



**INSTITUTO POLITÉCNICO NACIONAL**  
**Escuela Superior de Cómputo**

**Ingeniería en Sistemas Computacionales**



# **Proyecto fase 4: El rompecabezas**

**PRESENTAN**

- **Suárez Bautista José Manuel**
- **Sandoval Hernández Eduardo**
- **Renteria Arriaga Josue**
- **Montenegro Cervantes Tania Viridiana**

**Unidad de aprendizaje: Sistemas Operativos**

**Grupo: 4CM2**

**07 de noviembre de 2021**

## Resumen

# Índice

|   |    |
|---|----|
| Introducción .....                        | 6  |
| Objetivos .....                           | 8  |
| Objetivos específicos: .....              | 8  |
| Justificación .....                       | 9  |
| Marco teórico .....                       | 11 |
| a) Procesos .....                         | 11 |
| b) Hilos.....                             | 12 |
| c) Semáforos.....                         | 13 |
| d) Colas de mensaje .....                 | 15 |
| e) Memoria compartida .....               | 17 |
| f) Archivos.....                          | 19 |
| g) Dispositivos de entrada y salida ..... | 21 |
| Análisis .....                            | 24 |
| Diseño .....                              | 27 |
| Implementación y pruebas.....             | 33 |
| Conclusiones .....                        | 45 |
| Referencias.....                          | 48 |
| Anexo 1. Códigos completos.....           | 50 |
| Anexo 2. Protocolo.....                   | 71 |
| Resumen.....                              | 71 |
| Palabras clave.....                       | 71 |
| Introducción .....                        | 71 |
| Objetivos .....                           | 73 |
| Justificación .....                       | 73 |
| Productos o Resultados esperados .....    | 75 |
| Metodología .....                         | 76 |

## Índice de Figuras

|   |    |
|---|----|
| <b>Figura 1.</b> Contador de programa de los procesos.....  | 11 |
| <b>Figura 2.</b> Código para la creación de procesos usando la llamada al sistema fork [8]. ..... | 12 |
| <b>Figura 3.</b> Ejemplificación del funcionamiento de los hilos. ....                            | 12 |
| <b>Figura 4.</b> Código para la ejecución de hilos en Java .....                                  | 13 |
| <b>Figura 5.</b> Reglas de los Semáforos.....   | 13 |
| <b>Figura 6.</b> Código para manipular semáforos .....  | 14 |
| <b>Figura 7.</b> Código para iniciar la variable de un semáforo .....                             | 14 |
| <b>Figura 9.</b> Código para la ejecución de Semáforos.....                                       | 15 |
| <b>Figura 10.</b> Reglas de los Semáforos [17].....   | 16 |
| <b>Figura 11.</b> Colas de Mensajes en Python.....  | 16 |
| <b>Figura 12.</b> Colas de Mensajes en Java. ....   | 17 |
| <b>Figura 13.</b> Funcionamiento de la memoria compartida .....                                   | 18 |
| <b>Figura 14.</b> Estructura general del sistema. ....  | 24 |
| <b>Figura 15.</b> Interacción del control con el cliente. ....                                    | 24 |
| <b>Figura 16.</b> Interacción del cliente con el carrito. ....                                    | 25 |
| <b>Figura 17.</b> Interacción del cliente con el catálogo. ....                                   | 25 |
| <b>Figura 18.</b> Interacción del proveedor con el catálogo. ....                                 | 26 |
| <b>Figura 19.</b> Estructura de Cliente.....  | 27 |
| <b>Figura 20.</b> Estructura de Producto. ....  | 28 |
| <b>Figura 21.</b> Estructura de Carrito. ....   | 29 |
| <b>Figura 22.</b> Funciones de semáforos.....   | 30 |
| <b>Figura 23.</b> Funciones para modificar archivo de carritos.....                               | 32 |
| <b>Figura 24.</b> Código del programa Control. ....   | 33 |
| <b>Figura 25.</b> Funciones atender_cliente y atender_proveedor. ....                             | 34 |
| <b>Figura 26.</b> Código principal del programa Cliente. ....                                     | 35 |
| <b>Figura 27.</b> Código principal del programa Proveedor.....                                    | 36 |
| <b>Figura 28.</b> Sistema atendiendo el registro de dos clientes.....                             | 37 |
| <b>Figura 29.</b> Segundo cliente terminando su registro. ....                                    | 38 |
| <b>Figura 30.</b> Dos proveedores siendo atendidos. ....  | 38 |
| <b>Figura 31.</b> Segundo proveedor registrando su producto.....                                  | 39 |
| <b>Figura 32.</b> Primer proveedor consultando el producto con id: 1.....                         | 39 |
| <b>Figura 33.</b> Segundo proveedor modificando el stock del producto con id: 0. ....             | 40 |

|  |    |
|--|----|
| <b>Figura 34.</b> Segundo proveedor consultando el producto con id: 0.....     | 40 |
| <b>Figura 35.</b> Los dos clientes iniciando sesión.....                       | 41 |
| <b>Figura 36.</b> Dos clientes interactuando con el sistema. ....              | 42 |
| <b>Figura 37.</b> Primer cliente consultando por segunda vez el catálogo. .... | 43 |
| <b>Figura 38.</b> Reinicio del sistema e inicio de sesión de un cliente. ....  | 44 |

## Índice de Tablas

|   |    |
|---|----|
| <b>Tabla A.</b> Tipos de empresas en México .....                           | 6  |
| <b>Tabla 1.</b> Extensiones comunes de archivos. ....                       | 19 |
| <b>Tabla 2.</b> Atributos de archivos.....                                  | 21 |
| <b>Tabla 3.</b> Velocidades de transferencia de datos en dispositivos. .... | 22 |

## Introducción

Actualmente las actividades cotidianas se vuelven cada vez más sencillas de realizar con ayuda de la tecnología, existen un sinnúmero de aplicaciones y sistemas informáticos que se especializan en un sector en específico y son ideales para cumplir con una tarea en particular. Es necesario definir qué es un sistema informático para explicar mejor algunos puntos que se tratarán a continuación, de forma general es aquel que puede procesar cualquier tipo de información que se pueda representar de forma matemática, ya sean cálculos básicos hasta tratar de analizar el pensamiento humano [1], por esta razón son utilizados por grandes empresas de todo tipo para mejorar su organización. Existen diferentes tipos de empresas que gozan de los beneficios de un sistema informático, pero en este caso el enfoque son los sistemas informáticos que se basan en la gestión de ventas en línea para un pequeño establecimiento (tienda de conveniencia) cuyo principal problema se refiere a mejorar las búsquedas al catálogo de productos al que acceden los clientes, que también permita agregar artículos en existencia y mejore el proceso de añadir productos al carrito principalmente, posteriormente se explicará el funcionamiento en el apartado “Productos o Resultados esperados”.

Las necesidades que necesita cumplir un sistema varían dependiendo de quién lo va a ocupar, se requiere diferenciar entre los tipos de clientes que pueden solicitar este tipo de software para sus negocios como se muestra en la Tabla A.

| Tipo de cliente   | Cantidad de empleados | Volumen de ventas  |
|-------------------|-----------------------|--|
| Grandes empresas  | 101 hasta 251         | Superiores a los 250 millones de pesos                                     |
| Medianas empresas | 31 hasta 100          | Varían desde los 100 millones y pueden superar hasta 250 millones de pesos |
| Pequeñas empresas | 11 hasta 30           | Varían entre los 4 hasta los 100 millones de pesos                         |
| Microempresas     | Menos de 10           | Menos de 4 millones de pesos   |

*Tabla A. Tipos de empresas en México*

Una vez se conocen los tipos de empresas que existen en México, es importante resaltar que con base a la problemática de este proyecto referente a la creación de un sistema informático de ventas para una tienda de conveniencia, esta pertenece al sector de las microempresas, mismas que son fundamentales para la economía del país, ya que, según datos presentados por el último Censo Económico publicado por el Instituto Nacional de Estadística y Geografía (INEGI), del total de las empresas de México, el 95.2% son microempresas, a su vez generan el 45.6% del empleo y contribuyen con 15% del valor agregado de la economía

[2], por ello es fundamental que cuenten con un sistema informático de ventas adecuado para su correcta administración.

