

Trabajo Integrador de Arquitectura y Sistemas Operativos

Título: “Virtualización con VirtualBox”

Profesor:

Osvaldo Falabella

Tutor:

Patricio Costello

Alumnos:

Sol Yoon - solyoon90@gmail.com

Guido Damian Aguero - guidoaguero14@gmail.com

Comisión 23

Fecha de Entrega:

5 de junio de 2025

Índice

	Página
1. Introducción	2
2. Marco Teórico	3
3. Caso Práctico	7
4. Metodología Utilizada	14
5. Resultados Obtenidos	16
6. Conclusiones	17
7. Bibliografía	18

Introducción

El presente trabajo se enfoca en el uso de la virtualización mediante la herramienta VirtualBox para crear una máquina virtual con el sistema operativo Ubuntu. En este entorno se verificó la instalación de Python, y luego de confirmarse, se desarrolló un programa simple como prueba de funcionamiento mediante Visual Studio Code.

La elección del tema responde a la necesidad de contar con espacios controlados y seguros para prácticas de programación. La virtualización permite ejecutar un sistema operativo dentro de otro, facilitando la experimentación sin afectar el equipo físico. Además, el dominio de herramientas como VirtualBox representa una ventaja en el ámbito profesional.

Este trabajo busca aplicar conceptos teóricos sobre virtualización y entornos de desarrollo, fortaleciendo habilidades técnicas clave para la formación como técnico en programación.

Marco teórico

❖ Definición de Virtualización

La virtualización es una tecnología que permite crear versiones virtuales de recursos informáticos como sistemas operativos, servidores, dispositivos de almacenamiento o redes, que funcionan sobre una infraestructura física real. Según el Instituto Nacional de Estándares y Tecnología de EE. UU. (NIST), se define como “el proceso de simular una plataforma de hardware para permitir la ejecución de múltiples sistemas operativos en una misma máquina física”.

Esta tecnología es ampliamente utilizada en entornos de desarrollo, testing, servidores y nube, ya que permite aislar entornos, reducir costos y mejorar la escalabilidad.

❖ Tipos de Virtualización

Existen distintas clasificaciones de la virtualización, entre las cuales se destacan:

Tipo de Virtualización	Descripción
Virtualización de hardware	Simula todo el hardware para ejecutar sistemas operativos invitados.
Virtualización de sistema operativo	Crea contenedores que comparten el mismo kernel (como Docker).
Virtualización de almacenamiento	Consolida múltiples dispositivos físicos en una unidad lógica.
Virtualización de red	Permite crear redes virtuales, separadas del hardware físico.

❖ Hipervisores

El hipervisor, también conocido como monitor de máquina virtual (VMM, por sus siglas en inglés), es el componente fundamental que permite ejecutar múltiples sistemas operativos de forma simultánea en una misma máquina física, dividiendo y gestionando los recursos de hardware de manera eficiente.

Tipos de hipervisores

1. Hipervisor Tipo 1 (nativo o bare-metal)

Este tipo de hipervisor se instala directamente sobre el hardware físico del servidor, sin requerir un sistema operativo anfitrión. Tiene acceso directo a los recursos del sistema, lo que le otorga mejor rendimiento, menor latencia y mayor estabilidad. Es ampliamente utilizado en entornos empresariales y de centros de datos.

Ejemplos de hipervisores tipo 1:

- VMware ESXi
- Microsoft Hyper-V (modo nativo)
- Xen Project
- KVM (Kernel-based Virtual Machine, aunque técnicamente se ejecuta sobre Linux, es tratado como tipo 1 por su integración con el kernel)

2. Hipervisor Tipo 2 (hosteado o de usuario)

Este tipo de hipervisor se ejecuta sobre un sistema operativo ya instalado (como Windows, macOS o Linux), funcionando como una aplicación más. Si bien su rendimiento es algo inferior al de los tipo 1 debido a la capa adicional del sistema anfitrión, su facilidad de uso y configuración lo hace ideal para entornos de pruebas, educación y desarrollo personal.

Ejemplos de hipervisores tipo 2:

- Oracle VM VirtualBox
- VMware Workstation
- VMware Fusion (para macOS)
- Parallels Desktop

❖ VirtualBox

En este trabajo se utiliza Oracle VM VirtualBox, un hipervisor tipo 2 gratuito y multiplataforma. Es una herramienta muy utilizada en entornos educativos por su interfaz amigable, compatibilidad con múltiples sistemas operativos y facilidad para crear y gestionar máquinas virtuales con distintas configuraciones de red, almacenamiento y hardware simulado.

