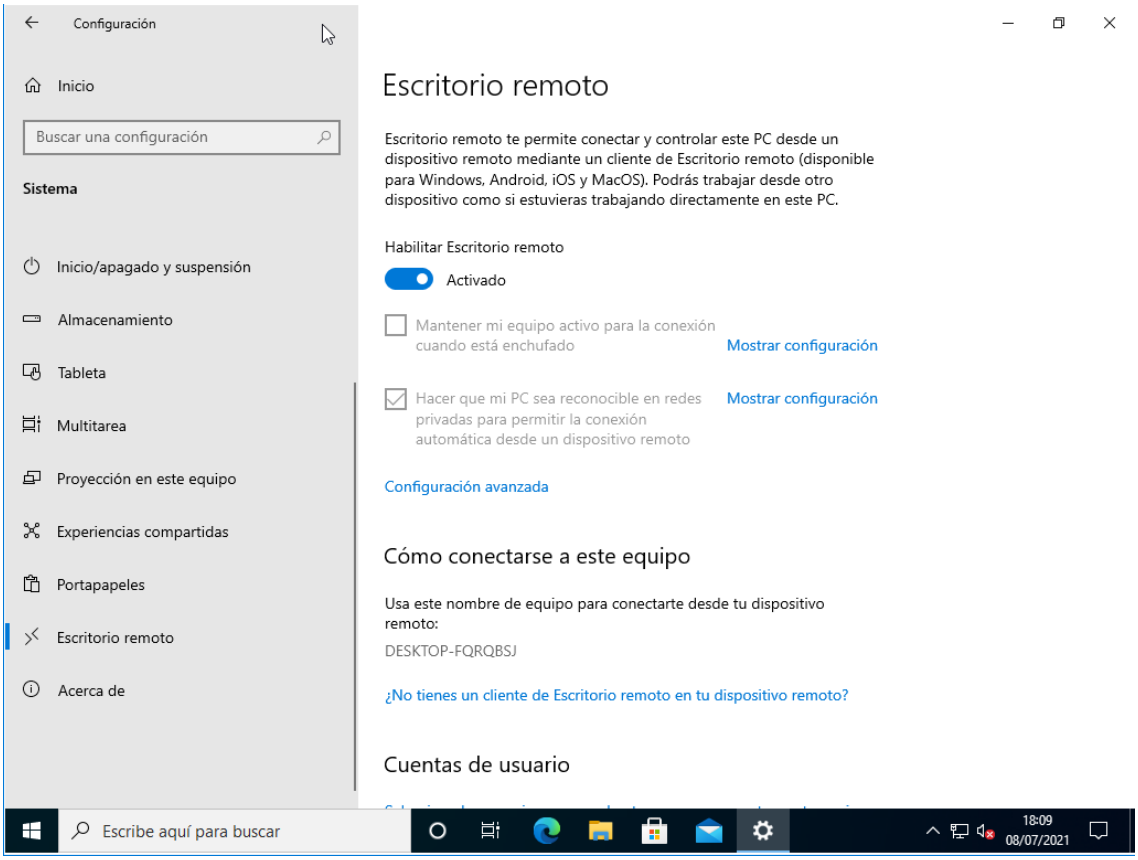

Figura 7. Activación de escritorio remoto para Windows

Una vez completados estos pasos las máquinas virtuales ya estas preparadas para ser utilizadas desde cualquier cliente que esté configurado para hacerlo. Esta configuración se explica en el capítulo 4.

Por último, se reduzco la cantidad de RAM que utilizan las máquinas virtuales cuando se ejecutan para poder ejecutar ambas maquinas al mismo tiempo en el servidor dadas las reducidas características del sistema.

3.3.- Configuración de Red

Con total seguridad, la configuración de la red de este proyecto ha sido la más difícil de entender, como de implementar. Se ha pasado por numerosas ideas hasta llegar a la implementación final deseada. El problema principal que se planteaba desde el principio era como crear una red privada que tuviera acceso a internet y que además todos los dispositivos estuvieran interconectados y se vieran entre ellos, lo que es parte fundamental para la virtualización.

