



INSTITUTO POLITÉCNICO NACIONAL
Escuela Superior de Cómputo
“ESCOM”
Ingeniería en Sistemas Computacionales



Unidad de Aprendizaje:
Sistemas Operativos

Proyecto fase No. 5
Prototipo No. 2

Integrantes:

Aguilar Chávez Alexis Daniel - 4CV4
Calva Vargas Oswaldo Enrique - 4CM2
Flores Flores Rodrigo Ernesto - 4CM2
Jiménez Luna Rodrigo Efren - 4CV4

Maestro:

Jiménez Benítez José Alfredo

Fecha de entrega:

04/06/2023

Resumen

El proyecto se centrará en los principales temas de la gestión de procesos y la comunicación entre procesos en sistemas operativos. Se explorarán los conceptos de procesos e hilos, sus diferencias y usos. Se analizarán las técnicas de sincronización de procesos, incluyendo semáforos y colas de mensajes, y se discutirán las ventajas y desventajas de cada una. También se examinará la memoria compartida y cómo se utiliza para compartir datos entre procesos. Además, se discutirá la gestión de archivos en sistemas operativos y cómo los procesos pueden compartir información mediante el uso de archivos. Se explicarán los diferentes tipos de archivos y las funciones que se utilizan para manipularlos. Por último, se discutirán los dispositivos de entrada y salida y cómo los procesos y los hilos interactúan con ellos. Se describirán los diferentes tipos de dispositivos de entrada y salida y se explorará cómo los sistemas operativos proporcionan controladores de dispositivos para permitir la comunicación con estos dispositivos. En general, el proyecto proporcionará una visión general de la gestión de procesos y la comunicación entre procesos en sistemas operativos y cómo estos conceptos son esenciales para el funcionamiento de los sistemas informáticos modernos. Se discutirán las técnicas y herramientas utilizadas en la gestión de procesos y se analizarán sus aplicaciones en diferentes contextos.

Índice

Contenido

Introducción

Descripción del proyecto	... 1
Objetivos que debe cumplir el proyecto	... 1
Objetivos específicos del proyecto	... 1
Justificación	... 2
Estado del arte	... 2
Productos esperados	... 3

Marco teórico

Procesos	... 3
Hilos	... 4
Semáforos	... 5
Aplicaciones de los semáforos	... 7
Colas de mensajes	... 7
Memoria compartida	... 8
Memoria compartida distribuida	... 9
Archivos	... 9
Dispositivos de entrada y salida	...11

Prototipo 1

Análisis y diseño	...12
Implementación y pruebas	...13

Prototipo 2

Análisis y diseño	...17
Implementación	...18
Conclusiones	...27
Referencias	...30
Anexos	
Códigos	...33

Figuras

Figura 1. Estados de los procesos	... 2
-----------------------------------	-------

Figura 2. Definición de hilos	... 3
Figura 3. Funcionamiento de la cola de mensajes	... 9
Figura 4. Representación del uso de memoria compartida	... 9
Figura 5. Memoria compartida distribuida	... 10
Figura 6. Tipos de archivos	...11
Figura 7. Diagrama de bloques del sistema	...13

Tablas

Tabla 1. Clasificación y ejemplos de dispositivos de entrada/salida	... 11
---	--------

Introducción

Descripción del proyecto

Las tecnologías de la información y la comunicación (TIC) son todas aquellas herramientas y programas que tratan, administran, transmiten y comparten la información mediante soportes tecnológicos. La informática, Internet y las telecomunicaciones son las TIC más extendidas, aunque su crecimiento y evolución están haciendo que cada vez surjan cada vez más modelos [1].

En la actualidad, debido a la accesibilidad del uso del internet a nivel mundial, ha generado nuevos modelos de negocios que permiten a las empresas realizar transacciones comerciales de manera electrónica, este es el caso del comercio electrónico o también llamado e-commerce, según Fernández et al. (2015) manifiestan que el comercio electrónico es la compraventa de productos o servicios que están siendo publicitados a través de redes informáticas.

Una de las principales ventajas de las tiendas en línea es la comodidad y la conveniencia que ofrecen a los consumidores. Las personas pueden realizar compras desde la comodidad de su hogar o desde cualquier lugar con acceso a internet, sin tener que desplazarse a una tienda física. Además, las tiendas en línea suelen tener una selección más amplia de productos y precios más competitivos en comparación con las tiendas físicas. [2]

La pandemia ha acelerado la adopción de las tiendas en línea, ya que muchas personas han recurrido a ellas como una forma segura y conveniente de adquirir bienes y servicios. Se espera que esta tendencia continúe en el futuro, ya que las tiendas en línea ofrecen una experiencia de compra cada vez más personalizada y mejorada, gracias a la utilización de inteligencia artificial y tecnologías de análisis de datos.

En general, la pandemia ha demostrado la importancia de las TIC en nuestra vida diaria y ha acelerado la adopción de tecnologías digitales. Se espera que las TIC sigan transformando la forma en que vivimos, trabajamos y nos relacionamos en el futuro. Sin embargo, es importante abordar la brecha digital para asegurarnos de que todas las personas tengan acceso a las oportunidades que ofrece la tecnología.

Por ello en este proyecto se busca realizar una aplicación no en línea con el objetivo de ver la funcionalidad de automatizar los procesos de venta y gestión de inventario, se puede reducir la necesidad de mano de obra y ahorrar en costos de alquiler de espacio físico. Además, una tienda puede integrarse con herramientas de marketing digital y analítica, lo que permite a las empresas recopilar datos valiosos sobre el comportamiento de compra de los consumidores y mejorar su estrategia de marketing.

Por otro lado, en esta fase 4 se presentará un prototipo, en el cual se dará una descripción del sistema mediante un diagrama de bloques; representación gráfica que muestra la estructura o interconexión de diversos componentes o etapas de un sistema o proceso, permitiendo simplificar la representación de sistemas complejos al dividirlos en bloques más pequeños y comprensibles.

Objetivos que debe cumplir el proyecto

Desarrollar un sistema que permita una interacción entre vendedor y cliente para la adecuada transacción de diversos productos electrónicos mediante el control y precisión de la información.

Objetivos específicos

- Generar de manera segura la compraventa de diversos artículos (componentes electrónicos) y así tener un control de las ventas.
- Mantener al usuario informado de manera clara sobre de los diferentes componentes electrónicos.
- Permitir al usuario un inicio de sesión.
- Permitir a los usuarios agregar productos seleccionados a su carrito de compras, donde podrán revisarlos.

Justificación

Sabemos que ya existan aplicaciones de ventas para grandes empresas, pero las microempresas se quedan atrás en este aspecto, ya que, a diferencia de las grandes empresas, las cuales cuentan con la infraestructura suficiente para implementar sus plataformas de estas, ponen a las microempresas o negocios pequeños en desventaja por no contar con estas tecnologías.

Por otro lado, la pandemia por Covid-19 favoreció el desarrollo tecnológico de los habitantes en México como consecuencia de las medidas de prevención del virus y del aislamiento social de la población [3], de esta manera la mayoría de los habitantes quedo muy acostumbrado al uso de tecnología para el desarrollo de sus diversas actividades, por lo que implementar una aplicación de ventas en un negocio beneficiaría demasiado a su crecimiento, ya que atraería más clientes gracias a su comodidad al momento de comprar. Si bien ya existen aplicaciones de este tipo, tienen múltiples deficiencias que no permiten el uso adecuado en cada una de sus interfaces, haciendo muy complicado elegir las compras de nuestro día a día de manera segura y rápida.

La aplicación de este proyecto busca facilitar a estos negocios, con el manejo de recursos de la aplicación de tienda, así entrarían a la competencia del mercado moderno donde la aplicación será más amigable con los usuarios y mejorando su funcionamiento mediante métodos y herramientas que se aplicaran en el código de la aplicación.

Estado del arte

El primer paso que se dio para lo que hoy conocemos como e-commerce, en el planeta, fueron las ventas por catálogos en los años 20 y 30 del siglo XX en Estados Unidos este modelo de negocio rompió con todo lo establecido, porque les permitía a los consumidores ordenar sus productos preferidos sin salir de casa.

Aquí la industria se dio cuenta de que el consumidor deseaba comprar, pero con la comodidad de evitar ir hasta una tienda física y llevar los productos consigo al hogar, la venta por catálogo evolucionó gracias a la expansión del teléfono por toda la nación estadounidense y lo que se usaba como medio de comunicación con familiares y amigos, se convirtió en la herramienta maestra de los vendedores.

Los pedidos telefónicos cambiaron el modelo de negocio de muchas empresas, ya que pudieron entender que había un mercado real y abundante en este sector, y que los métodos tradicionales no eran la única forma de hacer dinero. [4]

Los componentes electrónicos han ido evolucionando constantemente hasta nuestros días. En los inicios los componentes eran todos de gran tamaño, comparados con los utilizados ahora. Y todos eran del tipo que hoy se llaman PTH (Pin Through Hole), esto es: componentes cuyos terminales se introducen en agujeros en la PCB y se sueldan por el lado contrario al que se encuentra el componente.

La tecnología de montaje superficial supuso un cambio importante. Se desarrolló en los 60 y fue en los 80 cuando se empezó a utilizar de manera masiva. Esta tecnología se diferencia de la anterior en que los componentes se montan y se sueldan en la misma cara, sin necesidad de agujeros en el circuito impreso. Este tipo de tecnología se llama SMT (Surface mount technology) y los componentes que se sueldan de esa manera se llaman SMD (Surface mount devices).

A pesar de que este tipo de componentes SMD, se pueden ensamblar automáticamente mucho más rápido que los componentes PTH. No todos los componentes se han podido pasar a la tecnología SMD, como es el caso de algunos componentes de gran tamaño y potencia como transformadores o semiconductores de potencia con disipadores térmicos.

Dentro de la tecnología SMD la evolución ha ido encaminada a conseguir componentes de menor tamaño. Por una parte, ha ido empujando la necesidad de reducir costes y de mejorar el comportamiento en aplicaciones de alta frecuencia. Además, las máquinas de montaje superficial han ido mejorando en precisión y han permitido el uso, cada vez mayor, de componentes muy pequeños. [5]

Productos Esperados

El resultado que se pretende alcanzar al concluir el proyecto es la aplicación desarrollada, con un correcto funcionamiento, que cumpla con lo planteado en nuestro diagrama de bloques de nuestro sistema, que se muestra en la figura 7, la cual se encuentra en la página 13.

Marco teórico

Procesos

Cuando se habla de un proceso, hace referencia a un programa el cual es ejecutado dentro del núcleo del sistema operativo de una computadora. Estos procesos siguen una estructura de jerarquía donde existen procesos padre y procesos hijo; los procesos padres, pueden contar con múltiples hijos al mismo tiempo, mientras que un proceso hijo únicamente proviene de un solo padre [6].

Los procesos cuando se ejecutan pueden atravesar distintos estados desde que se inician hasta que finalizan. Cuando un proceso se inicia, primero se mete en una cola de trabajos; cuando es admitido por el sistema, pasa a una cola de procesos que están preparados y esperando para ejecutarse; cuando el procesador asigna tiempo de ejecución, el proceso pasa de estar preparado a ejecución, y cuando el proceso necesita alguna señal o dato, pasa al estado bloqueado (se lo introduce en la cola de bloqueados) [6].

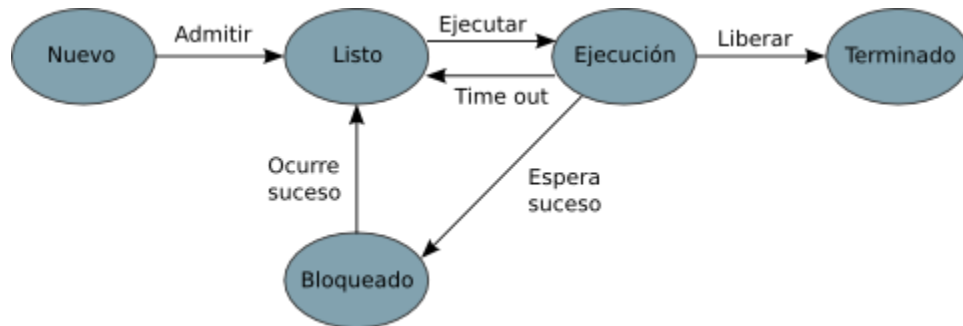


Figura 1. Estados de los procesos.

Fuente: [6]

Estos procesos se ejecutan dentro de un mismo procesador gracias al método de multitarea donde cada núcleo procesa una tarea a la vez. Otra de las facilidades que otorga la multitarea a los procesadores es que estos son capaces de elegir entre todos los procesos que se están ejecutando sin importar que estos hayan sido terminados, a este concepto se le conoce en inglés como “preemption”. A pesar de lo útil que puede parecer la multitarea, esta tiene efectos secundarios, ya que en la prioridad con la que los procesos son realizados, todos los relacionados con el usuario se encuentran primero en la jerarquía, por lo que algunos procesos del propio sistema operativo son colocados en segunda instancia. Los procesos cuentan con diversos tipos de estados por los cuales estos pasan durante su ejecución. En primera instancia el proceso se crea una vez que este pasa de un dispositivo de almacenamiento secundario tal como el disco duro a la memoria principal que en sistemas modernos encontramos como RAM, una vez ahí, se le otorga el estado de “listo”, donde se encuentra esperando para pasar al procesador; cuando el proceso pasa al procesador se ejecuta y cambia al estado de “correr”, puede suceder que este proceso necesite esperar a que algún recurso de la computadora retorne una respuesta y se coloca al proceso en estado de “bloqueado”; por último el proceso es enviado de nuevo a la memoria donde es destruido o en estado de “terminado” esperando a ser eliminado. [7].