❖ Sistema Operativo Ubuntu

Ubuntu es una distribución del sistema operativo Linux basada en Debian, de código abierto y ampliamente utilizada en entornos educativos, empresariales y de desarrollo. Se destaca por su estabilidad, seguridad y comunidad activa. Es ideal para pruebas en entornos virtuales por su bajo requerimiento de recursos y compatibilidad con herramientas de desarrollo.

❖ Imagen ISO

Una imagen ISO es un archivo que contiene una copia exacta (imagen) de un sistema de archivos óptico, como un CD o DVD. Es el formato comúnmente utilizado para distribuir sistemas operativos, como Ubuntu. Al montar o arrancar una máquina virtual desde una imagen ISO, es posible iniciar el proceso de instalación como si se tratara de un disco físico.

En este trabajo se utilizó la imagen **Ubuntu 20.04.6 LTS**, descargada desde el sitio oficial de Ubuntu (<https://ubuntu.com>). Esta versión fue elegida por su estabilidad y compatibilidad con entornos virtualizados.

❖ Lenguaje Python

Python es un lenguaje de programación de alto nivel, interpretado, multiplataforma y de sintaxis simple. Fue creado por Guido van Rossum en 1991. Su facilidad de aprendizaje y gran comunidad lo convierten en uno de los lenguajes más populares actualmente. Es ampliamente utilizado en automatización, ciencia de datos, desarrollo web y scripting.

❖ **Visual Studio Code**

Visual Studio Code (VS Code) es un entorno de desarrollo integrado (IDE) de código abierto desarrollado por Microsoft. Es una herramienta liviana y multiplataforma (disponible para Windows, Linux y macOS) que ofrece soporte para múltiples lenguajes de programación mediante extensiones.

En este trabajo fue utilizada dentro de la máquina virtual con Ubuntu como entorno para escribir y ejecutar un programa en Python.

Caso práctico

Descripción del problema

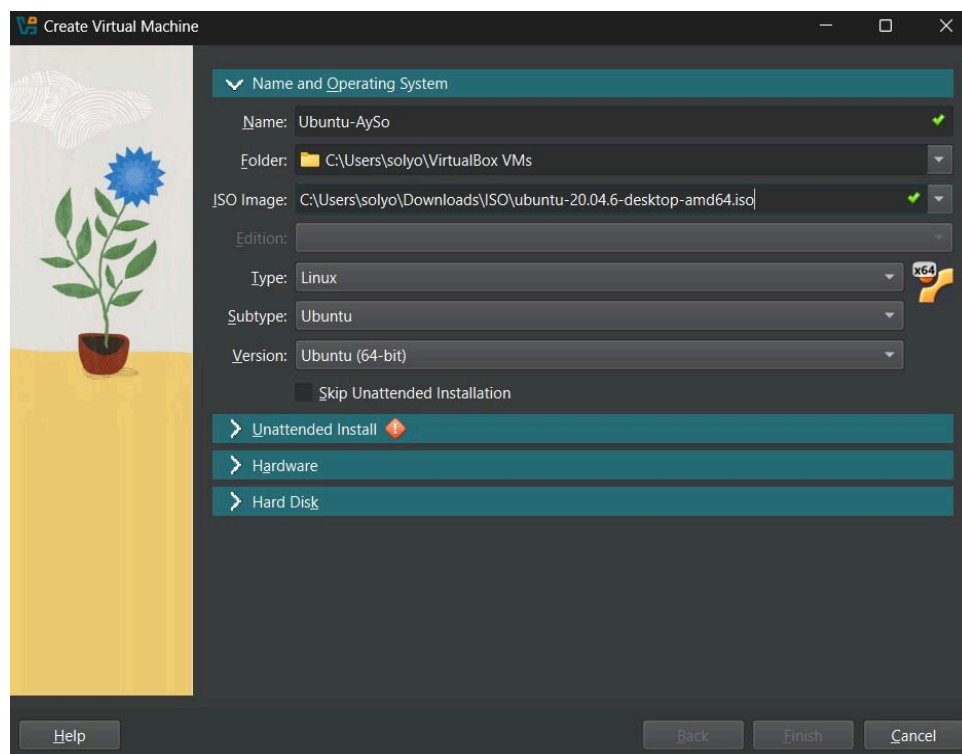
Con el objetivo de aplicar los conceptos teóricos de virtualización, se propone la creación de un entorno virtualizado de desarrollo. Para ello, se utilizará **Oracle VM VirtualBox** para instalar una máquina virtual con **Ubuntu 20.04.6 LTS**, dentro de la cual se instalará **Python 3** y se desarrollará un programa simple que imprima un mensaje en pantalla.

