

## **Trabajo Práctico Integrador**

### **Tema: Virtualización**

#### **“Instalando un Sistema Operativo con VirtualBox: el rol del Hypervisor Tipo 2”**

#### **Alumnos**

Santiago Octavio Varela ([santiagov.linked@gmail.com](mailto:santiagov.linked@gmail.com)) ,

Ximena Maribel Sosa ([ximenasosa44@gmail.com](mailto:ximenasosa44@gmail.com))

**Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.**

#### **Arquitectura y Sistemas Operativos**

##### **Docente Titular**

Osvaldo Faravella

##### **Docente Tutor**

Adriel Ezequías Herrera

##### **Fecha de entrega**

3 de Junio de 2025

### **Tabla de contenido**

○ Introducción	3
○ Marco Teórico	4
○ Caso Práctico	9
○ Metodología Utilizada	17
○ Resultados Obtenidos	21
○ Conclusiones	22
○ Bibliografía	23
○ Anexos	24

## 1. Introducción

La virtualización es una tecnología clave en el ámbito de la informática moderna, ampliamente utilizada en entornos de desarrollo, pruebas y producción. Es por eso que fue elegida a fines de realizar este trabajo, ya que es parte de la caja de herramientas de cualquier persona dedicada al desarrollo de software.

En este trabajo se abordará específicamente la instalación de un sistema operativo utilizando VirtualBox, un software de virtualización que funciona como Hypervisor de Tipo 2. La elección de este tema en específico, dentro del área temática de virtualización, responde a la necesidad de comprender cómo funciona la virtualización desde una perspectiva técnica aplicada y, además, cómo esta herramienta permite simular entornos operativos completos dentro de un sistema anfitrión, facilitando tareas fundamentales para la programación.

Este enfoque resulta relevante en la formación como técnico en programación, ya que brinda conocimientos prácticos sobre el uso de máquinas virtuales para la configuración de entornos de trabajo, la experimentación con distintos sistemas operativos y la ejecución segura y controlada de software en entornos aislados. Comprender el funcionamiento de un Hypervisor Tipo 2 como VirtualBox permite al estudiante identificar los recursos virtualizados involucrados (CPU, memoria, almacenamiento, red, dispositivos) y reconocer las implicancias de rendimiento y compatibilidad que conlleva esta capa de virtualización.

El objetivo principal del trabajo, entonces, es demostrar a través de una instalación práctica, cómo se configura y utiliza VirtualBox para ejecutar un sistema operativo invitado, distinto al que está permitiendo ejecutar VirtualBox. A su vez, se busca fundamentar técnicamente cada uno de los pasos realizados, explicando cómo interviene el hypervisor en la virtualización de los distintos recursos y destacando la importancia de la virtualización asistida por hardware desde BIOS/UEFI para garantizar el funcionamiento correcto de la máquina virtual. De este modo, se pretende lograr una integración entre el saber técnico y la práctica operativa, clave para la formación profesional en el campo de la programación.

URL Repositorio en GitHub: [https://github.com/xms44/ayso\\_integrador\\_VarelaSosa/tree/main](https://github.com/xms44/ayso_integrador_VarelaSosa/tree/main)

URL del video explicativo: <https://www.youtube.com/watch?v=oOn1xx24Z7Q>

## 2. Marco Teórico

### Virtualización: Concepto y Relevancia

La virtualización es una tecnología que permite crear versiones simuladas de recursos informáticos físicos, tales como sistemas operativos, servidores o dispositivos de red. A través de software especializado, estos recursos se abstraen del hardware subyacente y se presentan como entornos independientes.

Como se ha visto en los textos de cátedra (Tecnicatura Universitaria en Programación a distancia, 2025) en el contexto actual de la informática, la virtualización cumple un rol estratégico, tanto en desarrollo como en producción. Su utilización permite ejecutar múltiples entornos operativos en una única máquina física, lo cual resulta especialmente valioso para técnicos y programadores. Gracias a ella, es posible experimentar con diferentes sistemas operativos, probar software en entornos aislados y configurar laboratorios sin necesidad de múltiples equipos físicos.

Entre los principales beneficios se destacan:

- Eficiencia: Aprovechamiento óptimo del hardware disponible.
- Flexibilidad: Creación rápida de entornos personalizados para pruebas o desarrollo.
- Ahorro de costos: Menor necesidad de inversión en equipos físicos.
- Escalabilidad: Facilidad para adaptar los entornos virtuales según las necesidades.

Por otra parte, siguiendo el capítulo *Virtual Machines* (Silberschatz, Galvin, & Gagne, 2018, p. 704), hay tres principios fundamentales que se deben seguir para asegurar una virtualización efectiva:

- Fidelidad: el entorno virtual debe comportarse como el hardware físico real.
- Rendimiento: el impacto en la performance debe ser mínimo.
- Seguridad: el Virtual Machine Manager (VMM) o como se ha visto, el Hypervisor, debe tener el control total sobre los recursos del sistema.

### La importancia del CPU y el Hypervisor

Cómo señala el artículo *¿Qué es la Virtualización?* de IBM:

*“La virtualización de CPU (unidad central de procesamiento) es la tecnología fundamental que hace posibles los hipervisores, las máquinas virtuales y los sistemas operativos. Permite que una sola CPU se divida en varias CPU virtuales para su uso por varias máquinas virtuales.”* (IBM, s.f)

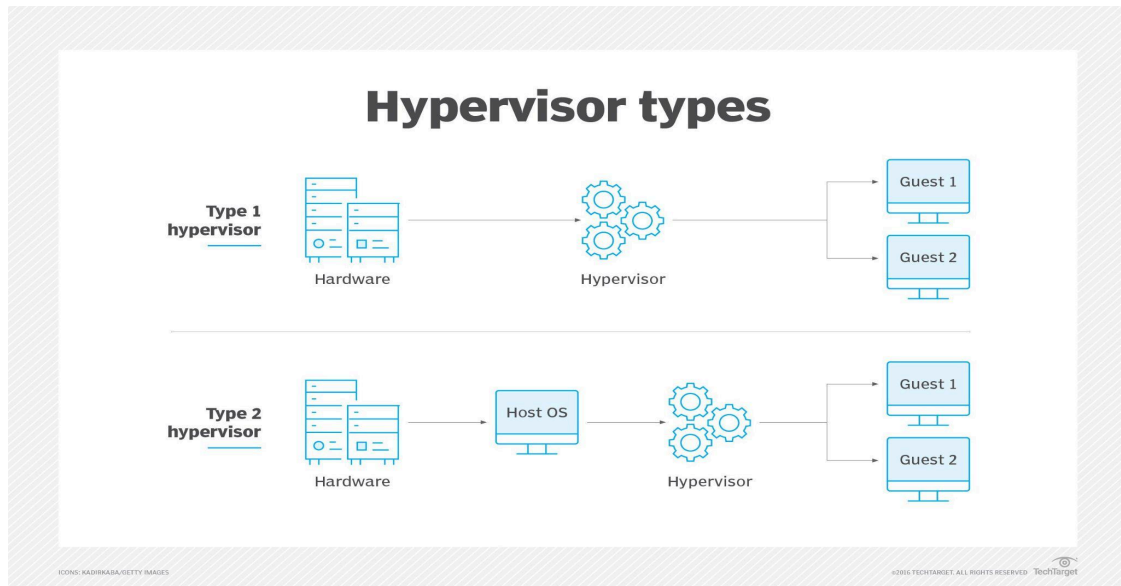
Es por esta razón que el avance de las herramientas de virtualización en el desarrollo de software está directamente relacionada con los cambios en los CPUs, ya que anteriormente estos no contaban con una configuración específica para la virtualización