Podemos definir diferentes tipos de procesos tomando diferentes criterios, por ejemplo, tomando en cuenta el diseño de este se pueden clasificar en “reutilizables” y “reentrantes”. En el primer caso, son procesos que se pueden reutilizar múltiples veces mientras que los procesos reentrantes no cuentan con datos, consisten únicamente en código que es utilizado por cada uno de los usuarios del sistema.

También podemos separar los procesos por su capacidad de acceder al procesador, donde existen procesos “apropiativos” los cuales acceden a los recursos y tienen la capacidad de elegir cuando abandonarlos, mientras que los “no apropiativos” son incapaces de acaparar los recursos y son cedidos a otros procesos. Desde el punto de vista de la ejecución se puede clasificar a los procesos como “residentes” o “intercambiables; los procesos

residentes se mantienen en la memoria durante todo el proceso de ejecución y los procesos intercambiables o conocidos como “swappable” en inglés, pueden ser transferidos a la memoria secundaria a pesar de que estos se estén ejecutando. Y por último un ángulo diferente de clasificar a los procesos es según los propietarios de estos, es decir, si estos provienen del usuario o del sistema, y se separan en dos clasificaciones que llevan estos nombres; los procesos de usuario tienen la característica de no estar en modo protegido mientras que los del sistema operativo si lo están [6, p. 15].

Hilos

Un hilo es una secuencia de código en ejecución dentro del contexto de un proceso, un hilo no puede ejecutarse ellos mismos, pues requieren de la “supervisión” de un proceso padre para correr. Dentro de cada proceso hay un hilo o varios hilos ejecutándose [9].

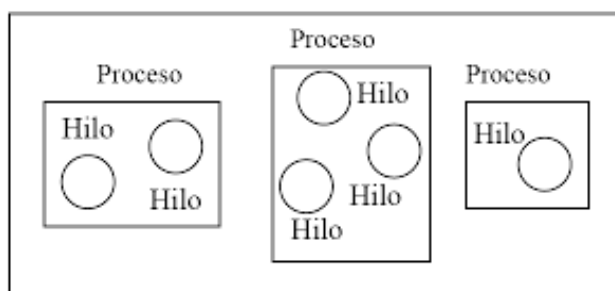


Figura 2. Definición de hilos.

Fuente: [9]

Los hilos de ejecución que comparten los mismos recursos, sumados a estos recursos, son en conjunto conocidos como un proceso. El hecho de que los hilos de ejecución de un mismo proceso compartan los recursos hace que cualquiera de estos hilos pueda modificar estos recursos. Cuando un hilo modifica un dato en la memoria, los otros hilos acceden a ese dato modificado inmediatamente. Lo que es propio de cada hilo es el contador de programa, la pila de ejecución y el estado de la CPU (incluyendo el valor de los registros) [10].

El proceso sigue en ejecución mientras al menos uno de sus hilos de ejecución siga activo. Cuando el proceso finaliza, todos sus hilos de ejecución también han terminado. Así mismo, en el momento en el que todos los hilos de ejecución finalizan, el proceso no existe más y todos sus recursos son liberados [10].

Una ventaja de la programación multihilo (Asignar múltiples hilos a una tarea) es que los programas operan con mayor velocidad en sistemas de computadores con múltiples CPU (sistemas Multiprocesador o a través de grupo de máquinas) ya que los hilos del programa se prestan verdaderamente para la ejecución concurrente. En tal caso el programador necesita ser cuidadoso para evitar condiciones de carrera (problema que sucede cuando diferentes hilos o procesos alteran datos que otros también están usando), y otros comportamientos no intuitivos. Los hilos generalmente requieren reunirse para procesar los datos en el orden correcto. Es posible que los hilos requieran de operaciones atómicas para impedir que los datos comunes sean cambiados o leídos mientras estén siendo modificados, para lo que usualmente se utilizan los semáforos [10].

De las partes de los hilos (Estado, Contexto del procesador, pila de ejecución, espacio de almacenamiento y acceso a los recursos de la tarea) destaca el Contexto del procesador, pues es el punto en el que estamos ejecutando la instrucción concretamente en la que nos hallamos. Es útil a la hora de reanudar un hilo que fue interrumpido con anterioridad, puesto que, al guardar el contexto, guardamos la última instrucción que ejecutamos, y así podemos conocer por donde tenemos que continuar la ejecución del hilo [11].

Existen dos tipos de hilos, los ULT (User level thread) y KLT (Kernel level thread). En una aplicación ULT pura, la aplicación se encarga de gestionar los hilos y el núcleo no tiene conocimiento de ellos. Si se desea una aplicación multihilo, se puede utilizar una biblioteca de hilos que proporciona el código necesario para crear y destruir hilos, intercambiar datos y mensajes entre ellos, planificar su ejecución y guardar/restaurar su contexto. Todas estas operaciones ocurren en el espacio de usuario del mismo proceso y el kernel continúa planificando el proceso como una sola unidad, asignándole un único estado. Algunas de sus ventajas son: El intercambio de

hilos no necesita los privilegios del modo kernel, Se puede hacer una planeación específica, se pueden ejecutar en cualquier sistema operativo [11].

Los KLT En una aplicación KLT pura, todo el trabajo de gestión de hilos lo realiza el kernel. En el área de la aplicación no hay código de gestión de hilos, únicamente un API (application programming interface) para la gestión de hilos en el núcleo. Windows 2000, Linux y OS/2 utilizan este método. Linux utiliza un método muy particular en que no hace diferencia entre procesos e hilos, para linux si varios procesos creados con la llamada al sistema «clone» comparten el mismo espacio de direcciones virtuales el sistema operativo los trata como hilos y lógicamente son manejados por el kernel [11].

Semáforos

Los semáforos son una herramienta de sincronización que ofrece una solución al problema de la sección crítica (porción de código de un programa de computador en la cual se accede a un recurso compartido que no debe ser accedido por más de un proceso o hilo en ejecución). Un semáforo provee una simple pero útil abstracción para controlar el acceso de múltiples procesos a un recurso común en programación paralela, o entornos multiusuarios. El concepto de semáforo fue inventado por el holandés Esdger W. Dijkstra [12].

Los semáforos sólo pueden ser manipulados usando las siguientes operaciones (éste es el código con espera activa):

Inicia(Semáforo s, Entero v)

```
{  
    s = v;  
}
```

En el que se iniciará la variable semáforo s a un valor entero v.

P(Semáforo s)

```
{  
    if(s>0)  
        s = s-1;  
    else  
        wait();  
}
```

La cual mantendrá en espera activa al regido por el semáforo si éste tiene un valor inferior o igual al nulo.

V(Semáforo s)

```
{  
    if(!procesos_bloqueados)  
        s = s+1;  
    else  
        signal();  
}
```

[12].

El por qué no se pueden usar directamente otras estructuras más clásicas, como por ejemplo usar una variable común para decidir si se puede o no acceder a un recurso, se debe a que estamos en un sistema multitarea: hacer esto implicaría realizar una espera activa (un bucle, comprobando constantemente si la variable está o no a 0, y así saber si podemos seguir ejecutando o no). Por otro lado, puede ocurrir algo mucho peor: supongamos que un proceso comprueba la variable, y ve que el recurso está libre, por lo que procedería a cambiar dicha variable de valor y seguir. Pues bien, si justo después de la comprobación, pero antes de que cambie el valor se conmuta de tarea (puede pasar, pues el sistema operativo puede hacerlo en cualquier momento), y el nuevo proceso comprueba la variable, como todavía no se ha actualizado, creará que el recurso está libre, e intentará tomarlo, haciendo que ambos programas fallen. Lo peor del caso es que se tratará de un error aleatorio: unas veces fallará (cuando se produzca cambio de tarea en ese punto) y otras no [13].

A continuación, cada vez que un thread o un proceso quiera acceder a dicho recurso (por ejemplo, un fichero), hará primero una petición con la primera de las llamadas disponibles. Cuando el S.O. ejecuta esa llamada, comprueba el valor que hay en la posición de memoria del semáforo, y si es distinta de cero, se limita a restarle 1 y devolver el control al programa; sin embargo, si ya es cero, duerme al proceso que hizo la petición y lo mete en la cola de procesos, en espera de que el semáforo se ponga a un valor distinto de cero [13].

Por último, cuando el proceso ha terminado el acceso al recurso, usa la segunda llamada para liberar el semáforo. Cuando el S.O. la ejecuta, comprueba si la cola del semáforo está vacía, en cuyo caso se limita a incrementar el valor del semáforo, mientras que, si tiene algún proceso, lo despierta, de modo que vuelve a recibir ciclos de CPU y sigue su ejecución. Si había varios procesos en espera, se irán poniendo en marcha uno tras otro a medida que el anterior va liberando el semáforo. Cuando termina el último, el semáforo se vuelve a poner a 1. Se trata, por tanto, del mismo proceso que seguiríamos con la variable, pero con la ventaja de que es un mecanismo estándar para todos los procesos, y como es una operación atómica (esto es, que durante su ejecución no se admiten cambios de tarea), no surge el problema de que una conmutación pueda producir errores aleatorios [13].

La mala sincronización puede desencadenar problemas como el DeadLock, que es cuando dos o más procesos están esperando una condición que solo puede ser causada por un proceso que también está esperando. Y la Espera indefinida, que es cuando un proceso en la lista de espera de un semáforo del que están entrando y saliendo procesos y listas de espera continuamente [13].

En resumen, podemos decir que la importancia de los semáforos radica en que permiten prevenir errores de concurrencia, como la condición de carrera, donde múltiples procesos intentan acceder simultáneamente a la misma sección crítica del código y pueden generar comportamientos inesperados. Los semáforos también permiten la comunicación y coordinación entre procesos, lo que facilita la implementación de algoritmos de sincronización más complejos.

Aplicaciones de los semáforos

Los semáforos se emplean para permitir el acceso a diferentes partes de programas (llamados secciones críticas) donde se manipulan variables o recursos que deben ser accedidos de forma especial. Según el valor con que son inicializados se permiten a más o menos procesos utilizar el recurso de forma simultánea [14].

Un tipo simple de semáforo es el binario, que puede tomar solamente los valores 0 y 1. Se inicializan en 1 y son usados cuando sólo un proceso puede acceder a un recurso a la vez. Son esencialmente lo mismo que los mutex. Cuando el recurso está disponible, un proceso accede y decreuenta el valor del semáforo con la operación P. El valor queda entonces en 0, lo que hace que si otro proceso intenta decrementarlo tenga que esperar. Cuando el proceso que decrementó el semáforo realiza una operación V, algún proceso que estaba esperando puede despertar y seguir ejecutando [14].

Para hacer que dos procesos se ejecuten en una secuencia predeterminada puede usarse un semáforo inicializado en 0. El proceso que debe ejecutar primero en la secuencia realiza la operación V sobre el semáforo antes del código que debe ser ejecutado después del otro proceso. Éste ejecuta la operación P. Si el segundo proceso en la secuencia es programado para ejecutar antes que el otro, al hacer P dormirá hasta que el primer proceso de la

secuencia pase por su operación V. Este modo de uso se denomina señalación (signaling), y se usa para que un proceso o hilo de ejecución le haga saber a otro que algo ha sucedido [14].

Los semáforos pueden ser usados para diferentes propósitos, entre ellos:

- Implementar cierres de exclusión mutua o locks
- Barreras
- Permitir a un máximo de N threads acceder a un recurso, inicializando el semáforo en N
- Notificación. Inicializando el semáforo en 0 puede usarse para comunicación entre threads sobre la disponibilidad de un recurso [14].

Colas de mensajes

Una cola de mensajes es una forma de comunicación asíncrona de servicio a servicio, un mecanismo mediante el cual los mensajes se retienen hasta que una aplicación está lista para atenderla y una vez que se cumple procede a desecharla. Al mencionar el termino asíncrono nos referimos a que en este tipo de comunicación no es necesario que la parte emisora y receptora interactúen al mismo tiempo, sino trabajan a su propio ritmo. Asimismo, es importante destacar que las colas de mensajes funcionan procesando un mensaje a la vez, por lo cual solo se dará solución a un mensaje por solicitud.

Las colas de mensajes se utilizan en sistemas operativos o aplicaciones para permitir que las aplicaciones se comuniquen entre sí. También se pueden utilizar para transferir mensajes entre sistemas informáticos [15]. En la arquitectura de la nube moderna, las aplicaciones se desacoplan en bloques pequeños e independientes que son más fáciles de desarrollar, implementar y mantener. Las colas de mensajes proporcionan la comunicación y la coordinación para estas aplicaciones distribuidas. Las colas de mensajes pueden simplificar de forma significativa la escritura de código para aplicaciones desacopladas y, a la vez, mejorar el rendimiento, la fiabilidad y la escalabilidad. En la figura _ se puede observar el funcionamiento de la cola de mensajes, desde que el productor añade un mensaje a la cola, el almacenamiento y finalmente el procesamiento que realiza el consumidor con el mismo [16].