Después de analizar qué tipo de clientes pueden solicitar un sistema informático, es importante explicar en qué consiste, se puede decir que es un lugar virtual donde un cliente ejecuta el pago de bienes o servicios y donde los impuestos sobre las ventas pueden llegar a ser pagaderos, se realiza principalmente utilizando una computadora o dispositivo electrónico móvil [3], además pueden brindar a los minoristas más oportunidades de micro mercado de categorías de productos específicas e influir en los consumidores para hacer más atractivos los productos que ofrecen.

Los sistemas de software de ventas electrónicos agilizan las operaciones minoristas al automatizar el proceso de compra [4], realizar un seguimiento de los datos de ventas importantes, coordinar los datos recopilados de las compras diarias, generar recibos de forma electrónica, administración de inventario, informes y análisis de ventas, entre otros.

Por ejemplo, el concepto de tienda de conveniencia de Amazon, Amazon Go, que implementa tecnologías que permiten a los compradores entrar, tomar artículos y salir sin pasar por una caja registradora, podría revolucionar los sistemas de venta tal y como se conocen al aumentar la comodidad, esto podría permitir que los sistemas de venta, la lealtad y los pagos se conviertan en una única experiencia centrada en el cliente.

Algunas de las ventajas de digitalizar un negocio son:

- Inversión más baja: El interesado solo tiene que invertir en una página web o un sistema informático, además se puede apoyar utilizando las redes sociales y los gastos de tener un negocio físico disminuyen [5].
- Disponibilidad de horario: Se proporciona flexibilidad de compra y gestión para los clientes.
- Conocer a los clientes: Con herramientas de analítica y marketing digital, además de consultar las bases de datos que utiliza el sistema, es posible acceder a la información referente a las preferencias de los clientes, sus hábitos de compra principalmente, todo esto para impulsar mejores estrategias y experiencias.
- Interacción inmediata: Es posible dar atención a varios clientes al mismo tiempo, además se pueden implementar aplicaciones para atender dudas y quejas, además se le puede sacar más ventajas a las redes sociales, chat o email.

También es importante considerar las fases de ventas en sistemas informáticos, son las siguientes, siempre tomando en cuenta que el objetivo principal es vender más:

### **Fase 1: Atraer clientes**

El primer paso para alcanzar el éxito de un sistema de ventas radica en conocer quiénes son esas personas que pueden estar interesadas en ocuparlo, esos potenciales clientes que pueden

llegar a interesarse en los productos, ese público que está esperando ser encontrado. Conocerlo puede ayudar a plantear mejor la estrategia de atracción de visitas al sistema y alcanzar resultados más rápidos y eficaces en el negocio.

### **Fase 2: Convertir en ventas**

Uno de los secretos para vender por internet con éxito consiste en ofrecer los productos a los consumidores de la forma más atractiva y completa posible. Apenas un potencial cliente visite el sistema de ventas, es clave engancharlo visualmente para llamar su atención, para generarle interés.

### **Fase 3: Retener a los clientes**

Una vez que tu consumidor concretó la compra, comienza la etapa de posventa. En esta etapa hay que asegurarte de haber cumplido con lo prometido tanto a nivel producto como tiempos de entrega, es decir, es necesario conocer la experiencia que tuvo el usuario utilizando el sistema.

Desarrollar un sistema de venta ya sea para un pequeño negocio o dedicado a una gran empresa, tiene varias similitudes con un entorno de compra físico, además es fácil, seguro y automatizado.

## Objetivos

Desarrollar un sistema informático sobre la correcta administración y gestión de los procedimientos relacionados con las ventas para una pequeña tienda de conveniencia.

### **Objetivos específicos:**

- Desarrollar un algoritmo eficaz para la búsqueda por producto.
- Utilizar las herramientas proporcionadas por el sistema operativo para gestionar procesos.
- Diseñar una interfaz agradable, intuitiva e interactiva para los usuarios.
- Implementar el uso de semáforo, archivos y memoria compartida dentro del sistema de ventas para facilitar su accesibilidad.
- Sincronizar de manera eficiente y correcta la información mientras se interactúa con ella desde diferentes usuarios como clientes o proveedores.
- Implementar un sistema multiusuario, que permita múltiples interacciones en momentos simultáneos.



## Justificación

En el sistema por desarrollar se propone resolver la correcta administración y gestión de los procedimientos relacionados con las ventas para una pequeña tienda de conveniencia. Principalmente con la digitalización de una interfaz agradable e interactiva, el desarrollo de un algoritmo eficaz para la búsqueda por productos mediante procesos, el uso de semáforos, la implementación de memoria virtual, la utilización de archivos y memoria compartida dentro del sistema de ventas para facilitar su accesibilidad; además de utilizar las herramientas proporcionadas por el Sistema Operativo para gestionar los procesos y procesos hijos, como los son el uso de hilos, memoria compartida, colas de mensajes, semáforos, etc.. Por lo mismo, se implementarán algunas características de un Sistema Seguro.

Con lo anterior, podremos resolver las siguientes dificultades que se presentan en una Tienda de conveniencia:

- **Comunicación:** En gran parte de los casos, cuando se tiene una tienda (pequeña o mediana) los gerentes son los encargados de gestionar la información de la tienda (inventarios, ventas de productos y empleados) con ayuda de sus empleados que registran cada una de sus ventas, en ocasiones este responsable no se encuentra en su área de trabajo, así que la información llega a acumularse o perderse. Esto es un problema para el responsable de gestión, pues no tiene un canal dedicado para gestionar dicha información.
- **Organización:** En gran parte de los casos, en las pequeñas y medianas tiendas no se lleva un gran control de información acerca de los productos, personal de trabajo y proveedores. Esto es un problema para los responsables de gestionar dicha información, ya que no cuentan con un canal dedicado para consultar la información de toda la tienda.
- **Trazabilidad:** Una vez que los proveedores dejan su mercancía correspondiente con el encargado de administrarla y gestionarla, se pierde el seguimiento de cada uno de los productos puestos a disposición con el encargado de la tienda, ya que no cuentan con su propio canal para administrar la información de productos recibidos.
- **Seguridad:** Al no contar con un Sistema eficiente para administrar el inventario de entradas de productos, salidas de productos y personal de la tienda, esto puede llegar a ser un gran problema, ya que dichos productos pueden ser robados sin que nadie se dé cuenta (ya sea por los clientes o por el personal).
- **Recursos:** Para guardar todos los datos de nuestra tienda sin un Sistema que nos ayude a hacerlo sería muy complicado, muy cansado y nos llevaría mucho tiempo hacerlo. Además de no administrar correctamente nuestra información.
- **Tiempo:** Para guardar los datos de los clientes, proveedores y empleado se necesita de un gran tiempo para guardar y recuperar la información de toda la tienda sin un utilizar un sistema que lo haga por nosotros. El realizar el guardado de información

de la manera tradicional solo nos aumenta el tiempo que nos llevara realizar dicha tarea, por ende, una pérdida de dinero.

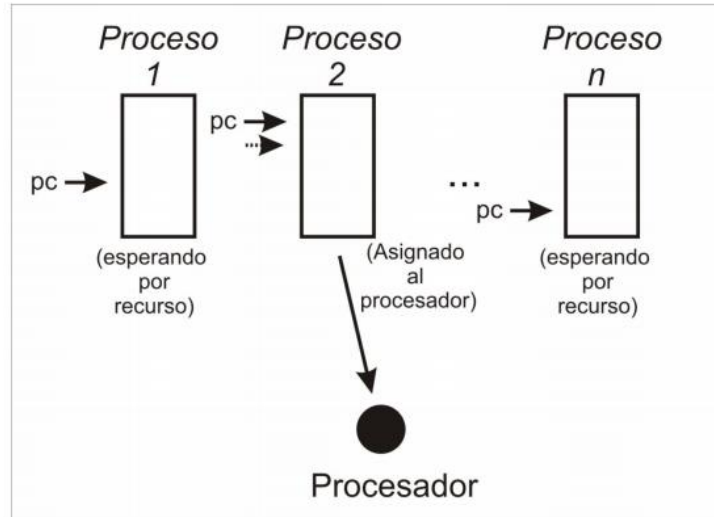
Todo lo anterior genera una pérdida de información importante que es difícil de recuperar, además de que genera un descontrol de información y de canales de comunicación.

