

# **Trabajo integrador**

## **Virtualización con VirtualBox**



**Alumnos:**

Villarruel Joaquin - joaquinvillarrul04@gmail.com

Silveira Santiago - theisas@hotmail.com

**Materia:** Arquitectura y Sistemas Operativos

**Profesor:** Martín Aristiaran

**Fecha de entrega:** 5 de junio de 2025

# Índice

Introducción.....	3
Marco Teórico.....	4
Tipo 1 – Nativo.....	4
Tipo 2 – Alojado.....	5
virtualización de la CPU.....	5
virtualización de la Memoria.....	6
Virtualización del Almacenamiento.....	6
Virtualización de la Red.....	6
Virtualización de Dispositivos:.....	6
Otros recursos de entornos virtualizados.....	8
Imágenes ISO.....	8
Snapshots:.....	8
Contenedores.....	8
Caso Práctico.....	9
Descarga de VirtualBox.....	10
Instalación de VirtualBox.....	10
Descarga de Ubuntu.....	11
Creación de la máquina virtual e Instalación de Ubuntu.....	12
Verificación de Python en Ubuntu.....	15
Instalación Visual Studio Code.....	15
Desarrollo y prueba de código.....	16
metodología utilizada.....	16
Resultados obtenidos.....	17
Conclusiones.....	18
Bibliografía.....	18

# Introducción

La virtualización es una herramienta muy útil, la cual permite ejecutar múltiples sistemas operativos sobre una misma computadora, con la ayuda de un software dedicado el cual permite gestionar los recursos del sistema. A lo largo del trabajo se analizará y aplicará esta tecnología.

En este informe se analizarán dos tipos de virtualización, la de tipo 1 que se ejecuta directamente sobre el hardware y la de tipo 2 que funciona sobre un sistema operativo anfitrión. La práctica será realizada utilizando la virtualización de tipo 2 con la ayuda de la herramienta VirtualBox.

Estos conceptos de virtualización son importantes para la formación de un técnico en programación ya que una de las características es que permite crear un entorno seguro de desarrollo y pruebas, sin afectar al sistema principal.

El objetivo del trabajo será profundizar en los conceptos básicos de la virtualización y realizar una experiencia práctica la cual incluirá la instalación de VirtualBox, la creación de una máquina virtual con Ubuntu y la utilización de esta para el desarrollo y ejecución de un código sencillo en Python.

## Marco Teórico

La virtualización permite abstraer recursos físicos del sistema utilizando software especializado esto nos da la posibilidad de crear entornos virtuales independientes. Estos entornos tienen la capacidad de alojar sistemas como si estos estuvieran en el hardware real.

Antes de continuar es necesario comprender cómo funcionan las máquinas virtuales, estas son como un sandbox, es decir un entorno aislado en el cual se puede ejecutar software, instalar sistemas operativos o realizar pruebas sin afectar al sistema anfitrión. El aislamiento es uno de los conceptos más importantes de la virtualización debido a que es el que nos permite trabajar en estos entornos seguros y controlados.

Uno de los principales componentes de la virtualización es el **hypervisor o hipervisor**, Este es el encargado de crear y gestionar las máquinas virtuales. Lo realiza actuando como intermediario entre el hardware físico y los sistemas operativos invitados, se ocupa de asignar y controlar el uso de la memoria, almacenamiento, CPU y demás recursos.

Tenemos dos tipos principales de hipervisores, los de tipo 1 y los de tipo 2.

### Tipo 1 – Nativo

Estos se instalan directamente sobre el hardware, es decir no dependen de un sistema anfitrión. Por su arquitectura nos dan un mejor rendimiento y mayor seguridad. Son utilizados comúnmente en servidores y entornos de producción.

Un ejemplo de esto es **Microsoft Hyper-V** el cual es el producto de virtualización proporcionado por Microsoft. Esta tecnología se puede encontrar en versiones de Windows Server, y en los Windows 10 y 11 profesionales. Brinda la posibilidad de ejecutar múltiples máquinas virtuales en forma simultánea, es compatible con arquitecturas x64, además cuenta con soporte para redes y discos virtuales y gestión remota. Una de sus mas grandes ventajas es la integración nativa con Windows.

## Tipo 2 – Alojado

Se instala como una aplicación dentro de un sistema operativo, el cual será conocido como anfitrión. Esto agrega una capa más entre el hipervisor y el hardware lo que podría derivar en un menor rendimiento, a su vez esto puede ser compensado con la facilidad de su instalación y uso.

Al iniciar una máquina virtual con un hipervisor de tipo 2 está realizará las solicitudes de hardware a través de dicho hipervisor, el cual se las va a transmitir al sistema operativo anfitrión y este finalmente al hardware, añadiendo de esta forma una cierta latencia.

Este tipo de hipervisor utiliza APIs y servicios del sistema anfitrión para gestionar los recursos que necesita la máquina virtual los cuales son la memoria, el almacenamiento y la CPU, por ejemplo, para lograr la virtualización de estas utiliza las siguientes técnicas:

### virtualización de la CPU

- **Traducción binaria:** Es utilizada en la virtualización de la CPU, interpreta y traduce las instrucciones privilegiadas para que de esta forma las máquinas virtuales no accedan al hardware físico. Las instrucciones privilegiadas son las que tienen control directo sobre el hardware, como podría ser modificar algún registro del procesador o acceder a las interrupciones. En caso de ejecutarse sin control en una máquina virtual podría llegar a afectar al sistema anfitrión o a otras máquinas virtuales.

Un ejemplo puede ser el registro CR3, el cual contiene la dirección de base de la tabla de páginas, la cual es fundamental para la gestión de memoria virtual. Si una máquina virtual lo intenta modificar la instrucción será interceptada y redirigida al sistema anfitrión, simulando el cambio sin permitir acceso al hardware físico.

