INSTITUTO POLITÉCNICO NACIONAL.

ESCUELA SUPERIOR DE CÓMPUTO.

APLICACIONES PARA COMUNICACIONES EN RED.

PRÁCTICA 1.

PROFESOR: AXEL ERNESTO MORENO CERVANTES.

GRUPO 6CM1.

ALUMNA: SÁNCHEZ VALERIANO ALEXANDRA.

FECHA: 29/09/2025

INTRODUCCIÓN.

En el mundo actual, donde gran parte de las interacciones comerciales se realizan en entornos digitales, la programación de aplicaciones cliente-servidor se ha convertido en una herramienta esencial para el desarrollo de sistemas capaces de reproducir experiencias reales en un entorno virtual. Uno de los principales retos de este tipo de aplicaciones no es únicamente mostrar información en pantalla, sino garantizar que dicha información sea confiable, se mantenga actualizada y pueda manejarse de manera simultánea entre múltiples usuarios. Con esta motivación, el presente proyecto busca simular una tienda en línea con funcionalidades básicas, pero con un nivel de complejidad suficiente para poner en práctica conceptos fundamentales como la programación de redes, la concurrencia, el diseño modular y la transferencia de datos multimedia.

Este reporte analiza una aplicación de comercio electrónico cliente-servidor desarrollada en Java. El sistema está compuesto por dos entidades principales: el servidor, encargado de gestionar el inventario de la tienda, y el cliente, que proporciona una interfaz gráfica para que el usuario final interactúe con los productos. La comunicación entre ambos se establece mediante el uso de sockets, lo que permite el envío y recepción de datos en tiempo real, asegurando la interacción continua entre los participantes del sistema.

El planteamiento inicial del proyecto fue claro: diseñar un sistema en el cual uno o varios clientes pudieran conectarse al servidor para consultar el catálogo de productos, realizar compras y recibir confirmaciones de transacciones. Más allá de mostrar un catálogo estático, el objetivo era implementar una dinámica de comunicación bidireccional en la que los cambios en el inventario se reflejarán de forma inmediata y coherente. Este enfoque plantea retos significativos relacionados con la sincronización de datos, el manejo de múltiples conexiones concurrentes, el diseño modular de las clases y la definición de un protocolo de comunicación que permitiera a cliente y servidor “hablar el mismo idioma” de forma ordenada.

Desde el punto de vista técnico, el sistema se apoya en el modelo cliente-servidor. El cliente es quien inicia las solicitudes, mientras que el servidor responde a ellas siguiendo un protocolo predefinido. Para implementar este esquema, se utilizaron las clases **java.net.Socket** y **java.net.ServerSocket**, que crean un canal de comunicación bidireccional entre ambas partes. El intercambio de información se lleva a cabo mediante flujos de entrada y salida (**InputStreamReader** y **PrintWriter**), lo que facilita la transferencia de datos en formato de texto y permite al cliente recibir respuestas línea por línea de manera estructurada.

DESARROLLO.

El proyecto se compone de cuatro clases principales:.

1. **SERVIDOR.JAVA - EL GESTOR DEL INVENTARIO.**

La clase Servidor es el *backend* de toda la operación de la tienda.

- **Gestión de Conexiones**: El método **iniciar()** crea un **ServerSocket** que escucha activamente en el puerto **9999**. Al detectar una nueva conexión entrante con **servidorSocket.accept()**, en lugar de manejarla directamente, delega la tarea a un nuevo hilo (**new ManejadorCliente(clienteSocket).start()**). Este enfoque **multi-hilo** es esencial para la concurrencia, permitiendo que el servidor atienda a varios clientes simultáneamente sin que uno bloquee a los demás, una característica fundamental para cualquier aplicación de red escalable.

- **Inicialización de Datos**: El método **generarProductos()** simula una base de datos. Crea un arreglo de objetos **Producto**, donde cada uno se inicializa con datos predefinidos. La parte crucial es la carga de imágenes: el servidor lee archivos JPG desde la carpeta de recursos **src/main/resources/imagenes_productos** utilizando un **InputStream**, convierte el flujo de bytes a un **byte[]** y lo asigna a la propiedad **imagen** del objeto **Producto**.

- **Protocolo de Aplicación**: El servidor define y sigue un protocolo simple para comunicarse con el cliente.

    - **Solicitud GET_CATALOG**: El servidor lee la solicitud del cliente. Si el comando es **"GET_CATALOG"**, entra en un bucle que recorre el arreglo de productos. Para cada producto, construye una cadena de texto que incluye el ID, nombre, precio, existencias, descripción y una cadena **Base64** de la imagen. La cadena se envía línea por línea al cliente y finaliza con la señal **"CATALOG_END"**.

    - **Solicitud BUY_PRODUCTS**: Si el comando es **"BUY_PRODUCTS"**, el servidor entra en un bucle para leer cada producto y su cantidad enviados por el cliente. Utiliza el ID y la cantidad para invocar a **actualizarExistencias()**, lo que decrementa el stock disponible. Finalmente, construye y envía un ticket de compra al cliente con la información de los productos y el total, finalizando con la señal **"TICKET_END".**

2. CLIENTE.JAVA - LA INTERFAZ DE USUARIO Y EL CONSUMIDOR DE DATOS.

El **Cliente** es la aplicación con la que el usuario final interactúa.