El siguiente elemento de relevancia es el Hypervisor, la capa de coordinación de la tecnología de virtualización (AWS, s.f). Se trata de un software que actúa como mediador entre el hardware físico y los sistemas operativos virtualizados, permitiendo la creación, ejecución y gestión de múltiples máquinas virtuales (VMs). Su función es asegurar que cada VM acceda de forma controlada a los recursos del sistema.

Existen dos tipos principales de Hypervisores:

- Tipo 1 (Bare-metal): Se instala directamente sobre el hardware, sin sistema operativo intermedio. Ejemplos: VMware ESXi, Microsoft Hyper-V, Xen. Brinda mayor rendimiento y seguridad, aunque su configuración puede ser más compleja. Puede verse un esquema de este tipo de Hypervisor en el anexo.
- Tipo 2 (Hosted): Se ejecuta como una aplicación dentro de un sistema operativo anfitrión. Ejemplos: VirtualBox, VMware Workstation. Es ideal para entornos de pruebas y desarrollo por su facilidad de uso, aunque con un rendimiento algo inferior debido a la capa adicional del sistema operativo.

La diferencia esencial entre ambos radica en la interacción con el hardware: el Tipo 1 accede directamente, mientras que el Tipo 2 lo hace a través del sistema anfitrión. Esto puede observarse en el siguiente esquema:



### **Funcionamiento Interno de VirtualBox (Hypervisor Tipo 2)**

VirtualBox representa un claro ejemplo de Hypervisor Tipo 2. Funciona como una aplicación dentro del sistema anfitrión (por ejemplo, Windows o Linux), utilizando sus servicios y controladores para gestionar recursos virtuales.

Este enfoque introduce desafíos técnicos particulares:

- Latencia adicional debido a la capa intermedia.
- Competencia por recursos entre el sistema anfitrión y las VMs.
- Overhead en la traducción de instrucciones o direcciones de memoria.

Aun así, su arquitectura facilita el uso en entornos educativos o de desarrollo, ya que permite crear y eliminar VMs con rapidez, probar configuraciones diversas y simular infraestructuras sin requerir hardware dedicado.

### **Virtualización de Recursos**

---

<sup>1</sup> URL de la imagen: [https://www.techtarget.com/rms/onlineImages/server\\_virt-hypervisor.jpg](https://www.techtarget.com/rms/onlineImages/server_virt-hypervisor.jpg)

Siguiendo el texto de cátedra (Tecnicatura Universitaria en Programación a Distancia, 2025), para ejecutar un sistema operativo dentro de una VM, el Hypervisor debe virtualizar los componentes esenciales del hardware. Entre los más importantes se encuentran:

- CPU: Se utilizan técnicas como la traducción binaria (modifica instrucciones sensibles antes de ejecutarlas) y la virtualización asistida por hardware (tecnologías como Intel VT-x o AMD-V, que permiten ejecutar instrucciones privilegiadas directamente en el hardware, aumentando el rendimiento).
- Memoria: Se aplican mecanismos como las Shadow Page Tables, que traducen las direcciones virtuales del sistema invitado a direcciones reales del anfitrión. Tecnologías como Second Level Address Translation (SLAT) optimizan este proceso a nivel de hardware.
- Almacenamiento: Cada VM tiene uno o más discos virtuales (archivos .vdi, por ejemplo) que actúan como unidades de almacenamiento independientes. El acceso al disco se realiza a través del sistema anfitrión.
- Red: Se emplean adaptadores virtuales que permiten a las VMs conectarse entre sí o con redes externas mediante diferentes modos, como NAT (predeterminado en VirtualBox), Bridged o Red Interna.
- Dispositivos: El Hypervisor puede emular periféricos (como tarjetas gráficas o de red), o bien permitir el acceso directo (passthrough) a dispositivos físicos específicos para un rendimiento superior.

### **Importancia de la Virtualización Asistida por Hardware**

Para lograr un funcionamiento óptimo de las VMs, es esencial habilitar las tecnologías de virtualización desde la BIOS o UEFI del equipo. Estas extensiones del procesador, como Intel VT-x o AMD-V, permiten dividir el procesamiento entre múltiples entornos virtuales, reducir la latencia y ejecutar instrucciones privilegiadas con mayor eficiencia.

Acceder a estas opciones depende del modelo del equipo, pero suele hacerse durante el arranque, presionando teclas como F2 o Del y seleccionando la configuración de la BIOS desde allí frente a otras opciones de inicio. Su activación es crucial: sin este soporte, muchas VMs no funcionarán correctamente o tendrán un rendimiento muy limitado.

En conclusión, el presente marco teórico expone los fundamentos esenciales para comprender el proceso de virtualización, centrado en la utilización de VirtualBox como herramienta pedagógica y técnica. Se destaca el rol del Hypervisor Tipo 2 como intermediario funcional, las capacidades de virtualización de recursos clave, y la importancia del soporte de hardware. Esta base conceptual permite interpretar de manera fundamentada los pasos requeridos para instalar y configurar un sistema operativo dentro de una máquina virtual, objetivo práctico del trabajo.



### 3. Caso Práctico

#### **Descripción del problema a resolver**

El caso práctico de este trabajo está centrado en simular un entorno de trabajo aislado, flexible y controlado para fines didácticos, utilizando tecnologías de virtualización. La principal tarea consiste en instalar y configurar una máquina virtual mediante el uso de VirtualBox (un Hypervisor Tipo 2) sobre un sistema anfitrión, instalando como sistema operativo invitado Linux Mint XFCE, una distribución liviana basada en Ubuntu que favorece el bajo consumo de recursos y la accesibilidad para quienes no están familiarizados con entornos Linux más complejos.