- **virtualización asistida por Hardware:** En los procesadores modernos de AMD e Intel podemos encontrar las tecnologías **Intel VT-x** y **AMD-V**, estas dividen el procesador en contextos separados para cada máquina virtual, esto permite que las instrucciones sensibles se ejecuten sin la necesidad de una traducción. Esta tecnología es especialmente útil para mejorar el rendimiento de las máquinas virtuales, evitando la sobrecarga que genera la traducción binaria, permitiendo una ejecución más directa y eficiente.

## virtualización de la Memoria

- **Tablas de Páginas Sombreadas:** Se mapean direcciones virtuales a direcciones físicas para cada máquina virtual, para así asegurar un aislamiento entre ellas. Por ejemplo, si una máquina virtual accede a la dirección 0x1000, el hipervisor traduce esa dirección a una física distinta, la cual podría ser 0x2000.
- **SLAT:** Esta tecnología delega la traducción al hardware, de esta forma reduciendo la carga del hipervisor y mejorando el rendimiento. En el caso de Intel esta se llama **EPT (Extended Page Tables)** y en AMD se la conoce como **NPT (Nested Page Tables)**.

## Virtualización del Almacenamiento

- **Discos virtuales:** Son archivos que sirven como discos duros para las máquinas virtuales. Contienen el sistema de archivos y el sistema operativo invitado, en el caso de VirtualBox el archivo tendrá formato .vdi.
- **Controladores Emulados:** El hipervisor simula interfaces de almacenamiento como IDE o SATA para que el sistema operativo invitado pueda comunicarse con el disco virtual como si fuera uno real.

## Virtualización de la Red

- **Adaptadores Virtuales:** Simulan interfaces de red físicas, permiten que las máquinas virtuales se conecten a redes externas. El adaptador virtual puede conectarse de distintos modos dependiendo de las necesidades, estos son **NAT (Network Address Translation)** en el cual las máquinas comparten la dirección IP del sistema anfitrión, **Red en puente (Bridged Networking)** la máquina virtual obtiene una dirección IP en la misma red física que el anfitrión. **Red interna (Internal Network)** Las máquinas virtuales pueden comunicarse entre sí, pero no tienen acceso al sistema anfitrión, **Solo anfitrión (Host-only)** La máquina virtual se comunica solo con el sistema anfitrión.

## Virtualización de Dispositivos:

- **Dispositivos emulados:** El hipervisor simula el hardware físico, para que de esta forma las máquinas virtuales puedan funcionar como si estuvieran conectadas a un dispositivo real.

- **Passthrough:** Algunas veces el hipervisor permite que una máquina virtual acceda a los dispositivos físicos del anfitrión. Esto se utiliza cuando se requiere un alto rendimiento.

Un ejemplo de hipervisor de tipo 2 es **VirtualBox**, el cual es gratuito, de código abierto y compatible con Windows, Linux y MacOS.

En el siguiente cuadro se resumen las diferencias, desventajas y beneficios de cada tipo de hipervisor.

Criterio	Tipo 1 – Nativo	Tipo 2 – Alojado
Instalación	Se instala sobre el hardware físico, requiere conocimientos técnicos	Se instala como aplicación en un sistema operativo, es fácil de instalar y usar
Rendimiento	Alto, debido a que accede directamente al hardware	Menor, tiene al sistema anfitrión en el medio
Complejidad de uso	Configuración más compleja	Fácil de administrar e intuitivo
Uso de recursos	Eficiencia alta, no hay consumo de otro sistema	Baja eficiencia, el sistema anfitrión consume parte de los recursos
Escalabilidad	Alta, se pueden administrar grandes cantidades de VMs	Limitada, depende del hardware del host
Seguridad	Ofrece un aislamiento mayor entre las máquinas virtuales	Expuesto a fallos por parte del sistema anfitrión
Casos de uso	Servidores, centro de datos	Entornos de desarrollo o pruebas, educación
Ejemplos	VMware ESXi, Microsoft Hyper-V, Xen	VirtualBox, VMware Workstation

## Otros recursos de entornos virtualizados

### Imágenes ISO

Es un archivo el cual contiene una copia exacta de un sistema de archivos, se utiliza para representar discos ópticos como CD o DVD. En el caso de la virtualización las imágenes ISO sirven para instalar sistemas operativos en la máquina virtual, simulando un medio de instalación físico. Si quisiéramos instalar Ubuntu en una máquina virtual debemos montar la imagen ISO en la unidad óptica virtual y se inicia desde allí el proceso de instalación.

### Snapshots:

Es una copia del estado actual de la máquina virtual, la misma incluye el contenido de la memoria, el estado del disco y la configuración del sistema. Esto nos permite guardar un punto de restauración al que podremos volver si nos encontramos con un error o con un cambio no deseado. Suelen ser muy útiles para entornos de prueba o desarrollo debido a que permiten experimentar sin riesgo.

A pesar de ser una herramienta muy útil, las snapshots deben ser utilizadas con un poco de precaución ya que crear varias podría ocupar mucho espacio y reducir el rendimiento de la máquina virtual. Una vez que no sean necesarios es recomendable eliminarlos.

## Contenedores

Además de las máquinas virtuales existen otras tecnologías que nos permiten aislar aplicaciones de forma liviana, sin tener que virtualizar un sistema operativo completo. Una de las más conocidas y utilizadas es **Docker**, la cual funciona mediante contenedores.

Un contenedor es un entorno aislado en la cual se ejecuta una aplicación junto a sus dependencias, pero con el mismo kernel del sistema anfitrión. Una de las diferencias que tiene con las máquinas virtuales es que estos no requieren de un hipervisor ni de un sistema operativo invitado completo.

Los contenedores no tienen el mismo aislamiento que una máquina virtual, pero su eficiencia y simplicidad compensan para convertirlos en una alternativa bastante utilizada.