- **Interfaz Gráfica (GUI)**: La interfaz está construida con **Swing**, una librería de Java para la creación de interfaces gráficas. La ventana se divide en dos paneles principales: un **JList** para mostrar el catálogo y un panel de detalles que se actualiza dinámicamente al seleccionar un producto.

- **Conexión y Petición de Datos**: El método **cargarCatalogoAutomaticamente()** establece una conexión con el servidor. Envía la cadena **"GET_CATALOG"** a través del **PrintWriter** y luego, en un bucle, lee línea por línea la respuesta del servidor con el **BufferedReader**.

- **Manejo de Imágenes**: Al recibir una línea que contiene los datos de un producto, el cliente utiliza **Base64.getDecoder().decode(imagenBase64)** para convertir la cadena de texto de la imagen de vuelta a su formato original de **byte[]**. Con estos bytes, crea un **ImageIcon** que luego se escala y se muestra en un **JLabel** de la interfaz.

- **Lógica del Carrito**: La clase **CarritoDialog** contiene la lógica del carrito de compras. El método **anadirProductoAlCarrito()** agrega una nueva instancia de **Producto** al carrito. La notable mejora, señalada en el código, es el nuevo método **eliminarUnaUnidad().** Esta función es crucial para la integridad de los datos. Al recibir un ID de producto, busca el producto en el carrito, decrementa su cantidad en uno y, si la cantidad llega a cero, lo elimina completamente. Lo más importante es que esta acción incrementa las existencias en la interfaz principal del catálogo, lo que evita desincronizaciones de stock entre la vista del usuario y el carrito.

3. PRODUCTO.JAVA Y TICKET.JAVA - LOS MODELOS DE DATOS

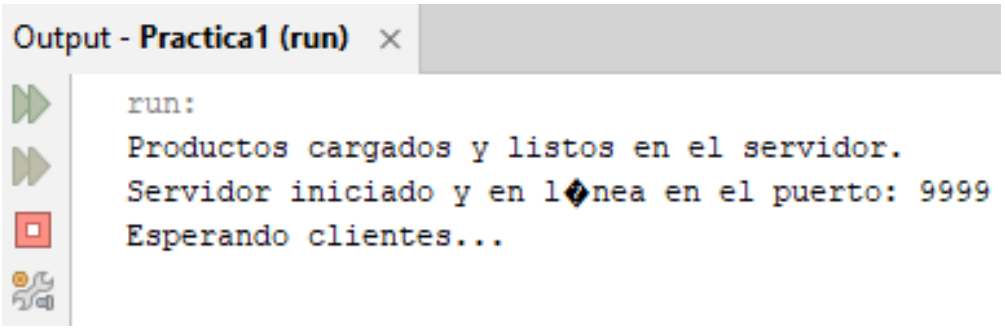
Estas clases son fundamentales para estructurar y transportar la información en el sistema.

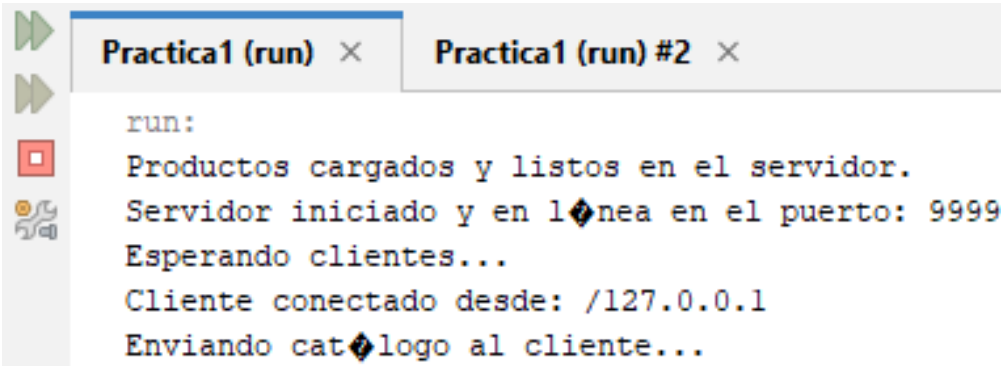
- **Producto:** Es un modelo de datos simple. Implementa **Serializable** para ser compatible con la transmisión a través de la red, aunque en este proyecto se optó por un protocolo basado en texto para la mayoría de los datos. El campo **private byte[] imagen;** es una mejora clave que permite almacenar los datos binarios de la imagen directamente dentro del objeto, lo que facilita su manejo en el cliente y servidor.