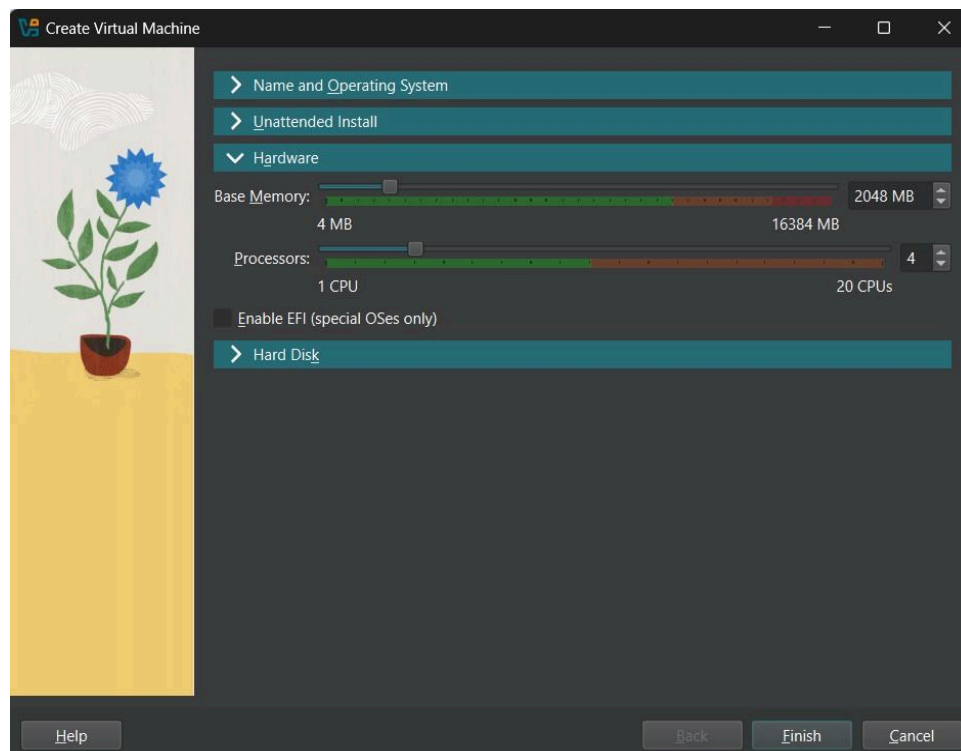
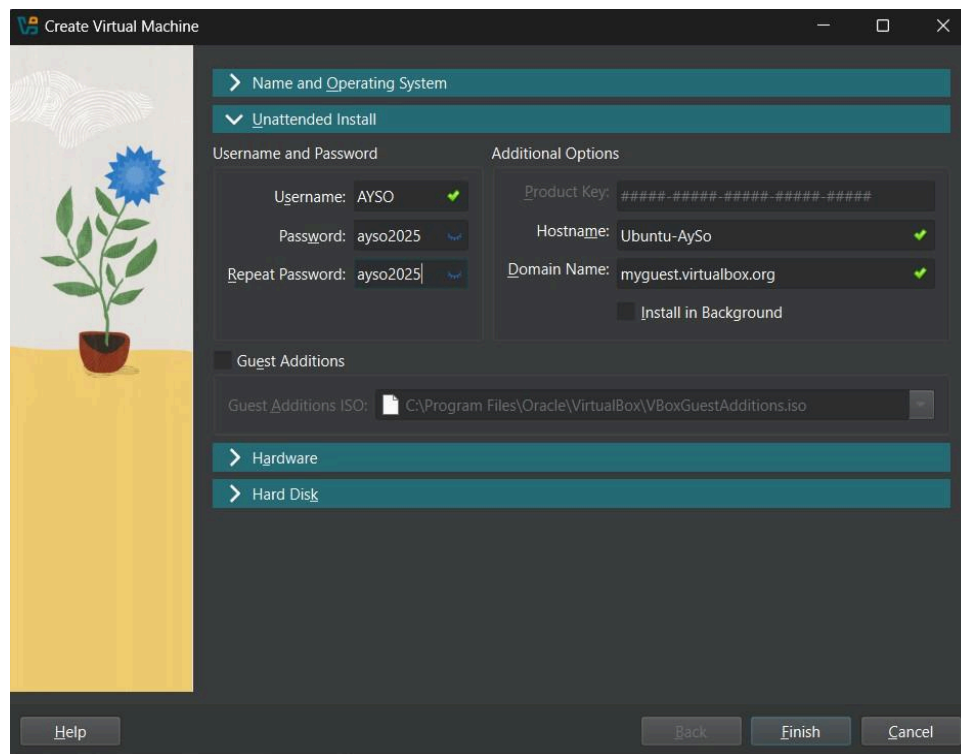
Este entorno permitirá al usuario simular una situación real de trabajo, en la que se necesita probar o desarrollar software dentro de un entorno aislado y controlado, sin afectar al sistema operativo principal.