A continuación, una tabla comparativa entre contenedores y máquinas virtuales:

Criterio	Contenedores	Máquinas Virtuales
Nivel de aislamiento	Bajo, comparten kernel del sistema anfitrión	Alto, cada VM tiene su propio sistema operativo
Hipervisor	No requiere	Requiere un hipervisor (Tipo 1 o 2)
Sistema Operativo Invitado	No incluye un sistema operativo completo	Cada VM ejecuta un sistema operativo completo
Uso de recursos	Bajo	Alto
Portabilidad	Alta, los contenedores pueden ejecutarse en cualquier sistema Docker	Menor portabilidad, depende del hipervisor y formato del disco
Aplicaciones típicas	Microservicios, pruebas rápidas	Simulación de entornos completos, servidores virtuales, pruebas complejas
Ejemplos de herramientas	Docker, Podman	VirtualBox, VMware

## Caso Práctico

El objetivo de este caso será utilizar los conceptos del marco teórico para crear y configurar una máquina virtual mediante **VirtualBox**, instalando el sistema operativo **Ubuntu**. Una vez hecho esto se utilizará como entorno para el desarrollo y prueba de un código sencillo en Python. El programa creará una lista y solicitará números al usuario para llenarla, luego calculará el promedio y finalmente lo mostrará por pantalla.

## Descarga de VirtualBox



Lo primero que se debe hacer es descargar **VirtualBox** desde la [página oficial](#).

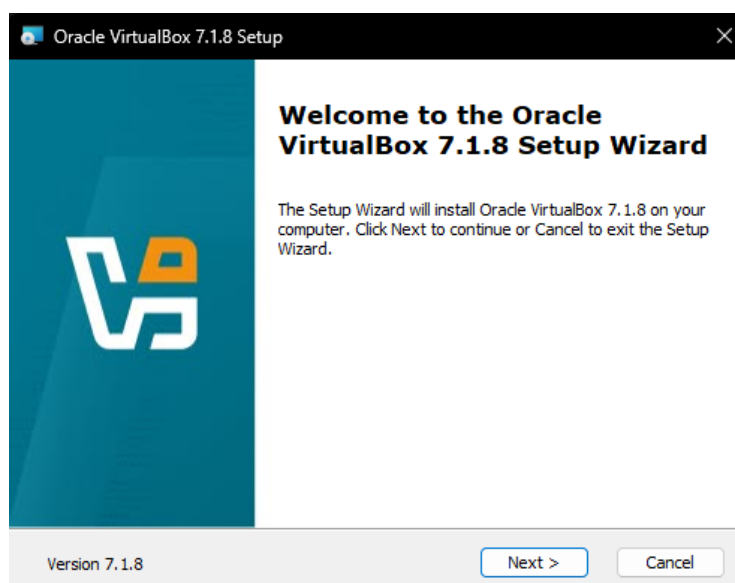


Una vez en la página de descarga debemos seleccionar la versión compatible con nuestro sistema operativo anfitrión, en este caso Windows 64 bits. Además, descargamos el paquete de extensión de VirtualBox.

## Instalación de VirtualBox

Una vez que descargamos ambos archivos abrimos primero el asistente de instalación de VirtualBox, el primer archivo que se ve en la foto.

Nombre	Fecha de modificación	Tipo	Tamaño
 VirtualBox-7.1.8-168469-Win.exe	2/6/2025 19:01	Aplicación	121.642 KB
 Oracle_VirtualBox_Extension_Pack-7.1.8.v...	2/6/2025 19:02	VirtualBox Extensi...	22.436 KB



Para terminar la instalación se debe seguir los pasos y aceptar los términos y condiciones.

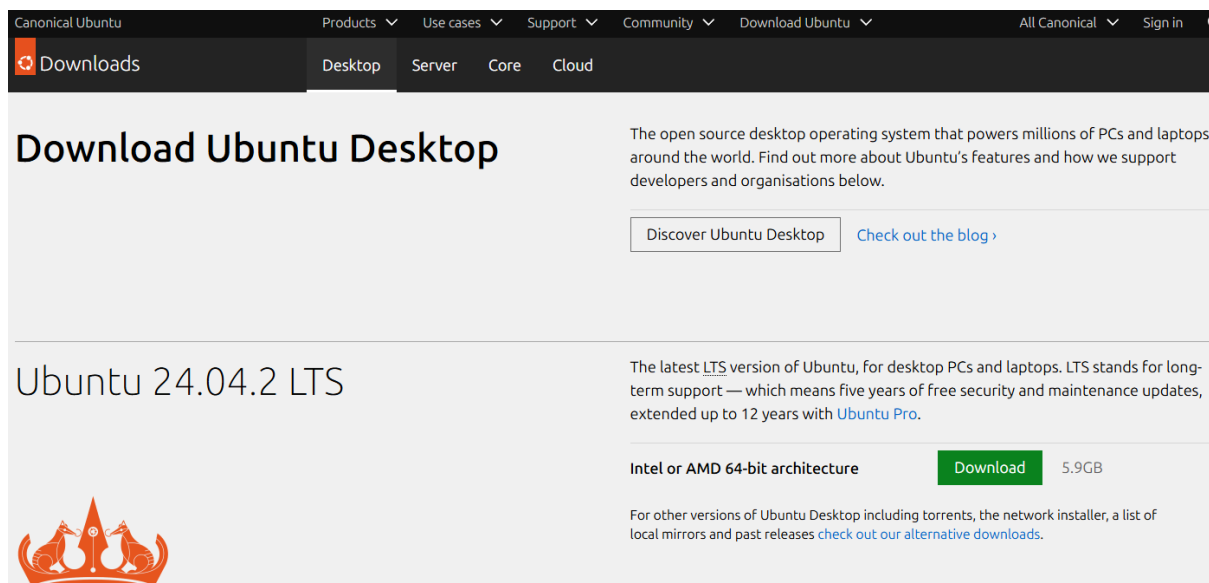
Una vez terminado ese proceso abrimos el pack de extensión y lo instalamos.



