

Trabajo Práctico – Virtualización con VirtualBox

Alumnos:

Agustin Miranda - agusmiranda1611@gmail.com

Tobias Correa - tobiasbcorrea@gmail.com

Comision 8

Materia: Arquitectura y Sistemas Operativos

Tutor: Oscar Londero

Profesor: Santos Sanchez

Fecha de Entrega: 22 de octubre de 2025

Índice	2
Introducción	3
Marco Teórico	4
Caso Práctico	6
Metodología Utilizada	10
Resultados Obtenidos	10
Conclusiones	11
Bibliografía	11

Links

[Repositorio GitHub](#)

[Video Youtube](#)

Introducción

Este Trabajo Práctico Integrador tiene como objetivo documentar el proceso completo de creación, configuración y puesta en marcha de un servidor de API dentro de un entorno virtualizado.

La idea principal es demostrar cómo, a través de la virtualización, es posible ejecutar distintos sistemas operativos de manera aislada sobre una misma computadora física, optimizando los recursos disponibles y permitiendo crear entornos de prueba seguros y controlados.

Para este trabajo se utilizó una máquina virtual (VM) con Ubuntu Server 22.04, la cual se instaló sobre el hipervisor de tipo 2 Oracle VirtualBox. Este sistema operativo fue elegido por su estabilidad, su bajo consumo de recursos y porque es muy utilizado en entornos de servidores reales.

Al tratarse de una versión sin entorno gráfico, todas las configuraciones y comandos se realizaron desde la terminal, lo que permite comprender mejor el funcionamiento interno del sistema.

Dentro del servidor, se instaló Python 3 junto con un entorno virtual (virtual environment), que se utilizó para gestionar las dependencias del proyecto sin afectar al sistema principal.

El núcleo del trabajo consiste en el despliegue de una API REST simple, desarrollada con el framework FastAPI, un entorno rápido para crear servicios web en Python. La API fue servida mediante Uvicorn, un servidor ligero y eficiente.

Finalmente, se configuró la red de la máquina virtual en modo “Puente” (Bridge Adapter), lo que le permite obtener una dirección IP dentro de la misma red que la computadora anfitriona. De esta forma, la API puede ser accedida desde el host o desde otros dispositivos conectados a la red local, simulando un entorno de desarrollo y despliegue similar al de un servidor real.

Este trabajo permitió comprender el proceso completo que va desde la instalación del sistema operativo hasta la publicación de un servicio web funcional, aplicando conceptos de virtualización, redes y desarrollo backend con Python.

Marco Teórico

Virtualización

La virtualización es un proceso que permite crear entornos o máquinas virtuales simuladas desde un solo equipo físico, compartiendo y distribuyendo los recursos del hardware (CPU, memoria y almacenamiento) entre varios sistemas operativos independientes.

Esto permite que múltiples sistemas operativos y aplicaciones se ejecuten simultáneamente en un mismo dispositivo físico, como si fueran máquinas

independientes.

La virtualización aporta varios beneficios:

- **Eficiencia de recursos:** antes de la virtualización, cada servidor de aplicaciones requería de un dispositivo físico dedicado. El personal de TI solía mantener únicamente una aplicación y sistema operativo por computadora por motivos de confiabilidad.
Con la virtualización, es posible ejecutar varias aplicaciones, cada una en su propia máquina virtual con su propio sistema operativo, sobre una sola computadora física sin sacrificar la confiabilidad. Esto representa un gran avance, ya que reduce costos, optimiza el espacio y aprovecha mejor la capacidad del hardware disponible.
- **Configuración más rápida:** la instalación y configuración de hardware físico para cada aplicación suele requerir mucho tiempo y esfuerzo manual, ya que implica adquirir los equipos, instalarlos, conectarlos y configurarlos de manera individual.
En cambio, las máquinas virtuales permiten un proceso mucho más ágil y automatizable, ya que se pueden crear, clonar o eliminar en pocos minutos desde el propio software de virtualización.
- **Gestión más sencilla:** reemplazar las computadoras físicas con máquinas virtuales definidas por software simplifica significativamente la gestión de las mismas.
Lo que antes suponía acceder físicamente a cada máquina para instalar, configurar o reparar algo, ahora se puede hacer desde una misma interfaz o programa central.
- **Tiempo de inactividad mínimo:** las fallas del sistema operativo o de las aplicaciones pueden provocar tiempos de inactividad que interrumpen los servicios y afectan la experiencia del usuario.
Gracias a la virtualización, es posible mantener máquinas virtuales redundantes que actúan como respaldo, permitiendo migrar los servicios automáticamente en caso de fallas.