Nuestro sistema como se observa en la figura 1, se compone de una red privada, con acceso a internet a través de una IP publica situada en el servidor. Este servidor actúa de *router*, haciendo *forwarding* y proveyendo de internet al resto de sistemas dentro de la red, conectadas a través de un *switch*. El servidor cuenta con dos tarjetas de red, una publica conectada a internet y otra conectada a la red privada. Todos los sistemas se

han configurado de manera manual estática porque se quería dar diferentes direcciones a cada dispositivo de la red y que no cambiaran.

Lo primero que se hizo, fue configurar todos los sistemas dándoles una IP privada salvo una de las tarjetas de red del servidor que se puso pública. De esta forma la configuración quedo de la siguiente manera.

- Red pública servidor – 193.144.198.205
- Red privada servidor – 192.168.0.20
- Cliente – 192.168.0.10
- Raspberry PI – 192.168.0.50
- VM Windows – 192.168.0.101
- VM CentOS – 192.168.0.102

Las máquinas virtuales son contenedores que por sí mismos no tienen ninguna conexión externa, por ello si queremos dotarlo de algún tipo de conexión tenemos que asociarlos a alguna red. Estas redes las crea el dominio principal o DOM0 y las asocia a las máquinas virtuales por medio de *bridges*. Estos puentes unen las propias máquinas virtuales con las redes que están asociadas a puertos físicos. Este mecanismo se realiza automáticamente cuando al crear una máquina virtual la asocias una red creada previamente.

El siguiente paso fue dotar de conexión a internet a todos estos dispositivos de la red privada. Para ello se realizó un *forwading* en el servidor siguiendo los siguientes pasos. Primero modifica permanentemente el fichero donde esta almacenada la variable que controla el *forwarding* en /etc/sysctl.conf añadiendo una línea con net.ipv4.ip_forward = 1. De esta manera el *forwarding* ya estaría activado.