Siendo conscientes de la situación que estamos viviendo (sociedad tecnológica), los sistemas deberán permitir establecer distintas reglas de negocios adaptadas a nuestro nuevo entorno de trabajo. Así podremos establecer un nuevo Sistema de trabajo que nos ayudara a administrar mejor la información de una Tienda de conveniencia.

## Marco teórico

### a) Procesos

Los procesos son programas en ejecución los cuales tienen cada uno un hilo (thread) de ejecución, los recursos del procesador son alternados entre los diversos procesos que existan en el sistema, de forma que se ejecutan en paralelo (multiprogramación). Los procesos tienen también un contador de programa el cual avanza cuando se le es asignado un recurso al procesador (Figura 1.). Los procesos también tienen un id para poder ser identificados [6].



*Figura 1. Contador de programa de los procesos.*

Los procesos se crean mediante otros procesos, a los procesos creadores se les llama procesos “padre” y a los nuevos procesos creados a partir de estos se les llama procesos “hijo”, teniendo de esta forma una jerarquía de procesos en el sistema. El sistema operativo es el que decide el momento en el que se crea un nuevo proceso, así como los recursos que utilizará y/o compartirá con el proceso padre. También se encarga de determinar lo que sucederá con los procesos hijo cuando finaliza el proceso padre [7].

Para crear un proceso se tiene la llamada al sistema *fork* la cual retorna al proceso padre el id del proceso hijo y a este último el valor de 0. Si no se ha podido crear al proceso hijo se retorna un -1 [8].

El código mostrado en la figura 2 es un ejemplo de cómo se puede crear un proceso usando la llamada al sistema *fork*, el programa muestra el mensaje “Este es el proceso padre” cuando *pid*=ID del proceso hijo, mientras que mostrará el mensaje “Este es el proceso hijo” cuando *pid*.

```

#include <unistd.h>
#include <stdio.h>

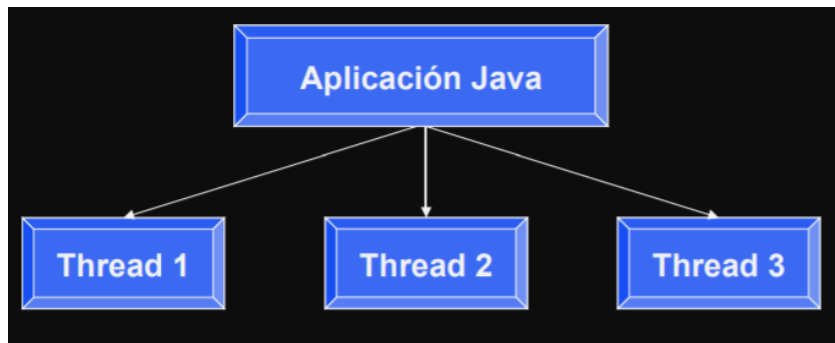
int main (){
    pid_t pid;
    pid = fork();
    if(pid == -1){
        printf("No se pudo crear el proceso hijo\n");
        return 0;
    }
    if(pid)
        printf("Este es el proceso padre\n");
    else
        printf("Este es el proceso hijo\n");
    return 0;
}

```

*Figura 2. Código para la creación de procesos usando la llamada al sistema fork [8].*

### **b) Hilos**

Un hilo (thread) es la menor de las estructuras lógicas de programación que funciona como una línea secuencial de ejecución por parte del planificador del sistema operativo [9]. Los hilos son más “ligeros” que los procesos, ya que muchos de los recursos que son necesarios para reservar y ejecutar un proceso, se comparten entre distintos hilos dentro de un mismo proceso. Un programa puede generar dos o más hilos de ejecución, los cuales siempre dependerán de la aplicación principal. Básicamente, las tareas controlan un único aspecto dentro de un programa y todas las tareas comparten los mismos recursos (Figura 3).



*Figura 3. Ejemplificación del funcionamiento de los hilos.*

Un hilo se define como “multitarea”, es cuando dos o más tareas se ejecutan a la “vez” dentro de un mismo programa [10]. Los hilos no nacieron para acelerar la ejecución de los programas, sino como forma de “simplificar” la forma en la que se diseñaban los programas más complejos, se puede observar un ejemplo en la Figura 4.

```

class TestHilos extends Thread{
    private String nombre;
    private int retardo;

    //constructor
    public TestHilos(String s, int d){
        nombre = s;
        retardo = d;
    }

    //funcion para la ejecucion del hilo
    public void run(){
        try{
            sleep(retardo);
        } catch (InterruptedException e){
            ;
        }
        System.out.println("hola mundo " + nombre + " " + retardo);
    }
}

public class hilos {

    public static void main(String[] args) {
        TestHilos t1,t2,t3;
        t1 = new TestHilos("hilo_1",(int)(Math.random()*2000));
        t2 = new TestHilos("hilo_2",(int)(Math.random()*2000));
        t3 = new TestHilos("hilo_3",(int)(Math.random()*2000));

        t1.start();
        t2.start();
        t3.start();
    }
}

```

*Figura 4. Código para la ejecución de hilos en Java*

### c) Semáforos

En la programación de computadoras, un semáforo es una técnica de señalización que utiliza variables especiales dentro de un lenguaje de programación de computadoras. Esta señal se usa para representar contadores y declaraciones de verdadero o falso. El uso de semáforos ha sido adoptado por la mayoría de los lenguajes de software. Es más frecuente dentro de la programación orientada a objetos, incluidos Java®, C # y Smalltalk [15].

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.

- Reglas para el uso de Semáforos:

Como podemos observar en la Figura 5, nos muestran las reglas principales de los semáforos.

- Reglas para semáforos generales:

$$\frac{(Q \wedge S > 0) \rightarrow R_s^{S-1}}{\{Q\} \text{ wait}(s) \{R\}}$$

$$\frac{Q \rightarrow R_s^{S+1}}{\{Q\} \text{ send}(s) \{R\}}$$

- Reglas para semáforos binarios:

$$\frac{(Q \wedge B > 0) \rightarrow R_B^{B-1}}{\{Q\} \text{ wait}(B) \{R\}}$$

$$\frac{(Q \wedge B < 1) \rightarrow R_B^{B+1}}{\{Q\} \text{ send}(B) \{R\}}$$

*Figura 5. Reglas de los Semáforos.*

Los semáforos se pueden usar para implementar protocolos de sección crítica [16].

- Operaciones con Semáforos:

Los semáforos sólo pueden ser manipulados usando las siguientes operaciones (éste es el código con espera activa Figura 6):

```
Inicia(Semáforo s, Entero v)
{
    s = v;
}
```

*Figura 6. Código para manipular semáforos*

En el que se iniciará la variable semáforo s a un valor entero v (Figura 7).

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

*Figura 7. Código para iniciar la variable de un semáforo.*

La cual mantendrá en espera activa al regido por el semáforo si este tiene un valor inferior o igual al nulo (Figura 8)

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

*Figura 8. Código de espera.*

- Utilidad y tipos de Semáforos:

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 decrementa 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 comienza a utilizar el recurso.

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. Este 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 [17].

En el siguiente ejemplo se crean y ejecutan n procesos que intentarán entrar en su sección crítica cada vez que puedan, y lo lograrán siempre de a uno por vez, gracias al uso del semáforo s inicializado en 1. El mismo tiene la misma función que un lock.

```
const int n    /* número de procesos */
variable semáforo s; /* declaración de la variable semáforo de valor entero*/
Inicia (s,1)  /* Inicializa un semáforo de nombre s con valor 1 */

void P (int i) {
    while (cierto) {
        P(s)    /* En semáforos binarios, lo correcto es poner un P(s) antes de entrar en
                 la sección crítica, para restringir el uso de esta región del código */
        /* SECCIÓN CRÍTICA */
        V(s)    /* Tras la sección crítica, volvemos a poner el semáforo a 1 para que otro
                 proceso pueda usarla */
        /* RESTO DEL CÓDIGO */
    }
}

int main() {
    Comenzar-procesos(P(1), P(2), ..., P(n));
}
```

*Figura 9. Código para la ejecución de Semáforos.*

#### **d) Colas de mensaje**

Una cola de mensajes es una forma de comunicación asíncrona de servicio a servicio que se usa en arquitecturas de microservicios y sin servidor. Los mensajes se almacenan en la cola hasta que se procesan y eliminan. Cada mensaje se procesa una vez sola, por un solo consumidor. Las colas de mensajes se pueden usar para desacoplar procesos pesados, para acumular trabajo y para clasificar cargas de trabajo [18].