Figura 3. Funcionamiento de la cola de mensajes.

Fuente: [15].

Aportan grandes beneficios, como lo son:

Garantía de entrega y orden: los mensajes se consumen, en el mismo orden que se llegaron a la cola, y son consumidos una única vez.

Redundancia: Las colas persisten los mensajes hasta que son procesados por completo.

Desacoplamiento: siendo capas intermedias de comunicación entre procesos, aportan la flexibilidad en la definición de arquitectura de cada uno de ellos de manera separada, siempre que se mantenga una interfaz común.

Escalabilidad: con más unidades de procesamiento, las colas balancean su respectiva carga.

Memoria compartida

La memoria compartida es el mecanismo de comunicación entre procesos más rápido. El sistema operativo asigna un segmento de memoria en el espacio de direcciones de varios procesos, de modo que varios procesos puedan leer y escribir en ese segmento de memoria sin llamar a las funciones del sistema operativo, es decir, los procesos y los subprocesos pueden comunicarse directamente entre sí compartiendo partes de su espacio de memoria y luego leyendo y escribiendo los datos almacenados en la memoria compartida [17].

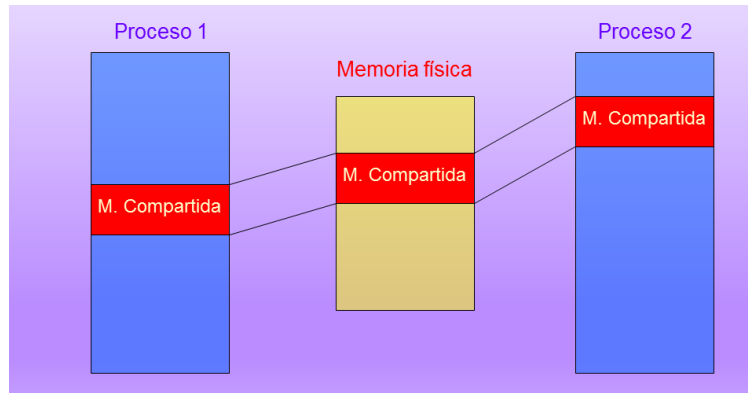


Figura 4. Representación del uso de memoria compartida.

Fuente: [17]

En la programación de computadoras, la memoria compartida es un método por el cual los procesos del programa pueden intercambiar datos más rápidamente que leyendo y escribiendo, esto es posible utilizando los servicios regulares del sistema operativo. Por ejemplo, el proceso del cliente puede tener datos para pasar a un proceso del servidor que el proceso del servidor es modificar y devolver al cliente [18].

La memoria compartida es más rápida porque los datos no se copian de un espacio de direcciones a otro, es decir la asignación de memoria se realiza una sola vez y la sincronización depende de los procesos que comparten la memoria. Esta memoria permite que múltiples elementos de procesamiento compartan la misma ubicación en la memoria, es decir ver lecturas y escrituras de los demás sin ninguna otra directiva especial [19].

Memoria compartida distribuida

Los sistemas de memoria compartida distribuida (DSM) representan la creación híbrida de dos tipos de computación paralelos: la memoria distribuida en sistemas multiprocesador y los sistemas distribuidos. Ellos proveen la abstracción de memoria compartida en sistemas con memorias distribuidas física y consecuentemente combinan las mejores características de ambos enfoques [20].

La memoria compartida distribuida (DSM) es una abstracción utilizada para compartir datos entre computadores que no comparten memoria física. Los procesos acceden a la DSM para leer y actualizar, dentro de sus espacios de direcciones, sobre lo que aparenta ser la memoria interna normal asignada a un proceso [21].

La principal característica de la DSM es que ahorra al programador todo lo concerniente al paso de mensajes al escribir sus aplicaciones, cuestión que en otro sistema debería tenerse muy presente. DSM es fundamentalmente una herramienta para aplicaciones paralelas o para aplicaciones o grupos de aplicaciones distribuidas en las que se puede acceder directamente a datos individuales que ellas comparten [22]. Una configuración de la DSM se muestra en la figura #.

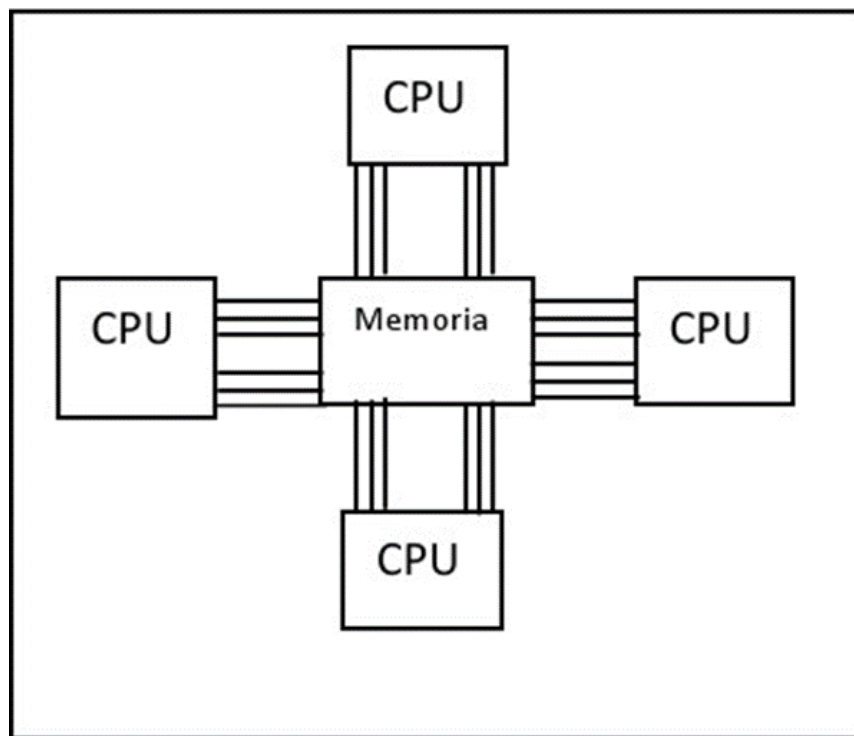


Figura 5. Memoria compartida distribuida.

Fuente: [21]

Archivos

Los archivos son una forma de almacenar información digitalmente, utilizando unidades de información conocidas como bits (unidad mínima de información que representa la contraposición de dos valores 0 y 1). Al igual que los archivos tradicionales de papel en una oficina, los archivos digitales tienen un nombre único y una extensión que indica su tipo y función.

Dentro de los archivos, la información se organiza en registros o líneas, y la forma en que se agrupan los datos puede variar según el creador del archivo, los archivos pueden ser manipulados de diversas maneras, incluyendo la creación, eliminación, renombramiento y ejecución, entre otras.

Es importante tener en cuenta que los archivos pueden contener diferentes tipos de información y que se necesitan programas específicos para acceder a ellos y manipularlos, evitando la pérdida de información, es necesario tomar precauciones y contar con un sistema adecuado de gestión de archivos; un archivo es un elemento que guarda un conjunto de datos y no es útil por sí solo, sino que necesita ser utilizado por el usuario final. Este puede ser cualquier elemento individual dentro de un ordenador [23].

Los archivos tienen un nombre y una extensión que define su tipo de archivo, la extensión suele ser una combinación de letras y números de 3 o 4 caracteres, los datos que conforman el archivo son necesarios para su correcto uso y si falta alguno de ellos, el archivo podría corromperse y no ser útil [24].

En cuanto a su ubicación, se guardan en subdirectorios o directorios y para poder acceder a ellos, la ruta está conformada por los diversos subdirectorios que lo contienen, hasta llegar al directorio contenedor, es importante destacar que además de la misma computadora se pueden ser almacenados en la nube, un sitio Share Point [24].

Los archivos informáticos pueden tener diversas utilidades, las cuales están determinadas por su extensión, no se puede definir la utilidad de un archivo en general, ya que cada extensión establece el uso específico de cada archivo. Si se modifica la extensión de un archivo simplemente cambiando su nombre, es muy probable que deje de funcionar ya que los datos almacenados en él están diseñados para ser interpretados de una manera específica. [25]

Los archivos pueden ser representables (tener un nombre); Únicos por directorio: dentro de una carpeta o directorio no puede haber más de un archivo con el mismo nombre; tienen tamaño: dado por kb, mb o gb y son modificables.[25]

Tipos de archivos

Los archivos se irán creando con una extensión denominada por 3 letras de igual forma dependerán de su finalidad y su contenido, en la figura 1 se muestran algunos de los tipos de archivos con extensiones comunes, si será un video tendrá extensión avi, mpg o mov; animación HTML; imagen con extensión GIF, Jpeg; archivos de texto otd, doc; archivos de sonido mp3, mp4 o m3u. [25]



Figura 6. Tipos de archivos.
Fuente: [25]

Dispositivos de entrada y salida

Los equipos físicos conectados a una computadora que permiten la comunicación de información entre el usuario y el sistema o manejan soportes de información son conocidos como unidades de entrada/salida. Comúnmente también se les llama periféricos de computadora o periféricos de entrada y salida, ya que se encuentran externos a la unidad central de procesamiento.

Los dispositivos de entrada/salida se clasifican en diferentes categorías en función de las funciones que pueden desempeñar. Estas categorías incluyen dispositivos de entrada, que permiten el ingreso de información a la computadora; dispositivos de salida, que presentan información procesada por la computadora al usuario; y dispositivos de entrada y salida, que realizan ambas funciones. [26]

La tabla 1 se describen los tipos de dispositivos en el que se adjuntan sus funciones y algunos ejemplos.

Tipo de dispositivo	Función	Ejemplos
Entrada	Encargados de introducir datos en la computadora para su uso, con estos dispositivos se transforma la información de entrada con señales eléctricas.	Teclado: alfabético, numérico, de control y teclas de función; ratón: movimiento del cursor en la pantalla; disco óptico: lectura y escritura de discos ópticos; micrófono: reciben señales de audio; cámara: introducen imágenes.
Salida	Presentan la información al usuario para que sea interpretada por ellos,	Pantalla: representación mediante configuraciones de puntos luminosos denominados píxeles; impresoras:

	a través de imágenes, texto, sonido o táctil.	salida de datos en papel; altavoz: transforman señales eléctricas en audio.
Mixtos	Se encargan de ambas funciones, entrada y salida de datos.	Pantalla táctil: pantalla que incluye un dispositivo que reconoce la zona de esta donde se ha realizado un pequeño contacto con el dedo; impresora multifunción: imprimen, escanean y fotocopian; terminales para operaciones financieras: cumplen función de cajero automático.

Tabla 1. Clasificación y ejemplos de dispositivos de entrada/salida
Fuente: elaboración propia.

Prototipo 1

Análisis y diseño

En esta etapa, vemos que para poder crear el sistema definitivo se debe realizar un análisis para su implementación considerando las múltiples complicaciones que podríamos tener a lo largo de esta fase. Una de las problemáticas más comunes es realizar el programa a manera de que lo puedan utilizar diferentes clientes (usuarios), para esto decidimos implementar un login, que nos servirá para identificar a cada usuario y acceder a los datos específicos de cada usuario.

En este caso, comenzamos realizando un diagrama de bloques donde planteamos las diferentes etapas que se llevarán a cabo en todos los procesos que serán ejecutados en este programa. Para entender mejor las bases del funcionamiento de este sistema, fue necesario explicarlo con un diagrama de bloques, el cual se muestra a continuación en la figura 7.

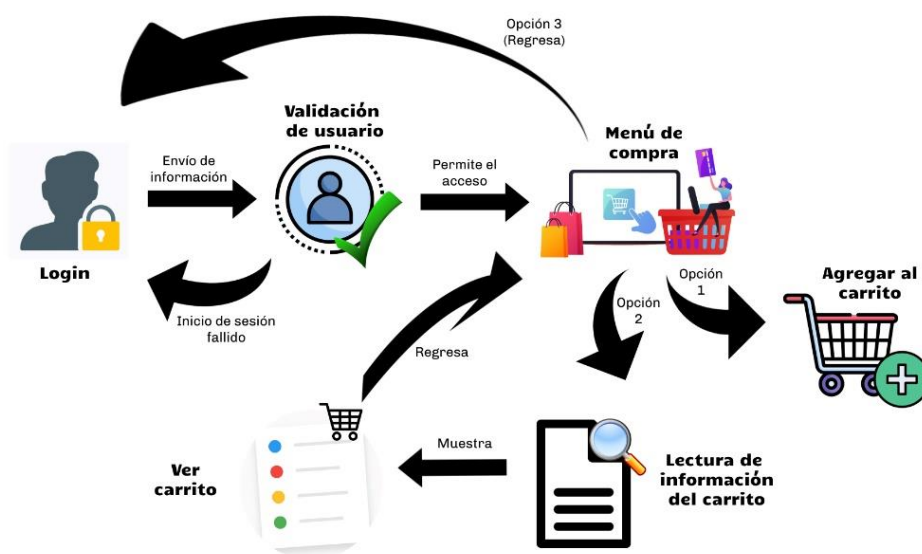


Figura 7. Diagrama de bloques del sistema.
Fuente: Elaboración propia.