Esto no soluciona el problema de internet, ya que faltaría por modificar el fichero de configuración de las tablas IP para evitar que se bloqueen los paquetes y que además se redireccionen de manera correcta. La regla que hace que funcione el *forwarding* es :FORWARD ACCEPT [0:0] ya que permite aceptar todos los paquetes que vengan por esta vía. Después de todos estos cambios, el fichero de configuración queda como muestra la figura 8.

```
# sample configuration for iptables service
# you can edit this manually or use system-config-firewall
# please do not ask us to add additional ports/services to this default configuration

*mangle
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT

*nat
:PREROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
# Masquerade all packets going from VMs to the LAN/Internet.
-A POSTROUTING -s 192.168.0.0/24 -o xenbr0 -j MASQUERADE
COMMIT


*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
# DHCP for host internal networks (CA-6996)
-A RH-Firewall-1-INPUT -p udp -m udp --dport 67 --in-interface xenapi -j ACCEPT
-A RH-Firewall-1-INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
# Linux HA hearbeat (CA-9394)
-A RH-Firewall-1-INPUT -m conntrack --ctstate NEW -m udp -p udp --dport 694 -j ACCEPT
-A RH-Firewall-1-INPUT -m conntrack --ctstate NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -m conntrack --ctstate NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -m conntrack --ctstate NEW -m tcp -p tcp --dport 443 -j ACCEPT
# dlm
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 21064 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m multiport --dports 5404,5405 -j ACCEPT
#-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Figura 8. Fichero de configuración de tablas IP

Después de esto tampoco había conexión a internet, porque faltaban los DNS, por lo que se configuraron de manera manual en todos los dispositivos de la red privada. La dirección que se les dio fue 193.144.193.11 y 193.144.193.22.

Esta ha sido la parte que más me ha costado de realizar de este proyecto ya que era la que menos controlaba y además ha sido la más difícil de diseñar dada la alta complejidad del sistema que se quería obtener. En lo que más tiempo se ha gastado en este apartado era en buscar cual era el problema que hacía que no funcionara el sistema al que queríamos llegar.

4.- Implementación del Cliente

Una vez realizada la configuración del *backend* del sistema es momento de preparar el *frontend*. Para ello se han modificado el sistema operativo de una Raspberry PI para hacer que ejecute de arranque un programa que proporciona acceso a las VM. El objetivo desde el principio fue crear una GUI que tuviera dos botones y que al presionar cada uno de ellos, se accediera a cada una de las máquinas virtuales.

En este capítulo se empezará por explicar cómo se ha hecho la aplicación, qué funcionalidades tiene, seguidamente se explicará las modificaciones hechas en el cliente para que pueda funcionar la aplicación correctamente y por último se describirá el prototipo que tenemos en funcionamiento.

4.1.- Desarrollo de la Aplicación cliente

La aplicación esta creada con WindowBuilder que es un programa de creación de interfaces basado en JFrame que hace más fácil la creación de interfaces. Con esta base se crea una pestaña sin bordes que se ejecuta en pantalla completa. Se busca que sea de esta manera porque al intentar que este proyecto sea un prototipo usable en docencia, no se quiere que el alumno tenga acceso al sistema operativo subyacente.

En esta pestaña, hay 3 botones. Los dos botones que acceden a cada una de las máquinas virtuales y un tercer botón que permite salir de la aplicación. Cada uno de estos 3 botones tiene asignado un método que al pulsarse ejecuta.

El botón de *Exit* cierra la pestaña permitiendo salir de la aplicación y acceder a la Raspberry PI de manera normal. Este botón está pensado para momentos especiales en los que se quiera salir de la aplicación para poder gestionar el cliente por el motivo que sea.

Los botones de Windows y CentOS son más complejos. Lo primero que hacen una vez pulsados es crear una conexión con el servidor. Esto se hace a través de una librería externa Jsch que permite hacer una conexión ssh al servidor. Una vez conectado al servidor, se ejecuta un comando sobre la consola que arranca la máquina virtual seleccionada.

En este punto todavía no se puede realizar la conexión con la máquina virtual porque ésta todavía no se ha terminado de arrancar. Por este motivo se espera unos segundos para que arranque la máquina utilizando un *Thread sleep*.

Una vez esperado ese tiempo, se hace una llamada al método de *Runtime* que ejecuta un comando. Este comando ejecuta el fichero de Remmina que abre una pestaña dónde se hace la virtualización del sistema si todo se ha ejecutado correctamente y no se ha producido ningún fallo.

El programa cada vez que intenta realizar una conexión, independientemente de que salga bien o no, se reinicia automáticamente para evitar fallos. Por lo tanto, sí al intentar la conexión falla, simplemente hay que cerrar la pestaña de Remmina que se ha cargado

para la virtualización y volver a seleccionar un botón para intentar conectarse de nuevo a la VM.

Una vez creado este programa se exporto como un .jar ejecutable para poder ejecutarlo en el cliente.

El programa completo se encuentra en https://github.com/mbc776/TFG.

4.2.- Optimización del cliente

Una vez creado el archivo ejecutable el siguiente paso es hacer que se ejecute de arranque en el sistema y que funcione todo correctamente.

Lo primero que hay que hacer para que se pueda ejecutar el programa es instalar los programas necesarios. El primero y más obvio es Java ya que el ejecutable está programado en ese lenguaje. Se descarga la última versión de internet y se instala de manera normal. También es necesario instalar Remmina para realizar la virtualización. Una vez instalado, se tienen que crear los ejecutables que utiliza el programa que hemos hecho para que funcione correctamente. Estos ejecutables tienen que estar guardados en un lugar accesible para el programa, porque de otra manera el programa fallaría.

Una vez que el programa funciona correctamente, el siguiente paso es que se ejecute de inicio. Para ello se crea un servicio como el que se ve en la imagen. Este servicio hace que se ejecute un script, que se puede observar en la figura 10, en el momento que se inicia el sistema, desde el cual se arranca el entorno gráfico diseñado.

