

**INSTITUTO POLITÉCNICO
NACIONAL**



ESCUELA SUPERIOR DE CÓMPUTO

Práctica 1: “Tienda en Línea”

Nombres:

Corona Hernández Martín Rafael

Quevedo Zepeda Ximena Vianney

Materia:

Aplicaciones para Comunicaciones en Red

Grupo:

6CM2

Profesor:

Axel Ernesto Moreno Cervantes

1. Introducción

La presente práctica tiene como objetivo implementar una aplicación cliente-servidor para la venta de artículos en línea, utilizando el **API de sockets de flujo y bloqueantes**.

Los **sockets** son mecanismos de comunicación que permiten el intercambio de datos entre procesos, ya sea en la misma máquina o en diferentes computadoras a través de una red.

En esta práctica, se emplean **sockets de flujo (TCP)**, los cuales garantizan una comunicación confiable, orientada a conexión y en orden. Además, se trabaja con **sockets bloqueantes**, lo que significa que cada operación de lectura o escritura espera hasta que el dato sea recibido o enviado correctamente antes de continuar.

2. Objetivos

Objetivo General

- Desarrollar una aplicación cliente-servidor que permita la compra de artículos en línea mediante la comunicación con sockets.

Objetivos Específicos

- Implementar un menú interactivo en línea de comandos o interfaz gráfica (GUI).
- Permitir la búsqueda y listado de artículos.
- Administrar un carrito de compras con opciones de agregar, eliminar y editar productos.
- Validar las existencias antes de realizar la compra.
- Generar un ticket al finalizar la transacción.

3. Marco Teórico

Los sistemas distribuidos requieren que varios procesos se comuniquen entre sí, ya sea dentro de la misma máquina o a través de una red. Para esto se utilizan **protocolos de comunicación**, los cuales definen cómo se envían y reciben datos.

Los **sockets** son una herramienta fundamental en este contexto, ya que permiten implementar dicha comunicación de manera estandarizada.

¿Qué es un socket?

Un **socket** es un punto final de comunicación entre dos procesos. Puede verse como un “enchufe lógico” donde un proceso cliente y uno servidor se conectan para intercambiar datos.

Cada socket se identifica con:

- **Dirección IP:** la máquina en la que se encuentra el proceso.
- **Puerto:** el número que identifica la aplicación.
- **Protocolo de transporte:** TCP o UDP.

Tipos de sockets

1. Sockets de flujo (Stream sockets):

- Basados en **TCP (Transmission Control Protocol)**.
- Orientados a conexión (primero se establece una conexión, luego se transmiten datos).
- Confiables: garantizan entrega y orden de los datos.
- Se usan en aplicaciones donde la pérdida de información no es tolerable, como compras en línea, mensajería o transferencias bancarias.

2. Sockets de datagramas (Datagram sockets):

- Basados en **UDP (User Datagram Protocol)**.
- No orientados a conexión.
- No garantizan entrega ni orden de los datos, pero son más rápidos.
- Se usan en aplicaciones como juegos en línea o videollamadas.

En esta práctica se usaron **sockets de flujo (TCP)** porque necesitamos seguridad y confiabilidad en las transacciones de la tienda en línea.

Sockets bloqueantes y no bloqueantes

- **Sockets bloqueantes:**

- Cada operación (**recv**, **send**) **detiene el programa** hasta que termine.
- Son más fáciles de implementar y entender.
- Limitación: si no hay respuesta, el programa queda “congelado” esperando.
- **Sockets no bloqueantes:**
 - No detienen el flujo del programa.
 - Requieren técnicas más avanzadas como multihilos, select o poll para manejar varios clientes al mismo tiempo.
 - Son más eficientes, pero también más complejos.

En esta práctica se trabajó con **sockets de flujo bloqueantes**, ya que simplifican la comunicación y son ideales para aprender el funcionamiento básico de cliente-servidor.

4. Desarrollo

La práctica se diseñó con una arquitectura **cliente-servidor**:

Servidor

Administra la base de datos de artículos (nombre, marca, tipo, precio, existencias). Asimismo se encarga del registro de usuarios, atiende las solicitudes de los clientes y actualiza las existencias después de cada compra.