Las colas de mensajes permiten a diferentes partes de un sistema comunicarse y procesar las operaciones de forma asíncrona. Una cola de mensajes ofrece un búfer ligero que almacena temporalmente los mensajes, y puntos de enlace que permiten a los componentes de software conectarse a la cola para enviar y recibir mensajes. Los mensajes suelen ser pequeños y pueden ser cosas como solicitudes, respuestas, mensajes de error o, sencillamente, información. Para enviar un mensaje, un componente llamado productor añade un mensaje a la cola. El mensaje se almacena en la cola hasta que otro componente, llamado consumidor, lo recupera y hace algo con él.



*Figura 10. Reglas de los Semáforos [17].*

Cuando se desarrolla una aplicación y ésta comienza a crecer, a menudo necesitamos interconectar distintos componentes. En estos casos se utiliza un middleware que nos permita comunicar las distintas piezas. Una opción es usar una cola de mensajes. No es el mecanismo más rápido, pero probablemente sí el más sencillo, y permite realizar acciones de forma asíncrona [18].

- Ejemplos de Colas de Mensajes:

Como podemos ver en la Figura 11 se encuentra una realización mensajes de colas en Python y en la Figura 12 se encuentra una realización de mensajes de colas de mensajes en Java [19].

```
1 import pika
2
3 connection = pika.BlockingConnection()
4 channel = connection.channel()
5 method_frame, header_frame, body = channel.basic_get('test')
6 if method_frame:
7     print method_frame, header_frame, body
8     channel.basic_ack(method_frame.delivery_tag)
9 else:
10    print 'No message returned'
11 connection.close()
```

*Figura 11. Colas de Mensajes en Python.*





*Figura 13. Funcionamiento de la memoria compartida*

La memoria compartida funciona de forma similar a las colas de mensajes durante su creación.

Un procesador comienza la ejecución de un programa. De forma clara, se encarga de abrir diferentes hilos de ejecución en diferentes procesadores. Requiere: zonas de memoria de acceso compartido y sincronización [12].

Para llevar a cabo una programación usando técnicas de memoria compartida será preciso tener en cuenta que:

- Cualquier procesador debe poder acceder a cualquier módulo de memoria.
- El acceso a un módulo de memoria por parte de un procesador no debe de privar a otro procesador del acceso a memoria.
- Hay que tratar de minimizar los tiempos de acceso a memoria para penalizar lo menos posible el rendimiento global de sistema [13].
- Mantener la coherencia de la memoria es imprescindible y la sincronización supone el mayor reto con el que será preciso lidiar.

El único problema al momento de usar memoria compartida es asegurar que no haya accesos concurrentes a ella, de forma que, si un proceso está estableciendo valores en la zona de memoria compartida, se debería esperar a que termine para comenzar la operación de lectura. Esto se puede hacer mediante semáforos o con el uso de bloque de registros [14].

## f) Archivos

No es más que la forma de abstraer información, proporcionando una manera de almacenarla en disco para su uso posterior evitando al usuario meterse en problemas de cómo y dónde está almacenada su información.

Un proceso creador de un nuevo archivo permite a este último asignarle un nombre, para posteriormente poder seguir siendo usado por otros procesos identificándolo por su nombre. En cuanto a la forma permitida de nombrar un archivo dependerá en gran medida del sistema en el que se le asigne, pero la mayoría permite cadenas de letras dígitos y caracteres especiales. En muchos sistemas se admiten nombres de archivos separados por un punto, donde el sufijo después del punto indicará la extensión del archivo y de su naturaleza:

| Extensión    | Significado   |
|--------------|---|
| archivo.bak  | Archivo de respaldo   |
| archivo.c    | Programa fuente en C  |
| archivo.gif  | Imagen en Formato de Intercambio de Gráficos de CompuServe              |
| archivo.hlp  | Archivo de ayuda  |
| archivo.html | Documento en el Lenguaje de Marcación de Hipertexto de World Wide Web   |
| archivo.jpg  | Imagen fija codificada con el estándar JPEG                             |
| archivo.mp3  | Música codificada en formato de audio MPEG capa 3                       |
| archivo.mpg  | Película codificada con el estándar MPEG                                |
| archivo.o    | Archivo objeto (producido por el compilador, no se ha enlazado todavía) |
| archivo.pdf  | Archivo en Formato de Documento Portable                                |
| archivo.ps   | Archivo de PostScript   |
| archivo.tex  | Entrada para el programa formateador TEX                                |
| archivo.txt  | Archivo de texto general  |
| archivo.zip  | Archivo comprimido  |

*Tabla 1. Extensiones comunes de archivos.*

En UNIX no es más que un tipo de convención. Este tipo de convención es útil cuando un programa puede manejar diferentes tipos de archivos. Sin embargo, en Windows adquiere significado como especificar para que programa posee una extensión.

### ○ Estructura:

#### 1. Secuencia de bytes:

El sistema no tiene conocimiento del contenido del archivo, solo ve bytes. De esta manera provee flexibilidad.

#### 2. Secuencia de registros:

Es una secuencia de registros de longitud fija, la lectura devuelve un registro y la escritura sobrescribe o agrega un registro.

3. **Árbol de registros:**

No todos son de la misma longitud, cada archivo del árbol contiene una llave en una posición fija del registro para una búsqueda rápida.

○ **Tipos:**

1. Regulares: los que contienen información del usuario, generalmente ASCII o binarios.
2. Directorios: sistemas de archivos para mantener la estructura.
3. Archivos especiales de caracteres: modela dispositivos de E/S: terminales, impresoras, redes.
4. Archivos especiales de bloques: para modelar discos.

○ **Acceso:**

1. Acceso secuencial:

Un proceso lee bytes/registros en un archivo de forma ordenada, desde el comienzo, no puede saltar líneas, pero puede rebobinar para leer las veces necesarias.

2. Acceso aleatorio:

Es posible leer fuera de orden, accede a registros por llave y no por posición.

○ **Atributos:**

| Atributo                       | Significado   |
|--------------------------------|---|
| Protección                     | Quién puede acceso al archivo y en qué forma                |
| Contraseña                     | Contraseña necesaria para acceder al archivo                |
| Creador                        | ID de la persona que creó el archivo                        |
| Propietario                    | El propietario actual                                       |
| Bandera de sólo lectura        | 0 para lectura/escritura; 1 para sólo lectura               |
| Bandera oculto                 | 0 para normal; 1 para que no aparezca en los listados       |
| Bandera del sistema            | 0 para archivos normales; 1 para archivo del sistema        |
| Bandera de archivo             | 0 si ha sido respaldado; 1 si necesita respaldarse          |
| Bandera ASCII/binario          | 0 para archivo ASCII; 1 para archivo binario                |
| Bandera de acceso aleatorio    | 0 para sólo acceso secuencial; 1 para acceso aleatorio      |
| Bandera temporal               | 0 para normal; 1 para eliminar archivo al salir del proceso |
| Banderas de bloqueo            | 0 para desbloqueado; distinto de cero para bloqueado        |
| Longitud de registro           | Número de bytes en un registro                              |
| Posición de la llave           | Desplazamiento de la llave dentro de cada registro          |
| Longitud de la llave           | Número de bytes en el campo llave                           |
| Hora de creación               | Fecha y hora en que se creó el archivo                      |
| Hora del último acceso         | Fecha y hora en que se accedió al archivo por última vez    |
| Hora de la última modificación | Fecha y hora en que se modificó por última vez el archivo   |
| Tamaño actual                  | Número de bytes en el archivo                               |
| Tamaño máximo                  | Número de bytes hasta donde puede crecer el archivo         |

*Tabla 2. Atributos de archivos.*

○ **Operaciones:**

1. Create. Se crea sin datos.
2. Delete. Libera el espacio en disco.
3. Open. Lo lleva de disco a memoria principal.
4. Close. Cerrar para liberar espacio en memoria.
5. Read. Lee el archivo. Debe especificar cuantos datos se necesitan y debe proporcionar un buffer para colocarlos.
6. Write. Se escriben datos.
7. Append. Forma restringida de write. Solo agrega datos al final del archivo.
8. Seek. En aleatorios se especifica de donde se tomarán los datos.
9. Get attributes.
10. Set attributes.
11. Rename. Nombra un archivo existente.