Una vez realizados estos pasos ya tenemos VirtualBox listo para instalar un sistema operativo.

## Descarga de Ubuntu

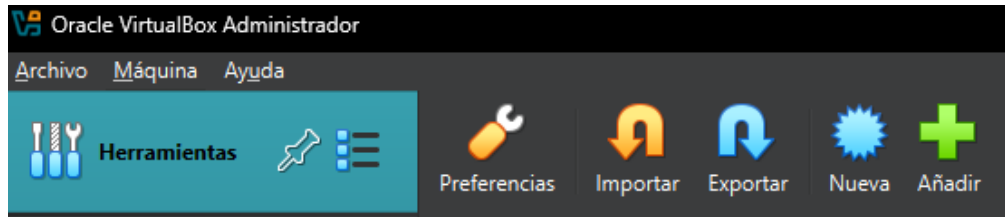
Descargamos la última versión de Ubuntu desde su [página oficial](#).



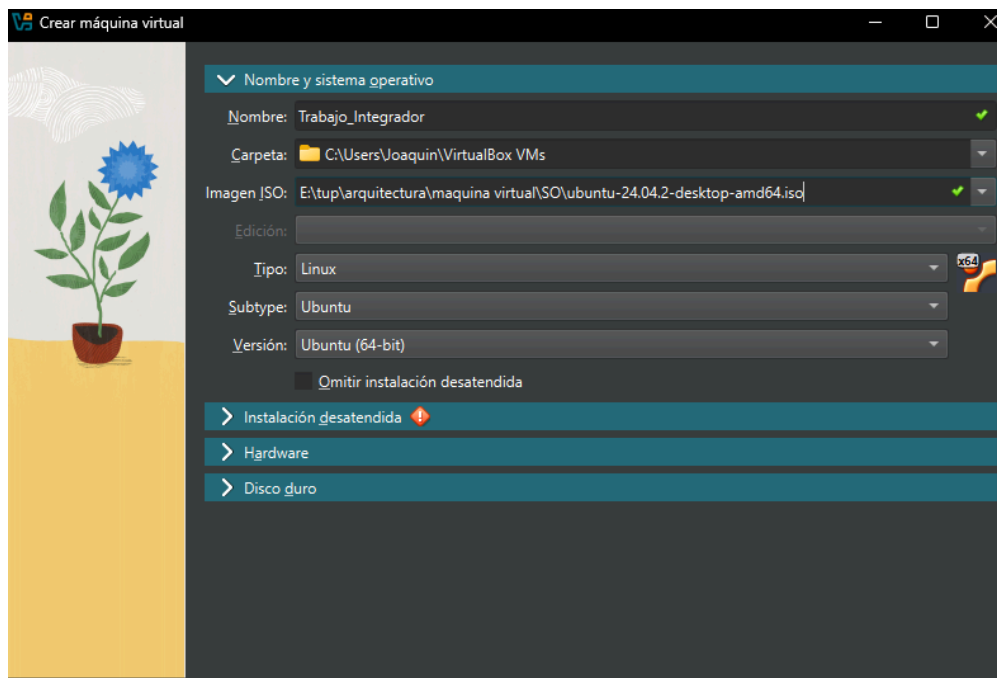
La página automáticamente nos ofrece la versión compatible con la arquitectura de nuestro procesador.

## Creación de la máquina virtual e Instalación de Ubuntu

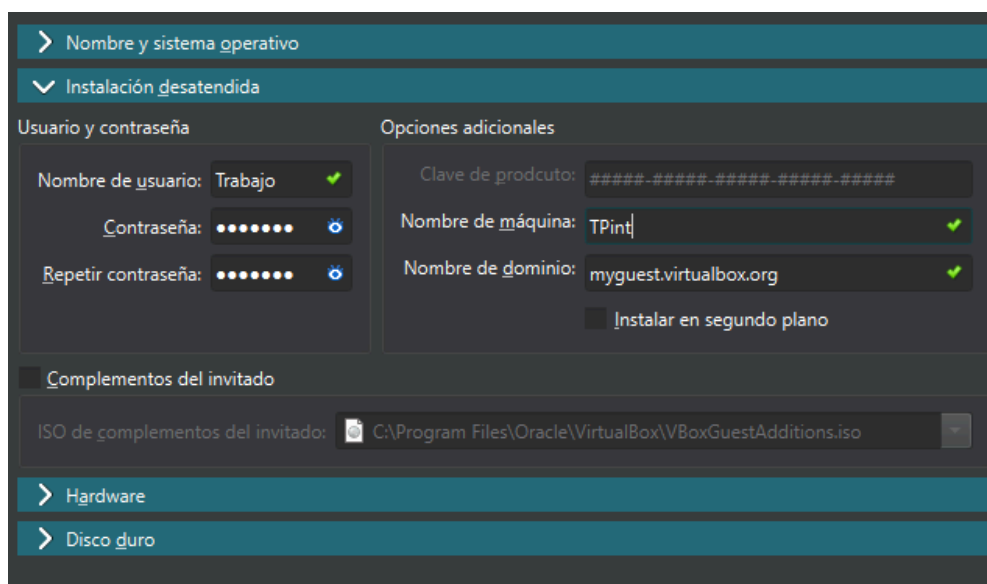
En la barra de opciones superior debemos hacer clic en **Nueva**.