El sistema por parte del servidor puede realizar las siguientes actividades:

- **Gestión de Cuentas de Usuario:** Permite a los clientes crear una cuenta nueva y acceder a ella de forma segura para realizar sus compras.
- **Administración del Catálogo de Productos:** Controla el inventario de productos. Cada artículo tiene detalles como nombre, marca, precio y la cantidad de unidades disponibles (stock).
- **Navegación y Búsqueda:** Los clientes pueden buscar productos específicos por nombre o marca, y también pueden explorar el catálogo filtrando por categorías (por ejemplo, "Laptops", "Accesorios", etc.).
- **Carrito de Compras:** Cada cliente dispone de un carrito de compras personal. Puede agregar productos que le interesen, ver el contenido de su carrito en cualquier momento y eliminar artículos si cambia de opinión.

- **Proceso de Compra (Checkout):** Cuando un cliente finaliza su compra, el sistema realiza dos acciones cruciales:
- **Actualiza el inventario:** Reduce automáticamente el stock de los productos comprados, asegurando que la tienda no venda más de lo que tiene.
- **Genera un recibo:** Crea un ticket de compra profesional en formato PDF con el detalle de la transacción, el total pagado y los datos del cliente.
- **Capacidad para Múltiples Usuarios:** Está diseñado para atender a numerosos clientes de forma simultánea. La compra de un usuario no interfiere ni ralentiza la experiencia de los demás.

Cliente

Es la interfaz de usuario o el punto de acceso para los clientes de la tienda en línea. Sirve para conectarse al sistema central de la tienda, explorar productos y realizar compras.

El sistema por parte del cliente puede realizar las siguientes actividades:

- **Conexión a la Tienda:** Al iniciar, el programa pide la dirección del servidor de la tienda. Esto es similar a escribir la dirección de una página web en un navegador para poder visitarla.
- **Acceso Seguro:** Antes de poder comprar, el sistema requiere que el usuario se identifique. Se le presentan dos opciones: crear una nueva cuenta de cliente (registrarse) o ingresar con sus credenciales existentes (iniciar sesión). Este paso es obligatorio para garantizar la seguridad.
- **Menú Principal :** Una vez dentro, el usuario ve un menú principal claro con todas las acciones disponibles, como:
 - Buscar productos
 - Listar artículos por tipo
 - Agregar al carrito
 - Ver el carrito
 - Eliminar del carrito
 - finalizar la compra
 - Salir
 - Mostrar todos los productos
- **Gestión del Carrito de Compras:** Puede agregar productos que le interesan, revisar el contenido y el costo total en cualquier momento, y eliminar artículos si cambia de parecer.
- **Finalización de la Compra:** Cuando el usuario está listo para pagar, selecciona la opción correspondiente. La aplicación se comunica con el servidor para procesar el pedido y, a continuación, muestra en pantalla un resumen detallado de la compra, a modo de ticket digital.
- **Salida Segura:** Al terminar, el cliente puede salir del programa, lo que finaliza su sesión y lo desconecta del servidor de la tienda.

Validaciones

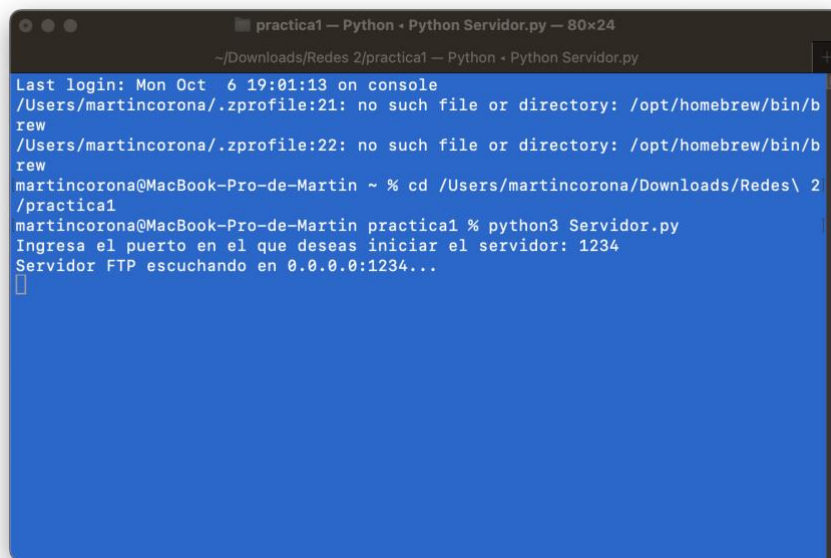
- Antes de agregar un artículo al carrito, se verifica que exista en inventario.
- Al confirmar la compra, se descuentan las existencias en el servidor.