Para iniciar, el sistema comienza con un login, para que de esta manera nuestro programa sea compatible con diferentes usuarios y cada uno tenga su inicio de sesión con sus propios datos, sin necesidad de compartirlos con otros usuarios, con esto guardamos la privacidad de cada usuario que acceda a este sistema. Una vez que

un usuario ingrese sus datos de inicio de sesión, este envía esa información a un archivo con el que se verifican que sean correctos los datos, en caso de ser correctos, en ese momento ese usuario tendrá acceso a este sistema. Por otro lado, si en esta etapa los datos de inicio de sesión son incorrectos, será rechazado el acceso al sistema para ese usuario, es decir, tendremos un inicio de sesión fallido.

Si la validación es correcta, nos permite el acceso al menú de compra, que podríamos decir que es el menú principal, en donde tendremos tres opciones:

1. Agregar al carrito
2. Ver carrito
3. Cerrar sesión

Si seleccionamos la opción 1, nos dirige a una pantalla donde se nos presentarán diferentes opciones para agregar los diferentes artículos a el carrito. Con la opción 2, accederá al archivo donde tendremos la información de los carritos de compra de cada cliente, una vez que hace esta revisión, la cual el usuario no la percibirá, nos dirige a una pantalla donde nos imprime la lista de artículos que se encuentran en el carrito del usuario y algunos datos relacionados con esto, como el monto total, entre otras cosas. Desde esta pantalla podremos regresar al menú de compra.

Finalmente, con la opción tres, simplemente se cierra la sesión del usuario y nos redirige a la pantalla del login.

Implementación y pruebas

Para comenzar con el funcionamiento del programa, debemos tener el login, como se menciona en el diagrama a bloques anterior, para esto definimos un tamaño máximo de usuario y contraseña, como se puede ver en la línea 4 y 5 de la figura 8, después definimos nuestra estructura de usuario, que contara con el nombre de usuario y contraseña, después tenemos la función de login, que recibe como argumentos la estructura de usuario como apuntador, la cantidad de usuarios que hay registrados, y apuntadores al usuario y contraseña de la estructura. En esta fase no implementamos el registro de usuarios desde el programa, para dar de alta a un usuario debemos hacerlo desde un archivo de texto. En esta función comparamos las cadenas de usuario en la iteración que va el bucle for, con el usuario de la estructura, además tenemos que comparar la contraseña del bucle en la iteración para la estructura junto con la del apuntador, que más adelante veremos de donde viene. Tenemos que usar el operador AND (&&) pues la única forma de iniciar sesión es cuando el usuario Y la contraseña son válidos, en caso de que se cumpla esa condición retornamos 1, si no se cumple retornamos 0.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  #define TAM_USUARIO 20
5  #define TAM_CONTRA 20
6
7  struct User {
8      char username[TAM_USUARIO];
9      char password[TAM_CONTRA];
10 };
11
12 int login(struct User *users, int numUsers, const char *username, const char *password) {
13     for (int i = 0; i < numUsers; i++) {
14         if (strcmp(users[i].username, username) == 0 && strcmp(users[i].password, password) == 0) {
15             return 1; // Inicio de sesión exitoso
16         }
17     }
18     return 0; // Inicio de sesión fallido
19 }
```

Figura 8. Función Login. Fuente: Elaboración propia.


```

21  int readUsersFromFile(struct User *users) {
22      FILE *file = fopen("usuarios.txt", "r");
23
24      int numUsers = 0;
25      char linea[TAM_USUARIO + TAM_CONTRA + 2];
26
27      while (fgets(linea, sizeof(linea), file) != NULL) {
28          char *username = strtok(linea, " \t\n");
29          char *password = strtok(NULL, " \t\n");
30
31          if (username != NULL && password != NULL) {
32              strcpy(users[numUsers].username, username);
33              strcpy(users[numUsers].password, password);
34              numUsers++;
35          }
36      }
37
38      fclose(file);
39      return numUsers;
40  }

```

Figura 9. Función para leer archivo de texto. Fuente: Elaboración propia.

Como se muestra en la figura 9, la función para leer el archivo de texto que contiene a los usuarios y sus contraseñas recibe 1 parámetro, que es un apuntador a la estructura de usuario, después abre el archivo de texto con los permisos de lectura "r", después inicializamos un contador de usuarios en 0, y medimos el tamaño de cada línea en el archivo de texto que será de la suma y contraseña y nombre de usuario más 2 (espacio que separa a estos y el salto de línea), después con el bucle while comenzamos a guardar las líneas leídas del archivo de texto con la llamada fgets y en la variable línea, mientras que las líneas del archivo de texto no sean vacías (NULL) partiremos las cadenas leídas del txt con la llamada strtok, lo que usaremos para separarlas serán los espacios, tabulaciones y saltos de línea (figura 12), después con el if, validamos que el usuario y contraseña no sean NULL y cada que se termine de leer una línea aumentamos el contador de usuarios en 1. posteriormente cerramos el archivo de texto y retornamos la cantidad de usuarios leídos.

```

42  int main() {
43      struct User users[100];
44      int numUsers;
45
46      numUsers = readUsersFromFile(users);
47      if (numUsers == 0) {
48          return 1;
49      }
50
51      char inputUsername[TAM_USUARIO];
52      char inputPassword[TAM_CONTRA];
53
54      printf("Ingrese el nombre de usuario: ");
55      scanf("%s", inputUsername);
56      printf("Ingrese la contraseña: ");
57      scanf("%s", inputPassword);
58
59      if (login(users, numUsers, inputUsername, inputPassword)) {
60          printf("%s, iniciaste sesion :D\n", inputUsername);
61      } else {
62          printf("Usuario o contraseña incorrectos D:\n");
63      }
64  }

```

Figura 10. Función main del login. Fuente: Elaboración propia.

En la figura 10 tenemos la función main del login, primero declaramos la cantidad de usuarios posibles a estar registrados e igualamos a la función de leer usuarios, pues esta nos retorna la cantidad de usuarios leídos, después declaramos dos cadenas, la contraseña y nombre de usuarios a escribir por el usuario, después de la línea 54 a 57 pedimos estas cadenas y las guardamos, después llamamos a la función login con los parámetros que necesita, que como vemos, estamos mandando el nombre de usuario y contraseña que fueron previamente registrados, si la función de login valida estos parámetros, entonces se habrá iniciado sesión y pasaremos a la siguiente parte del programa el menu de inicio de sesión exitoso se puede ver en la figura 11.

```

Ingrese el nombre de usuario: Efren
Ingrese la contraseña: 123
Efren, iniciaste sesion :D

```

Figura 11. Inicio de sesión exitoso. Fuente: Elaboración propia.

```

usuarios.txt
1  alexis contraseña
2  daniel 123
3  Efren 123
4

```

Figura 12. Estructura del archivo con usuarios y contraseñas. Fuente: Elaboración propia.

En la figura 12 se muestra el archivo de texto usado como base de datos con los pares de usuario y contraseña separados entre si mediante espacios y separados uno del otro con saltos de línea.

Una vez hemos terminado en el menú de login, el programa procede a ir al menú principal donde se desplegará una pantalla similar a la que se muestra en la figura 13. Como se puede observar esta muestra una lista de artículos los cuales corresponden a los productos registrados en la base de datos, en este caso el archivo “products.txt”.

```

===== Bienvenido Efren =====
Articulos 1 - 10 / 11
1 Diodo LED Diodo LED color Rojo 5
2 Transistor Transistor 2N222 10
3 Diodo Zener Diodo Zener de 1.5V 10
3 Diodo Zener Diodo Zener de 1.5V 10
3 Diodo Zener Diodo Zener de 1.5V 10
3 Diodo Zener Diodo Zener de 1.5V 10
3 Diodo Zener Diodo Zener de 1.5V 10
3 Diodo Zener Diodo Zener de 1.5V 10
3 Diodo Zener Diodo Zener de 1.5V 10
3 Diodo Zener Diodo Zener de 1.5V 10
3 Diodo Zener Diodo Zener de 1.5V 10

1. Agregar al carrito
2. Ir al carrito
3. Siguiete pagina
4. Cerrar sesion

```

Figura 13. Vista del menú principal con la lista de productos en la base de datos. Fuente: Elaboración propia.

```

11
1,Diodo LED,Diodo LED color Rojo,5
2,Transistor,Transistor 2N222,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10
3,Diodo Zener,Diodo Zener de 1.5V,10

```

Figura 14. Contenido de la base de datos “products.txt”. Fuente: Elaboración propia.

Como se puede observar en la figura 14, la base de datos comienza con un valor entero el cual indica la cantidad de productos registrados en la base de datos, en este caso para casos prácticos se añadió el mismo producto múltiples veces. En el menú principal mostrado en la figura 15. Observamos que se despliegan diferentes opciones: agregar al carrito, ir al carrito, siguiente página, cerrar sesión. También en esta pantalla se muestra en la parte superior la cantidad de productos que se están desplegando y la cantidad de productos totales. Si se elige la opción de siguiente página, debido a que hay 11 productos y solo se muestran 10, en la siguiente página se mostraran los que faltan, tal como se muestra en la figura 15.

```

===== Bienvenido Efren =====
Articulos 11 - 20 / 11
3 Diodo Zener Diodo Zener de 1.5V 10

1. Agregar al carrito
2. Ir al carrito
3. Pagina anterior
4. Cerrar sesion

```

Figura 15. Página siguiente de los productos. Fuente: Elaboración propia.

Prototipo 2

Análisis y diseño

Para el segundo prototipo, este se basó en la funcionalidad del anterior, con los mismos menús, pero mejorando diferentes opciones, resolviendo diferentes bugs que se presentaron y agregando una nueva funcionalidad mediante un modo administrador el cual contiene un panel de administración; la figura 16 muestra el diagrama de bloques que describe esta nueva funcionalidad.

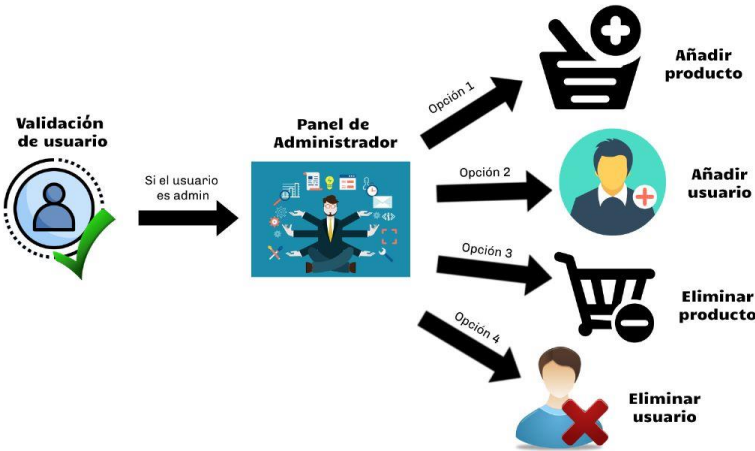


Figura 16. Diagrama de bloques que describe el comportamiento del panel de administración. Fuente: Elaboración propia.

Dentro de las mejoras que se agregaron en este prototipo con respecto al anterior, primero se agregó la funcionalidad completa del carrito. Ahora, una vez se elige la opción dos en el menú principal, la cual corresponde al carrito del usuario, esta mostrará todos los productos que se han agregado, con la opción de eliminar algún producto o una cantidad específica de este. Otra de las mejoras importantes que se agregaron en este prototipo fue la parte de la administración de las bases de datos. En este prototipo aún se conservan tres bases de datos que se encuentran en los archivos “users.txt”, “cart.txt” y “products.txt”, las cuales corresponden a los usuarios, los carritos de los usuarios y los productos de la tienda. Posteriormente se explicará en forma más detallada como es que la administración de estas bases se modificó.

Con respecto al panel de administrador, este es un usuario con privilegios superiores a los demás usuarios y tiene la facultad de realizar cuatro diferentes acciones: añadir o eliminar productos de la base de datos y añadir o eliminar usuarios del sistema sin la necesidad de saber su contraseña, esto con el objetivo de regular el acceso a la plataforma ya que en una versión posterior se planea agregar la opción de registrarse como nuevo usuario. Por lo que el administrador debe ser capaz de poder eliminar a algún usuario en caso de que este realice alguna acción que comprometa al sistema.

Por último, con respecto al diseño de este sistema, se optó por organizar el proyecto en carpetas con la estructura que muestra la figura 17, con el objetivo de una mejor organización en el equipo, así como una mayor facilidad al momento de arreglar errores en el código, esto apoyado en que el proyecto esta dividido en diferentes archivos cabecera.

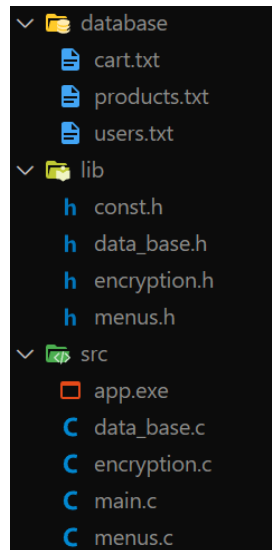


Figura 17. Estructura el proyecto en diferentes carpetas. Fuente: Elaboración propia.

Implementación

Administración de Bases de datos

Para comenzar, en la administración de las bases de datos, se cambió la lógica con las que esta se hacía. En la versión anterior la base de datos de los usuarios se veía tal como lo muestra la figura 12, es decir, cada línea en el archivo correspondía a un usuario, y en esta se encontraba su nombre, contraseña e ID del carrito. Si bien esta forma de manejar la base de datos funcionaba, implicaba un código más complejo con muchas operaciones para separar cada uno de los parámetros y almacenarlos en una estructura llamada “User” mostrada en la figura 18. Para cambiar esto, se optó por almacenar la estructura completa dentro del archivo de texto mediante el uso de “fwrite()”, función la cual permite almacenar una estructura en forma de bytes y recuperarla con la función “fread()”, simplificando las operaciones a realizar para la administración de la base de datos con la diferencia de que se necesitan unos cuantos más de bytes para almacenar los datos del usuario, pero a pesar de ello, comparando el peso en bytes del archivo anterior con respecto a este, los dos ocupan 1KB de almacenamiento. Este mismo proceso de almacenar una estructura completa se repite para las otras dos, haciendo que si se abre alguno de estos archivos observamos lo que se muestra en la figura 19.