Hipervisores

Un hipervisor es un software que agrupa recursos informáticos y los asigna entre las máquinas virtuales. Esta tecnología permite crear y gestionar máquinas virtuales, actuando como intermediario entre el hardware físico y los sistemas operativos que se ejecutan en las máquinas virtuales.

Tipos de hipervisores

- **Tipo 1 (bare-metal):** se ejecuta directamente sobre el hardware del host. Se suele usar en los centros de datos empresariales o entornos basados en servidores.
Estos ofrecen un mayor rendimiento y una mayor eficiencia para gestionar recursos, pero tienen una curva de aprendizaje más elevada y requieren de un hardware dedicado.
Ejemplos de hipervisores tipo 1: KVM, Microsoft Hyper-V.
- **Tipo 2 (alojado):** se ejecuta sobre un sistema operativo convencional como si fuera una aplicación. Suelen usarse por usuarios individuales que quieren

ejecutar varios SO en una computadora personal. Su instalación y uso es más fácil, siendo ideales para pruebas y desarrollo, pero tienen un rendimiento significativamente menor debido a la capa adicional del sistema operativo anfitrión.

Ubuntu Server

Ubuntu Server es una variante del sistema operativo Ubuntu diseñada específicamente para entornos de servidor. Basado en Linux, Ubuntu Server se caracteriza por no incluir una interfaz gráfica por defecto, utilizándose principalmente la línea de comandos para su gestión, lo que lo hace más liviano y eficiente en recursos que la versión de escritorio. Está orientado a alojar sitios web, ejecutar aplicaciones, administrar servicios de red y otras funciones propias de servidores. Su instalación facilita configurar distintos servicios como servidores web, de correo o de archivos.

Redes Virtuales NAT y Bridge

Las redes virtuales NAT (Network Address Translation) y Bridge son dos modos comunes para conectar máquinas virtuales a una red:

- En modo NAT, la máquina virtual comparte la conexión de red del host y utiliza una dirección IP privada en una subred interna creada por el software de virtualización. La máquina puede acceder a Internet y a la red externa, pero no es accesible directamente desde allí.
Es más seguro y sencillo para el acceso saliente a Internet sin exponer la VM directamente.
- En modo Bridge, la máquina virtual se conecta directamente a la red física a través del adaptador del host, actuando como un dispositivo independiente con una IP asignada dentro de la misma red local. Esto permite que la VM sea visible y accesible en la red como si fuera un equipo físico más, con acceso completo a la red local y posibilidad de recibir conexiones entrantes sin configuraciones adicionales.

En nuestro caso práctico, deberemos configurar la máquina virtual en modo Bridge (Puente), para así poder acceder al servidor desde nuestra PC.