Flujo de la aplicación

1. El cliente se conecta al servidor.
2. El usuario selecciona opciones desde el menú.
3. El servidor responde con la información solicitada.
4. Al finalizar la compra, se genera un **ticket digital** con el detalle de los artículos, cantidades, precios y total.

5. PRUEBAS

SERVIDOR



```
practica1 — Python · Python Servidor.py — 80x24
~/Downloads/Redes 2/practica1 — Python · Python Servidor.py
Last login: Mon Oct 6 19:01:13 on console
/Users/martincorona/.zprofile:21: no such file or directory: /opt/homebrew/bin/brew
rew
/Users/martincorona/.zprofile:22: no such file or directory: /opt/homebrew/bin/brew
rew
martincorona@MacBook-Pro-de-Martin ~ % cd /Users/martincorona/Downloads/Redes\ 2 /practica1
martincorona@MacBook-Pro-de-Martin practica1 % python3 Servidor.py
Ingresa el puerto en el que deseas iniciar el servidor: 1234
Servidor FTP escuchando en 0.0.0.0:1234...
█
```

Se pide un número de puerto para que se pueda conectar un cliente.

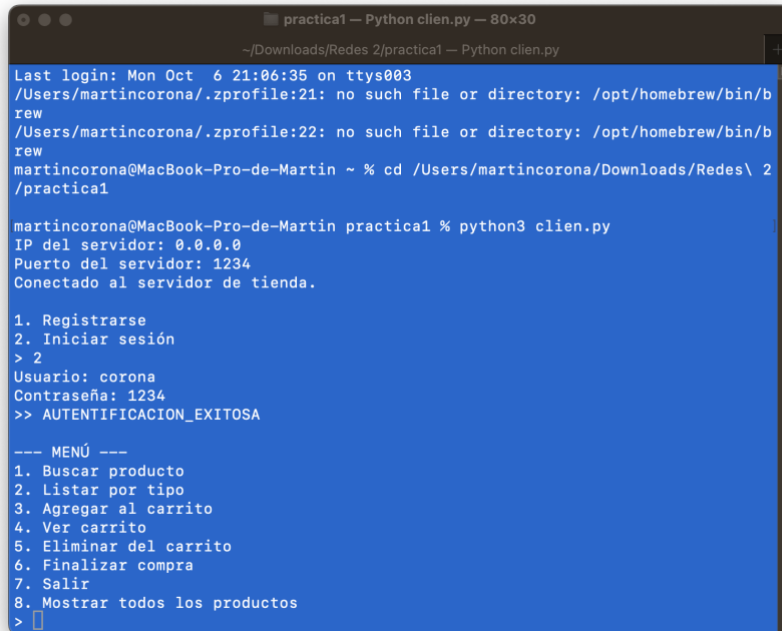
```
practica1 — Python • Python Servidor.py — 80x24
~/Downloads/Redes 2/practica1 — Python • Python Servidor.py
Last login: Mon Oct 6 19:01:13 on console
/Users/martincorona/.zprofile:21: no such file or directory: /opt/homebrew/bin/brew
/Users/martincorona/.zprofile:22: no such file or directory: /opt/homebrew/bin/brew
martincorona@MacBook-Pro-de-Martin ~ % cd /Users/martincorona/Downloads/Redes\ 2 /practica1
martincorona@MacBook-Pro-de-Martin practica1 % python3 Servidor.py
Ingresa el puerto en el que deseas iniciar el servidor: 1234
Servidor FTP escuchando en 0.0.0.0:1234...
Conexión establecida con ('127.0.0.1', 50729)
```