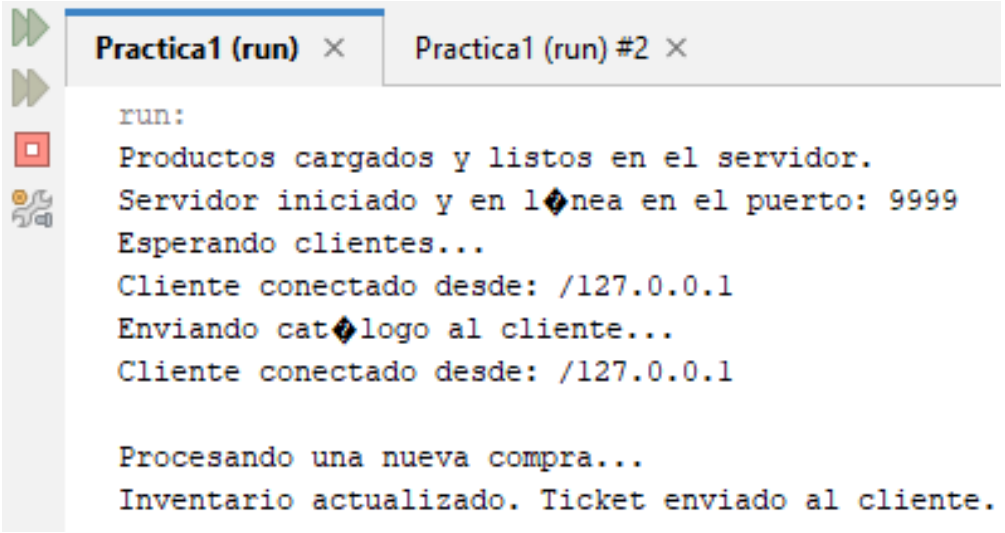
- **Ticket:** Una clase de conveniencia que encapsula la información de una compra finalizada. Al igual que **Producto**, es **Serializable**, lo que facilita su manejo y potencial transporte en el futuro.

PRUEBAS.