API

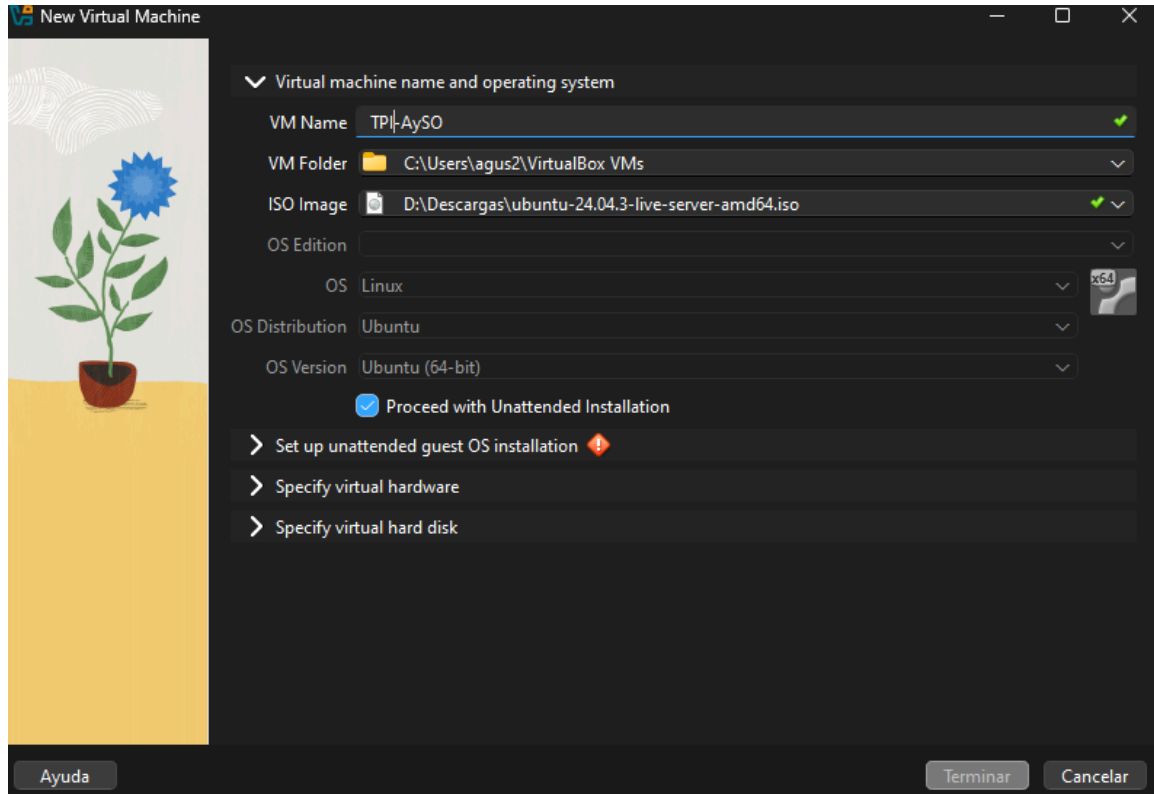
Una API REST (Representational State Transfer) es un tipo de interfaz de programación que permite la interacción entre sistemas informáticos a través de la web de manera sencilla, escalable y eficiente. En esencia, REST se basa en la idea de recursos, que pueden ser datos o servicios accesibles mediante URLs específicas. Las operaciones sobre estos recursos se realizan usando métodos HTTP estándar como GET (obtener datos), POST (crear), PUT (actualizar) y DELETE (eliminar).

JSON

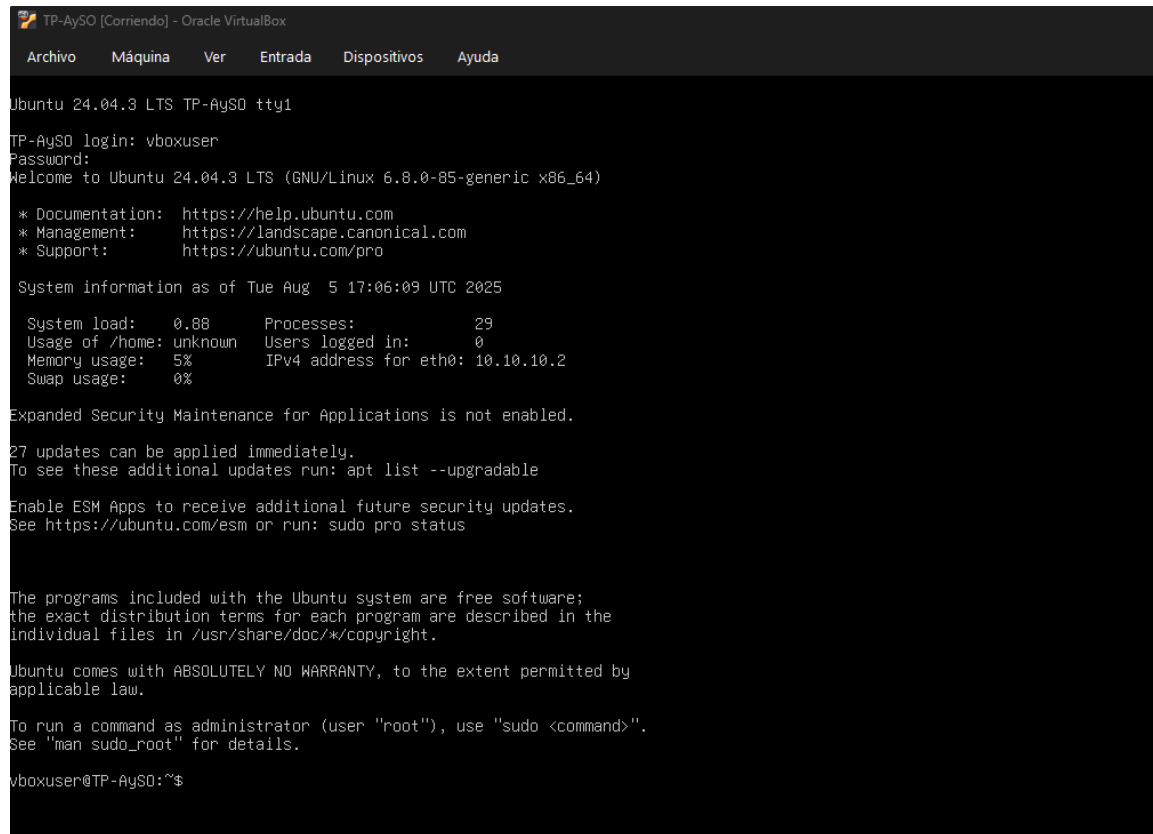
JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos muy utilizado en APIs, especialmente en APIs REST. JSON representa datos en una estructura de texto que es fácil de leer y escribir tanto para humanos como para máquinas. En el contexto de una API REST, JSON se utiliza para enviar y recibir información entre el cliente y el servidor de manera estándar y eficiente.