**g) Dispositivos de entrada y salida**

Un sistema operativo debe tener la capacidad de controlar todos los dispositivos de E/S. Emitir comandos, captar interrupciones y manejar errores, además de proporcionar una interfaz para estos y el resto del sistema.

- **Dispositivos de E/S:**

- 1. De bloque:

- Almacena información de bloques de tamaño fijo y propia dirección. En estos se tiene la posibilidad de leer y escribir de forma independiente: discos duros, CD-ROMs, y memorias USBs son ejemplos de estos.

- 2. De carácter:

- Envía o acepta flujos de caracteres, sin importar la estructura del bloque: impresoras, interfaces de red y ratones son ejemplos de estos.

| Dispositivo             | Velocidad de transferencia de datos |
|-------------------------|-------------------------------------|
| Teclado                 | 10 bytes/seg                        |
| Ratón                   | 100 bytes/seg                       |
| Módem de 56K            | 7 KB/seg                            |
| Escáner                 | 400 KB/seg                          |
| Cámara de video digital | 3.5 MB/seg                          |
| 802.11g inalámbrico     | 6.75 MB/seg                         |
| CD-ROM de 52X           | 7.8 MB/seg                          |
| Fast Ethernet           | 12.5 MB/seg                         |
| Tarjeta Compact Flash   | 40 MB/seg                           |
| FireWire (IEEE 1394)    | 50 MB/seg                           |
| USB 2.0                 | 60 MB/seg                           |
| Red SONET OC-12         | 78 MB/seg                           |
| Disco SCSI Ultra 2      | 80 MB/seg                           |
| Gigabit Ethernet        | 125 MB/seg                          |
| Unidad de disco SATA    | 300 MB/seg                          |
| Cinta de Ultrium        | 320 MB/seg                          |
| Bus PCI                 | 528 MB/seg                          |

*Tabla 3. Velocidades de transferencia de datos en dispositivos.*

- **Controladores:**

La interfaz entre el controlador y el dispositivo es a menudo de muy bajo nivel. El trabajo del controlador es convertir el flujo de bits serial en un bloque de bytes y realizar cualquier corrección de errores necesaria.

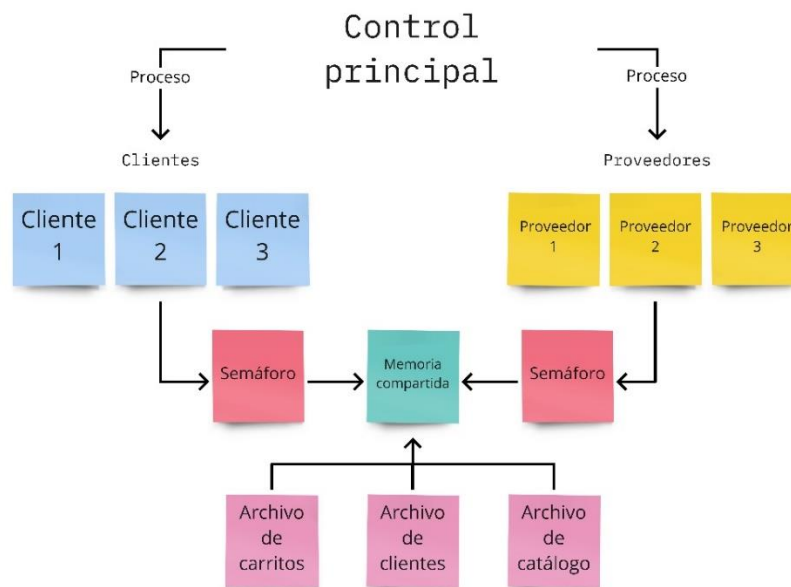
- **E/S por asignación de memoria:**

Cada controlador tiene unos cuantos registros que se utilizan para comunicarse con la CPU. Al escribir en ellos, el sistema operativo puede hacer que el dispositivo envíe o acepte datos, se encienda o se apague, o realice cualquier otra acción. Al leer de estos registros, el sistema operativo puede conocer el estado del dispositivo, si está preparado o no para aceptar un nuevo comando, y sigue procediendo de esa manera. Además de los registros de control, muchos dispositivos tienen un búfer de datos que el sistema operativo puede leer y escribir. Por ejemplo, una manera común para que las computadoras muestren píxeles en la pantalla es tener una RAM de video, la cual es básicamente sólo un búfer de datos disponible para que los programas o el sistema operativo escriban en él.

Existen dos alternativas para la comunicación entre la CPU y los registros de control.

1. A cada registro de control se le asigna un número de puerto de E/S, un entero de 8 o 16 bits. El conjunto de todos los puertos de E/S forma el espacio de puertos de E/S y está protegido de manera que los programas de usuario ordinarios no puedan utilizarlo (sólo el sistema operativo puede).
2. Asignar todos los registros de control al espacio de memoria. A cada registro de control se le asigna una dirección de memoria única a la cual no hay memoria asignada. Este sistema se conoce como E/S con asignación de memoria (mapped-memory). Por lo general, las direcciones asignadas se encuentran en la parte superior del espacio de direcciones.

## Análisis



*Figura 14. Estructura general del sistema.*

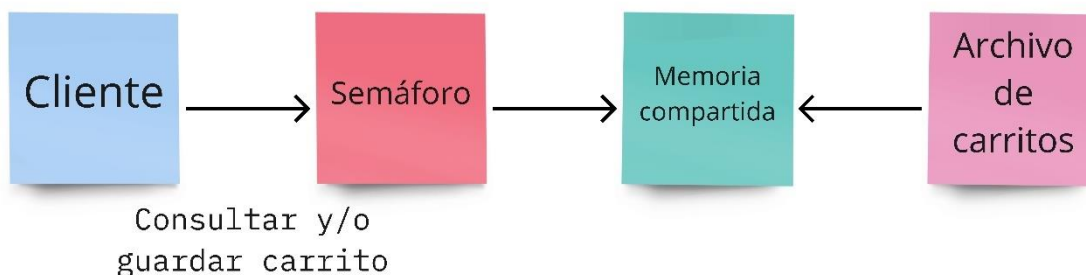
La figura 14 muestra el cómo será la estructura general del sistema a desarrollar, de forma general se muestra cómo interactuarán los diferentes módulos del sistema entre sí. Estos módulos son: Control principal, aquél que va a gestionar los procesos que atienden a los clientes y proveedores, además de inicializar los semáforos; Proveedores, aquellos que surtirán los productos; Clientes, los usuarios del sistema que podrán adquirir los productos; Archivos, donde se guardarán los datos de los usuarios que se han registrado en el sistema, los productos disponibles en el catálogo y los carritos de compras de los clientes; semáforos, los que nos permitirán sincronizar el acceso a la memoria compartida; la memoria compartida, el método por el cual podremos leer y/o escribir en los archivos.