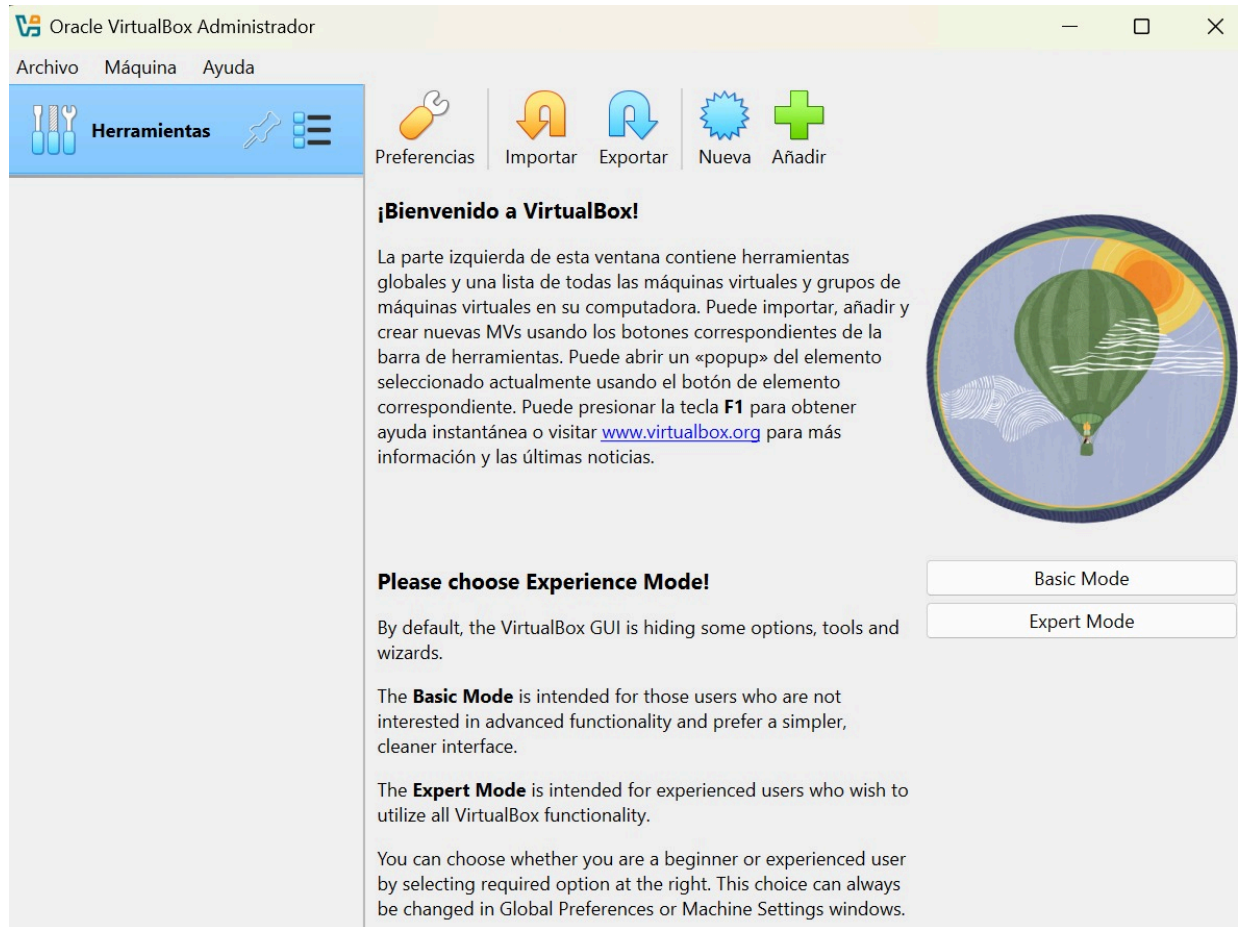
El propósito principal es adquirir experiencia práctica en los procesos técnicos asociados a la virtualización, incluyendo:

- La descarga, instalación y configuración de VirtualBox.
- La creación de una máquina virtual con parámetros específicos (memoria, almacenamiento, red, etc.).
- La instalación de un sistema operativo open source como Linux Mint XFCE.
- La interacción básica con el sistema operativo invitado.
- La ejecución de un programa en Python dentro del entorno virtualizado, como verificación mínima del correcto funcionamiento del sistema operativo y de sus capacidades básicas.

Este ejercicio permite comprender, desde la práctica, cómo los recursos del sistema anfitrión son compartidos y gestionados por el Hypervisor, y cómo se comporta un sistema operativo invitado en un entorno controlado. Además, ilustra las ventajas de la virtualización para el desarrollo y la experimentación segura, sin alterar el sistema operativo anfitrión ni comprometer su estabilidad (aunque esto depende del hardware con el que se cuente).

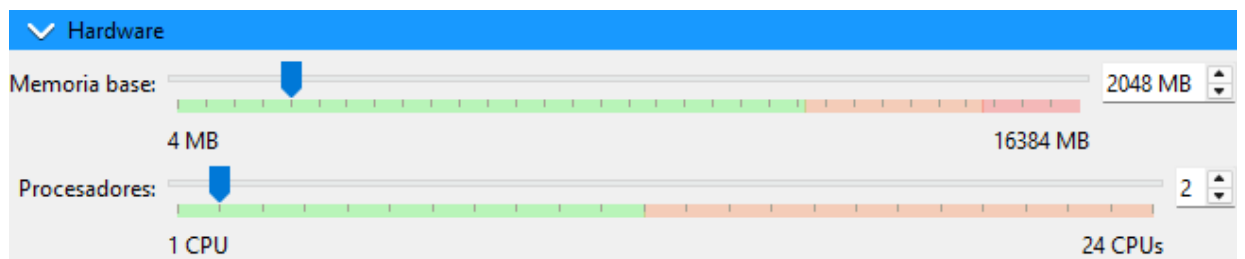
## Instalación de VirtualBox

Se procedió a instalar VirtualBox desde el sitio oficial<sup>2</sup>. Se aceptaron los valores por defecto:

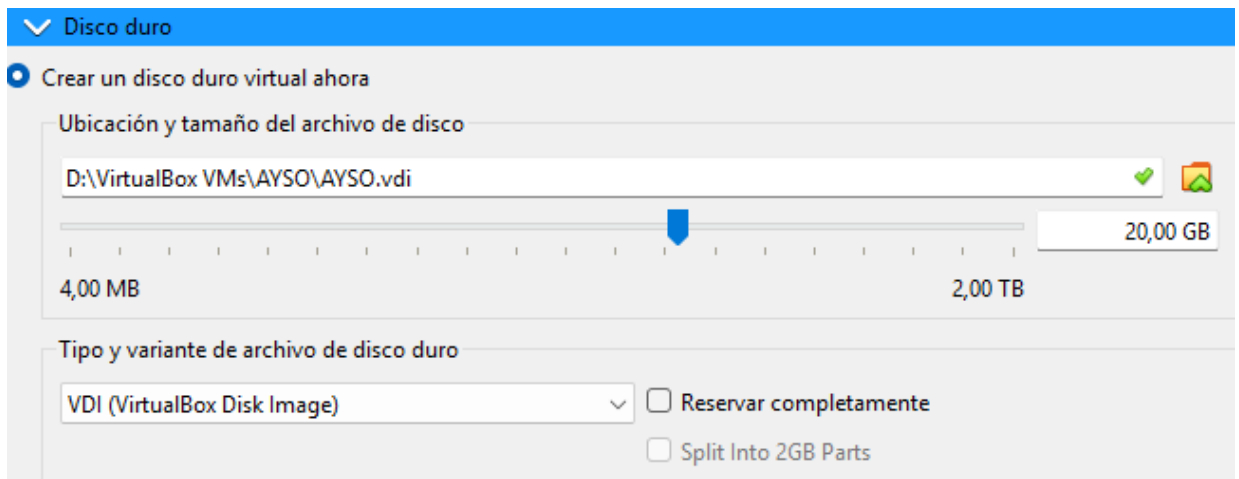


### 1. Creación de una nueva máquina virtual

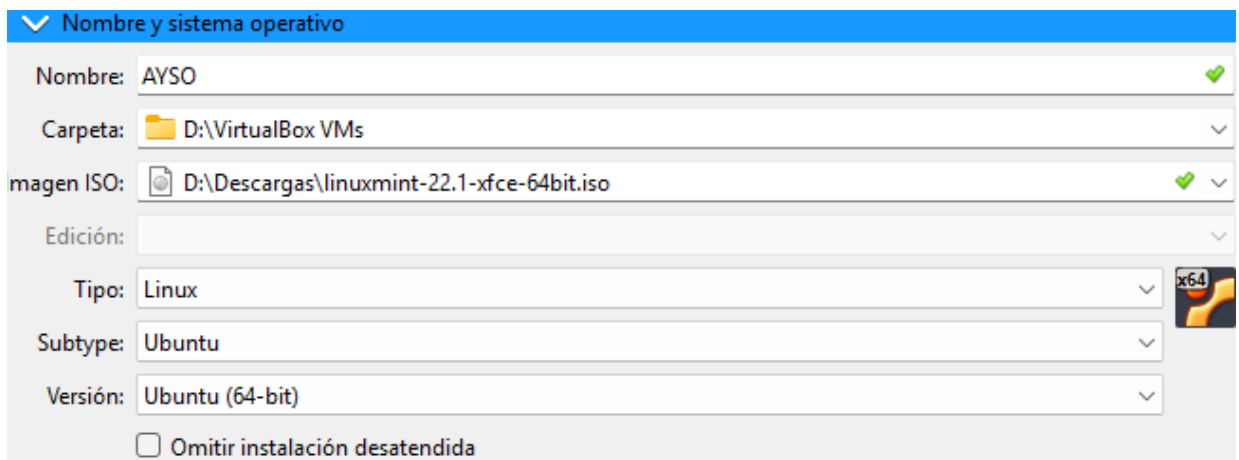
Se asignaron 2048 MB de RAM y 20 GB de disco dinámico:



<sup>2</sup> URL del sitio oficial de VirtualBox: <https://www.virtualbox.org/>



## 2. Montaje de la ISO e inicio de instalación



## 3. Instalación finalizada y acceso al sistema

En la siguiente imagen se aprecia que inició correctamente en modo Live:

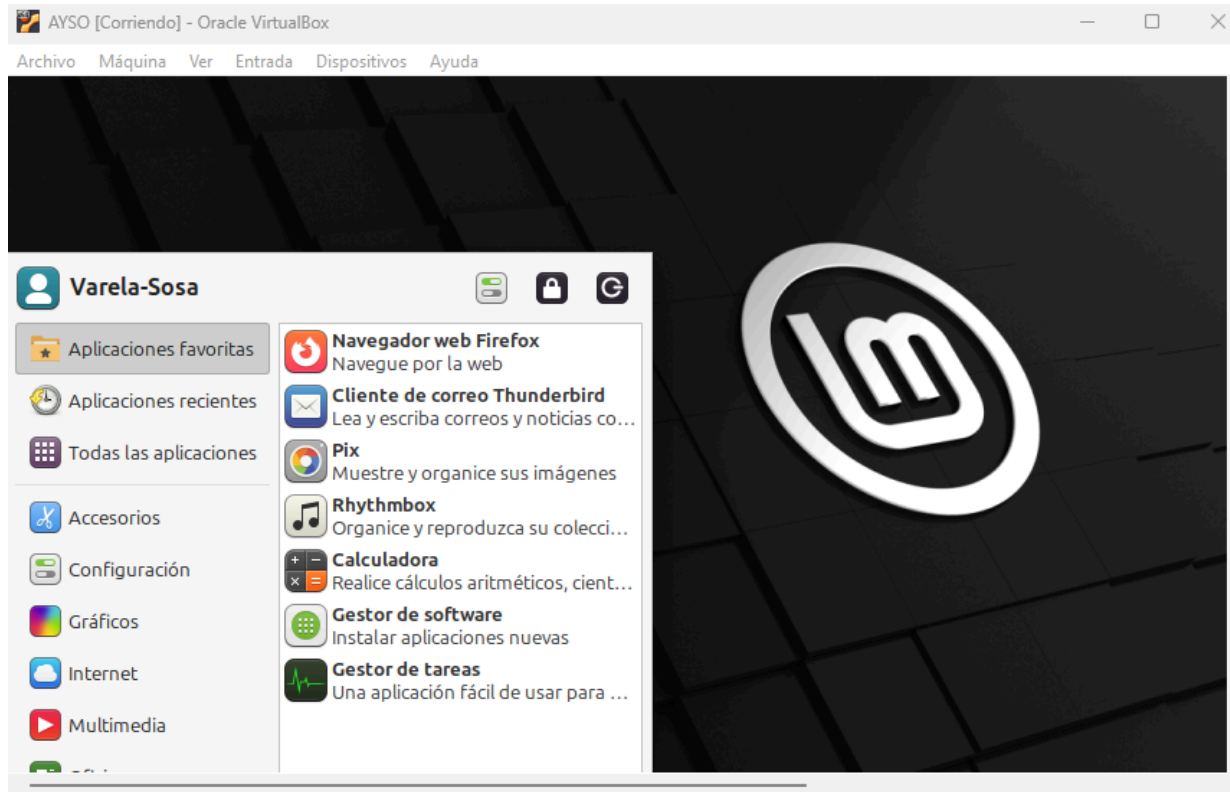


### **Validación del funcionamiento**

Para validar el correcto funcionamiento del entorno virtualizado propuesto, se definieron una serie de criterios observables y verificables que confirman la instalación exitosa de los componentes involucrados y el cumplimiento de los objetivos del caso práctico. Estos criterios fueron evaluados al finalizar cada etapa principal del procedimiento.

#### **1. Inicio correcto de la máquina virtual**

- La máquina virtual fue creada en VirtualBox e inicia sin errores ni bloqueos.
- El sistema operativo Linux Mint XFCE es cargado y operativo desde el entorno virtualizado.
- Se verifica la interacción fluida con la interfaz gráfica de Linux Mint.