Caso Práctico

1. Creamos una VM con Ubuntu Server con 2GB de RAM y 1 CPU



2. Iniciamos sesión en la máquina virtual



```
TP-AySO [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

Ubuntu 24.04.3 LTS TP-AySO tty1
TP-AySO login: vboxuser
Password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-85-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Tue Aug 5 17:06:09 UTC 2025

System load:  0.88      Processes:      29
Usage of /home: unknown  Users logged in:  0
Memory usage:  5%      IPv4 address for eth0: 10.10.10.2
Swap usage:    0%

Expanded Security Maintenance for Applications is not enabled.

27 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

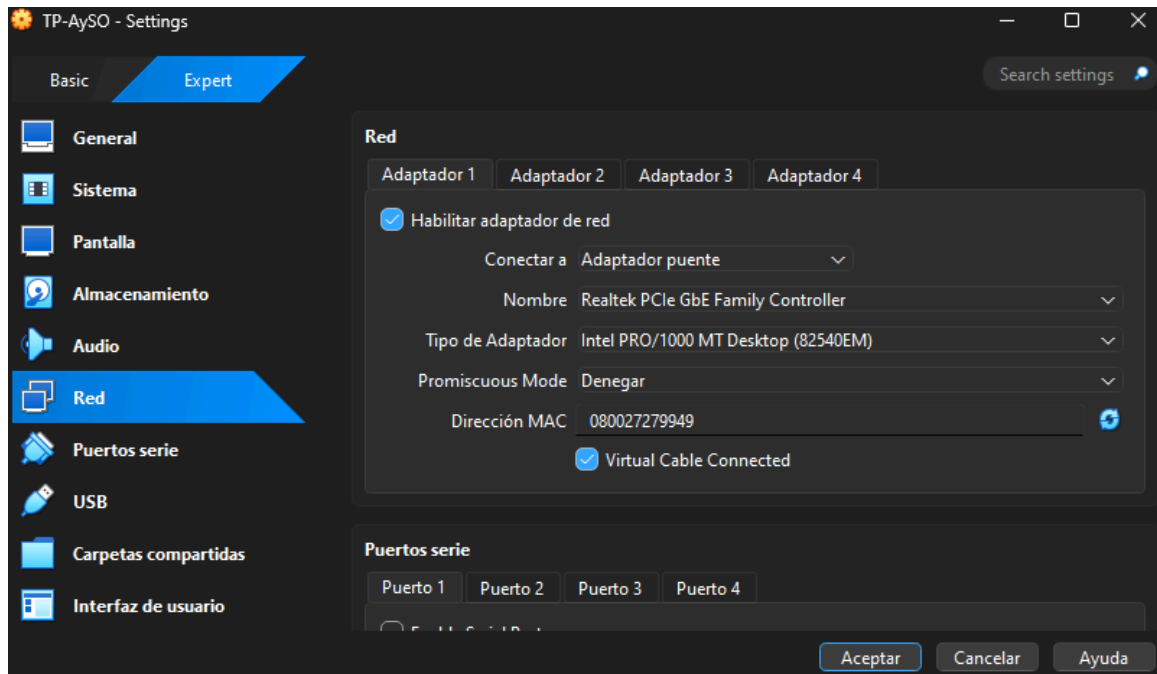
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

vboxuser@TP-AySO:~$
```