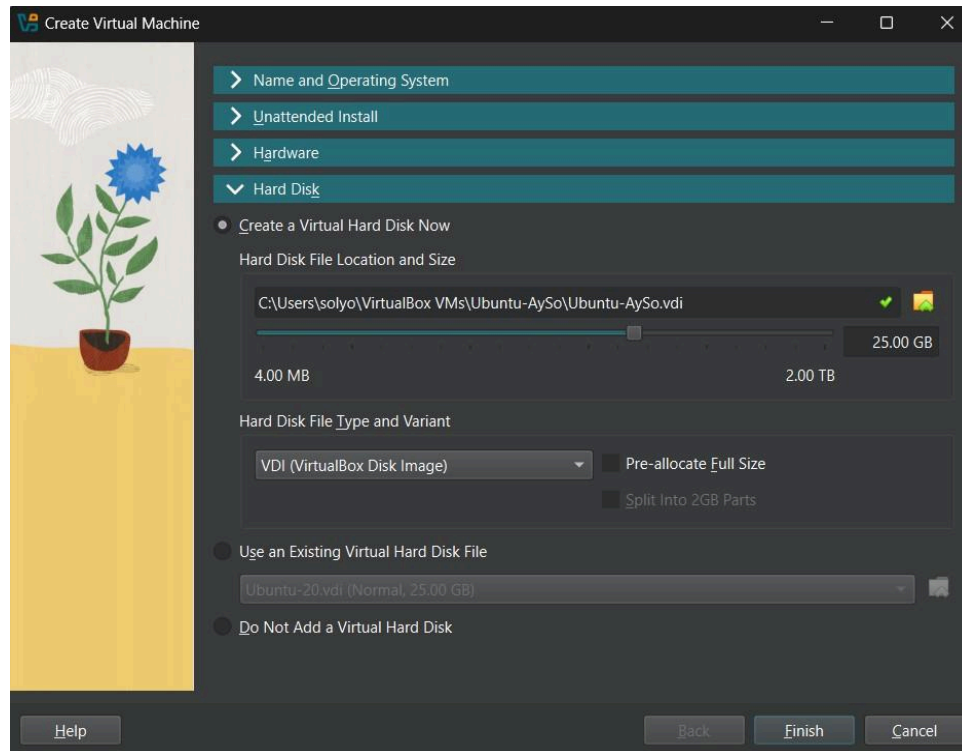
Pasos realizados

1. Creación de la máquina virtual

- ◆ Se abre VirtualBox y se selecciona la opción "Nueva".
- ◆ Se asigna el nombre "Ubuntu-AySo", tipo "Linux" y versión "Ubuntu (64-bit)".
- ◆ Se configura la memoria RAM (2048 MB) y se crea un disco duro virtual de 25 GB.
- ◆ Se monta el archivo ISO de Ubuntu versión 20.04.6 LTS para iniciar la instalación.