*Figura 15. Interacción del control con el cliente.*

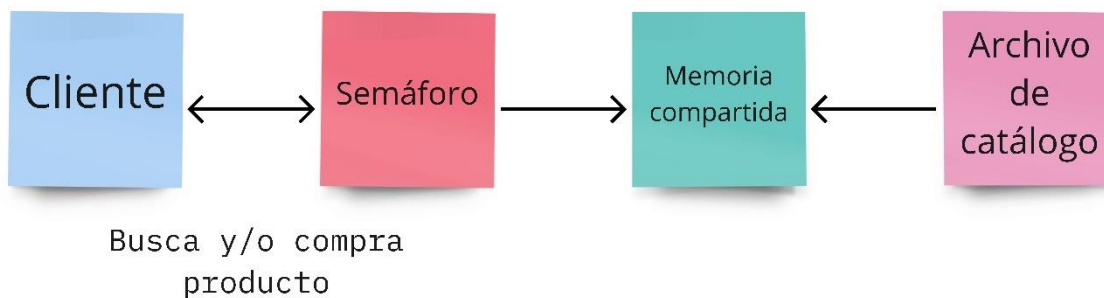


La figura 15 muestra la principal interacción que tendrá el módulo de Control principal con los clientes, mediante un proceso hijo, el proceso Control atenderá al cliente, este último puede registrarse en el sistema y/o iniciar sesión.



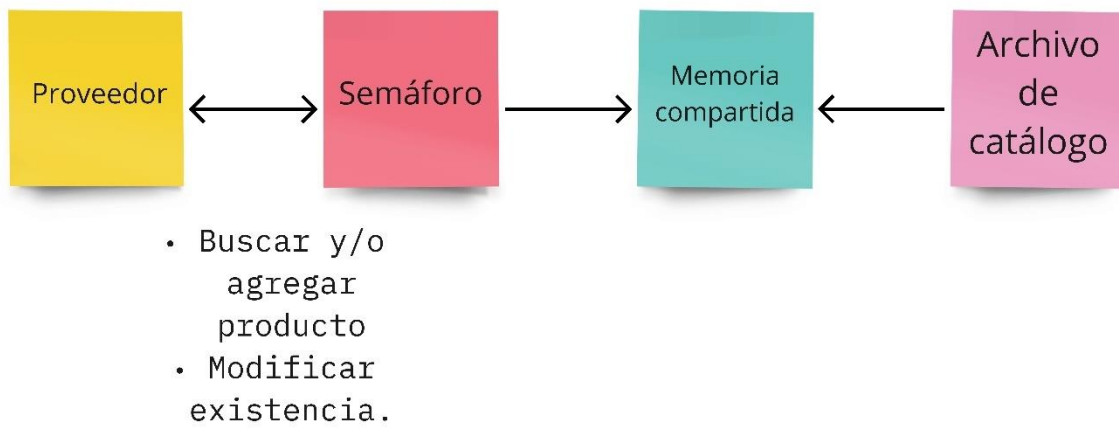
*Figura 16. Interacción del cliente con el carrito.*

La figura 16 muestra la interacción que tendrán los clientes con sus respectivos carritos de compra, el cliente podrá consultar y/o modificar su carrito de compras en el archivo de carritos, se utilizará un semáforo para sincronizar el acceso a la memoria y con ello evitar que ocurran fallos al momento de modificar el archivo de carritos.



*Figura 17. Interacción del cliente con el catálogo.*

La figura 17, similar a la figura anterior, muestra la interacción del cliente con el catálogo de productos, el cliente podrá ver todos los productos y agregar los que desee al carrito de compras, de igual forma se utilizará un semáforo para que el acceso a los datos del catálogo sea sincronizado y se eviten problemas.



*Figura 18. Interacción del proveedor con el catálogo.*

La figura 18 muestra la interacción del proveedor con el catálogo de productos, con ello se puede buscar y/o agregar productos a los catálogos y modificar la existencia de estos, igualmente utilizará un semáforo para sincronizar el acceso al catálogo.

## Diseño

Para los clientes, se creó una estructura de nombre Cliente, entre sus campos se encuentran dos enteros que sirven para guardar el id del cliente y su edad, además tiene 5 arreglos de caracteres para almacenar su correo, su contraseña, nombre(s), apellido paterno y apellido materno. Para la implementación se crearon dos arreglos globales de clientes llamados Aux y Reset con tamaño de 64 además de una variable global tipo Cliente donde se almacenarán los datos del usuario que inicie sesión y otra de tipo entero donde se guarda la cantidad de clientes registrados en el sistema. El arreglo Aux servirá para guardar los datos de los clientes leídos desde el archivo de clientes mientras que el arreglo Reset funciona como auxiliar en caso de que se quiera borrar a todos los clientes del archivo para hacer más pruebas, la figura 19 muestra el código realizado para este propósito.

```
typedef struct NodoCliente{
    int id_cliente;
    int edad;
    char correo[64];
    char password[64];
    char nombre_s[64];
    char apellido_paterno[64];
    char apellido_materno[64];
} Cliente;

Cliente Aux[64];
Cliente Reset[64];
Cliente Cliente_Actual;

int NumClientes = 0;
```

*Figura 19. Estructura de Cliente.*

Para el catálogo de productos se creó una estructura similar a la de cliente, entre sus campos se encuentran 3 enteros que almacenan el id del producto, el precio y stock disponible de este, al igual que con los clientes, cuenta con dos arreglos globales de tipo producto de tamaño 64, Aux\_Productos y Reset\_Productos, el primero cumple una función similar al Aux anterior ya que este almacenará los productos que se leen desde el archivo de catálogo y el

segundo servirá para reiniciar los datos de los productos, además tendrá una variable entera global para almacenar el número total de productos que se han registrado, la figura 20 muestra el código realizado para esta parte.

```
typedef struct NodoProducto{
    int id_producto;
    char producto_nombre[64];
    int precio;
    int stock;
} Producto;

Producto Aux_Productos[64];
Producto Reset_Productos[64];

int NumProductos = 0;
```

*Figura 20. Estructura de Producto.*

Para los carritos de compras de los clientes, también se creó una estructura, entre sus campos se encuentran 2 enteros donde se almacenan el id del cliente al que pertenece el carrito y el número de productos en el carrito, también cuenta con un arreglo de enteros donde se almacenarán los id de los productos que el cliente a agregado al carrito (figura 21). También se contará con 2 arreglos de tipo Carrito: Aux\_Carrito y Reset\_Carrito de tamaño 64 que cumplirán funciones similares a los anteriores sobre almacenar los carritos leídos desde el archivo de carritos y reiniciar los carritos pero, además existe un tercer arreglo de nombre Set\_Carrito, esto debido a que la creación de los carritos está ligada al registro de los clientes, por tanto si se reinician los carritos sin reiniciar los clientes ocurrirán errores en el sistema, y es por eso que Set\_Carrito servirá para reiniciar los carritos en función a los clientes registrados.

```
typedef struct NodoCarrito{
    int id_cliente;
    int Num_Productos_Carrito;
    int id_productos[64];
} Carrito;

Carrito Aux_Carrito[64];
Carrito Reset_Carrito[64];
Carrito Set_Carrito[64];
```

*Figura 21. Estructura de Carrito.*

Los semáforos se manipularán mediante las funciones descritas en Semaforo.h y que se muestran en la figura 22, la función createsem recibirá como primer parámetro un entero el cual es la llave para generar el semáforo y como segundo parámetro otro entero que será para inicializar el semáforo, esta función regresará el id del semáforo generado. La función ereasesem recibe como parámetro el id del semáforo que se desea eliminar, las funciones semwait y semsignal reciben como parámetro el id del semáforo sobre el cual se quiere operar, estas funciones son las que se encargan de hacer la sincronización del acceso a la región crítica y por tanto se utilizarán cada vez que se requiera leer o escribir en los archivos de clientes, carritos o catálogo.

```

#include<sys/sem.h>
#include<sys/types.h>
#include<sys/ipc.h>

#define SEM_ID int

void semwait(int semid){
    struct sembuf s;
    s.sem_num=0;
    s.sem_op=-1;
    s.sem_flg=SEM_UNDO;

    semop(semid,&s,1);
    return;
}

void semsignal(int semid){
    struct sembuf s;
    s.sem_num=0;
    s.sem_op=1;
    s.sem_flg=SEM_UNDO;

    semop(semid,&s,1);
    return;
}

int createsem(int key, int value){
    int semid;
    semid = semget(key,1,0666|IPC_CREAT);
    semctl(semid,0,SETVAL,value);
    return(semid);
}

void ereasesem(int semid){
    semctl(semid,0,IPC_RMID,0);
    return;
}