3. Configuramos la RED en modo “Puente” para posteriormente poder acceder a ella desde nuestra PC.



4. Instalamos Python 3, que nos va a permitir desarrollar nuestra app y pip, el gestor de paquetes de Python.

```
root@TP-AySO:/home/vboxuser# sudo apt install python3 python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
python3 set to manually installed.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 cpp cpp-13 c
  g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13 gcc-13-base gcc-13-x86
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan8 lib
  libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-13-dev libgomp1 libgprofng0 l
  libjs-underscore liblsan0 libmpc3 libpython3-dev libpython3.12-dev libquadmath0 libsf
  python3-dev python3-wheel python3.12-dev zlib1g-dev
Suggested packages:
```

5. Creamos un entorno virtual para desarrollar la API y poder instalar las dependencias sin comprometer el sistema.

```
root@TP-AySO:/home/vboxuser# python3 -m venv api
root@TP-AySO:/home/vboxuser# ls
api
```

```
root@TP-AyS0:/home/vboxuser# source api/bin/activate
(api) root@TP-AyS0:/home/vboxuser#
```

7. Instalamos FastAPI, esta va a ser la herramienta que nos permitirá desarrollar la API de forma rápida y Uvicorn, que en este caso actúa como el servidor encargado de ejecutar y servir nuestra API desarrollada con FastAPI.

```
(api) root@TP-AyS0:/home/vboxuser# pip install fastapi uvicorn
```

8. Creamos una API simple que retorne un JSON de una lista de juegos.

```
app = FastAPI()

@app.get("/")
def main():
    return {
        "Dark Souls": {
            "Fecha de lanzamiento": 2011,
            "Plataformas": ["PC", "PS3", "XBOX360"]
        },
        "Dark Souls 2": {
            "Fecha de lanzamiento": 2014,
            "Plataformas": ["PC", "XBOX360", "XBOX ONE", "PS3", "PS4"]
        },
        "Dark Souls 3": {
            "Fecha de lanzamiento": 2016,
            "Plataformas": ["PS4", "PC", "XBOX ONE"]
        }
    }
```

9. Corremos nuestra API corriendo el Servidor de Uvicorn en el puerto 8000

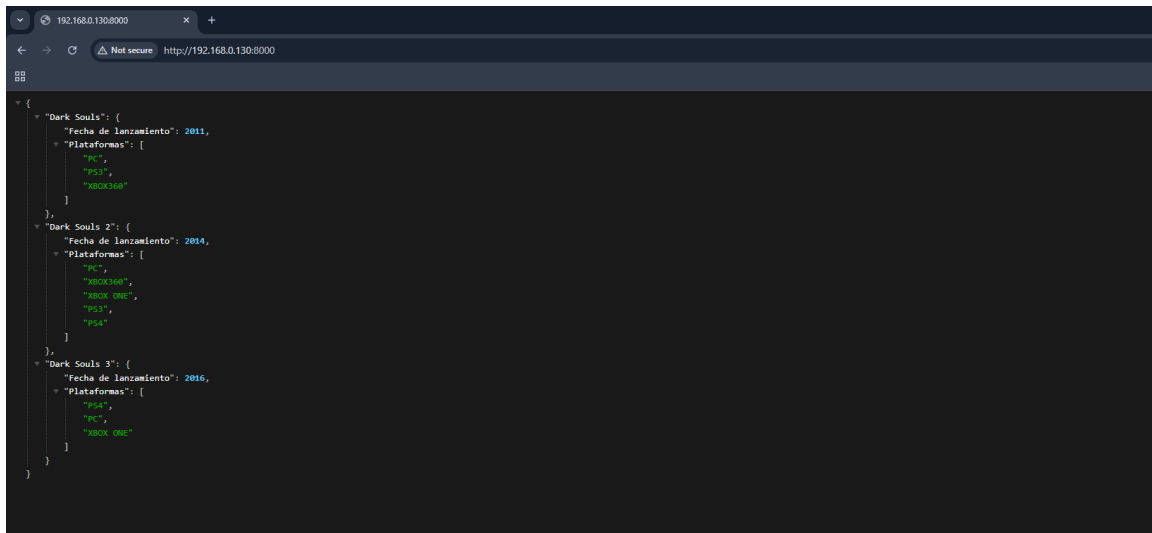
```
(api) vboxuser@TP-AyS0:~$ uvicorn api:app --host 0.0.0.0 --port 8000
INFO: Started server process [908]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

10. Utilizamos el comando ip a para saber cual es la ip de nuestra máquina virtual y poder conectarnos a nuestro servidor en el navegador de nuestra PC.