```
[Unit]
Description=Script de virtualizacion
After=systemd-journald.socket

[Service]
#Enviroment=DISPLAY=:0
ExecStart=bash /home/pi/virtualizacion.sh
Restart=always
WorkingDirectory=/home/pi
User=pi
Group=pi


[Install]
WantedBy=multi-user.target
```

Figura 9. Servicio que ejecuta el script para iniciar el programa

Lo único destacable de este servicio es que se tiene que ejecutar después del systemd-journald.socket porque si se ejecuta antes de este proceso produce un error la primera vez que se inicia la Raspberry PI que hace que falle la interfaz gráfica y haya que reiniciar el programa. Además, esta puesta la opción de Restart=always que hace que cada vez

que se cierre la interfaz, se abra otra nueva. Esta opción esta para que no se pueda usar la Raspberry PI para otra cosa que no sea acceder al servidor.

```
#!bin/bash
echo $DISPLAY >> file.txt
export DISPLAY=:0.0
java -jar /home/pi/Prueba.jar
~
~
```

Figura 10. Script para iniciar el programa

Después de esto el proceso nos daba un fallo porque no detectaba bien la pantalla por la que debía arrancar el programa. La solución fue cambiar la variable de entorno DISPLAY a el valor :0.0 con el comando export $DISPLAY=:0.0 que significa que se ejecuta en la primera pantalla de tu dispositivo.

4.3.- Prototipo

A lo largo de todo el proyecto hemos ido explicando las diferentes partes de las que consta. En este apartado se explicará el modelo que tenemos en funcionamiento.

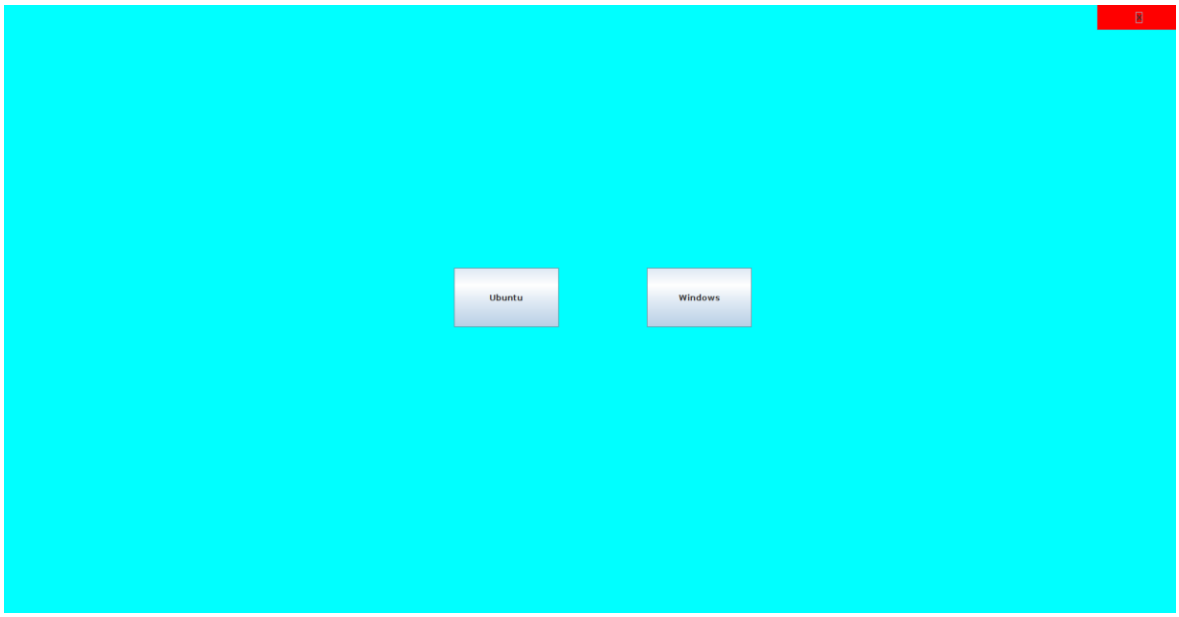

Figura 11. Captura de la interfaz gráfica del programa en funcionamiento