```
typedef struct {
    char user_name[20];
    char password[20];
    int ID_cart;
} User;
```

Figura 18. Estructura “User”. Fuente: Elaboración propia.



```
1 admin;
efren;
alexis;
Papoi;
usr;
89;
```

Figura 19. Archivo “users.txt” donde se almacenan las estructuras completas. Fuente: Elaboración propia.

Para el caso de almacenar los usuarios, se agregó una forma de seguridad, encriptando las contraseñas de los usuarios, para que esta permanezca oculta incluso para alguien que tenga acceso a la base de datos y no pueda obtener los datos de los usuarios. Para ello se agregó una función llamada “encrypt_password()”, la cual está inspirada en el algoritmo SHA-256 del cuál se puede encontrar más información en la siguiente referencia [27], algoritmo que se utiliza actualmente para encriptar contraseñas en las bases de datos de diferentes aplicaciones. En este caso se utilizó un método muy simplificado de este algoritmo. Tal como se observa en la figura 20, la función toma el valor en ASCII de cada carácter escrito en la contraseña y realiza un corrimiento a la derecha a los bits que representan este valor en ASCII y los reescribe en el apuntador de la contraseña. Este método a pesar de ser sencillo es mejor que simplemente realizar una suma al valor ASCII y poner el nuevo valor, ya que este método es susceptible para un ataque de fuerza bruta, si bien el método implementado aquí también, este tomaría más iteraciones para poder descifrarlo.

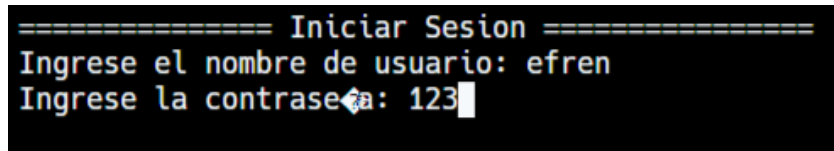
```
char* encrypt_password(char* password){
    for(int i = 0; i < PASSWORD_SIZE; i++)
        password[i] >>= 1;
}
```

Figura 20. Función “encrypt_password” basada en el SHA-256. Fuente: Elaboración propia.

Inicio de Sesión

El menú de “login”, el cual no es muy diferente al anterior, solo se agregaron mejoras visuales en lo que respecta a la terminal, tal como lo muestra la figura 21. Observando el código que se muestra en la figura 22, este implementa la función “encrypt_password()” en la línea 26 después de obtener la lectura del usuario para después llamar a la función “get_user” en la línea 30 la cuál confirma que el usuario se encuentre en la base de datos y retorna un valor entero, este tiene valor de 1 en caso de un inicio de sesión exitoso, -1 en caso de iniciar sesión como administrador (función que se explicará más adelante) y 0 cuando el usuario no se encuentra en la

base de datos. En este último caso se imprime en pantalla el mensaje “Usuario o contraseña incorrectos D:” tal como lo muestra la figura 23.



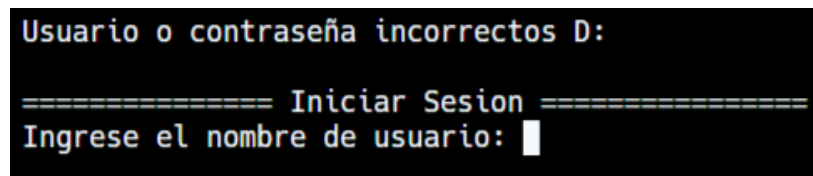
```
==== Iniciar Sesion =====
Ingrese el nombre de usuario: efren
Ingrese la contraseña: 123
```

Figura 21. Pantalla de “login”. Fuente: Elaboración propia.



```
12 int login() {
13     char input_User_name[USER_SIZE];
14     char input_Password[PASSWORD_SIZE];
15     int login_check = 0;
16
17     system("clear");
18
19     while (1) {
20         printf("==== Iniciar Sesion =====\n");
21         printf("Ingrese el nombre de usuario: ");
22         scanf("%s", input_User_name);
23         printf("Ingrese la contraseña: ", 164);
24         scanf("%s", input_Password);
25
26         encrypt_password(input_Password);
27
28         getchar();
29
30         login_check = get_user(&current_user, input_User_name, input_Password);
31
32         if (login_check == 1 || login_check == -1) {
33             system("clear");
34             return login_check;
35         } else {
36             system("clear");
37             printf("Usuario o contraseña incorrectos D:\n\n");
38         }
39     }
40 }
```

Figura 22. Código de la función “login”. Fuente: Elaboración propia.



```
Usuario o contraseña incorrectos D:
==== Iniciar Sesion =====
Ingrese el nombre de usuario:
```

Figura 23. Mensaje de usuario o contraseña incorrectos en la pantalla de “login”. Fuente: Elaboración propia.

Modo Administrador

La figura 24 es un caso particular de un inicio de sesión, al iniciar con las credenciales “admin”, “12345” como usuario y contraseña, una vez en este menú podemos realizar 4 acciones de administrador, que son: Agregar producto, Agregar Usuario, Eliminar producto, Eliminar usuario. Opciones que se muestran a continuación cada una por separado obviando la función de salir del panel de administrador que nos lleva al menú de inicio de sesión principal.

```
===== Panel de Administrador =====
Opciones:

1. Agregar producto
2. Agregar Usuario
3. Eliminar producto
4. Eliminar Usuario
5. Salir del panel de Administrador
```

Figura 24. Inicio de sesión como Administrador. Fuente: Elaboración propia.

```
int admin_menu() {
    int access = 0;
    int ID, price;
    char product_name[20], description[100];
    char user_name[USER_SIZE], password[PASSWORD_SIZE];
    char option;

    while (1) {
        system("clear");
        printf("===== Panel de Administrador =====\n");
        printf("Opciones: \n\n");
        printf("1. Agregar producto\n");
        printf("2. Agregar Usuario\n");
        printf("3. Eliminar producto\n");
        printf("4. Eliminar Usuario\n");
        printf("5. Salir del panel de Administrador\n");

        option = getchar();
```

Figura 25. Menú del administrador código. Fuente: Elaboración propia.

En la figura 25 se muestra el código del menú del administrador, se muestran las constantes declaradas para el id y precio del producto a agregar, las longitudes de las cadenas para el nombre de los productos y su descripción, además de las cadenas de usuario y contraseña que se agregará después, además de la opción que nos permite seleccionar el menú del administrador.


```

190 ~ if (option == '1') {
191     system("clear");
192     printf("Ingrese el ID del producto: ");
193     scanf("%d", &ID);
194     printf("Ingrese el precio del producto: ");
195     scanf("%d", &price);
196     getchar();
197     printf("Ingrese el nombre del producto [20 caracteres max]: ");
198     scanf("%[^\n]*c", product_name);
199     getchar();
200     printf("Ingrese la descripcion del producto [100 caracteres max]: ");
201     scanf("%[^\n]*c", description);
202     getchar();
203
204     add_product(ID, product_name, description, price);

```

Figura 26. Opción de agregar producto. Fuente: Elaboración propia.

En la figura anterior, se muestra el funcionamiento de la opción de agregar producto desde el menú del administrador, lo más relevante de esta figura es la línea 204, que llama a la función “add_product” con los parámetros ID, nombre, descripción y precio que previamente asignamos al producto que queríamos ingresar, a continuación, se muestra el funcionamiento de la función add_product en la figura ()

```

168 ~ void add_product(const int ID, const char* name, const char* description, const int price){
169     FILE *fp = fopen("../database/products.txt", "r");
170     FILE *temp = fopen("../database/temp.txt", "a");
171     Product aux;
172     int found = 0;
173     char option;
174
175 ~ while(fread(&aux, sizeof(Product), 1, fp)){
176 ~     if(aux.ID == ID){
177         printf("El producto ya existe, desea actualizarlo? [y/n]\n");
178         scanf("%c", &option);
179
180 ~         if(option == 'y'){
181             strcpy(aux.name, name);
182             strcpy(aux.description, description);
183             aux.price = price;
184
185             found = 1;
186         }
187     }else if(aux.ID == -ID)
188         continue;
189
190     fwrite(&aux, sizeof(Product), 1, temp);
191 }

```

Figura 27. Funcion add_product. Fuente: Elaboración propia.

En la figura anterior se muestra la definición de la función para agregar un producto, recibe los parámetros antes mencionados, en la línea 169 y 170 se abren los archivos de texto que funcionan como nuestra base de datos, además de un “Producto” auxiliar, la constante “found” que nos sirve para indicar que un producto fue encontrado (más adelante se explica que es para validar que un producto que se quiere dar de alta no exista antes) y la opción que nos sirve para decidir si actualizar el producto que existía para el id que se ingresó a la función de agregar. De la línea 175 a la 187 se valida que el producto no exista, en caso de que si exista y el usuario desee actualizar la información de ese producto se actualiza la información del producto de la línea 180 a la 185, y cambiamos el valor de “found” dentro del if para indicar que el producto ha sido encontrado en la búsqueda. En la línea 190 escribimos sobre la base de datos el producto que tenemos guardado en au

Agregar producto

En este menú se deben ingresar 4 datos, El id del producto que más adelante identifica a los productos en la página para agregarlos al carrito de compra, el precio que tendrá el producto, y su descripción como se muestra en la figura 25. Una vez que agregamos este producto desde el menú de admin, podemos ver que la base de datos se ha actualizado con este nuevo producto, en la figura 26 se muestra la base de datos con el producto encriptado con el método explicado previamente, las cadenas de nombre y descripción de producto tienen un límite, en caso de superar ese límite se desborda la memoria asignada para ese dato. Una vez dado de alta el componente, ahora cualquier usuario puede agregarlo a su carrito para después comprarlo.

```
Ingrese el ID del producto: 3
Ingrese el precio del producto: 20
Ingrese el nombre del producto [20 caracteres max]: AND gate
Ingrese la descripcion del producto [100 caracteres max]: Logic gate
```

Figura 25. Inicio de sesión como Administrador. Fuente: Elaboración propia.

```
1 Diode LED rojo 1.5V
2 Diode Zener 5.1V 5W
3 Jumper metro
4 Cableado la Variable
5 AND Gate tro
6 Logic gate Variable
```

Figura 26. Base de datos de productos actualizada. Fuente: elaboración propia.

Para esta función se implementó una función llamada “add_product()” la cuál funciona para actualizar cualquier producto registrado en la base de datos “products.txt”. Tal como se muestra en la figura 27, esta función no solo puede agregar productos, si no también editar sus valores y hasta eliminarlos de la base de datos. En la línea 177 de la figura 27, se observa que si el producto que desea registrar ya se encuentra en la base de datos, se preguntará al usuario si se desea actualizar su información antes indicada.

```
175 while(fread(&aux, sizeof(Product), 1, fp)){
176     if(aux.ID == ID){
177         printf("El producto ya existe, desea actualizarlo? [y/n]\n");
178         scanf("%c", &option);
179
180         if(option == 'y'){
181             strcpy(aux.name, name);
182             strcpy(aux.description, description);
183             aux.price = price;
184
185             found = 1;
186         }
187     }else if(aux.ID == -ID)
188         continue;
189
190     fwrite(&aux, sizeof(Product), 1, temp);
191 }
```

Figura 27. Función que edita la base de datos “products.txt”. Fuente: Elaboración propia.

Agregar Usuario

En esta opción agregamos un usuario, inicialmente esta función estaría en el menú principal o al errar la contraseña al iniciar sesión, esto se espera agregar en la próxima fase del proyecto, por ahora la única forma de registrar un usuario es mediante el menú de admin, esta opción pide ingresar un nombre de usuario y contraseña como se muestra en la figura 2. En caso de que el nombre de usuario ingresado ya se encuentre registrado, se pregunta si se desea cambiar la contraseña por la ingresada anteriormente como se muestra en la figura 29.