```
(api) vboxuser@TP-AyS0:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:27:99:49 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.130/24 metric 100 brd 192.168.0.255 scope global dynamic enp0s3
        valid_lft 604662sec preferred_lft 604662sec
    inet6 fe80::a00:27ff:fe27:9949/64 scope link
        valid_lft forever preferred_lft forever
(api) vboxuser@TP-AyS0:~$ _
```

11. Accedemos a la API desde nuestra PC

Resultado final:



The screenshot shows a web browser window with the address bar displaying 'http://192.168.0.130:8000'. The browser is showing a JSON response from an API. The JSON data is as follows:

```
{
  "Dark Souls": {
    "Fecha de lanzamiento": 2011,
    "Plataformas": [
      "PC",
      "PS3",
      "XBOX360"
    ]
  },
  "Dark Souls 2": {
    "Fecha de lanzamiento": 2014,
    "Plataformas": [
      "PC",
      "XBOX360",
      "XBOX ONE",
      "PS3",
      "PS4"
    ]
  },
  "Dark Souls 3": {
    "Fecha de lanzamiento": 2016,
    "Plataformas": [
      "PS4",
      "PC",
      "XBOX ONE"
    ]
  }
}
```

Metodología Utilizada

- Descarga de imagen ISO oficial de Ubuntu Server.
- Instalación de máquina virtual en VirtualBox con 2GB de RAM y 1 CPU
- Configuración de red en modo bridge para acceso externo.
- Creación de la API simple en Python
- Ejecución del servidor dentro de la VM
- Validación del servicio web desde la PC.

Resultados Obtenidos

- El servidor dentro de la VM se ejecutó correctamente.
- La PC se conectó con éxito a la API.
- Se comprendió la interacción entre el sistema host y el invitado.

Conclusiones

Este proyecto fue muy útil y didáctico, en él pudimos aplicar de forma integral toda la teoría sobre virtualización, desde lo más básico como descargar la imagen ISO y desplegarla, hasta la configuración de un servidor y el acceso a él desde otra PC.

También fue muy importante para aplicar conceptos de redes, ya que algo crucial fue configurar la red de la VM en modo puente, permitiendo que podamos acceder a ella desde la PC anfitriona.

En cuanto al último tema que estamos viendo, arquitectura de aplicaciones, tuvimos en cuenta que lo que hicimos fue una clara demostración de la arquitectura cliente-servidor, donde la VM actúa como el servidor, quién contiene la API y es consultado por uno o varios clientes (en este caso nuestra PC anfitriona), devolviendo una respuesta.

Como dificultad principal, encontramos el manejo del servidor sin entorno gráfico, ya que trabajar únicamente desde la terminal, utilizar combinaciones de teclas y mantener el formato correcto del código fue un reto al principio. Sin embargo, con práctica y paciencia, logramos adaptarnos y ejecutar correctamente el programa.

En resumen, nos gustó mucho llevar a cabo este trabajo y, mediante la práctica, comprender cómo todos estos conceptos que aprendimos no se quedan en la teoría, sino que son clave para llevar a cabo nuestra tarea como programadores.

Bibliografía

Amazon Web Services. (s. f.). *¿Qué es la virtualización?* Recuperado el 18 de octubre de 2025, de <https://aws.amazon.com/es/what-is/virtualization/>

IBM. (s. f.). *Virtualización*. Recuperado el 18 de octubre de 2025, de <https://www.ibm.com/mx-es/think/topics/virtualization>

Radic, S. (2023, 30 de agosto). *Ubuntu Server: Instalación, configuración y beneficios*. Contabo blog. <https://contabo.com/blog/es/ubuntu-server-instalacion-configuracion-y-beneficios/>

Red Hat. (s. f.). *¿Qué es la virtualización?* Recuperado el 18 de octubre de 2025, de <https://www.redhat.com/es/topics/virtualization>

Red Hat. (s. f.). *¿Qué es un hipervisor?* Recuperado el 18 de octubre de 2025, de <https://www.redhat.com/es/topics/virtualization/what-is-a-hypervisor>

Red Hat. (s. f.). *¿Qué es una API de REST?* Recuperado el 18 de octubre de 2025, de <https://www.redhat.com/es/topics/api/what-is-a-rest-api>

Tapp, F. (2023, 20 de septiembre). *What's the Difference Between NAT, Bridge, and Host-Only Network Modes?* MakeUseOf. <https://www.makeuseof.com/whats-the-difference-nat-bridge-host-only-network-modes/>

Universidad Tecnológica Nacional. (s. f.). *UNIDAD 7: Virtualización que es un hipervisor* [Material de curso]. TUPAD.