El prototipo en funcionamiento es el que se muestra en la figura 11, en él se puede observar los dos botones que acceden a cada una de las máquinas virtuales además de un botón que cierra la aplicación sobre un fondo azul claro. En el momento en el que se pulsa en uno de los dos botones se abre una pantalla de carga como la que se observa en la figura 12.

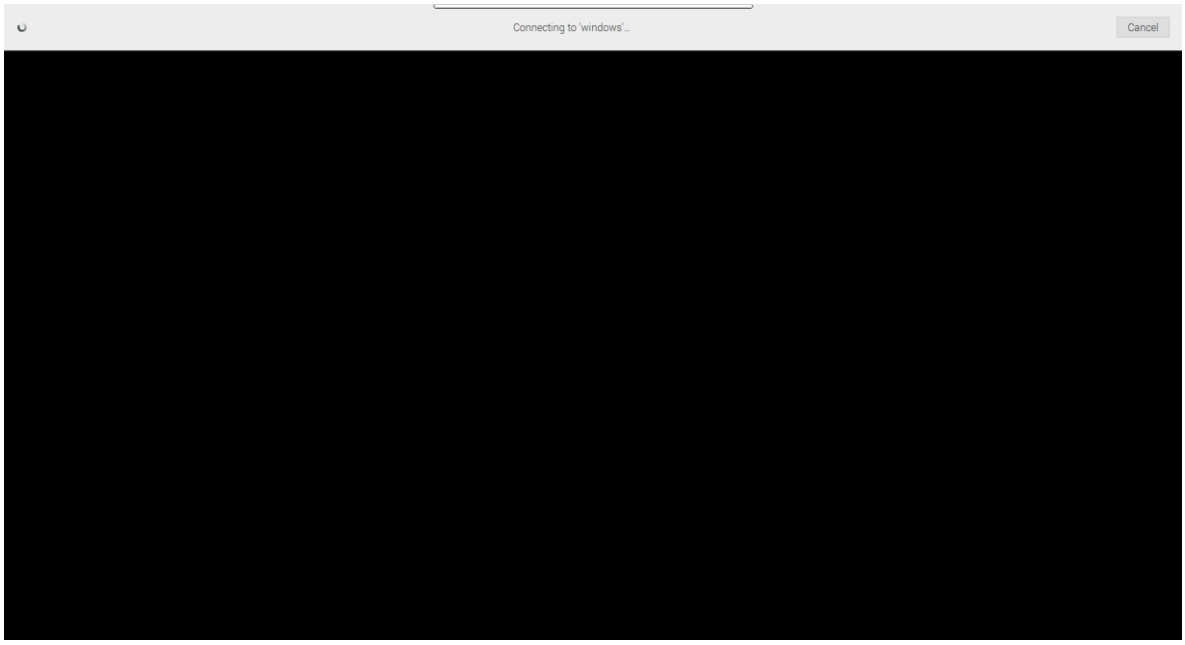

Figura 12. Captura de la pantalla de carga del programa en funcionamiento

En este punto hay dos opciones la primera es que la máquina virtual carga correctamente, pero también hay una posibilidad de que falle la conexión. Si esto sucede se cerraría la pestaña de conexión y el programa se reiniciaría automáticamente evitando fallos y permitiendo volver a acceder a cualquiera de las dos máquinas de virtuales de manera normal. Si no hay ningún fallo en la conexión lo que veríamos a continuación sería lo que se puede observar en la figura 13 y 14.

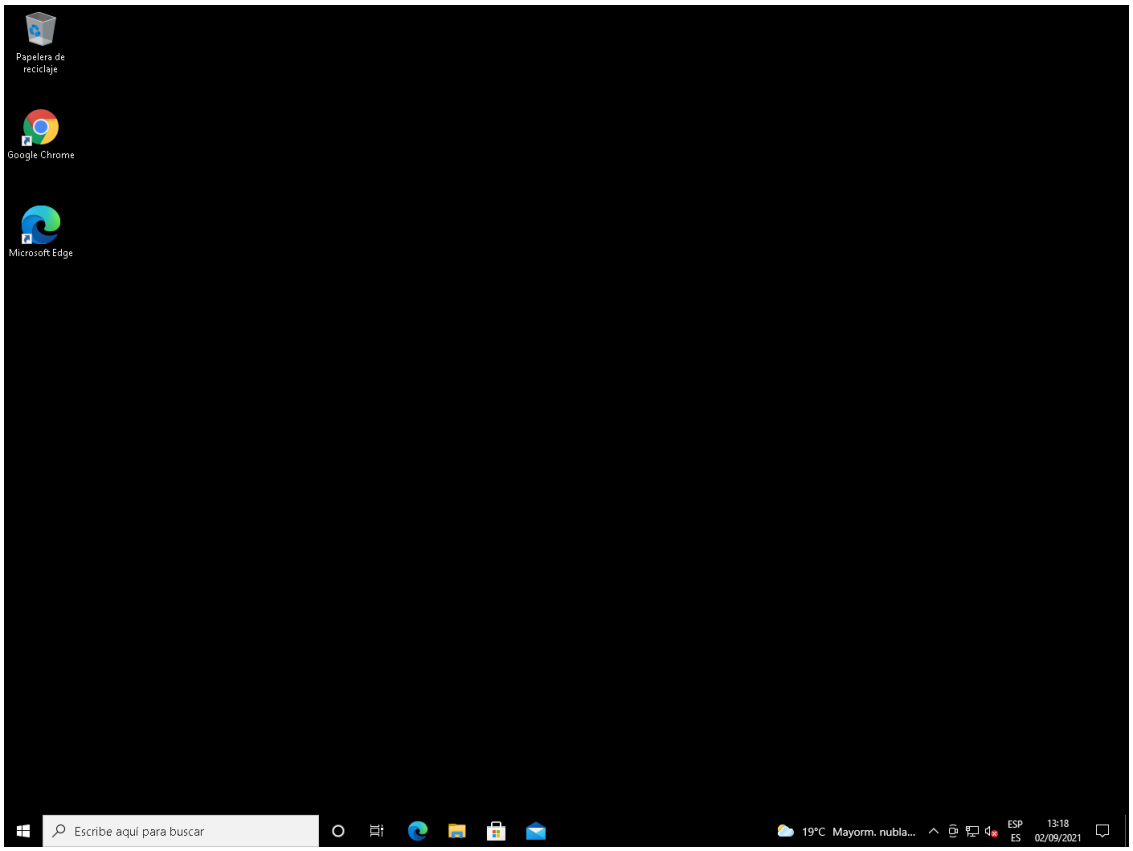

Figura 13. Captura de la pantalla de la VM de Windows del programa en funcionamiento

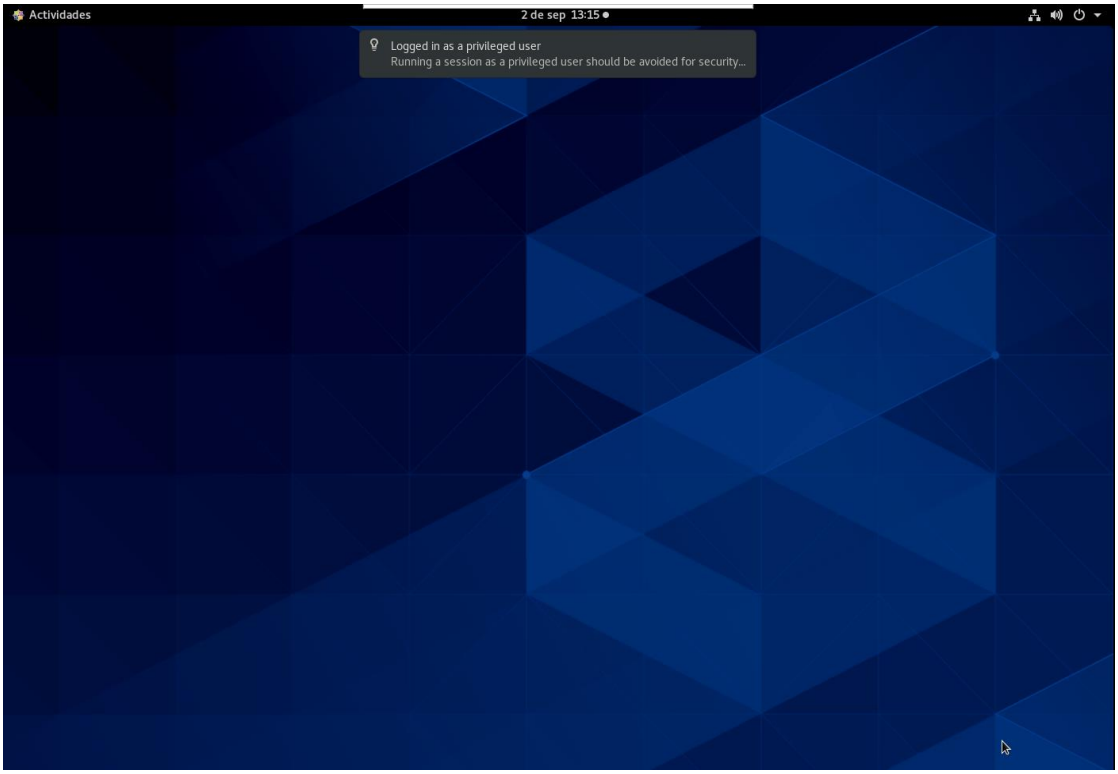

Figura 14. Captura de la pantalla de la VM de CentOS del programa en funcionamiento

Una vez accedido a las máquinas éstas cuentan con toda la funcionalidad del sistema operativo y por lo tanto se puede utilizar en su totalidad. Una vez finalizado su uso se pueden cerrar de 2 maneras tanto apagando la máquina virtual cómo cerrando la conexión. Cualquiera de estas opciones es válida, porque el programa se reinicia automáticamente cuando cualquiera de estas dos cosas sucede evitando fallos. La única diferencia entre ambas es que si se cierra la conexión la máquina virtual permanece encendida, de la otra forma se apaga.