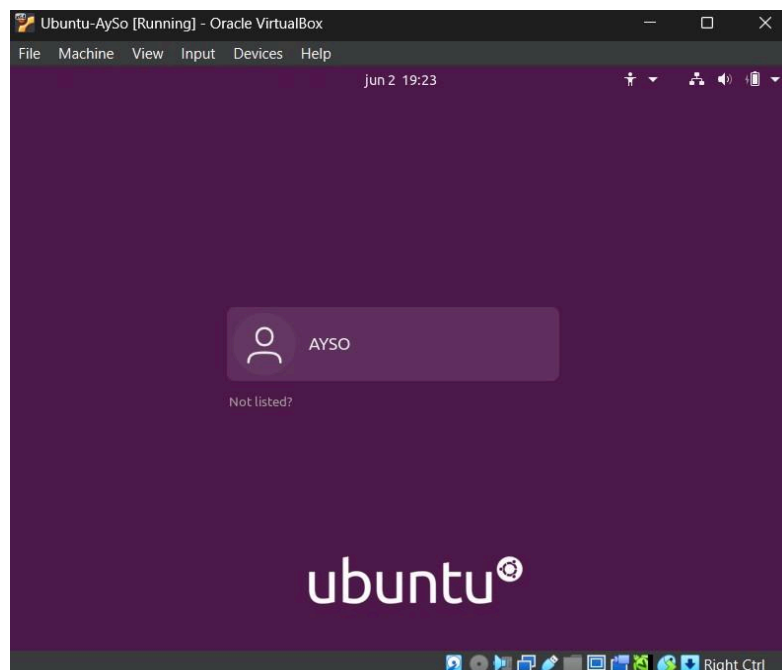


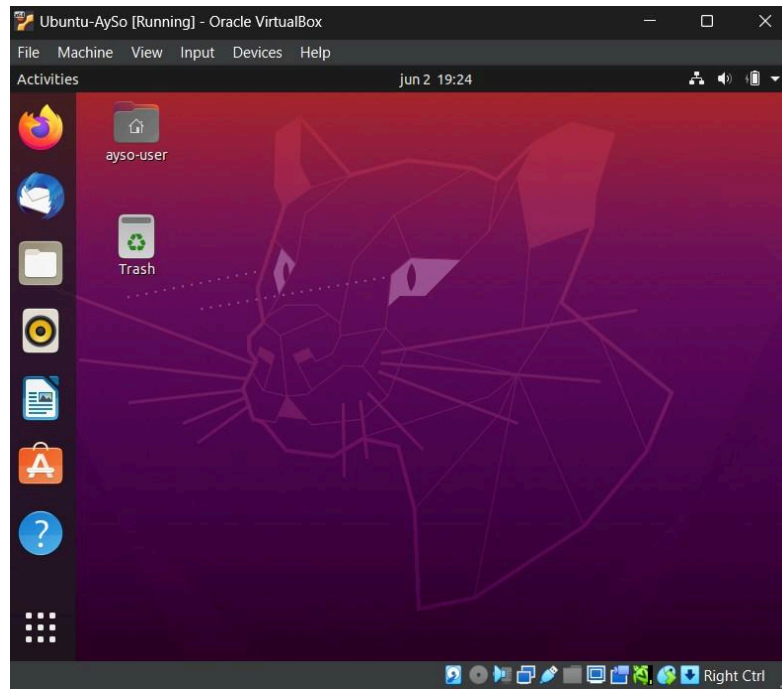




2. Instalación de Ubuntu

- ◆ Se sigue el asistente de instalación de Ubuntu, seleccionando idioma, distribución de teclado, nombre de usuario y contraseña.
- ◆ Una vez finalizada la instalación, se reinicia la máquina virtual y se inicia sesión en el entorno Ubuntu.





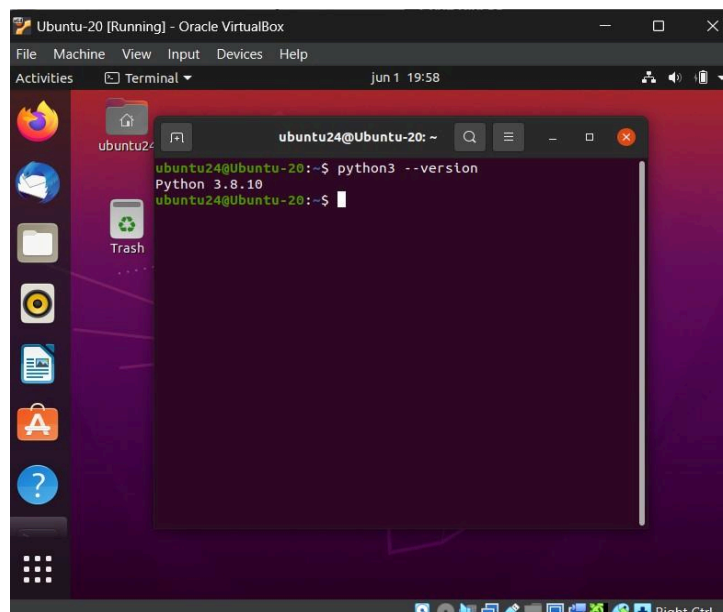
3. Instalación de Python

Ubuntu ya incluye Python preinstalado, pero se verifica su versión con:

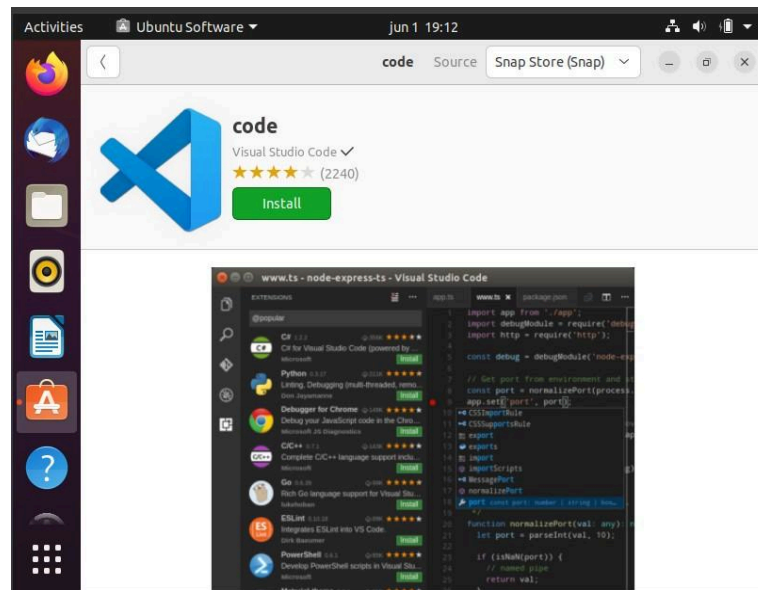
```
python3 --version
```

En caso de no estar instalado o querer actualizarlo, se puede usar:

```
sudo apt update  
sudo apt install python3
```