```
Ingrese el nombre del Usuario [20 caracteres max]: Elon
Ingrese la contraseña [20 caracteres max]: 1234
```

Figura 28. Dada de alta de un usuario. Fuente: Elaboración propia.

```
Ingrese el nombre del Usuario [20 caracteres max]: Elon
Ingrese la contraseña [20 caracteres max]: 1234
El Usuario ya existe, desea cambiar la contraseña? [y/n]
```

Figura 29. Cambio de contraseña de usuario existente. Fuente: Elaboración propia.

En caso de que demos de alta un usuario que ya existía, si decimos que si a la opción de cambiar contraseña, la contraseña que se quedara es la última escrita.

Eliminar Producto

Esta opción elimina un producto que exista, simplemente necesitamos especificar el ID del producto a eliminar de la base de datos como se muestra en la figura 30, al dar enter nos regresa al menú del administrador.

```
Ingrese el ID del producto: 5
```

Figura 30. Eliminando un producto con su id. Fuente: Elaboración propia.

Eliminar Usuario

Esta función recibe el nombre del usuario a borrar, y al igual que la función de borrar producto, solo damos enter y se da de baja el usuario de la base de datos. A continuación, se muestra la base de datos previa a la eliminación de un usuario en la figura 31, después la ejecución del comando para borrar usuarios en la figura 32 y por último la base de datos (encriptada) con el usuario eliminado figura 33.

```
1 admin Elon 5 defren alexis
```

Figura 31. Base de datos inicial. Fuente: Elaboración propia.

```
Ingrese el nombre del usuario: Elon
```

Figura 32. Eliminación del usuario Elon. Fuente: Elaboración propia.

```
admin defren alexis
```

Figura 33. Base de datos actualizada. Fuente: Elaboración propia.

Menú principal

Una vez iniciada sesión como un usuario con permisos limitados, el menú que vemos es el de la figura 34, explicaremos lo que cada opción de este menú hace, sus opciones son Agregar al carrito, ira al carrito, siguiente página, página anterior, cerrar sesión. A continuación, explicaremos lo que hace cada opción.

```

===== Bienvenido alexis =====
Articulos 1 - 10 / 3
1 Diodo Diodo LED rojo 1.5V $10
2 Diodo Zener Diodo Zener 5.1V $5
4 AND gate Logic gate $20

1. Agregar al carrito
2. Ir al carrito
3. Siguiente pagina
4. Pagina Anterior
5. Cerrar sesion

```

Figura 34. Menú principal al iniciar sesión. Fuente: Elaboración propia.

Al agregar al carrito del usuario, se modifica la estructura de datos asociada a dicho usuario, más adelante se explica el funcionamiento completo de la opción de ver el carrito del usuario, pues la opción 1 solo modifica la base de datos, las opciones 3 y 4 no funcionan en esta versión del código, y aún están a discusión de mantenerla, pues como se muestra en la figura 34, la primera mitad de la pantalla nos muestra los componentes que están a la venta, las opciones 3 y 4 se planeó que funcionaran para navegar entre listas de productos y mejorar la interfaz, pero tomando en cuenta el alcance del proyecto, no serán necesarias varias páginas por lo que la continuidad de estas opciones no se asegura para la siguiente fase del proyecto. La opción 5 nos regresa al menú de iniciar sesión

Menú del carrito del usuario

Una vez que elegimos la opción número dos en el menú principal, este nos llevará al carrito único del usuario, el cuál mostrará la lista de todos los productos que este ha agregado, su ID, el nombre del producto, descripción, precio por unidad y la cantidad de piezas agregadas. Al final de esta lista también se muestra el monto total del carrito tal como se observa en la figura 35. En caso de que el carrito de muestra tal como se observa en la figura 36. Esto con el objetivo para en una versión posterior agregar la opción de comprar todos los artículos que se encuentran en el carrito.

```

===== Tu carrito efren =====
1 Diodo $10 | 20 pzs.
2 Diodo Zener $5 | 10 pzs.

Total: $250

1. Eliminar producto
2. Regresar al menu principal

```

Figura 35. Carrito del usuario con dos productos. Fuente: Elaboración propia.

```

===== Tu carrito efren =====
Tu carrito esta vacio!
Agrega productos en el menu principal

Presione <Enter> para regresar

```

Figura 36. Carrito vacío del usuario “efren”. Fuente: Elaboración propia.

Una vez en el carrito observamos en la figura 35, que se nos dan dos opciones, la primera de eliminar algún producto del carrito y la segunda para regresar al menú principal. La primera opción es la más interesante ya que esta nos permite editar con mayor flexibilidad el carrito. Cuando se selecciona esta opción se solicitará al usuario que ingrese el ID del producto y la cantidad a eliminar como en la figura 37. Al ingresar el ID del producto y la cantidad debemos tomar en cuenta que si el ID no se encuentra en la base de datos de la tienda, se nos mostrará un mensaje de error como se muestra en la figura 38. En caso contrario, el carrito se actualizará

con la nueva cantidad de ese producto en específico, otra observación es que si se indica que se elimine una cantidad mayor a la que tenemos en el carrito, este se eliminará sin mostrar una cantidad de piezas negativa.

```

===== Tu carrito efren =====
1 Diodo $10 | 20 pzs.
2 Diodo Zener $5 | 10 pzs.

Total: $250

1. Eliminar producto
2. Regresar al menu principal
1

Indique el ID del producto a eliminar: 1

Indique la cantidad a eliminar: 10

```

Figura 37. Opción uno del carrito para eliminar algún producto de este. Fuente: Elaboración propia.

```

El producto que indico no existe, intentar de nuevo!
===== Tu carrito efren =====
1 Diodo $10 | 20 pzs.
2 Diodo Zener $5 | 10 pzs.

Total: $250

1. Eliminar producto
2. Regresar al menu principal

```

Figura 38. Se muestra mensaje de error cuando se ingresa un ID no disponible. Fuente: Elaboración propia.

Por último, la eliminación de productos en el carrito se realiza utilizando la función “add_cart_item()”, función a la cual se le envía el valor de “quantity” en la línea 145 como se muestra en la figura 394ero con valor negativo para que este se reste a la cantidad de productos que existen en el carrito, actualizando así este.