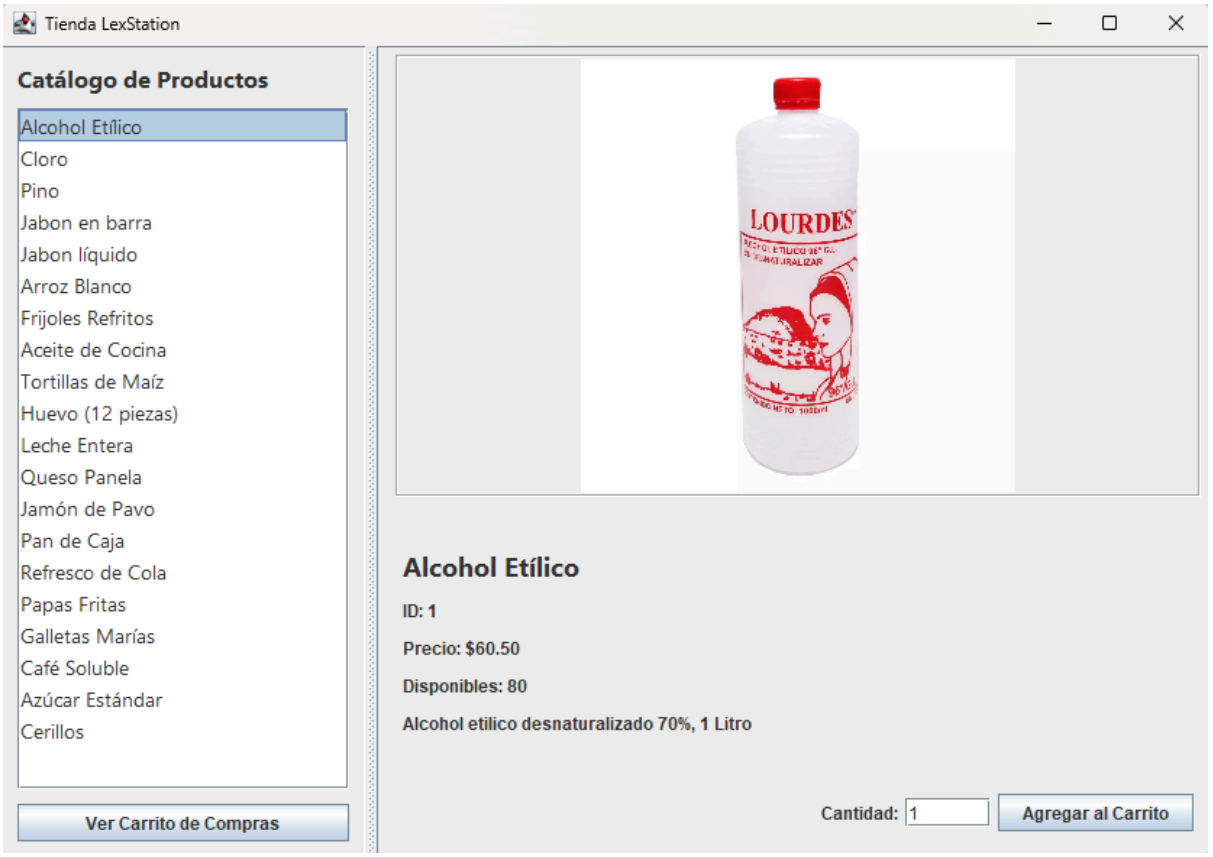
- **SERVIDOR.**

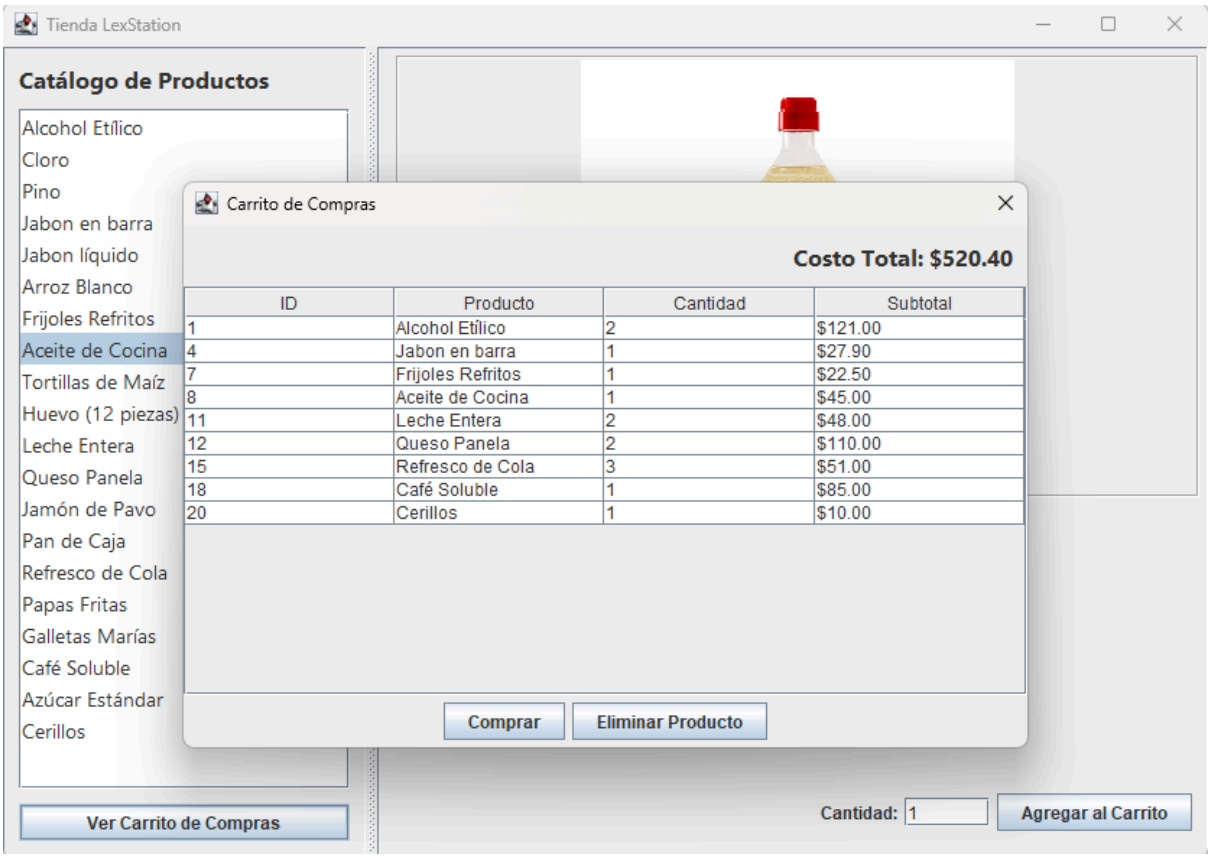


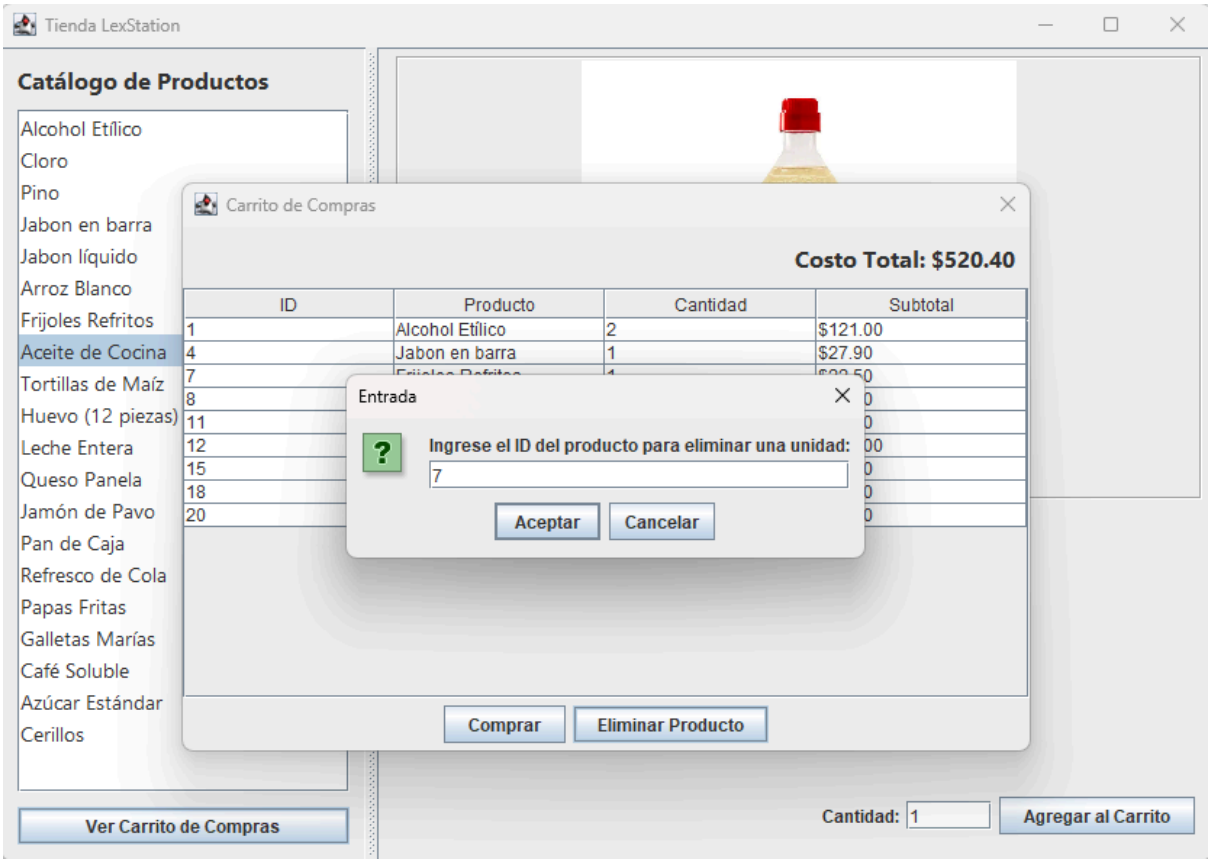


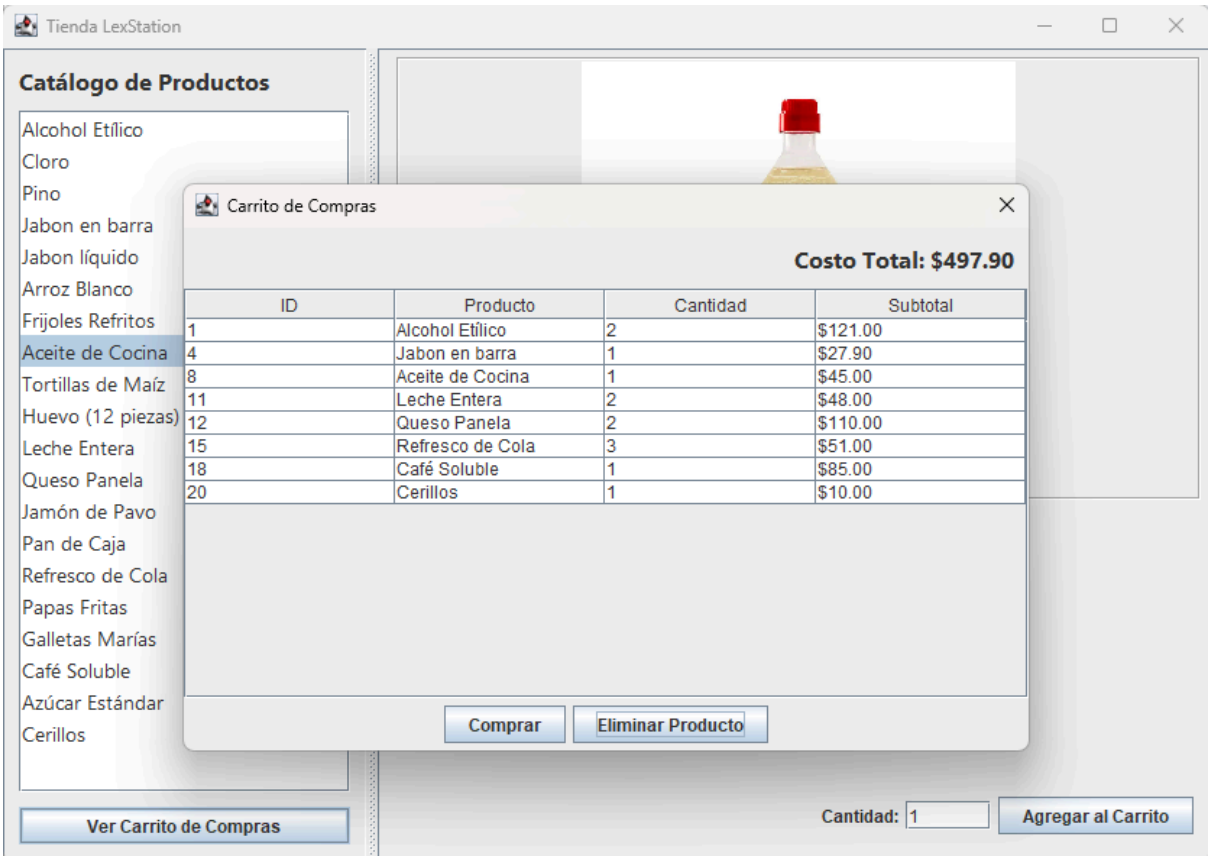


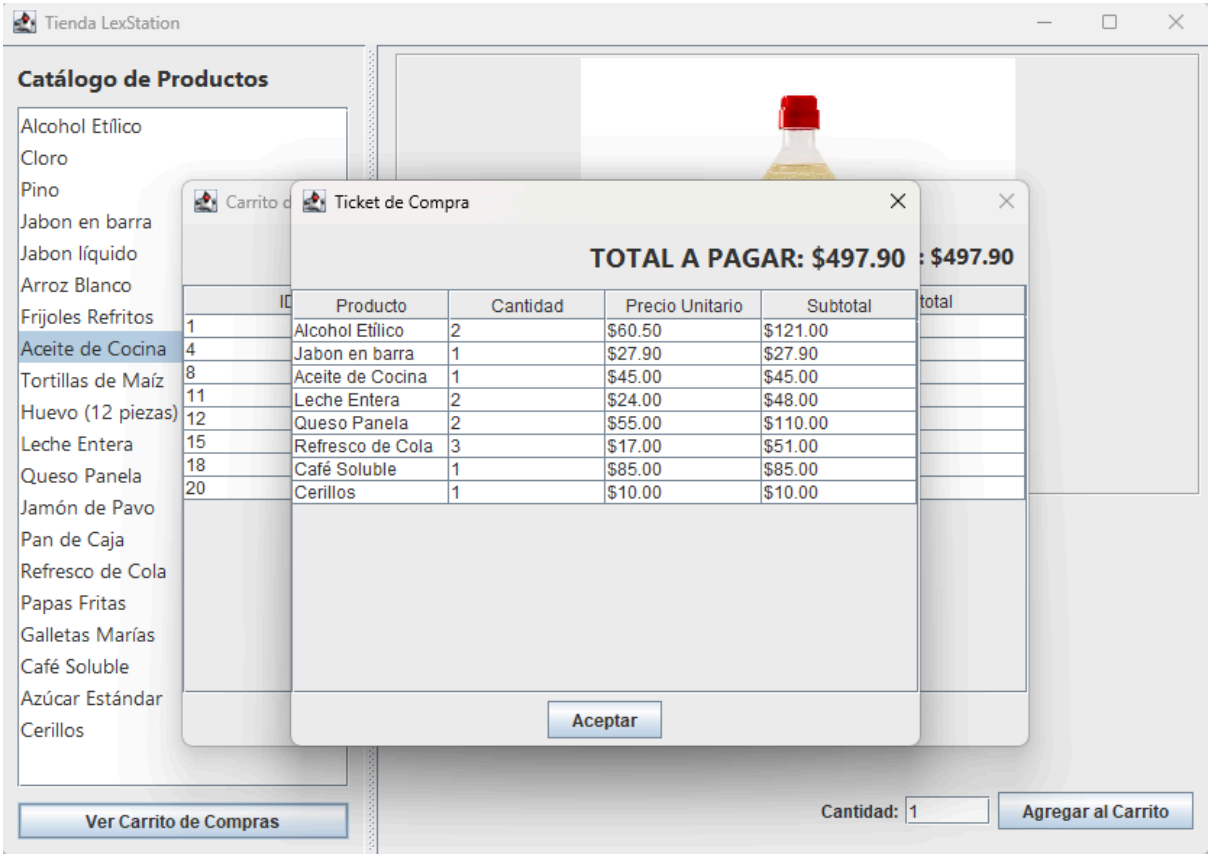
- **CLIENTE.**

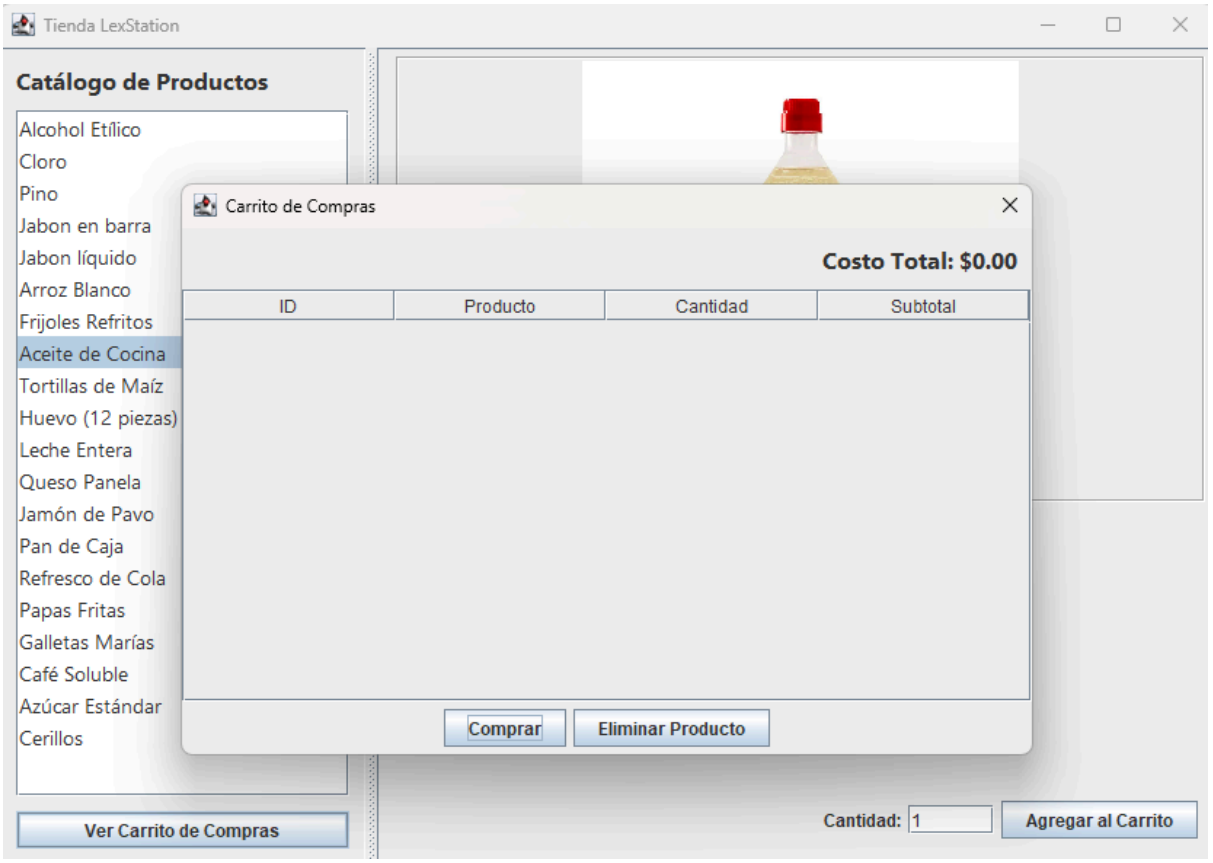

Tienda LexStation
Catálogo de Productos
Alcohol Etílico
Cloro
Pino
Jabon en barra
Jabon líquido
Arroz Blanco
Frijoles Refritos
Aceite de Cocina
Tortillas de Maíz
Huevo (12 piezas)
Leche Entera
Queso Panela
Jamón de Pavo
Pan de Caja
Refresco de Cola
Papas Fritas
Galletas Marías
Café Soluble
Azúcar Estándar
Cerillos
Ver Carrito de Compras
LOURDES
Alcohol Etílico
ID: 1
Precio: $60.50
Disponibles: 80