5.- Conclusiones y Líneas futuras

Desde el principio del planteamiento de este proyecto, se tenía una la idea sobre él bastante amplia. El punto final al que se pretendía llegar no estaba claro, por lo que optó por comenzar, dejándolo abierto. Durante su desarrollo, se fue perfilando y finalmente se decidió hasta donde se quería llegar. El objetivo fue hacer un prototipo para poder seguir desarrollándolo en futuros proyectos.

A lo largo de toda esta investigación se han tenido que superar las numerosas dificultades que se fueron presentando, solventándolas y finalmente logrando los principales puntos que se buscaban, esto es, obtener un prototipo fácilmente ampliable, investigar y conocer todas las tecnologías existentes en el campo de la virtualización con el fin de decidir que tecnologías eran las más idóneas para nuestro proyecto, y como añadido que pudiese ser aplicado en el ámbito de la docencia.

Con este proyecto, al tratarse de un prototipo, de cara a futuro, se podrá ampliar y optimizar. El objetivo que se pretendió es que este proyecto, se pudiese desarrollar con el tiempo y conseguir dar cobertura a más usuarios (que acceden al sistema de manera concurrente, con las dificultades que todo esto conlleva).

Lo primero que se debería abordar es la escalabilidad del servidor, porque las capacidades con las que hemos trabajado han sido bastante limitadas, sobre todo en cuanto al número de máquinas virtuales que se pueden ejecutar a la vez. Como se mencionó en el apartado de creación y configuración de VM, se tuvo que reducir el tamaño de RAM utilizado, al no poderse ejecutar las dos máquinas al mismo tiempo.