## 2. Reconocimiento y uso de recursos

- Instalamos `htop` desde los repositorios de Ubuntu para una mejor precisión del seguimiento en el uso de los recursos.
- Se comprueba que el sistema operativo invitado reconoce los recursos asignados: CPU, memoria RAM, almacenamiento y conexión de red.
- Desde `htop` o el administrador de tareas por defecto de Linux Mint se visualiza el uso de CPU y memoria en tiempo real.

```

Terminal - varela-sosa@varelasosa-VirtualBox: ~
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
^Cvarela-sosa@varelasosa-VirtualBox:~$ sudo apt install htop
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  htop
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 270 no actualizados.
Se necesita descargar 171 kB de archivos.
Se utilizarán 434 kB de espacio de disco adicional después de esta operación.
Des:1 http://archive.ubuntu.com/ubuntu noble/main amd64 htop amd64 3.3.0-4build1
[171 kB]
Descargados 171 kB en 54s (3.174 B/s)
Seleccionando el paquete htop previamente no seleccionado.
(Leyendo la base de datos ... 456630 ficheros o directorios instalados actualmen
te.)
Preparando para desempaquetar .../htop_3.3.0-4build1_amd64.deb ...
Desempaquetando htop (3.3.0-4build1) ...
Configurando htop (3.3.0-4build1) ...
Procesando disparadores para mailcap (3.70+nmulubuntu1) ...
Procesando disparadores para desktop-file-utils (0.27-2build1) ...
Procesando disparadores para hicolor-icon-theme (0.17-2) ...
Procesando disparadores para gnome-menus (3.36.0-1.1ubuntu3) ...
Procesando disparadores para man-db (2.12.0-4build2) ...
varela-sosa@varelasosa-VirtualBox:~$ sudo apt update

```

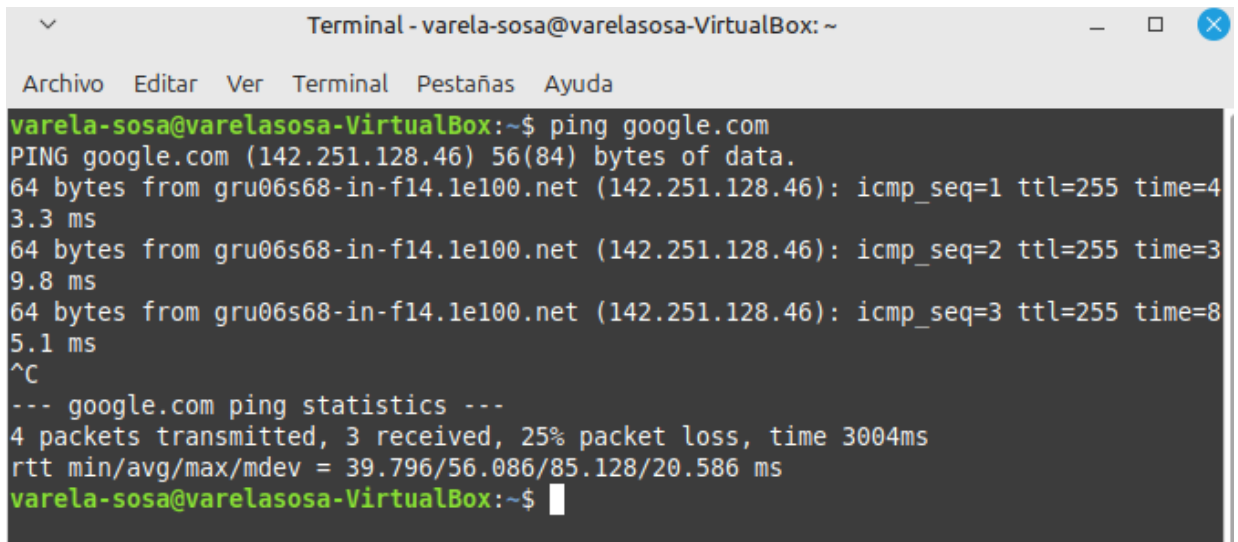
```

AYSO [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
Main  I/O
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
4269 varela-sos 20 0 14176 5120 3584 R 2.0 0.3 0:00.64 htop
1508 root 20 0 348M 100M 52104 S 0.7 5.1 0:08.68 /usr/lib/xorg/Xorg -core :0 -seat seat0
1 root 20 0 22620 13520 9424 S 0.0 0.7 0:01.67 /sbin/init splash
262 root 19 -1 50932 17056 15616 S 0.0 0.8 0:00.19 /usr/lib/systemd/systemd-journald
326 root 20 0 29928 7920 4848 S 0.0 0.4 0:00.22 /usr/lib/systemd/systemd-udevd
457 systemd-re 20 0 21576 13056 10752 S 0.0 0.6 0:00.09 /usr/lib/systemd/systemd-resolved
458 systemd-ti 20 0 91212 7936 7040 S 0.0 0.4 0:00.04 /usr/lib/systemd/systemd-timesyncd
463 systemd-ti 20 0 91212 7936 7040 S 0.0 0.4 0:00.00 /usr/lib/systemd/systemd-timesyncd
577 root 20 0 308M 8000 7232 S 0.0 0.4 0:00.03 /usr/libexec/accounts-daemon
580 avahi 20 0 8608 4352 3968 S 0.0 0.2 0:00.02 avahi-daemon: running [varelasosa-Virtu
582 root 20 0 12040 2688 2560 S 0.0 0.1 0:00.00 /usr/sbin/cron -f -P
583 messagebus 20 0 11124 6656 4480 S 0.0 0.3 0:00.77 @dbus-daemon --system --address=systemd
589 root 20 0 82920 3968 3712 S 0.0 0.2 0:00.11 /usr/sbin/irqbalance
604 root 20 0 82920 3968 3712 S 0.0 0.2 0:00.00 /usr/sbin/irqbalance
605 polkitd 20 0 384M 12412 8016 S 0.0 0.6 0:00.36 /usr/lib/polkit-1/polkitd --no-debug
606 root 20 0 308M 7552 6912 S 0.0 0.4 0:00.01 /usr/libexec/power-profiles-daemon
608 root 20 0 305M 6784 6272 S 0.0 0.3 0:00.01 /usr/libexec/switcheroo-control
609 root 20 0 18132 9092 7808 S 0.0 0.5 0:00.12 /usr/lib/systemd/systemd-logind
611 root 20 0 458M 13624 11320 S 0.0 0.7 0:00.03 /usr/libexec/udisks2/udisksd
614 root 20 0 308M 8000 7232 S 0.0 0.4 0:00.00 /usr/libexec/accounts-daemon
615 root 20 0 308M 8000 7232 S 0.0 0.4 0:00.05 /usr/libexec/accounts-daemon
616 avahi 20 0 8420 1288 1024 S 0.0 0.1 0:00.00 avahi-daemon: cchroot helper
618 root 20 0 308M 7552 6912 S 0.0 0.4 0:00.00 /usr/libexec/power-profiles-daemon
619 root 20 0 308M 7552 6912 S 0.0 0.4 0:00.00 /usr/libexec/power-profiles-daemon
620 root 20 0 305M 6784 6272 S 0.0 0.3 0:00.00 /usr/libexec/switcheroo-control