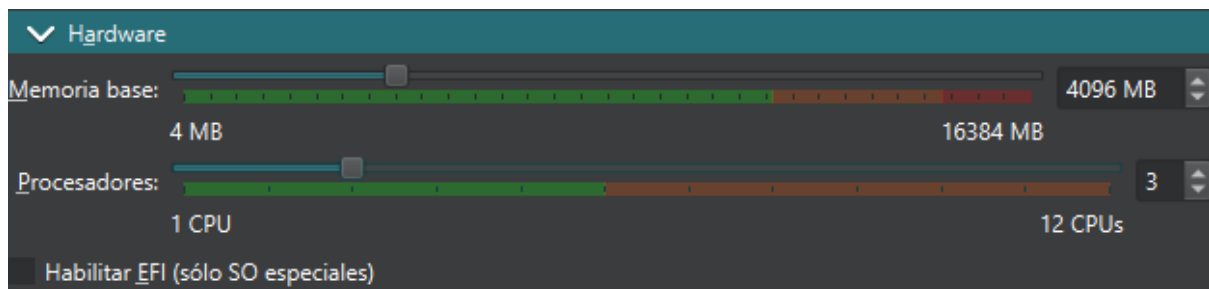
Luego le asignamos un nombre a la máquina virtual y seleccionamos nuestra imagen ISO.



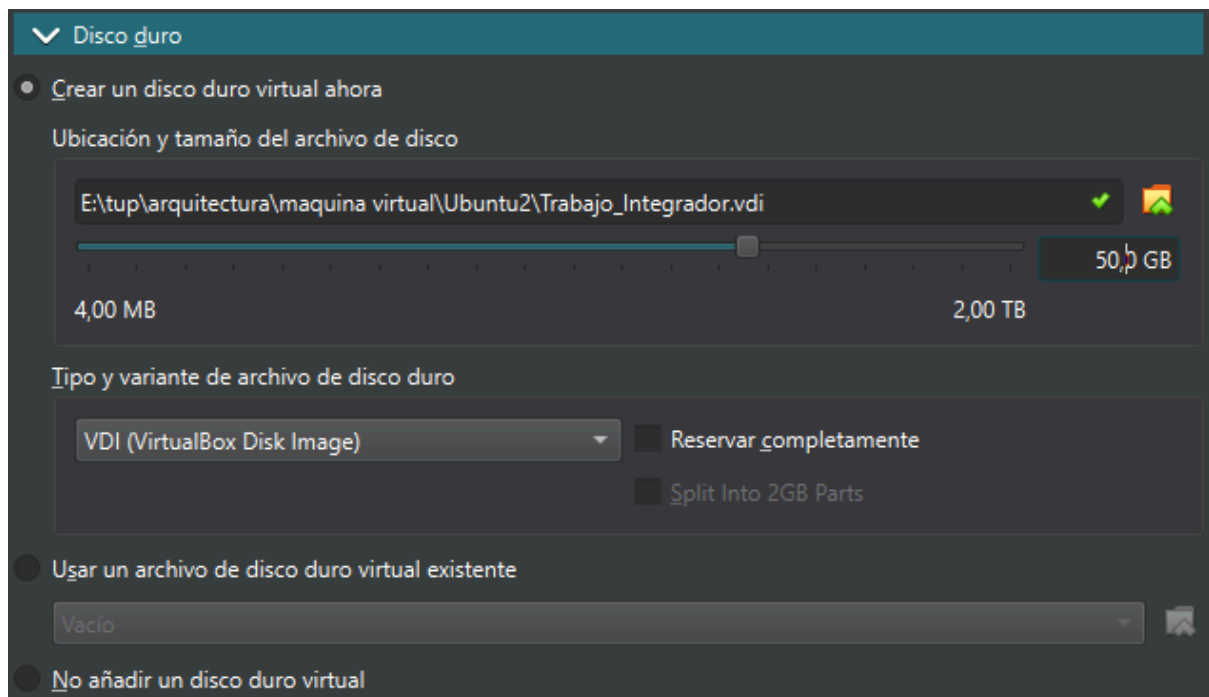
En el menú de **instalación desatendida** asignamos el nombre de usuario y la contraseña.



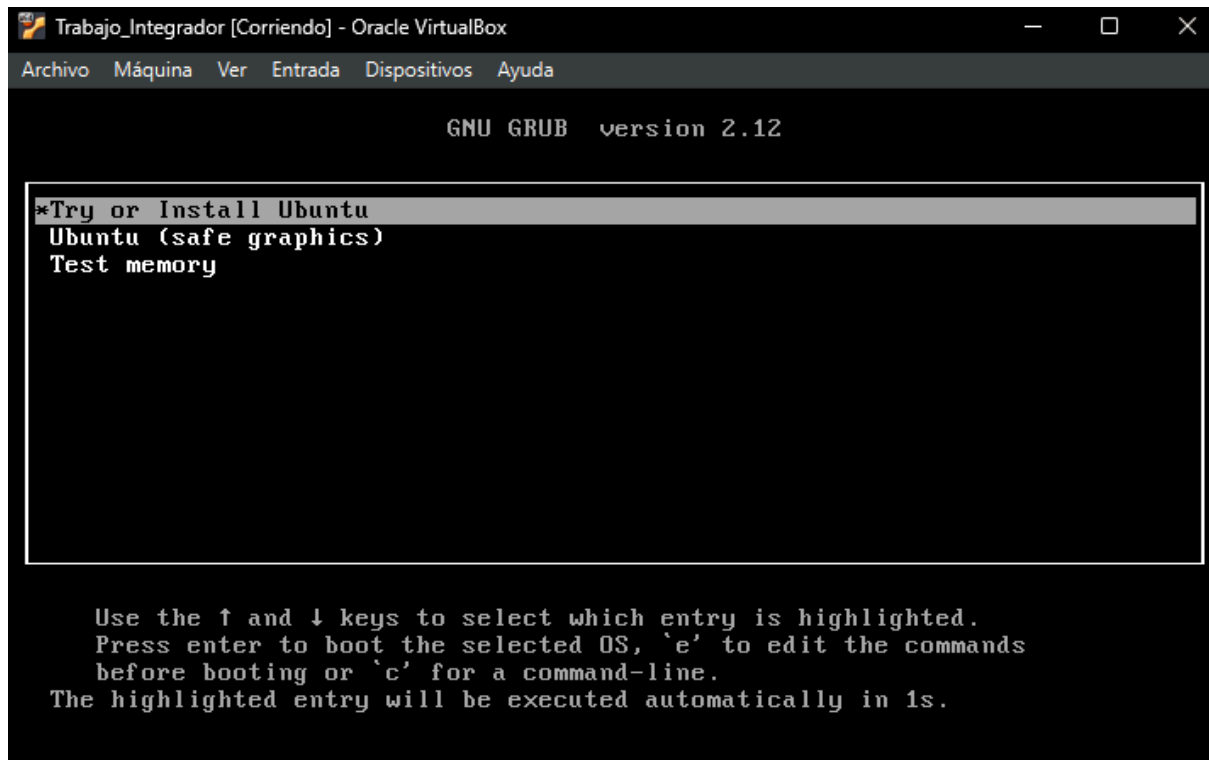
En el menú de **Hardware** asignaremos los recursos necesarios para el funcionamiento de la máquina virtual, siempre teniendo en cuenta no excedernos para que el sistema operativo anfitrión pueda continuar funcionando correctamente.