Cuando se amplía un proyecto como este, hay que tener en cuenta los problemas añadidos que se generan, como puede ser, la seguridad, la administración de usuarios y el balanceo de carga.

Otro aspecto que queda pendiente para seguir trabajando en este proyecto de cara al futuro es la autenticación de usuarios. Este problema abordable, dotaría al proyecto de seguridad y privacidad de cara a su implementación en un laboratorio real, ya no siendo un prototipo.

La interfaz gráfica que se ha creado es básica y dependiendo del como se quiera enfocar el proyecto a futuro, puede llegar a ser muy diferente. Se podría cambiar por completo, desde lo que se ve en la pantalla hasta las librerías que se utilizan para realizar la conexión al servidor. También se podría elegir otro lenguaje de programación que se pudiese considerar más idóneo.

La gestión de la conexión al servidor y la creación de la virtualización son los procesos considerados a mantener para futuros proyectos de desarrollo de este trabajo.

Referencias

1) Fisher, J. (8 de junio de 2018) "*Virtualización de escritorio, reduce costes, aumenta la seguridad y la movilidad*." https://web.archive.org/web/20080908073951/http://www.virtualizationconference.com/node/642945

2) Punt informàtic. (s.f.) *¿Qué es el VDI?*. https://puntinformatic.com/que-es-el-vdi/

3) Klaus, L. (15 de Septiembre de 2020) "*¿Qué es el RDP, y cómo usarlo con seguridad?*" https://nordvpn.com/es/blog/acceso-remoto-rdp/