Alcohol etilico desnaturalizado 70%, 1 Litro
Cantidad: 1
Agregar al Carrito


Tienda LexStation
Catálogo de Productos
Alcohol Etílico
Cloro
Pino
Jabon en barra
Jabon líquido
Arroz Blanco
Frijoles Refritos
Aceite de Cocina
Tortillas de Maíz
Huevo (12 piezas)
Leche Entera
Queso Panela
Jamón de Pavo
Pan de Caja
Refresco de Cola
Papas Fritas
Galletas Marías
Café Soluble
Azúcar Estándar
Cerillos
Ver Carrito de Compras
Carrito de Compras
Costo Total: $520.40
ID
Producto
Cantidad
Subtotal
1 Alcohol Etílico 2 $121.00
4 Jabon en barra 1 $27.90
7 Frijoles Refritos 1 $22.50
8 Aceite de Cocina 1 $45.00
11 Leche Entera 2 $48.00
12 Queso Panela 2 $110.00
15 Refresco de Cola 3 $51.00
18 Café Soluble 1 $85.00
20 Cerillos 1 $10.00
Comprar
Eliminar Producto
Cantidad: 1
Agregar al Carrito


Tienda LexStation
Catálogo de Productos
Alcohol Etílico
Cloro
Pino
Jabon en barra
Jabon líquido
Arroz Blanco
Frijoles Refritos
Aceite de Cocina
Tortillas de Maíz
Huevo (12 piezas)
Leche Entera
Queso Panela
Jamón de Pavo
Pan de Caja
Refresco de Cola
Papas Fritas
Galletas Marías
Café Soluble
Azúcar Estándar
Cerillos
Ver Carrito de Compras
Carrito de Compras
Costo Total: $520.40
ID
Producto
Cantidad
Subtotal
1
Alcohol Etílico
2
$121.00
4
Jabon en barra
1
$27.90
7
8
11
12
15
18
20
Entrada
Ingrese el ID del producto para eliminar una unidad:
7
Aceptar
Cancelar
Comprar
Eliminar Producto
Cantidad: 1
Agregar al Carrito


Tienda LexStation
Catálogo de Productos
Alcohol Etílico
Cloro
Pino
Jabon en barra
Jabon líquido
Arroz Blanco
Frijoles Refritos
Aceite de Cocina
Tortillas de Maíz
Huevo (12 piezas)
Leche Entera
Queso Panela
Jamón de Pavo
Pan de Caja
Refresco de Cola
Papas Fritas
Galletas Marías
Café Soluble
Azúcar Estándar
Cerillos
Ver Carrito de Compras
Carrito de Compras
Costo Total: $497.90
ID
Producto
Cantidad
Subtotal
1
Alcohol Etílico
2
$121.00
4
Jabon en barra
1
$27.90
8
Aceite de Cocina
1
$45.00
11
Leche Entera
2
$48.00
12
Queso Panela
2
$110.00
15
Refresco de Cola
3
$51.00
18
Café Soluble
1
$85.00
20
Cerillos
1
$10.00
Comprar
Eliminar Producto
Cantidad: 1
Agregar al Carrito