Una vez que se conecta un cliente, aparece un mensaje el cual muestra que la conexión fue exitosa, y la información del mismo.

```
practica1 — Python • Python Servidor.py — 80x25
~/Downloads/Redes 2/practica1 — Python • Python Servidor.py
Last login: Mon Oct 6 19:01:13 on console
/Users/martincorona/.zprofile:21: no such file or directory: /opt/homebrew/bin/brew
/Users/martincorona/.zprofile:22: no such file or directory: /opt/homebrew/bin/brew
martincorona@MacBook-Pro-de-Martin ~ % cd /Users/martincorona/Downloads/Redes\ 2 /practica1
martincorona@MacBook-Pro-de-Martin practica1 % python3 Servidor.py
Ingresa el puerto en el que deseas iniciar el servidor: 1234
Servidor FTP escuchando en 0.0.0.0:1234...
Conexión establecida con ('127.0.0.1', 50729)
Ticket generado: ticket_corona_20251006_212015.pdf
```

Una vez que un cliente logra completar una compra aparece un mensaje de que se ha generado un documento en PDF.

CLIENTE



```
practica1 — Python clien.py — 80x30
~/Downloads/Redes 2/practica1 — Python clien.py
Last login: Mon Oct  6 21:06:35 on ttys003
/Users/martincorona/.zprofile:21: no such file or directory: /opt/homebrew/bin/brew
/Users/martincorona/.zprofile:22: no such file or directory: /opt/homebrew/bin/brew
martincorona@MacBook-Pro-de-Martin ~ % cd /Users/martincorona/Downloads/Redes\ 2 /practica1
martincorona@MacBook-Pro-de-Martin practica1 % python3 clien.py
IP del servidor: 0.0.0.0
Puerto del servidor: 1234
Conectado al servidor de tienda.

1. Registrarse
2. Iniciar sesión
> 2
Usuario: corona
Contraseña: 1234
>> AUTENTICACION_EXITOSA

--- MENÚ ---
1. Buscar producto
2. Listar por tipo
3. Agregar al carrito
4. Ver carrito
5. Eliminar del carrito
6. Finalizar compra
7. Salir
8. Mostrar todos los productos
> 
```

Como primer paso para el cliente se le pide una dirección IP, la cual sería como una url de una tienda en línea.

Después se le pide al usuario registrarse o iniciar sesión según sea el caso, una vez se inicia sesión sale el menú principal, el cual permite realizar las consultas o compra de productos.


```
practical1 — Python clien.py — 86x52
~/Downloads/Redes 2/practical1 — Python clien.py

6. Eliminar del carrito
6. Finalizar compra
7. Salir
8. Mostrar todos los productos
> 8

--- TODOS LOS PRODUCTOS DISPONIBLES ---
[1] Laptop HP Pavilion - HP - Computadora - $12000 (stock 4)
[2] Mouse Logitech MX Master 3 - Logitech - Accesorio - $1800 (stock 20)
[3] Monitor Samsung Odyssey G5 - Samsung - Pantalla - $7500 (stock 10)
[4] Teclado Redragon Kumara K552 - Redragon - Accesorio - $700 (stock 15)
[5] Tablet Lenovo Tab P11 - Lenovo - Tablet - $4500 (stock 8)
[6] MacBook Air M3 - Apple - Computadora - $24500 (stock 7)
[7] Audifonos Sony WH-1000XM5 - Sony - Audio - $6500 (stock 18)
[8] Webcam Logitech C920 - Logitech - Accesorio - $1500 (stock 25)
[9] Impresora Epson EcoTank L3250 - Epson - Impresora - $4200 (stock 12)
[10] Disco Duro Externo Seagate 2TB - Seagate - Almacenamiento - $1300 (stock 30)
[11] Smartwatch Garmin Forerunner 55 - Garmin - Accesorio - $3800 (stock 9)
[12] iPhone 16 Pro - Apple - Celular - $28500 (stock 10)
[13] Tarjeta de Video NVIDIA RTX 4070 - NVIDIA - Componente PC - $11500 (stock 6)
[14] Monitor Gamer LG UltraGear 27 pulgadas - LG - Pantalla - $6800 (stock 11)
[15] Bocina Bluetooth JBL Flip 6 - JBL - Audio - $2600 (stock 22)
[16] Consola Xbox Series X - Microsoft - Consola - $11000 (stock 9)
[17] SSD Interno Samsung 980 Pro 1TB - Samsung - Almacenamiento - $1900 (stock 28)
[18] Router WiFi 6 TP-Link Archer AX55 - TP-Link - Accesorio - $1600 (stock 15)
[19] Drone DJI Mini 3 - DJI - Dron - $10500 (stock 7)
[20] Silla Gamer Corsair T3 Rush - Corsair - Accesorio - $5500 (stock 13)

--- MENÚ ---
1. Buscar producto
2. Listar por tipo
3. Agregar al carrito
4. Ver carrito
5. Eliminar del carrito
6. Finalizar compra
7. Salir
8. Mostrar todos los productos
> 6

--- TICKET ---
Usuario: corona
Fecha: 2025-10-06 21:20:15
Productos:
- Laptop HP Pavilion - $12000
Total: $12000

--- MENÚ ---
1. Buscar producto
2. Listar por tipo
3. Agregar al carrito
4. Ver carrito
```