```

### 3. Conectividad

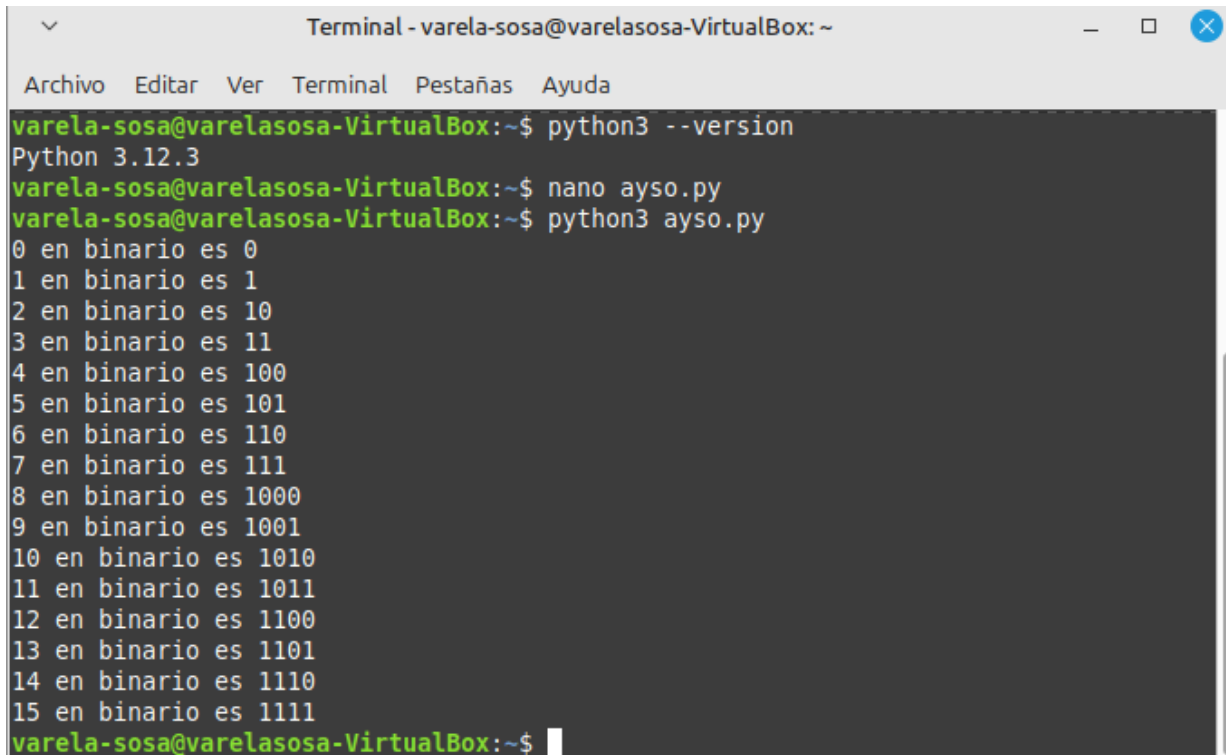
- Se realiza un ping desde la terminal hacia un sitio externo (por ejemplo: ping google.com) para confirmar la conectividad a internet.



```
Terminal - varela-sosa@varelasosa-VirtualBox: ~
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
varela-sosa@varelasosa-VirtualBox:~$ ping google.com
PING google.com (142.251.128.46) 56(84) bytes of data.
64 bytes from gru06s68-in-f14.1e100.net (142.251.128.46): icmp_seq=1 ttl=255 time=43.3 ms
64 bytes from gru06s68-in-f14.1e100.net (142.251.128.46): icmp_seq=2 ttl=255 time=39.8 ms
64 bytes from gru06s68-in-f14.1e100.net (142.251.128.46): icmp_seq=3 ttl=255 time=85.1 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3004ms
rtt min/avg/max/mdev = 39.796/56.086/85.128/20.586 ms
varela-sosa@varelasosa-VirtualBox:~$
```

### 4. Instalación y ejecución de software

- Se instala Python. En Linux Mint viene preinstalado, aunque con limitaciones para la instalación de librerías pip para evitar conflictos con apt.
- Se ejecuta correctamente el siguiente script en Python:  
[https://github.com/santiagovOK/UTN-TUPaD-P1/blob/main/mate\\_semanaIntegracion1/tp\\_semana-integracion-01.py](https://github.com/santiagovOK/UTN-TUPaD-P1/blob/main/mate_semanaIntegracion1/tp_semana-integracion-01.py)

A screenshot of a terminal window titled "Terminal - varela-sosa@varelasosa-VirtualBox: ~". The window has a menu bar with "Archivo", "Editar", "Ver", "Terminal", "Pestañas", and "Ayuda". The terminal shows the following commands and output:

```
varela-sosa@varelasosa-VirtualBox:~$ python3 --version
Python 3.12.3
varela-sosa@varelasosa-VirtualBox:~$ nano ayso.py
varela-sosa@varelasosa-VirtualBox:~$ python3 ayso.py
0 en binario es 0
1 en binario es 1
2 en binario es 10
3 en binario es 11
4 en binario es 100
5 en binario es 101
6 en binario es 110
7 en binario es 111
8 en binario es 1000
9 en binario es 1001
10 en binario es 1010
11 en binario es 1011
12 en binario es 1100
13 en binario es 1101
14 en binario es 1110
15 en binario es 1111
varela-sosa@varelasosa-VirtualBox:~$
```

### 5. Estabilidad general

La VM se mantiene estable durante la ejecución del sistema y del software de prueba. No se presentan errores críticos ni cierres inesperados.

### 6. Registro de evidencia

- Se generan y documentan capturas de pantalla para cada una de las etapas de validación mencionadas.

La gran mayoría de la evidencia está disponible en el repositorio de GitHub<sup>3</sup>, incluyendo el paso a paso del proceso de instalación; las comprobaciones funcionales de cada uno de los pasos; el código fuente utilizado y, por último, algunas observaciones de todo el proceso.

---

<sup>3</sup> URL Repositorio en GitHub: [https://github.com/xms44/ayso\\_integrador\\_VarelaSosa/tree/main](https://github.com/xms44/ayso_integrador_VarelaSosa/tree/main)



#### 4. Metodología Utilizada

##### Investigación previa

En cuanto a la investigación previa y búsqueda de fuentes, dados los tiempos, además de basarnos en el material de cátedra proporcionado, fueron planteadas búsquedas a partir de los aspectos conceptuales de la virtualización, desde los cuales parte nuestra idea para este trabajo práctico (que no son todos los que están presentes en la unidad, sino una parte).

En primer lugar, se pensó en la documentación oficial, principalmente la de VirtualBox que, como la gran mayoría del software de estas características, cuenta con documentación oficial en su página web. También es el caso de VMware y de empresas de hardware y software que publican entradas de blog sobre virtualización y temas relacionados.

Por otro lado, la realidad es que las herramientas de búsqueda potenciadas con inteligencia artificial, como Perplexity o cualquiera de los grandes chatbots con su respectiva función de búsqueda y modelos (LLMs) corriendo detrás, son de gran ayuda frente a la búsqueda inducida desde lo conceptual. Aunque no siempre igualan la precisión de los buscadores tradicionales como Google, que indexan y devuelven enlaces directos, estas herramientas destacan en guiar búsquedas conceptuales mediante razonamientos más elaborados.