```

136 ~ if (option == '1') {
137     printf("\nIndique el ID del producto a eliminar: ");
138     scanf("%d", &ID);
139     printf("\nIndique la cantidad a eliminar: ");
140     scanf("%d", &quantity);
141     getchar();
142     get_product(cart_item, ID);
143     system("clear");
144 ~ if (cart_item->ID == ID) {
145     add_cart_item(current_user.ID_cart, ID, -quantity);
146 ~ } else {
147     printf("\nEl producto que indico no existe, intentar de nuevo!\n");
148 }
149 ~ } else if (option == '2') {
150     getchar();
151     return 1;
152 }else
153     system("clear");

```

Figura 39. Proceso de eliminar un producto en el carrito. Fuente: Elaboración propia.

Conclusiones

Aguilar Chávez Alexis Daniel

La investigación de los temas anteriores me permitió entender la complejidad que hay detrás de estos componentes y cómo trabajan juntos para lograr un funcionamiento adecuado del sistema.

Cada uno de estos componentes es importante por sí solos, pero juntos son cruciales para lograr la efectividad en la gestión de los recursos del sistema operativo y en la interacción con los usuarios y el hardware. Los procesos son la entidad básica de ejecución que permite la realización de múltiples tareas simultáneamente, y los hilos permiten que los procesos realicen varias tareas en paralelo, lo que mejora el rendimiento del sistema.

Por su parte, los semáforos son una herramienta útil para controlar el acceso a recursos compartidos y asegurarse de que los procesos no interfieran entre sí en el acceso a los recursos. Las colas de mensajes y la memoria compartida son mecanismos que permiten a los procesos comunicarse y compartir datos de manera eficiente, mejorando así la coordinación entre ellos.

Los archivos y dispositivos de entrada y salida son vitales para que los usuarios interactúen con el sistema operativo y para que el sistema operativo interactúe con el hardware. La memoria compartida y los archivos también son importantes para el almacenamiento de datos y la transferencia de información, lo que garantiza la consistencia y la integridad de los datos en el sistema.

En conclusión, la investigación nos ha permitido comprender la importancia de cada uno de estos componentes y cómo se relacionan entre sí para lograr un funcionamiento adecuado del sistema operativo. Los procesos, hilos, semáforos, colas de mensajes, memoria compartida, archivos y dispositivos de entrada y salida trabajan juntos para garantizar el funcionamiento adecuado del sistema y proporcionar una experiencia de usuario efectiva. El conocimiento de estos componentes es crucial para cualquier persona involucrada en el desarrollo, la programación o el mantenimiento de un sistema operativo, y permitirá la implementación de soluciones eficientes y efectivas para garantizar el correcto funcionamiento del sistema.

Una aplicación de tienda básica con inicio de sesión, visualización de artículos y función de agregar al carrito es una solución práctica y conveniente para los usuarios que desean realizar compras en línea, aunque en este proyecto no será una tienda en línea seguirá los mismos métodos en lenguaje c, esta aplicación proporciona una experiencia de compra, permitiendo explorar y seleccionar productos desde la comodidad de sus dispositivos.

Calva Vargas Oswaldo Enrique

Cada uno de los temas tratados tienen un gran impacto para el funcionamiento de cualquier computadora, por ejemplo, los procesos e hilos permiten a los sistemas informáticos realizar múltiples tareas de manera simultánea y mejorar el rendimiento, lo que se traduce en aplicaciones más eficientes y escalables. Además, su uso efectivo permite un mejor control y gestión de los recursos del sistema, lo que aumenta la productividad y eficacia del sistema.

En un mundo cada vez más interconectado, los procesos e hilos se han convertido en una herramienta indispensable en el desarrollo de sistemas informáticos, al permitir la ejecución simultánea de múltiples tareas, los procesos e hilos permiten la creación de aplicaciones de alta calidad que satisfacen las necesidades de los usuarios, además el uso de estos es fundamental en la optimización de la velocidad y capacidad de procesamiento del sistema.

Por otro lado, los semáforos se utilizan para garantizar el acceso exclusivo a recursos compartidos y prevenir conflictos de acceso entre procesos e hilos; los semáforos son una herramienta esencial en la creación de sistemas concurrentes y en la prevención de condiciones de carrera y deadlock, de igual forma, permiten la comunicación entre procesos e hilos, lo que los hace útiles para la sincronización de procesos y la implementación de mecanismos de control de acceso a recursos compartidos.

Las colas de mensajes, por su parte, dan la comunicación entre procesos e hilos mediante la transferencia de mensajes, as colas de mensajes permiten la comunicación asincrónica y permiten a los procesos e hilos trabajar de manera independiente, lo que aumenta la eficiencia y escalabilidad del sistema; son útiles en situaciones en las que la sincronización de procesos no es necesaria, pero se requiere la transferencia de datos entre procesos e hilos.

En las siguientes secciones del marco teórico trata sobre la memoria, los archivos que de alguna forma están involucrados en esta y los dispositivos de entrada/salida que son los que permiten la interacción usuario y computadora, además, estos periféricos igual logran el almacenamiento como mencione con anterioridad cada uno estos son importantes, no hay uno más que otro.

Es de suma importancia que nosotros como futuros ingenieros tengamos el manejo de cada uno de estos conceptos tratados, ya que, de alguna forma estamos totalmente involucrados en cada uno de ellos, por ello debemos considerar cuidadosamente los requisitos de entrada/salida de los usuarios y diseñar interfaces intuitivas y fáciles de usar para garantizar la eficiencia del sistema.

En resumen, el conocimiento y la aplicación adecuada de estos temas son esenciales para el desarrollo y mantenimiento de sistemas informáticos de alta calidad de igual forma sin estos sería imposible trabajar con una computadora.

Los semáforos desempeñan un papel fundamental en la sincronización y el control de acceso a los recursos compartidos, evitando condiciones de carrera y asegurando una ejecución ordenada de los procesos, se utilizan para garantizar la coherencia y la integridad de los datos, así como para prevenir conflictos y garantizar la exclusión mutua cuando varios usuarios intentan acceder a los mismos recursos.

Flores Flores Rodrigo Ernesto

Siempre que comenzamos un proyecto es de suma importancia tener un conocimiento previo sobre las herramientas que necesitamos para desarrollarlo, por lo cual hacer una investigación previa es fundamental para tener un buen comienzo. El hecho de tener la idea general de los recursos será de gran beneficio para encontrar de la mejor manera la metodología correcta a seguir. En este proyecto es importante conocer una gran variedad de información, pero principalmente sobre procesos y como se llevan a cabo dentro de nuestro sistema, así como los diferentes recursos de los que disponen para que su desarrollo no sea tan complejo.

Ahora, los hilos serán una línea de ejecución de cualquier proceso, es decir son los pasos dentro del proceso. Entre los beneficios que nos da usar hilos es que los tiempos de respuesta son mejores, ya que el hecho de que una parte de algún código este bloqueado no impide que otro pueda estar ejecutándose. Usualmente los procesos necesitaran estar comunicados entre sí, de preferencia de una forma bien estructurada, pero debemos tener en cuenta diversas situaciones como por ejemplo que algunos procesos no se interpongan entre sí, así mismo tomar en cuenta las dependencias entre procesos, para esto podemos utilizar los semáforos y las colas de mensajes.

Es fundamental conocer por completo los recursos con los que podemos trabajar tanto en software como en hardware, ya que si en nuestro planteamiento comenzamos queriendo utilizar recursos con los que no contamos podría convertirlo en un proceso más complejo al momento de adaptar todo a los recursos con los que contamos, por lo tanto es de suma importancia saber con qué recursos contamos e implementar la forma más óptima para que con los recursos que contamos lleguemos a los resultados esperados.

Además de tener total conocimiento sobre los recursos necesarios para realizar nuestro proyecto, es necesario delimitar los alcances de nuestro proyecto, es decir, no podemos empezar desarrollando un proyecto enorme a nivel de una empresa, por lo que concientizando sobre la falta de recursos que tienen los negocios pequeños fue que decidimos plantear una solución para lograr una inserción de estos al mercado moderno, queremos que negocios pequeños cuenten con las nuevas tecnologías, para que sean más atractivos para los clientes, además de que estén actualizados y eso los impulse a motivarse a crecer como negocio. Claramente esperamos que con la implementación de este proyecto tendrá un gran impacto en la economía y el crecimiento de estos pequeños negocios. De igual manera consideramos que con la ejecución de este proyecto también estaremos influyendo en el desarrollo tecnológico de la sociedad, con lo que aportamos nuestro granito de arena al impulso por mejorar la calidad de vida de las diferentes comunidades.

En conclusión, la implementación de una aplicación de tienda básica de componentes electrónicos utilizando archivos, memoria compartida, semáforos e hilos ofrece una solución robusta y eficiente para satisfacer las necesidades de los usuarios en el entorno de comercio electrónico. Esta aplicación aprovecha estas tecnologías avanzadas para lograr un rendimiento óptimo, una gestión eficaz de recursos y una experiencia de usuario fluida.

Jiménez Luna Rodrigo Efrén

Durante el desarrollo del protocolo, se abarco la situación actual acerca de la venta de componentes electrónicos a través de herramientas online. En el año 2020, una pandemia azotaría al mundo y cambiaría totalmente la forma en que veríamos los objetivos a largo plazo. A causa de la cuarentena, gran variedad de escenarios llenos de depresión, ansiedad, enojo, etc. Situaciones donde la salud mental y física se vieron comprometidas.

Además de esto, la pandemia también provoco múltiples estragos en la industria, desde la automotriz hasta la agropecuaria. En este caso el proyecto tiene como objetivo resolver uno de los problemas que se generaron, el cuál fue la gran escasez de componentes electrónicos, problema que, a su vez, fue consecuencia de la poca producción del silicio, así como las pocas empresas que tienen los recursos económicos y tecnológicos para la manufactura y manejo de estos componentes.

Actualmente, la complejidad de los componentes electrónicos ha crecido de forma exponencial; la batalla por crear un transistor cada más vez más pequeño ha llegado al punto de rozar la barrera donde las leyes de la física limitan el sueño de contar con computadoras más potentes. Añadido a esto, la demanda de componentes también ha ido en aumento, no solo en componentes específicos sino también los que se utilizan para aplicaciones más generales. Siendo los principales clientes estudiantes, empresarios, ingenieros, etc. Es por ello que la implementación de proyectos como este, que ayudan a mejorar la forma en la que se gestionan diversos tipos de ventas a través de internet.

Tomando en cuenta que la solución que se propuso en este protocolo no cuenta con un alcance muy extenso como el de una tienda online tal como las que conocemos hoy en día, con este proyecto se busca proponer soluciones diferentes a las ya existentes. En este caso, un sistema para el control de una tienda que se encargue de distribuir componentes electrónicos de diversos tipos, ofreciendo un amplio abanico de posibilidades para conseguir los materiales que permitirán a futuros ingenieros, estudiantes o público en general puedan realizar sus proyectos de una forma rentable y funcional.

Se optó por utilizar el sistema operativo Linux como base para el proyecto ya que supone una mejor implementación, así como facilitar el proceso, esto ya que Linux permite una mejor gestión de procesos a diferencia de otros sistemas operativos (Windows o MacOS). Linux también cuenta con otras ventajas al ser un sistema "open source" el cuál al tener una comunidad activa que contribuye día a día en parches de seguridad y mayor número de actualizaciones, dando como resultado que nuestro sistema sea también más seguro además de permitir un fácil mantenimiento de este a través del tiempo.

En general, una aplicación de tienda básica de componentes electrónicos que emplea archivos, memoria compartida, semáforos e hilos combina eficientemente estas tecnologías para proporcionar una plataforma segura, rápida y escalable, al aprovechar al máximo estas herramientas, se logra una gestión eficaz de los recursos, una comunicación fluida y una experiencia de usuario satisfactoria, mejorando así la eficiencia de las operaciones comerciales y brindando un entorno de compra confiable y atractivo para los usuarios.

Referencias

[1] "Que son las tic", Bmn. [En línea]. Disponible en: <http://www.bmnns.sld.cu/que-son-las-tic> [Accedido el 28 de abril de 2023]

- [2] “Las TIC’s como herramienta para el comercio electrónico”, Digital Publisher. [En línea]. Disponible en: https://www.593dp.com/index.php/593_Digital_Publisher/article/download/605/707/5095 [Accedido el 28 de abril de 2023]
- [3] M. Martínez-Domínguez, “El comercio electrónico durante la pandemia de Covid-19”, Cátedra Conacyt-CIESAS-Pacífico Sur. [En línea]. Disponible en: <https://ichan.ciesas.edu.mx/el-comercio-electronico-durante-la-pandemia-de-covid-19/>. [Accedido el 28 de abril de 2023]
- [4] “Comercio electrónico”, rockcontent. [En línea]. Disponible en: <https://rockcontent.com/es/blog/historia-del-comercio-electronico/> [Accedido el 01 de junio de 2023]
- [5] “Evolución componentes electrónicos”, Sergio Soriano. [En línea]. Disponible en: <https://www.thesergioscorner.com/post/2015/06/18/evoluci%C3%B3n-de-los-componentes-electr%C3%B3nicos> [Accedido el 01 de junio de 2023]
- [6] “Procesos”, IBM, 2021, abril. [En línea]. Disponible en: <https://www.ibm.com/docs/es/aix/7.2?topic=processes->. [Accedido: 2-abr-2023]
- [7] “Proceso (informática)”, AcademiaLab, 2023. [En línea]. Disponible en: <https://academia-lab.com/enciclopedia/proceso-informatica/>. [Accedido: 2-abr-2023]
- [8] “PROCESOS”, *Department of Computer Science and Information Technologies of the University of A Coruña*, pp. 42 2023. [En línea]. Disponible en: https://www.dc.fi.udc.es/~so-grado/2_PROCESOS.pdf. [Accedido: 1-abr-2023]
- [9] “XConcepto Hilos”.UAM MX, PDF, México. 2023. [En línea]. Disponible en: <http://aisii.azc.uam.mx/areyes/archivos/licenciatura/sd/U2/ConceptoHilos.pdf> [Accedido: 2-abr-2023]
- [10] "Hilo (Informática) - Sutil Web". Sutil Web. Sutil Web. [En línea]. Disponible en: <https://sutilweb.com/2021/10/14/hilo-informatica/>. [Accedido: 2-abr-2023].
- [11] ZHilos". E. Lozada. Sistemas Operativos. [En línea]. Disponible en: <https://edwardlozadaso.wordpress.com/hilos/>. [Accedido: 3-abr-2023].
- [12] "Sistemas Operativos: Semáforos", D. Pozo. Ciencia de la Computación. <https://danielpozoblog.wordpress.com/2016/09/28/sistemas-operativos-semaforos/>. [Accedido: 3-abr-2023]
- [13] "Concepto de Semáforo". RasterSoft. Raster's personal web. <https://www.rastersoft.com/OS2/CURSO/SEMAFORO.HTM>. [Accedido: 2 de abril de 2023].

[14] "SISTEMAS OPERATIVOS, PROCESOS CONCURRENTES-Unidad III" .K. Suzely. AIU | Online Adult Education | Create the future you deserve. <https://www.aiu.edu/spanish/publications/student/spanish/180-207/sistemas-operativos-procesos-concurren-tes-unidad-iii.html> (accedido el 2 de abril de 2023).

[15] "¿Qué es una cola de mensajes?", Icy Science. [En línea]. Disponible en: <https://es.theastrologypage.com/message-queue> [Accedido el 2 de abril de 2023]

[16] "¿Qué es una cola de mensajes?", Amazon Web Services. [En línea]. Disponible en: <https://aws.amazon.com/es/message-queue/> [Accedido el 2 de abril de 2023]

[17] "¿Qué es la memoria compartida en el sistema operativo?", CompuHoy. [En línea]. Disponible en: <https://www.compuhoy.com/que-es-la-memoria-compartida-en-el-sistema-operativo/> [Accedido el 2 de abril de 2023]

[18] "IBM Documentation", IBM. [En línea]. Disponible en: <https://www.ibm.com/docs/es/power8?topic=memory-shared> [Accedido el 2 de abril de 2023]

[19] "¿Qué es la memoria compartida?", COMPUTAEX, 2021. [En línea]. Disponible en: <https://www.cenits.es/faq/preguntas-generales/que-es-la-memoria-compartida> [Accedido el 2 de abril de 2023]

[20] "Memoria Compartida Distribuida", Memoria Compartida Distribuida, 2015. [En línea]. Disponible en: <https://iggvavl.wordpress.com/> [Accedido el 2 de abril de 2023]

[21] "Configuraciones Memoria Compartida Distribuida", Google Sites. [En línea]. Disponible en: <https://sites.google.com/site/soiiunidad4y5/home/4-1> [Accedido el 2 de abril de 2023]

[22] "Archivo informático". Concepto de. [En línea]. Disponible en <https://concepto.de/archivo-informatico/> .(Accedido el 30 de marzo de 2023).

[23] "¿Qué es un archivo?". Geeknetic. [En línea]. Disponible en: <https://concepto.de/archivo-informatico/>. [Accedido el 02 de abril de 2023].

[24] "Definición archivo". Sistemas Master. [En línea]. Disponible en <https://sistemas.com/archivo.php> . [Accedido el 31 de marzo de 2023].

[25] "Tipos de archivo". Tipos. [En línea]. Disponible en <https://www.tipos.co/tipos-de-archivo/> . [Accedido el 31 de marzo de 2023].

[26] “Dispositivos de entrada y salida”. Diferenciador. [En línea]. Disponible en: <https://www.diferenciador.com/dispositivos-de-entrada-y-salida/#:~:text=Los%20dispositivos%20de%20entrada%20y,manejar%20un%20soporte%20de%20informaci%C3%B3n> . [Accedido el 31 de marzo de 2023].

[27] Alvy, “El algoritmo SHA-256 explicado y visualizado paso a paso, bit a bit”, *Microsiervos*, 2022. [En línea]. Disponible en: <https://www.microsiervos.com/archivo/seguridad/algoritmo-sha-256-explicado-visualizado-paso-a-paso-bit-a-bit.html> [Accedido el 4 de junio de 2023].

Anexos

Códigos

Menus.h

```
#ifndef MENUS_H_
#define MENUS_H_

int login();
int menu();
int cart();
int purchase();
int admin_menu();
```

```
#endif
```

Const.h

```
#ifndef CONST_H_
#define CONST_H_

#define VALID_PTR(x) ((x) == NULL)

#define USER_SIZE 20
#define PASSWORD_SIZE 20

typedef struct {
    char user_name[20];
    char password[20];
    int ID_cart;
} User;

typedef struct{
    char name[20];
    char description[100];
    int ID;
    int price;
} Product;

typedef struct{
    int cart_ID;
    int product_ID;
```

```

    int quantity;
} cart_Item;

```

```

#endif

```

Data_base.h

```

#ifndef DATA_BASE_H_
#define DATA_BASE_H_

```

```

#include "const.h"

```

```

int get_user(User* current_user, const char* user_name, const char* password);

```

```

Product* get_product(Product* current_product, int ID);

```

```

int* get_cart_item(const int cart_ID, int* item_ID, int* quantity, int index);

```

```

int get_product_count();

```

```

int get_cart_count();

```

```

void add_cart_item(const int ID_cart, const int Item_ID, const int quantity);

```

```

void add_product(const int ID, const char* name, const char* description, const int price);

```