```

*Figura 22. Funciones de semáforos.*

Para la manipulación de archivos se utilizarán las funciones dentro de Archivos.h, para la explicación en esta sección se utilizarán como ejemplo las funciones que manipulan el archivo de carritos que se muestran en la figura 23. Estas funciones son: escribir\_carritos, leer\_carritos, reset\_archivo\_carritos, reset\_carritos y dos funciones únicas para el archivo de

carritos las cuales son `set_archivo_carritos` y `set_carritos`. La primera función se encarga de abrir el archivo con el nombre “`Archivo_Carritos`” en modo de escritura binaria, posteriormente comprueba si el archivo realmente existe y si no lo crea y guarda en él los datos del arreglo de carritos `Aux_Carrito`, finalmente regresa un 1 si ha tenido éxito la función. La siguiente función realiza el mismo procedimiento que la anterior con la diferencia de que esta vez abre el archivo en modo lectura binaria y copia los datos del arreglo guardado en el archivo al arreglo `Aux_Carrito`. La tercera función hace lo mismo que la primera con la diferencia de que el arreglo que guardará en el archivo es `Reset_Carritos` el cual siempre está vacío y la función 4 se encarga de invocar esta función para después invocar la función `leer_carritos` y de esa forma reiniciar el arreglo `Aux_Carrito`. Estas 4 funciones tienen implementaciones similares para los archivos de clientes y de catálogo, variando en el nombre de estas, el nombre de los arreglos y el nombre de los archivos. Como ya se comentó anteriormente, para el caso del archivo de carritos si no se reinician también los clientes se debe utilizar el arreglo `Set_Carritos` para reiniciar solo los carritos, las últimas dos funciones realizan un proceso similar a las funciones 3 y 4 con la diferencia de que utilizan el arreglo `Set_Carritos` en lugar de `Reset_Carritos`.

```

//Archivo de carritos

int escribir_carritos(){
    FILE *archivo;
    archivo = fopen("Archivo_Carritos","wb");
    if(archivo){
        fwrite(Aux_Carrito,sizeof(struct NodoCarrito),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int leer_carritos(){
    FILE *archivo;
    archivo = fopen("Archivo_Carritos","rb");
    if(archivo){
        fread(Aux_Carrito,sizeof(struct NodoCarrito),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int reset_archivo_carritos(){
    FILE *archivo;
    archivo = fopen("Archivo_Carritos","wb");
    if(archivo){
        fwrite(Reset_Carrito,sizeof(struct NodoCarrito),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int reset_carritos(){
    reset_archivo_carritos();
    leer_carritos();
    return 1;
}

int set_archivo_carritos(){
    for(int i=0 ; i<get_NumClientes() ; i++)
        Set_Carrito[i].id_cliente = i;
    FILE *archivo;
    archivo = fopen("Archivo_Carritos","wb");
    if(archivo){
        fwrite(Set_Carrito,sizeof(struct NodoCarrito),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int set_carritos(){
    set_archivo_carritos();
    leer_carritos();
    return 1;
}
//*****

```

*Figura 23. Funciones para modificar archivo de carritos.*



## Implementación y pruebas

Para esta implementación, se hizo que el proceso Control sea el encargado de crear e iniciar los semáforos necesarios para la sincronización del acceso a la memoria compartida y los archivos, además despliega al usuario un menú con 3 opciones: 1) Clientes, 2) Proveedores y 3) Salir, una vez que el usuario selecciona cualesquiera de las dos opciones, con ayuda de las funciones atender\_cliente y atender\_proveedor, además de mostrar cuantos clientes o proveedores se han atendido, el proceso Control utilizará la llamada al sistema fork() para crear un proceso hijo que atienda ya sea al cliente o proveedor, para atenderlos se hará con la llamada al sistema execlp la cual abrirá otra terminal que ejecutará ya sea el programa Cliente y/o Proveedor, de igual manera le enviará como argumentos las llaves necesarias para los semáforos.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include"Semaforos.h"
#include"Control.h"

int main(){
    id_sem_cliente = createsem(0x1234,1);
    id_sem_proveedor = createsem(0x0234,1);
    id_sem_carrito = createsem(0x0034,1);
    id_sem_catalogo = createsem(0x0004,1);
    int menu;
    do{
        menu = imp_menu_control();
        seleccion(menu);
    }while(menu!=3);

    return 0;
}
```

*Figura 24. Código del programa Control.*

La figura 24 muestra como el programa control crea e inicia los semáforos, posteriormente despliega el menú con las opciones antes descritas.

```

int atender_cliente(){
    NumClientesAtendidos++;
    printf("Clientes atendidos: %d\n",NumClientesAtendidos);
    if(fork() == 0){
        sprintf(string_sem_cliente,"%d",id_sem_cliente);
        sprintf(string_sem_carrito,"%d",id_sem_carrito);
        sprintf(string_sem_catalogo,"%d",id_sem_catalogo);
        int err = execlp("gnome-terminal","gnome-terminal","--",
            "./Cliente",string_sem_cliente,string_sem_carrito,
            string_sem_catalogo,(char*)NULL);
    }
}

int atender_proveedor(){
    NumProveedoresAtendidos++;
    printf("Proveedores atendidos: %d\n",NumProveedoresAtendidos);
    if(fork() == 0){
        sprintf(string_sem_proveedor,"%d",id_sem_proveedor);
        int err = execlp("gnome-terminal","gnome-terminal","--",
            "./Proveedor",string_sem_proveedor,(char*)NULL);
    }
}

```

*Figura 25. Funciones atender\_cliente y atender\_proveedor.*

La figura 25 muestra las funciones `atender_cliente` y `atender_proveedor` dentro del archivo `Control.h` y las cuales son invocadas por el proceso `Control`, en ellas se utilizan las llamadas al sistema `fork` para crear un proceso hijo donde con la llamada al sistema `execlp` se ejecutará el programa de `Cliente` o `Proveedor`, entre los argumentos de la función `execlp` también se encuentran las llaves para los semáforos generadas por el proceso `Control`.

La figura 26 muestra el código principal del programa `Cliente`, lo primero que hace es leer los archivos de clientes, carritos y catalogo para guardarlos en los arreglos `Aux`, `Aux_Carrito` y `Aux_Producto` para poder trabajar con ellos, después obtiene el número de clientes ya registrados mediante la función `get_NumClientes` descrita en `Cliente.h`, posteriormente asigna a 3 variables enteras globales el valor de los id de los semáforos para la manipulación de los 3 archivos, lo siguiente es mostrarle al usuario un menú con 3 opciones: 1) Registrar cliente, 2) Iniciar sesión y 3) Salir. Si se selecciona la primera opción se le preguntará al usuario los datos del cliente que desea agregar, una vez que se registra al cliente regresa al

mismo menú donde puede registrar otro cliente o iniciar sesión de uno ya registrado, si se decide iniciar sesión se le mostrará al usuario le pedirá que ingrese su correo electrónico y su contraseña, si no coinciden mostrará un mensaje de error y volverá a pedirle los datos, si logra iniciar sesión mostrará las siguientes opciones: 1) Mostrar productos, 2) Agregar producto al carrito, 3) Mostrar productos en carrito y 4) Salir, la primera opción es necesaria para que el cliente conozca qué productos existen en el sistema, cuál es su id, el precio y la cantidad disponible para la compra. Al seleccionar la segunda opción le pedirá al cliente que ingrese el id del producto y la cantidad que desea comprar, la tercera opción mostrará los productos que el cliente ha agregado al carrito, para esta implementación se utilizó la misma función que muestra todos los datos del producto en la primera opción y por tanto se puede mejorar más adelante.