De allí fue tomado, por ejemplo, el libro *Operating System Concepts*, mientras que este trabajo se apoyó puntualmente en el capítulo *Virtual Machines* (Silberschatz, Galvin, & Gagne, 2018), con la finalidad de contar con un material algo más complejo y profundo en el cual fundamentar lo referido a los sistemas operativos en general.

En cuanto al caso y su puesta en práctica, no hubo grandes dificultades ya que ambos integrantes del grupo conocían con anterioridad, al menos a grandes rasgos, el uso de VirtualBox y las características de Linux Mint XFCE como sistema operativo Linux, entre otras distribuciones existentes. Si bien el entorno de escritorio principal de Linux Mint es Cinnamon, que a nivel de gasto de recursos se asemeja más a GNOME, la variante de XFCE fue seleccionada por ser mucho más adecuada en ese sentido para un entorno virtualizado, principalmente en cuanto a consumo de RAM y espacio ocupado en el disco duro.

### Etapas de diseño y prueba del código

Dado que el enfoque del trabajo práctico está centrado en la virtualización y no en el desarrollo de software, la etapa de prueba de código fue sencilla pero funcional. Se reutilizó un programa en Python previamente desarrollado en el marco del primer trabajo práctico de Matemática<sup>4</sup>, cuyo objetivo era simular un contador binario que imprime los números del 0 al 15 junto a su correspondiente conversión binaria, agregando un retardo de un segundo entre cada número para imitar el comportamiento de un circuito.

El código, si bien simple, es adecuado para verificar el funcionamiento del entorno virtualizado, ya que requiere del correcto funcionamiento del intérprete de Python, del sistema de archivos y de la salida estándar. Se eligió intencionalmente un programa ya probado, con el objetivo de centrarse en evaluar la capacidad de ejecución del sistema operativo virtualizado y no en aspectos algorítmicos complejos.

### Herramientas y recursos utilizados

Para el desarrollo del trabajo y como ya se ha visto en los apartados anteriores, se recurrió a un conjunto de herramientas que facilitaron tanto la elaboración del informe como la ejecución del caso práctico. Google Docs fue utilizado como plataforma principal de redacción colaborativa para la construcción del informe, permitiendo edición en tiempo real entre los integrantes del equipo.

En cuanto al entorno de desarrollo, se utilizó Visual Studio Code, tanto para la escritura del pequeño programa en Python como para tomar notas y redactar el archivo README.md incluido en el repositorio. Para la virtualización del sistema operativo, se eligió VirtualBox, uno de los Hypervisores Tipo 2 más conocidos y utilizados en entornos educativos, además de estar referenciado en los materiales de estudio de la asignatura.

---

<sup>4</sup>URL original del repositorio:

[https://github.com/santigovOK/UTN-TUPaD-P1/blob/main/mate\\_semanaIntegracion1/tp\\_semana-integracion-01.py](https://github.com/santigovOK/UTN-TUPaD-P1/blob/main/mate_semanaIntegracion1/tp_semana-integracion-01.py)

La máquina virtual se configuró con una ISO de Linux Mint XFCE, descargada desde su sitio oficial<sup>5</sup>, por tratarse de una distribución liviana, estable y basada en Ubuntu, lo cual facilitó la instalación y las pruebas.

Por último, se emplearon Git y GitHub para el control de versiones y la publicación del repositorio asociado al trabajo, garantizando el seguimiento ordenado de los avances y la disponibilidad pública de los materiales desarrollados.

### Reparto de tareas y trabajo colaborativo

En el desarrollo de este trabajo, ambas personas que conforman el grupo asumieron responsabilidades complementarias que permitieron llevar adelante cada etapa con claridad y eficiencia.

Santiago se encargó de definir el enfoque específico del tema dentro del marco general de la virtualización, iniciando también la búsqueda y organización de las fuentes teóricas utilizadas. Tomó la iniciativa en la creación de la estructura formal del documento, estableciendo tanto las pautas de formato como el esquema general del archivo de texto. Desarrolló un primer borrador de la introducción y del marco teórico, y posteriormente realizó tareas de revisión en cuanto a redacción, coherencia y ortografía. En relación al caso práctico, elaboró la descripción del problema y propuso los criterios de validación iniciales durante su ejecución, valiéndose de una instalación previa de VirtualBox con Linux Mint XFCE ya funcional. Además, profundizó en su parte del guión individual para el video, la presentación de fondo usando Whimsical<sup>6</sup> y fue el responsable de la creación, edición y publicación final del mismo.

Ximena, por su parte, se ocupó de registrar en detalle el desarrollo del caso práctico mediante capturas de pantalla. Estas imágenes incluyen tanto el proceso de instalación de VirtualBox y Linux Mint XFCE como la ejecución del código en Python desde el sistema virtualizado, así como comandos ilustrativos dentro del entorno (por ejemplo, htop, ping, etc.). También asumió la creación y organización del repositorio en GitHub, estableciendo su estructura general. A la par de esta tarea, realizó mejoras en el sistema de citado y en el desarrollo de secciones

---

<sup>5</sup> <https://linuxmint.com/edition.php?id=320>

<sup>6</sup>Whimsical es una aplicación de espacio de trabajo digital diseñada para capturar, dar forma y compartir ideas de manera colaborativa. Sitio oficial: <https://whimsical.com/>

clave del trabajo escrito, como el marco teórico, los resultados obtenidos y las conclusiones. Finalmente, fue quien redactó el guión general para el video de presentación, el cual luego cada integrante adaptó y amplió para su intervención personal.

## 5. Resultados Obtenidos

El desarrollo del caso práctico permitió cumplir con los objetivos planteados en torno a la virtualización de sistemas operativos y la ejecución de software dentro de entornos virtualizados. A través del uso de VirtualBox como hipervisor tipo 2 y la instalación de Linux Mint XFCE, logramos configurar exitosamente una máquina virtual funcional, ligera y estable en los dispositivos de ambos integrantes de este trabajo. La elección de esta distribución, como se viene reafirmando, se justificó tanto por su bajo consumo de recursos como por su base en Ubuntu, que facilita la compatibilidad y la instalación de herramientas.

Durante el proceso de instalación y configuración de la máquina virtual, se verificaron aspectos clave como la asignación de recursos (memoria RAM, CPU, almacenamiento), gracias a las herramientas que vienen preinstaladas en la distribución, o que bien pueden instalarse a través de los repositorios de Ubuntu.

Una vez creado y configurado el entorno, se procedió a instalar Visual Studio Code y ejecutar Python dentro de él, que ya venía preinstalado. Esto permitió replicar condiciones similares a un entorno de desarrollo real. En este entorno virtual se ejecutó exitosamente un pequeño programa en Python, el contador binario, desarrollado en una instancia previa. La correcta ejecución del programa evidenció que la máquina virtual disponía de acceso al sistema de archivos y podía gestionar procesos, confirmando así el funcionamiento adecuado del sistema operativo invitado.