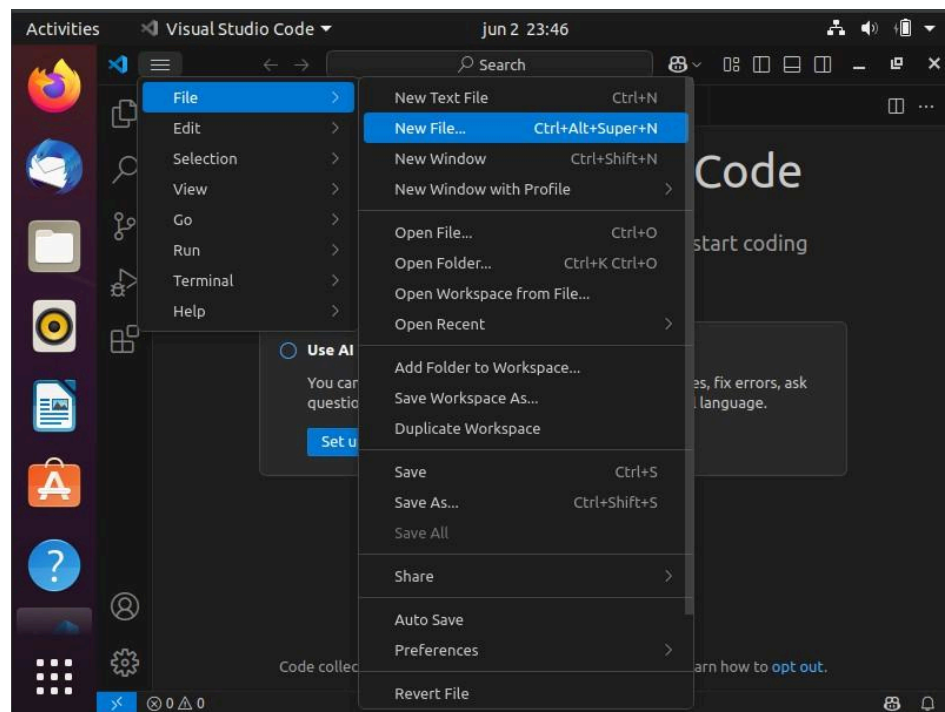
4. Instalación de Visual Studio Code

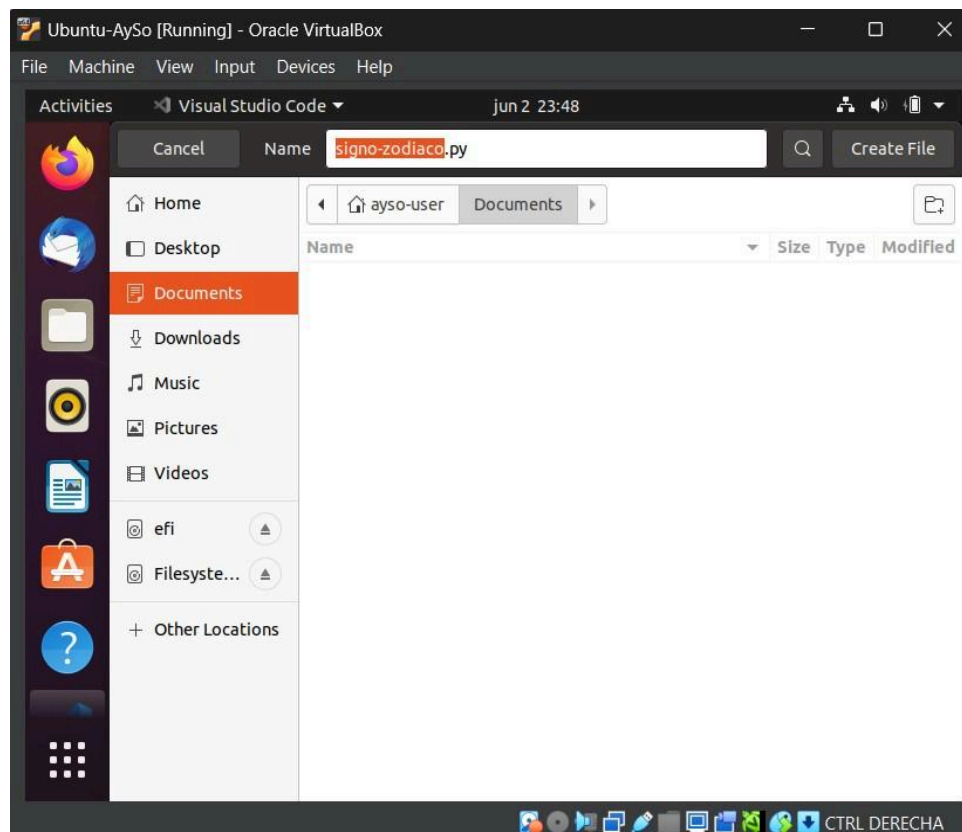


Se instala Visual Studio Code desde Ubuntu Software.