Tienda LexStation
Catálogo de Productos
Alcohol Etílico
Cloro
Pino
Jabon en barra
Jabon líquido
Arroz Blanco
Frijoles Refritos
Aceite de Cocina
Tortillas de Maíz
Huevo (12 piezas)
Leche Entera
Queso Panela
Jamón de Pavo
Pan de Caja
Refresco de Cola
Papas Fritas
Galletas Marías
Café Soluble
Azúcar Estándar
Cerillos
Ver Carrito de Compras
Ticket de Compra
TOTAL A PAGAR: $497.90
Producto | Cantidad | Precio Unitario | Subtotal
Alcohol Etílico | 2 | $60.50 | $121.00
Jabon en barra | 1 | $27.90 | $27.90
Aceite de Cocina | 1 | $45.00 | $45.00
Leche Entera | 2 | $24.00 | $48.00
Queso Panela | 2 | $55.00 | $110.00
Refresco de Cola | 3 | $17.00 | $51.00
Café Soluble | 1 | $85.00 | $85.00
Cerillos | 1 | $10.00 | $10.00
Aceptar
Cantidad: 1
Agregar al Carrito


Tienda LexStation
Catálogo de Productos
Alcohol Etílico
Cloro
Pino
Jabon en barra
Jabon líquido
Arroz Blanco
Frijoles Refritos
Aceite de Cocina
Tortillas de Maíz
Huevo (12 piezas)
Leche Entera
Queso Panela
Jamón de Pavo
Pan de Caja
Refresco de Cola
Papas Fritas
Galletas Marías
Café Soluble
Azúcar Estándar
Cerillos
Ver Carrito de Compras
Carrito de Compras
Costo Total: $0.00
ID | Producto | Cantidad | Subtotal
Comprar
Eliminar Producto
Cantidad: 1
Agregar al Carrito

CONCLUSIONES.

La experiencia de desarrollar este proyecto fue mucho más que sentarse a programar y hacer que el código funcionara. En realidad, lo que más me dejó fue entender cómo pequeños detalles, como la coherencia de datos entre cliente y servidor, pueden marcar la diferencia en la percepción de un sistema. El hecho de que un usuario viera un producto disponible mientras que otro ya lo había comprado reflejaba un problema real que cualquier aplicación podría enfrentar en la práctica. Resolver esa desincronización me llevó a valorar la importancia de la sincronización como un pilar fundamental en el diseño de aplicaciones distribuidas. No se trataba únicamente de arreglar un bug, sino de pensar en el sistema como un todo, donde lo que ocurre en un cliente debe reflejarse de inmediato en los demás para que la experiencia sea confiable.

El servidor, al manejar múltiples clientes con hilos, fue otra parte clave de la experiencia. En un inicio parecía un reto técnico más, pero poco a poco entendí que este tipo de soluciones son las que sostienen cualquier aplicación real en internet. Que cada cliente pueda conectarse sin bloquear a los demás es algo que damos por sentado como usuarios, pero que detrás requiere un diseño cuidadoso. Implementar esta lógica me ayudó a reforzar lo aprendido en mis estudios sobre concurrencia, llevándome de la teoría a la práctica. También me dio un vistazo a cómo las aplicaciones modernas resuelven estos problemas a gran escala, con arquitecturas distribuidas mucho más complejas.

Otro punto interesante fue enfrentarme a la transmisión de información multimedia. Aprender a manejar imágenes a través de la codificación Base64 fue más que un simple recurso técnico: fue entender cómo circulan los datos binarios en la red y cómo se transforman en algo visible y útil para el usuario. Ver cómo una secuencia de bytes terminaba mostrándose como una imagen dentro de la interfaz gráfica del cliente fue un recordatorio de que, detrás de cada aplicación que usamos todos los días, hay procesos similares que convierten lo abstracto en algo tangible. Fue también una oportunidad para consolidar mis conocimientos de redes, entendiendo no solo cómo transmitir texto o números, sino también cómo enviar y recibir datos más complejos.

En lo personal, el proyecto me permitió reforzar mis conocimientos de hilos, algo que ya había trabajado en la formación vocacional, pero que aquí encontré en un contexto más práctico y desafiante. Además, me impulsó a reflexionar sobre la diferencia entre un programa que funciona y un sistema que es confiable. Pude comprobar que muchas veces los verdaderos retos no son de sintaxis, sino de diseño, coordinación y comunicación. También aprendí que la interfaz de usuario no es algo separado del backend, sino que

ambos deben trabajar en sintonía para ofrecer una experiencia fluida. Cada vez que el carrito se actualizaba correctamente o que el stock se reflejaba de forma inmediata, entendí la importancia de esos pequeños detalles que hacen que un sistema se perciba como bien hecho.

REFERENCIAS BIBLIOGRÁFICAS.

- Deitel, P., Deitel, H. (2018). *Java: How to Program, Early Objects*. Pearson Education.
- Oracle. (s. f.). *Java SE Documentation*. Obtenido de https://docs.oracle.com/javase/8/docs/api/index.html
- RFC 4648. (2006). *The Base16, Base32, and Base64 Data Encodings*. Obtenido de https://tools.ietf.org/html/rfc4648
- Tanenbaum, A. S., & van Steen, M. (2016). *Distributed Systems: Principles and Paradigms*. Pearson Education.