En esta imagen se puede observar la totalidad de productos, en este ejemplo se selecciona un producto el cual es una computadora HP la cual se logra la compra, de primera vista se aprecia que genera un ticket breve, con la descripción del producto o productos y el monto a pagar.

Ticket de Compra

Usuario: corona
Fecha: 2025-10-06 21:20:15

Producto	Precio
Laptop HP Pavilion	\$12000.00
TOTAL	\$12000.00

Pero en realidad se genera un documento en PDF.

6. CONCLUSIONES (INDIVIDUALES)

Quevedo Zepeda Ximena Vianney

En esta práctica se desarrolla una aplicación cliente-servidor que simula el funcionamiento de una tienda en línea. Para la comunicación entre cliente y servidor se utilizan **sockets de flujo bloqueantes**, que permiten el intercambio confiable de mensajes en una conexión TCP.

El objetivo principal es comprender cómo aplicar el modelo cliente-servidor en la construcción de un sistema sencillo de comercio electrónico, donde el cliente pueda buscar artículos, listarlos, agregarlos a un carrito de compras, editarlos y finalizar la compra. Por su parte, el servidor se encarga de administrar las existencias y generar el ticket final.

Como estudiante de ESCOM, considero que esta práctica es un buen acercamiento al desarrollo de aplicaciones distribuidas, ya que nos muestra de forma clara cómo funcionan los mecanismos básicos de comunicación en red que posteriormente se aplican en sistemas más grandes como servicios web o aplicaciones empresariales. Además, trabajar con sockets bloqueantes me ayudó a entender la importancia de la sincronización entre cliente y servidor, reforzando los fundamentos que son base de la programación de sistemas abiertos.

Corona Hernández Martín Rafael

El uso de sockets de flujo bloqueante en la implementación de una tienda en línea permite comprender de manera práctica los fundamentos de la comunicación cliente-servidor y la transmisión confiable de datos mediante el protocolo TCP (Transmission Control Protocol). Este enfoque posibilita el establecimiento de una conexión persistente entre el cliente y el servidor, garantizando la integridad de la información intercambiada como el registro de usuarios, las solicitudes de productos y las confirmaciones de compra gracias al control de flujo y los mecanismos de verificación que caracterizan al protocolo TCP.

El desarrollo de este tipo de sistemas ayuda a el entendimiento de procesos esenciales como la sincronización de eventos, la gestión de concurrencia y la serialización de datos para su transmisión en red. Sin embargo, también ayuda a identificar las limitaciones inherentes a los sockets bloqueantes, como la dependencia del tiempo de respuesta del servidor y la posible ineficiencia en entornos con múltiples clientes simultáneos, donde cada conexión requiere un hilo o proceso independiente.

7.REFERENCIAS

Python Software Foundation. (s.f.). *socket — Low-level networking interface*.

Python 3.12.4 documentation. Recuperado el 5 de octubre de 2025, de

<https://docs.python.org/3/library/socket.html>

Real Python. (s.f.). *Socket programming in Python (guide)*. Recuperado el 4 de

octubre de 2025, de <https://realpython.com/python-sockets/>

Oracle. (s.f.). *Socket (Java SE 21 & JDK 21)*. Java® Platform, Standard Edition & Java

Development Kit version 21 API specification. Recuperado el 4 de octubre de

2025, de

<https://docs.oracle.com/en/java/javase/21/docs/api/java.net.Socket.html>