5. Creación del programa

Se crea un archivo con extensión **.py** desde VS Code:






Dentro del archivo se escribe el siguiente código:

```
# Función para obtener el signo zodiaco
def obtener_signo_zodiaco(dia, mes):
    if (mes == 3 and dia >= 21) or (mes == 4 and dia <= 19):
        return "Aries"
    elif (mes == 4 and dia >= 20) or (mes == 5 and dia <= 20):
        return "Tauro"
    elif (mes == 5 and dia >= 21) or (mes == 6 and dia <= 20):
        return "Géminis"
    elif (mes == 6 and dia >= 21) or (mes == 7 and dia <= 22):
        return "Cáncer"
    elif (mes == 7 and dia >= 23) or (mes == 8 and dia <= 22):
        return "Leo"
    elif (mes == 8 and dia >= 23) or (mes == 9 and dia <= 22):
        return "Virgo"
    elif (mes == 9 and dia >= 23) or (mes == 10 and dia <= 22):
        return "Libra"
    elif (mes == 10 and dia >= 23) or (mes == 11 and dia <= 21):
        return "Escorpio"
    elif (mes == 11 and dia >= 22) or (mes == 12 and dia <= 21):
```

```
# Se pide día y mes de nacimiento al usuario
dia = int(input("Ingresá tu día de nacimiento (1-31): "))
mes = int(input("Ingresá tu mes de nacimiento (1-12): "))

signo = obtener_signo_zodiaco(dia, mes)

# Se muestra el resultado
print(f"Tu signo del zodiaco es: {signo}")
```



The screenshot shows a terminal window with a dark background. At the top, there is a tab labeled "Python: signo-zodiaco" with standard window controls. Below the tab bar, a menu bar contains "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (which is selected and underlined), and "PORTS". The terminal content shows two runs of the script `/bin/python3 /home/ayso-user/Documents/signo-zodiaco.py`. The first run prompts for birth day (1-31) and month (1-12), resulting in the sign "Acuario". The second run prompts for birth day (1-31) and month (1-12), resulting in the sign "Tauro". The prompt `ayso-user@Ubuntu-AySo:~$` is visible at the end of each command line.

```
Python: signo-zodiaco + - [ ] [ ] ... ^ X
/bin/python3 /home/ayso-user/Documents/signo-zodiaco.py
ayso-user@Ubuntu-AySo:~$ /bin/python3 /home/ayso-user/Documents/signo-zodiaco.py
Ingrese su día de nacimiento (1-31): 25
Ingrese su mes de nacimiento (1-12): 1
Su signo del zodiaco es: Acuario
ayso-user@Ubuntu-AySo:~$ /bin/python3 /home/ayso-user/Documents/signo-zodiaco.py
Ingrese su día de nacimiento (1-31): 17
Ingrese su mes de nacimiento (1-12): 5
Su signo del zodiaco es: Tauro
ayso-user@Ubuntu-AySo:~$
```

La ejecución del programa se realizó correctamente, mostrando el mensaje por consola dentro del sistema operativo virtualizado. Esto demuestra que el entorno fue configurado correctamente, que Python funciona sin inconvenientes, y que el sistema es apto para el desarrollo de scripts y programas en lenguaje Python.

13

Metodología utilizada

Para el desarrollo del trabajo se siguió una metodología práctica basada en el uso de herramientas de virtualización y programación. A continuación, se describen las etapas y recursos utilizados:

- **Descarga de VirtualBox:**

Se accedió al sitio oficial de Oracle (<https://www.virtualbox.org>) para descargar e instalar el software Oracle VM VirtualBox en un sistema operativo anfitrión (Host) Windows 11. Esta herramienta permite crear y gestionar máquinas virtuales de manera sencilla.

- **Investigación previa:**

Se consultaron fuentes oficiales como la documentación de Oracle VirtualBox, el sitio web de Ubuntu y materiales teóricos proporcionados por los docentes de la asignatura, con el fin de comprender los conceptos fundamentales de virtualización, los tipos de hipervisores y el funcionamiento del sistema operativo Ubuntu en entornos virtuales. Además, se tomaron en cuenta videos tutoriales disponibles en YouTube, que resultaron útiles especialmente para la instalación del sistema operativo y la configuración inicial de la máquina virtual.