Finalmente, en el apartado de **Disco duro** crearemos un disco virtual. Esto también dependerá del hardware de la computadora. También hay que tener en cuenta el almacenamiento mínimo que necesita el sistema operativo.

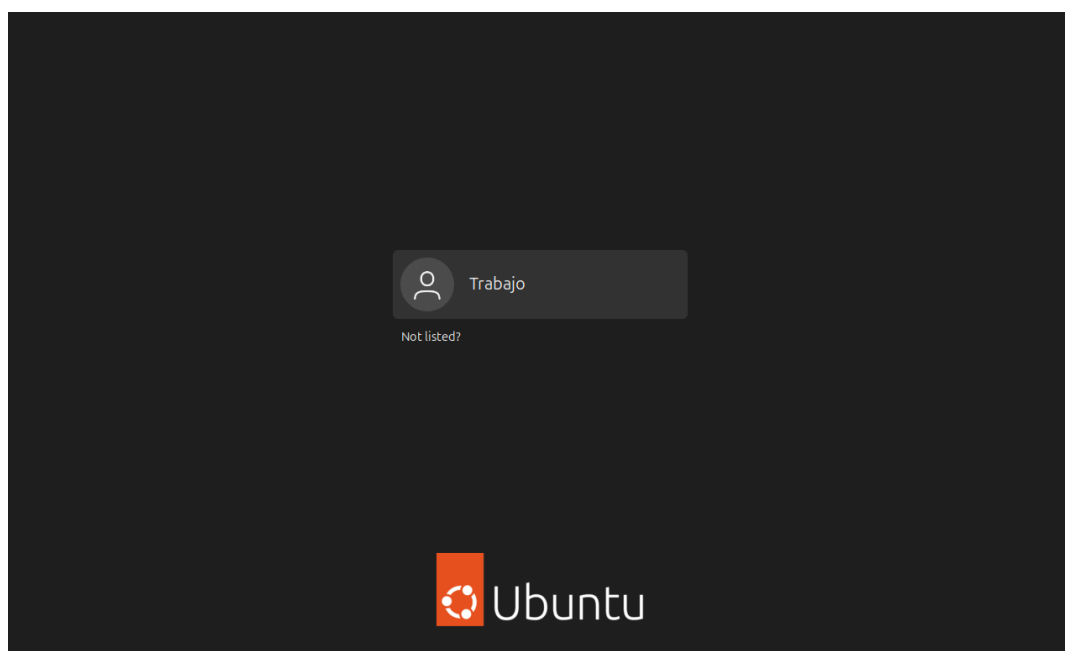


Una vez configurado esto hacemos clic en terminar y se empezará a ejecutar la máquina virtual.



Una vez nos aparezca este menú presionamos Enter en **Try or Install Ubuntu**.