4) Teradici. (s.f.) *What is PCoIP technology?*. https://www.teradici.com/pcoip-technology/what-is-pcoip

5) Martín, R. (1 de febrero de 2012) "Un Thin Client que consume 60% menos de energía que un PC tradicional" https://directortic.es/networking/thin-client-que-consume-60-menos-energia-que-pc-tradicional-201202012247.htm

6) Servicio de informática de la universidad de Cantabria (s.f.) *Unican Labs.* https://sdei.unican.es/Paginas/servicios/salas_aulas/UNICANLabs.aspx

7) VirtualBox (s.f.) *Welcome to VirtualBox.org!* https://www.virtualbox.org/

8) VMware (s.f.) *VMware Workstation Pro*. https://www.vmware.com/es/products/workstation-pro.html

9) Microsoft (31 de mayo de 2018) *Microsoft Virtual Server*. https://docs.microsoft.com/es-es/previous-versions/windows/desktop/msvs/microsoft-virtual-server-portal

10) XenProyect (s.f.) *What is the xen project?* https://xenproject.org/about-us/

11) Oracle (s.f.) *Descripción general de Oracle VM Server for x86*. https://docs.oracle.com/cd/E37929_01/html/E36580/ovmx86overview.html

12) XenProject Wiki (Ultima modificación 28 de junio de 2018) *Credit Scheduler* https://wiki.xenproject.org/wiki/Credit_Scheduler

13) XenProject Wiki (Ultima modificación 18 de julio de 2017) *Credit2 Scheduler* https://wiki.xenproject.org/wiki/Credit2_Scheduler

14) XenProject Wiki (Ultima modificación 14 de agosto de 2016) *RTDS-Based-Scheduler.* https://wiki.xenproject.org/wiki/RTDS-Based-Scheduler

15) RaspberryPi (s.f.) *Raspberry Pi 3 Model B*. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

16) Archlinux (1 de agosto de 2021) *LightDM (Español)*. https://wiki.archlinux.org/title/LightDM_(Espa%C3%B1ol)