- **Descarga de la imagen ISO de Ubuntu:**

Se accedió al sitio oficial de Ubuntu y se descargó la imagen ISO de la versión Ubuntu 20.04.6 LTS, elegida por su estabilidad y compatibilidad con entornos virtualizados. Inicialmente se intentó instalar la versión Ubuntu 24.04.02, pero durante el proceso se presentaron errores relacionados con el controlador gráfico, lo que motivó el cambio a una versión anterior más confiable para el uso en VirtualBox.

- **Preparación del entorno de virtualización:**

Se creó una máquina virtual en VirtualBox con Ubuntu 20.04.6 LTS como sistema invitado (Guest). Se utilizó como configuración memoria de base de 2048 MB, 4 procesadores, disco duro virtual de 25 GB, lo cual resultó suficiente para una instalación estable y sin errores.

- **Instalación y configuración del entorno de desarrollo:**

Dentro de la máquina virtual se verificó si Python se encontraba preinstalado utilizando APT en la Terminal de Ubuntu. Luego se instaló Visual Studio Code (VS Code) como entorno de desarrollo integrado (IDE), permitiendo escribir, editar y ejecutar scripts en Python de manera cómoda y eficiente.

- **Desarrollo y prueba del código:**

Se elaboró un programa simple en Python utilizando VS Code y se verificó su correcto funcionamiento ejecutándolo desde la terminal. Esto permitió comprobar que el entorno virtualizado era funcional para tareas básicas de programación.

- **Herramientas utilizadas:**

- Oracle VirtualBox (virtualización)
- Ubuntu 20.04.6 LTS (sistema invitado)
- Python 3
- Visual Studio Code (IDE)
- Terminal de Ubuntu

Resultados obtenidos

El caso práctico permitió simular exitosamente un entorno virtual de desarrollo, cumpliendo con los objetivos propuestos. Se lograron los siguientes resultados:

- Instalación funcional de Ubuntu en VirtualBox.
- Configuración del entorno de programación con Python 3 y prueba de código sin inconvenientes en Visual Studio Code (IDE)
- Verificación del funcionamiento del script Python mediante ejecución desde la terminal.
- Dificultades encontradas:
 - Problemas con la instalación de Ubuntu 24.04.2 por congelamiento y errores gráficos (VMWGFX).
 - Se corrigió cambiando a la versión 20.04.6 LTS, más estable y sin errores de instalación.

Conclusiones

A lo largo del desarrollo de este trabajo, se pudo afianzar los conceptos fundamentales de virtualización y comprender su aplicación práctica mediante la creación de una máquina virtual con Ubuntu. Se aprendió a utilizar herramientas como VirtualBox para gestionar entornos virtuales y a trabajar con Visual Studio Code como entorno de desarrollo.

Entre las principales dificultades se presentaron errores gráficos al intentar instalar una versión más reciente de Ubuntu (24.04.2), lo que llevó a tomar la decisión de utilizar una versión anterior (20.04.6 LTS), conocida por ser muy estable. También se ajustaron configuraciones según los requerimientos mínimos que se fueron investigando hasta lograr una instalación exitosa. Estas instancias fueron valiosas para ejercitar la resolución de problemas técnicos de manera autónoma.

Como posibles mejoras o extensiones futuras, se considera la implementación de un proyecto más complejo en Python o la conexión de múltiples máquinas virtuales para simular una red local. También se plantea continuar explorando otros sistemas operativos libres y configuraciones avanzadas dentro del entorno de VirtualBox.

En resumen, este trabajo permitió integrar conocimientos teóricos y prácticos, fortaleciendo competencias clave para la formación como técnicos en programación, y nos motivó a seguir explorando el uso de tecnologías de virtualización y software libre.

Bibliografía

- Oracle VM VirtualBox Documentation. Disponible en:
<https://www.virtualbox.org/manual/> (Accedido el 1 de junio de 2025)
- Ubuntu Official Website. Disponible en: <https://ubuntu.com> (Accedido el 1 de junio de 2025)
- Python Official Documentation. Disponible en: <https://docs.python.org/> (Accedido el 1 de junio de 2025)
- National Institute of Standards and Technology (NIST). Virtualization Definition. Disponible en: <https://csrc.nist.gov> (Accedido el 1 de junio de 2025)
- VMware Glossary – What is a Hypervisor? Disponible en: <https://www.vmware.com/> (Accedido el 1 de junio de 2025)
- Microsoft. *Visual Studio Code – Code Editing. Redefined* en:
<https://code.visualstudio.com/> (Accedido el 1 de junio de 2025)