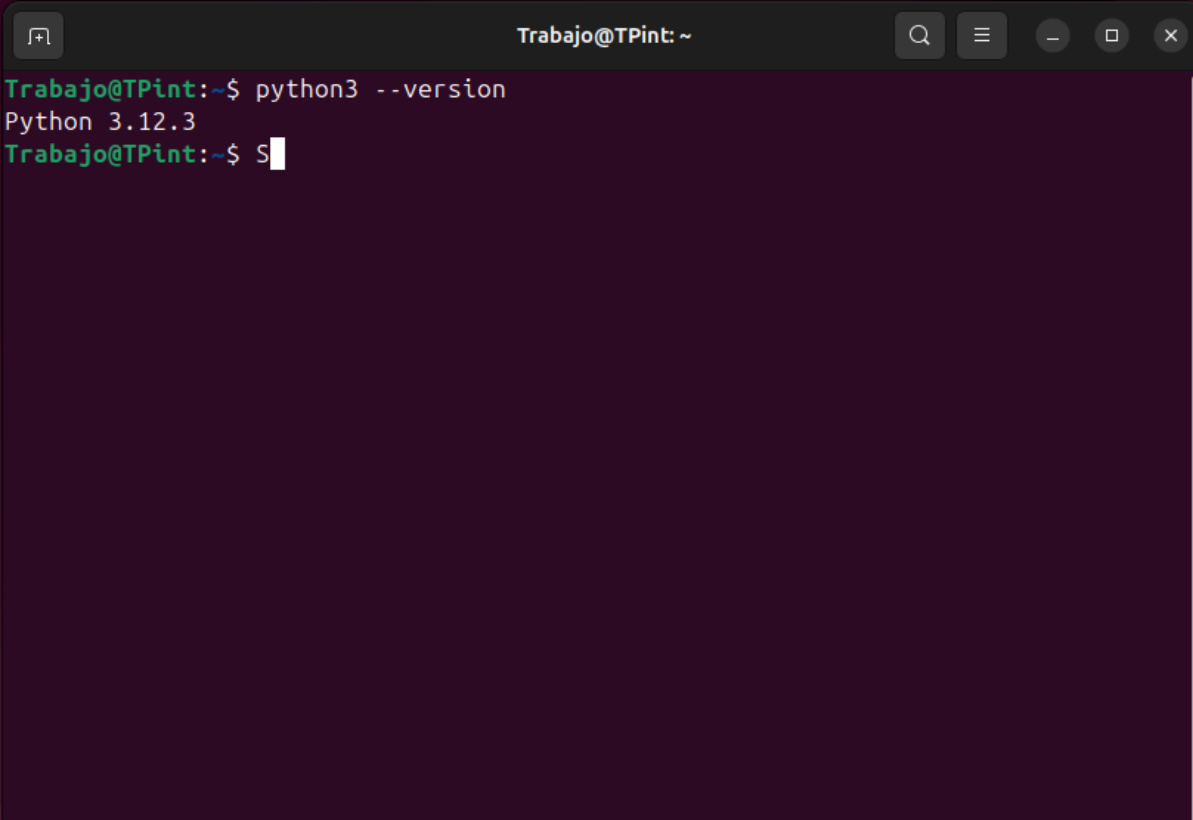
Luego de esto nos queda espera mientras VirtualBox realiza la configuración del sistema operativo.



Una vez veamos esta pantalla significa que terminó el proceso de instalación de Ubuntu y está listo para utilizar.

## Verificación de Python en Ubuntu

El sistema operativo Ubuntu 24.04 suele traer instalado Python, para comprobar esto vamos a ejecutar en comando **python3 --version** en la terminal.

A screenshot of a terminal window titled 'Trabajo@TPint: ~'. The terminal has a dark purple background. The prompt 'Trabajo@TPint:~\$' is shown in green. The command 'python3 --version' is entered in white. The output 'Python 3.12.3' is displayed in green. Below the output, the prompt 'Trabajo@TPint:~\$' is shown again with a white cursor. The terminal window has standard Ubuntu window controls (search, menu, zoom, close) in the top right corner.

Efectivamente vemos que la terminal nos devuelve la versión de Python que se encuentra instalada. De esta forma comprobamos que estamos en condiciones de utilizar la máquina virtual como entorno de desarrollo.