```

void add_user(const char* name, const char* password);

```

```

void delete_cart(const char* name);

```

```

#endif

```

Encryption.h

```

#ifndef ENCRYPTION_H_
#define ENCRYPTION_H_

```

```

char* encrypt_password(char* password);

```

```

#endif

```

Main.c

```

#include "../lib/menus.h"

```

```

#include "stdlib.h"

```

```

int main() {

```

```

    int state = 0;

```

```

    while(1){

```

```

        if(state == -1){

```

```

            state = admin_menu();

```

```

            system("clear");

```

```

        }else if(state == 0){

```

```

            state = login();

```

```

            system("clear");

```

```

        }else if(state == 1){

```

```

            state = menu();

```

```

            system("clear");

```

```

    }else if(state == 2){
        state = cart();
        system("clear");
    }
}

return 0;
}

```

Menus.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

```

#include "../lib/const.h"
#include "../lib/data_base.h"
#include "../lib/encryption.h"
#include "../lib/menus.h"

```

```

User current_user;

```

```

int login() {
    char input_User_name[USER_SIZE];
    char input_Password[PASSWORD_SIZE];
    int login_check = 0;

    system("clear");

    while (1) {
        printf("===== Iniciar Sesion =====\n");
        printf("Ingrese el nombre de usuario: ");
        scanf("%s", input_User_name);
        printf("Ingrese la contrase%ca: ", 164);
        scanf("%s", input_Password);

        encrypt_password(input_Password);

        getchar();

        login_check = get_user(&current_user, input_User_name, input_Password);

        if (login_check == 1 || login_check == -1) {
            system("clear");
            return login_check;
        } else {
            system("clear");
            printf("Usuario o contraseña incorrectos D:\n\n");
        }
    }
}

```

```

int menu() {

```

```

char option;
int first = 1, interval = 10;
int total = get_product_count();
int ID, quantity;

while (1) {
    Product *list_product = (Product *)malloc(sizeof(Product));

    printf("===== Bienvenido %s =====\n",
        current_user.user_name);
    printf("Articulos %d - %d / %d\n", first, first + interval - 1, total);

    for (int i = 0; i < interval; i++) {
        get_product(list_product, i + first);
        if (i + first <= total)
            printf("%d %s %s $%d\n", list_product->ID, list_product->name,
                list_product->description, list_product->price);
    }

    printf("\n1. Agregar al carrito\n");
    printf("2. Ir al carrito\n");
    printf("3. Siguiente pagina\n");
    printf("4. Pagina Anterior\n");
    printf("5. Cerrar sesion\n");

    option = getchar();

    if (option == '1') {
        system("clear");
        printf("Indique el ID del producto a agregar: ");
        scanf("%d", &ID);
        printf("Indique la cantidad: ");
        scanf("%d", &quantity);

        getchar();

        get_product(list_product, ID);

        if (list_product->ID == ID) {
            add_cart_item(current_user.ID_cart, ID, quantity);
        } else {
            system("clear");
            printf("\nEl producto que se indico no existe, intentar de nuevo!\n");
        }
    } else if (option == '2') {
        system("clear");
        getchar();
        return 2;
    } else if (option == '3' && first + interval - 1 < total) {
        system("clear");
        first += interval;
    } else if (option == '4' && first > 1) {
        system("clear");
        first -= interval;
    }
}

```

```

    } else if (option == '5')
        return 0;
    else
        system("clear");

    free(list_product);
}
}

int cart() {
    int total;
    int total_cart_amount;
    int ID, quantity;
    char option;

    while (1) {
        total = get_cart_count(current_user.ID_cart);
        printf("===== Tu carrito %s =====\n",
            current_user.user_name);

        if (total > 0) {
            Product *cart_item = (Product *)malloc(sizeof(Product));
            total_cart_amount = 0;

            for (int i = 0; i < total; i++) {
                get_cart_item(current_user.ID_cart, &ID, &quantity, i);
                get_product(cart_item, ID);
                total_cart_amount += cart_item->price * quantity;
                printf("%d %s $%d | %d pzs.\n", cart_item->ID, cart_item->name,
                    cart_item->price, quantity);
            }

            printf("\n\t\tTotal: $%d\n", total_cart_amount);

            printf("\n1. Eliminar producto\n");
            printf("2. Regresar al menu principal\n");

            option = getchar();

            if (option == '1') {
                printf("\nIndique el ID del producto a eliminar: ");
                scanf("%d", &ID);
                printf("\nIndique la cantidad a eliminar: ");
                scanf("%d", &quantity);
                getchar();
                get_product(cart_item, ID);
                system("clear");
                if (cart_item->ID == ID) {
                    add_cart_item(current_user.ID_cart, ID, -quantity);
                } else {
                    printf("\nEl producto que indico no existe, intentar de nuevo!\n");
                }
            } else if (option == '2') {
                getchar();
            }
        }
    }
}

```

```

        return 1;
    }else
        system("clear");

    free(cart_item);
} else {
    printf("Tu carrito esta vacio!\nAgrega productos en el menu principal\n");

    printf("\nPresione <Enter> para regresar\n");

    option = getchar();

    if (option == '\n')
        return 1;
    }
}
}

int purchase() { return 0; }

int admin_menu() {
    int access = 0;
    int ID, price;
    char product_name[20], description[100];
    char user_name[USER_SIZE], password[PASSWORD_SIZE];
    char option;

    while (1) {
        system("clear");
        printf("===== Panel de Administrador =====\n");
        printf("Opciones: \n\n");
        printf("1. Agregar producto\n");
        printf("2. Agregar Usuario\n");
        printf("3. Eliminar producto\n");
        printf("4. Eliminar Usuario\n");
        printf("5. Salir del panel de Administrador\n");

        option = getchar();

        if (option == '1') {
            system("clear");
            printf("Ingrese el ID del producto: ");
            scanf("%d", &ID);
            printf("Ingrese el precio del producto: ");
            scanf("%d", &price);
            getchar();
            printf("Ingrese el nombre del producto [20 caracteres max]: ");
            scanf("%[^\\n]*c", product_name);
            getchar();
            printf("Ingrese la descripcion del producto [100 caracteres max]: ");
            scanf("%[^\\n]*c", description);
            getchar();

            add_product(ID, product_name, description, price);

```



```

} else if (option == '2') {
    system("clear");
    printf("Ingrese el nombre del Usuario [%d caracteres max]: ", USER_SIZE);
    scanf("%s", user_name);
    getchar();
    printf("Ingrese la contrase%ca [%d caracteres max]: ", 164,
        PASSWORD_SIZE);
    scanf("%s", password);
    getchar();

    encrypt_password(password);

    add_user(user_name, password);
} else if (option == '3') {
    system("clear");
    printf("Ingrese el ID del producto: ");
    scanf("%d", &ID);
    getchar();

    add_product(-ID, "", "", 0);
} else if (option == '4') {
    system("clear");
    printf("Ingrese el nombre del usuario: ");
    scanf("%s", user_name);

    delete_cart(user_name);
    add_user(user_name, "delete");
} else if (option == '5')
    return 0;
else
    system("clear");
}
}

```

Data_base.c

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "../lib/data_base.h"

int get_user(User* current_user, const char* user_name, const char* password){
    FILE *fp = fopen("../database/users.txt", "r");

    if(VALID_PTR(fp)){
        fprintf(stderr, "Error while opening user database!\n");
        exit(1);
    }

    while(fread(current_user, sizeof(User), 1, fp)){
        if(!strcmp(current_user->user_name, user_name) && !strcmp(current_user->password, password)){
            if(!strcmp(current_user->user_name, "admin")){
                fclose(fp);

```

```

        return -1;
    }
    fclose(fp);
    return 1;
}

fclose(fp);
return 0;
}

Product* get_product(Product* current_product, int ID){
FILE *fp = fopen("../database/products.txt", "r");

    if(VALID_PTR(fp)){
        fprintf(stderr, "Error while opening product database!\n");
        exit(1);
    }

    while(fread(current_product, sizeof(Product), 1, fp)){
        if(current_product->ID == ID)
            break;
    }

    fclose(fp);
};

int get_product_count(){
    FILE *fp = fopen("../database/products.txt", "r");
    Product aux;
    int count = 0;

    while(fread(&aux, sizeof(Product), 1, fp)){
        count++;
    }

    fclose(fp);

    return count;
}

int get_cart_count(const int ID_cart){
    FILE *fp = fopen("../database/cart.txt", "r");

    if(VALID_PTR(fp)){
        fprintf(stderr, "Error while opening cart database!\n");
        exit(1);
    }

    int count = 0;
    cart_Item aux;

    while(fread(&aux, sizeof(cart_Item), 1, fp)){
        if(aux.cart_ID == ID_cart)

```

```

        count++;
    }

    fclose(fp);
    return count;
}

void add_cart_item(const int ID_cart, const int Item_ID, const int quantity){
    FILE *fp = fopen("../database/cart.txt", "r");
    FILE *temp = fopen("../database/temp.txt", "a");
    cart_Item aux;
    int found = 0;

    while(fread(&aux, sizeof(cart_Item), 1, fp)){
        if(aux.cart_ID == ID_cart && aux.product_ID == Item_ID){
            aux.quantity += quantity;
            found = 1;

            if(aux.quantity <= 0)
                continue;
        }

        fwrite(&aux, sizeof(cart_Item), 1, temp);
    }

    if(!found && quantity > 0){
        aux.cart_ID = ID_cart;
        aux.product_ID = Item_ID;
        aux.quantity = quantity;

        fwrite(&aux, sizeof(cart_Item), 1, temp);
    }

    fclose(fp);
    fclose(temp);

    remove("../database/cart.txt");
    rename("../database/temp.txt", "../database/cart.txt");
}

int get_user_ID_cart(const char *user_name){
    FILE *fp = fopen("../database/users.txt", "r");
    User current_user;

    while(fread(&current_user, sizeof(User), 1, fp)){
        if(!strcmp(current_user.user_name, user_name)){
            fclose(fp);
            return current_user.ID_cart;
        }
    }

    fclose(fp);
}

```

```

void delete_cart(const char* name){
    int ID_cart = get_user_ID_cart(name);
    FILE *fp = fopen("../database/cart.txt", "r");
    FILE *temp = fopen("../database/temp.txt", "a");
    cart_Item aux;

    while(fread(&aux, sizeof(cart_Item), 1, fp)){
        if(aux.cart_ID == ID_cart){
            continue;
        }

        fwrite(&aux, sizeof(cart_Item), 1, temp);
    }

    fclose(fp);
    fclose(temp);

    remove("../database/cart.txt");
    rename("../database/temp.txt", "../database/cart.txt");
}

int* get_cart_item(const int cart_ID, int* item_ID, int* quantity, int index){
    FILE *fp = fopen("../database/cart.txt", "r");
    cart_Item aux;

    int i = 0;

    while(fread(&aux, sizeof(cart_Item), 1, fp)){
        if(cart_ID == aux.cart_ID){
            if(i == index){
                *item_ID = aux.product_ID;
                *quantity = aux.quantity;
                break;
            }else
                i++;
        }
    }

    fclose(fp);
}

void add_product(const int ID, const char* name, const char* description, const int price){
    FILE *fp = fopen("../database/products.txt", "r");
    FILE *temp = fopen("../database/temp.txt", "a");
    Product aux;
    int found = 0;
    char option;

    while(fread(&aux, sizeof(Product), 1, fp)){
        if(aux.ID == ID){
            printf("El producto ya existe, desea actualizarlo? [y/n]\n");
            scanf("%c", &option);

            if(option == 'y'){

```

```

        strcpy(aux.name, name);
        strcpy(aux.description, description);
        aux.price = price;

        found = 1;
    }
} else if(aux.ID == -ID)
    continue;

fwrite(&aux, sizeof(Product), 1, temp);
}

if(!found && ID > 0){
    aux.ID = ID;
    strcpy(aux.name, name);
    strcpy(aux.description, description);
    aux.price = price;

    fwrite(&aux, sizeof(Product), 1, temp);
}

fclose(fp);
fclose(temp);

remove("../database/products.txt");
rename("../database/temp.txt", "../database/products.txt");
}

void add_user(const char* name, const char* password){
    FILE *fp = fopen("../database/users.txt", "r");
    FILE *temp = fopen("../database/temp.txt", "a");
    User aux;
    int found = 0, deleted = 0;
    char option;

    while(fread(&aux, sizeof(User), 1, fp)){
        if(!strcmp(aux.user_name, name) && strcmp(password, "delete")){
            printf("El Usuario ya existe, desea cambiar la contrase%ca? [y/n]\n", 164);
            scanf("%c", &option);

            if(option == 'y'){
                strcpy(aux.password, password);
                found = 1;
            }
        } else if(!strcmp(aux.user_name, name) && !strcmp(password, "delete") && !deleted){
            deleted = 1;
            continue;
        }
    }

    fwrite(&aux, sizeof(User), 1, temp);
}

if(!found && !deleted && strcmp(password, "delete")){
    strcpy(aux.user_name, name);

```

```

    strcpy(aux.password, password);
    aux.ID_cart += 1;

    fwrite(&aux, sizeof(User), 1, temp);
}

fclose(fp);
fclose(temp);

remove("../database/users.txt");
rename("../database/temp.txt", "../database/users.txt");
}

```

Encryption.c

```

#include "../lib/encryption.h"
#include "../lib/const.h"

char* encrypt_password(char* password){
    for(int i = 0; i < PASSWORD_SIZE; i++)
        password[i] >>= 1;
}

```