Este resultado valida no sólo la instalación técnica del sistema, sino también su operatividad para tareas básicas de programación. Además, a través de la utilización de herramientas como Git y GitHub, se gestionó el repositorio del proyecto de forma ordenada y transparente. El informe fue redactado colaborativamente en Google Docs, lo que facilitó la organización del contenido y la distribución de tareas entre integrantes del equipo.

## 6. Conclusiones

Realizar este trabajo permitió a los integrantes de este grupo consolidar y profundizar varios conceptos clave sobre virtualización, tanto desde una perspectiva técnica como teórica. Se aprendió aún más sobre diseñar e implementar un entorno virtual completo que simula un escenario real de trabajo, desde la instalación de una distribución GNU/Linux hasta la ejecución de código en Python dentro de ese entorno. A su vez, fueron reforzadas habilidades vinculadas al uso de herramientas de desarrollo (VS Code, Git/GitHub) y a la organización colaborativa del trabajo mediante plataformas como Google Docs. La búsqueda de información teórica también nos llevó a reflexionar sobre cómo se construyen los conocimientos técnicos: no sólo desde la documentación oficial y los manuales, sino también desde enfoques guiados más conceptualmente por herramientas modernas basadas en modelos de lenguaje.

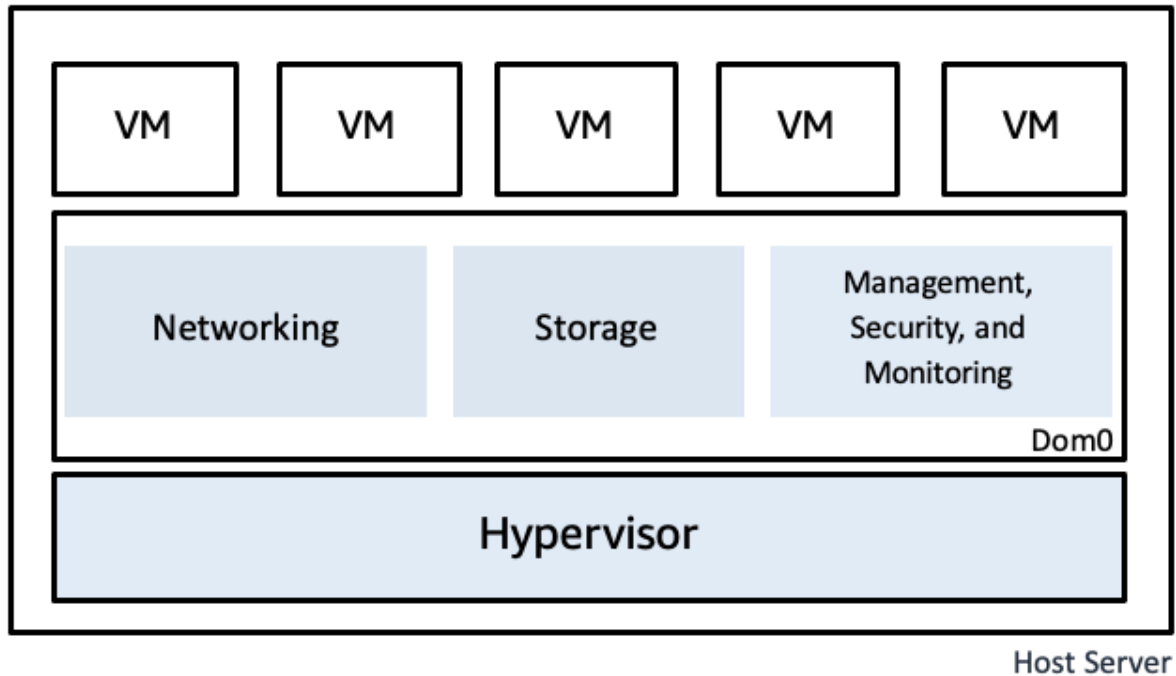
Como se viene mencionando, dado que ambos integrantes de este grupo manejaban estas herramientas y la virtualización de distribuciones Linux en general, no hubo grandes dificultades para desarrollar el caso práctico. Quizás es pertinente mencionar que, si bien la mayoría de las capturas de pantalla son de VirtualBox usado sobre Windows como sistema anfitrión, también fue ejecutado VirtualBox sobre Debian 12 Openbox con tint2 como panel para este caso. Dado que este segundo equipo no contaba con espacio en su disco duro SSD NVMe principal, el archivo .vdi fue alojado en un disco duro mecánico (HDD) secundario, lo que perjudicaba el rendimiento del SO virtualizado ligeramente. Es por eso que este detrimento fue compensado con una mayor asignación de recursos en CPU (de 2 a 4 núcleos) y RAM (de 2 a 4 GB) para evitar problemas.

Por último, entre las posibles mejoras o extensiones que se podrían considerar, se encuentra la incorporación de otros Hypervisores para comparar rendimiento y usabilidad (por ejemplo, VMware o alguno de Tipo 1). También sería interesante desarrollar programas en Python que interactúen más directamente con recursos del sistema (como el uso de múltiples hilos o acceso a archivos), para así testear aún más las capacidades del sistema operativo virtualizado. Por último, una instancia futura podría incluir la virtualización anidada, o incluso pruebas de sistemas operativos más exigentes o con otros fines, como Ubuntu Server.

## 7. Bibliografía

- Amazon Web Services. (s.f.). *¿Cuál es la diferencia entre los hipervisores de tipo 1 y de tipo 2?* AWS. Recuperado el 30 de mayo de 2025, de <https://aws.amazon.com/es/compare/the-difference-between-type-1-and-type-2-hypervisors/>
- IBM. (s.f.). *¿Qué es la virtualización?* IBM. Recuperado el 27 de mayo de 2025, de <https://www.ibm.com/mx-es/topics/virtualization>
- Oracle. (s.f.). *VirtualBox User Manual*. VirtualBox. Recuperado el 27 de mayo de 2025, de <https://www.virtualbox.org/manual/>
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Virtual machines. En Operating system concepts (10.<sup>a</sup> ed., pp. 701–733). Wiley.*
- *Tecnicatura Universitaria en Programación a Distancia. Arquitectura y Sistemas Operativos. Unidad 7: Virtualización* (2025). (1° ed.). Universidad Tecnológica Nacional.
- VMware. (s.f.). *What is a hypervisor?* VMware. Recuperado el 27 de mayo de 2025, de <https://www.vmware.com/topics/glossary/content/hypervisor>

## 8. Anexos



Este esquema, recuperado del sitio de Amazon Web Services (AWS, s.f), representa la arquitectura de un hipervisor tipo 1 (bare-metal), que se ejecuta directamente sobre el hardware del servidor. Encima del hipervisor se encuentran las máquinas virtuales (VMs), gestionadas a través de una partición privilegiada llamada Dom0, la cual se encarga de funciones críticas como red, almacenamiento y administración del sistema.