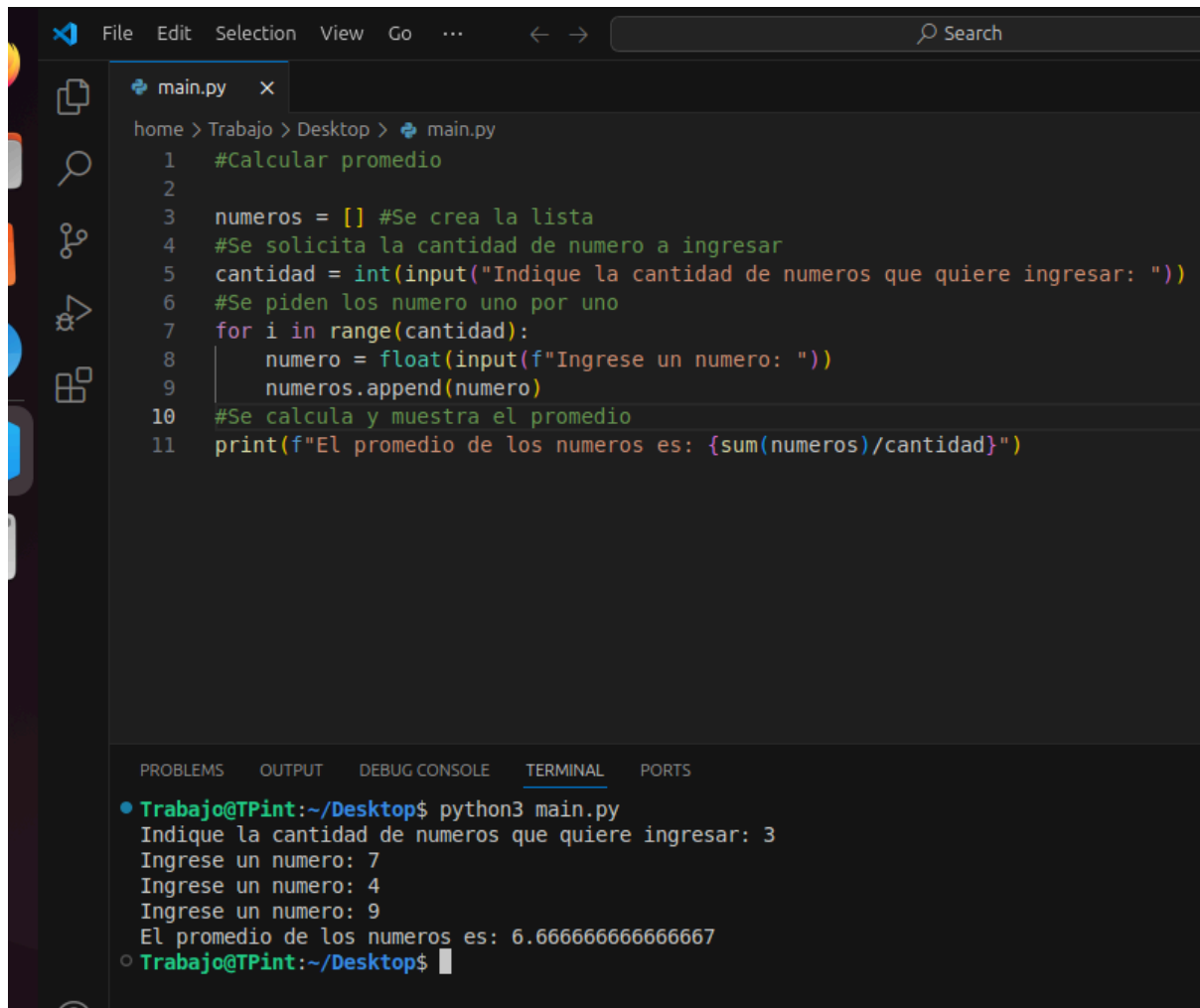
## Instalación Visual Studio Code

Para facilitar el desarrollo del código se decidió instalar Visual Studio Code, la instalación de este se realizó desde la tienda de aplicaciones que proporciona Ubuntu, aunque también se puede realizar desde la terminal usando el repositorio oficial de Microsoft. Esta última alternativa es útil en caso de que no se cuente con una interfaz gráfica.

## Desarrollo y prueba de código

Durante esta etapa se realizó el archivo de código en Python. El programa le solicita al usuario que ingrese una cantidad de números, los guarda en una lista, calcula el promedio y finalmente muestra el promedio.

A continuación se puede ver el código desarrollado junto a una prueba de funcionamiento.



The image shows a code editor window with a file named `main.py` open. The code is a Python script that calculates the average of a list of numbers. The script is as follows:

```
1 #Calcular promedio
2
3 numeros = [] #Se crea la lista
4 #Se solicita la cantidad de numero a ingresar
5 cantidad = int(input("Indique la cantidad de numeros que quiere ingresar: "))
6 #Se piden los numero uno por uno
7 for i in range(cantidad):
8     numero = float(input(f"Ingrese un numero: "))
9     numeros.append(numero)
10 #Se calcula y muestra el promedio
11 print(f"El promedio de los numeros es: {sum(numeros)/cantidad}")
```

Below the code editor, there is a terminal window showing the execution of the script. The terminal output is as follows:

```
Trabajo@TPint:~/Desktop$ python3 main.py
Indique la cantidad de numeros que quiere ingresar: 3
Ingrese un numero: 7
Ingrese un numero: 4
Ingrese un numero: 9
El promedio de los numeros es: 6.666666666666667
Trabajo@TPint:~/Desktop$
```

## metodología utilizada

Durante el desarrollo del trabajo se utilizó una metodología práctica, se aplicaron progresivamente los conceptos estudiados durante el cursado.

El proceso de desarrollo se llevó a cabo de la siguiente forma:

1. **Selección de tema:** Elegimos el tema de virtualización debido a que nos resultó entretenido y muy útil para lograr desarrollos en un entorno seguro.



2. **Recolección de información:** Se utilizó la bibliografía proporcionada por la cátedra, además de la documentación oficial de VirtualBox, Microsoft Hyper-V, VirtualBox, Ubuntu y Visual Studio Code
3. **Preparación del entorno:** Se descargó e instaló VirtualBox, se descargó la imagen ISO de Ubuntu y se creó y configuró una máquina virtual, asignando los recursos necesarios.
4. **instalación de herramienta para el desarrollo:** Se descargo Visual Studio Code desde la tienda de aplicaciones de Ubuntu.
5. **Desarrollo y prueba de código:** Se creó un código sencillo en Python y luego se ejecutó y corroboró su correcto funcionamiento, todo dentro de la máquina virtual.
6. **Documentación del proceso:** Cada uno de los pasos quedó documentado mediante capturas de pantalla junto a sus respectivas explicaciones.

## Resultados obtenidos

Se logró configurar de forma satisfactoria un entorno de virtualización del tipo 2 con la ayuda de VirtualBox. Se instaló el sistema operativo Ubuntu dentro de la máquina virtual. Realizar esta experiencia nos dejó verificar que una máquina virtual de este tipo es fácil de utilizar y sirve como un entorno seguro.

En este entorno se instaló Visual Studio Code y se desarrolló y probó un código en Python, de esta forma cumpliendo el objetivo de poner en práctica conceptos teóricos antes vistos.

Además, comprobamos que:

- La máquina virtual puede ejecutarse correctamente en el sistema anfitrión sin interferir en su funcionamiento, manteniendo el aislamiento.
- Los recursos asignados pueden asignarse según las necesidades del entorno de desarrollo.
- La instalación de herramientas se realiza de manera similar a un sistema físico.
- El código se ejecutó de forma exitosa, lo cual nos dice que la máquina virtual era funcional.

## Conclusiones

Durante este trabajo se aplicaron los conocimientos teóricos sobre virtualización vistos durante el cursado, a través de la instalación y configuración de una máquina virtual con Ubuntu se logró comprender mejor el funcionamiento de los hipervisores del tipo 2 y como estos gestionan los recursos virtuales.

Además, experimentamos como una máquina virtual funciona en un entorno aislado para el desarrollo y prueba de código. Realizar esta experiencia nos permitió afianzar los conceptos claves cómo lo son el aislamiento, el uso eficiente de recursos y la versatilidad de los entornos virtuales.

Surgieron dificultades durante la configuración de la máquina virtual, pero pudimos superarlas consultado los videos proporcionados por la cátedra y la documentación de VirtualBox

## Bibliografía

- **Documentación Oracle VirtualBox:** <https://www.virtualbox.org/manual/>
- **Ubuntu Downloads:** <https://ubuntu.com/download/desktop>
- **Documentación Microsoft Hyper-V:**  
<https://learn.microsoft.com/es-es/windows-server/virtualization/virtualization>
- **Documentación Visual Studio Code:** <https://code.visualstudio.com/docs>
- **Material bibliográfico y videos proporcionados por la cátedra**