```
int sem_id_cliente;
int sem_id_carrito;
int sem_id_catalogo;

int main(int argc, char *argv[]){
    int menu = 0;
    leer_clientes();
    leer_carritos();
    leer_catalogo();
    NumClientes = get_NumClientes();
    //sem_id_cliente = createsem(0x1234,1);
    sem_id_cliente = atoi(argv[1]);
    sem_id_carrito = atoi(argv[2]);
    sem_id_catalogo = atoi(argv[3]);
    //printf("Sem id: %d\n",sem_id_cliente);
    do{
        menu = imp_menu_cliente();
        seleccion_cliente(menu);
    }while(menu!=3);

    return 0;
}
```

*Figura 26. Código principal del programa Cliente.*

En caso de que se haya iniciado el programa Proveedor, lo primero que hará será leer el catálogo de productos y obtener el número de productos existentes, además de asignar el valor del id del semáforo necesario para la manipulación del catálogo a una variable entera global, al no haber registro de proveedores el programa simplemente pedirá al usuario que

ingrese la contraseña de proveedores la cuál es: 741963, esto con el motivo de restringir el ingreso, de igual forma se puede mejorar más adelante. Una vez que se ingresa la clave, el programa mostrará al usuario las siguientes opciones: 1) Buscar producto por id, 2) Agregar producto, 3) Modificar producto y 4) Salir, si se selecciona la primera opción el programa pedirá al usuario ingresar el id del producto, si este existe le mostrará al usuario los datos del producto, si no, mostrará datos vacíos, si se decide agregar un producto, el programa pedirá al usuario los datos del producto, como nombre, precio, y stock disponible, si se decide modificar un producto, el programa mostrará otro menú con las opciones: 1) Modificar nombre, 2) Modificar precio, 3) Modificar Stock y 4) No modificar nada, al seleccionar cualquiera de las primeras 3 opciones se le pedirá al usuario ingresar el id del producto a modificar y el valor del campo que se desea modificar, una vez modificado el producto o en caso de seleccionar la 4 opción se regresará al usuario al menú anterior, la figura 27 muestra el código principal del programa proveedor.

```
int sem_id_proveedor;

int main(int argc, char *argv[]){
    int menu = 0;
    leer_catalogo();
    NumProductos = get_NumProductos();
    //sem_id_proveedor = createsem(0x0234,1);
    sem_id_proveedor = atoi(argv[1]);
    //printf("Sem id: %d\n",sem_id_proveedor);
    login_proveedor();
    do{
        menu = imp_menu_proveedor();
        seleccion_proveedores(menu);
    }while(menu!=4);

    return 0;
}
```

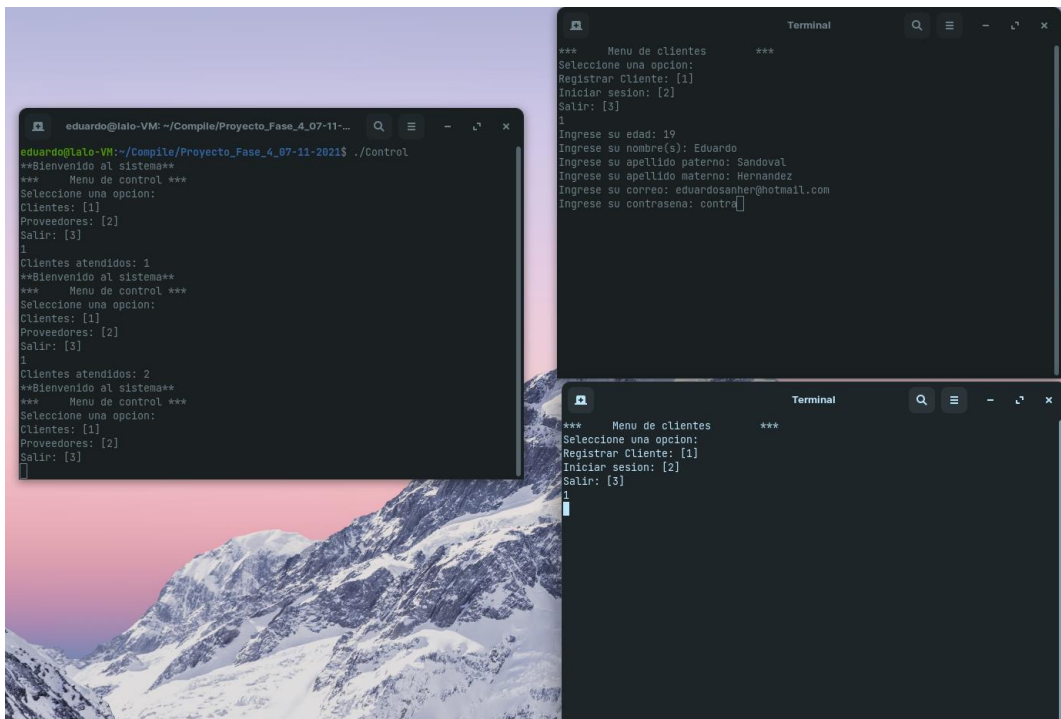
*Figura 27. Código principal del programa Proveedor.*

Cabe aclarar que, debido al uso de semáforos, solo se puede registrar un cliente a la vez, es decir, cuando dos clientes se quieren registrar, uno tendrá que esperar a que el otro termine, lo mismo al modificar los carritos y los productos del catálogo.

Para compilar y ejecutar el sistema se hará de la siguiente manera:

- gcc Cliente.c -o Cliente
- gcc Proveedor.c -o Proveedor
- gcc Control.c -o Control
- ./Control

La figura 28 muestra el sistema ya en ejecución, para ello se atendieron a dos clientes y se intentó registrar un cliente nuevo en el sistema, como resultado se muestra que mientras el primer cliente está ingresando sus datos al sistema, el segundo está esperando a que termine el registro, esto a causa de los semáforos que le impiden modificar el archivo de clientes para evitar conflictos.



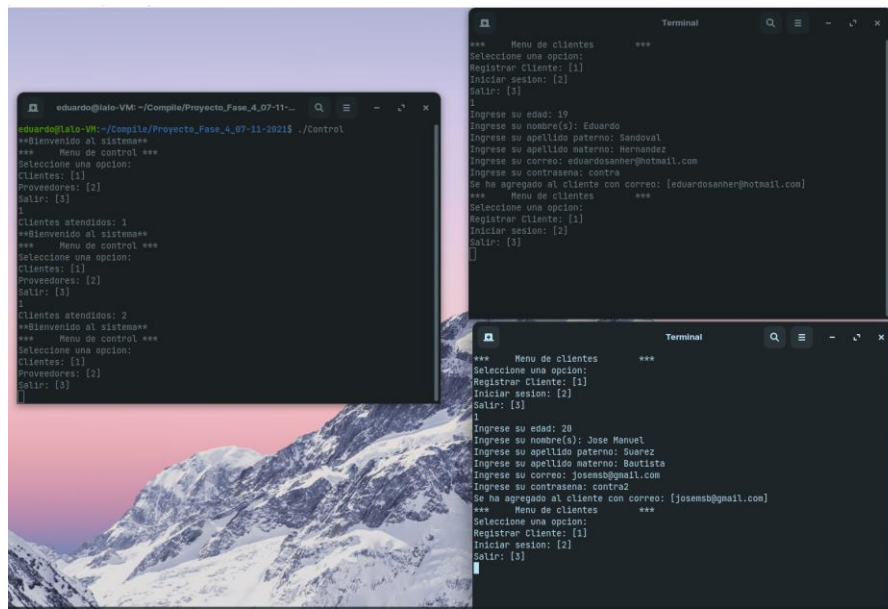
```

eduardo@lalo-VM: ~/Compile/Proyecto_Fase_4_07-11-2021$ ./Control
*** Menu de control ***
Seleccione una opcion:
Clientes: [1]
Proveedores: [2]
Salir: [3]
1
Clientes atendidos: 1
**Bienvenido al sistema**
*** Menu de control ***
Seleccione una opcion:
Clientes: [1]
Proveedores: [2]
Salir: [3]
1
Clientes atendidos: 2
**Bienvenido al sistema**
*** Menu de control ***
Seleccione una opcion:
Clientes: [1]
Proveedores: [2]
Salir: [3]
1
*** Menu de clientes ***
Seleccione una opcion:
Registrar Cliente: [1]
Iniciar sesion: [2]
Salir: [3]
1
Ingrese su edad: 19
Ingrese su nombre(s): Eduardo
Ingrese su apellido paterno: Sandoval
Ingrese su apellido materno: Hernandez
Ingrese su correo: eduardosanher@hotmail.com
Ingrese su contraseña: contra

```

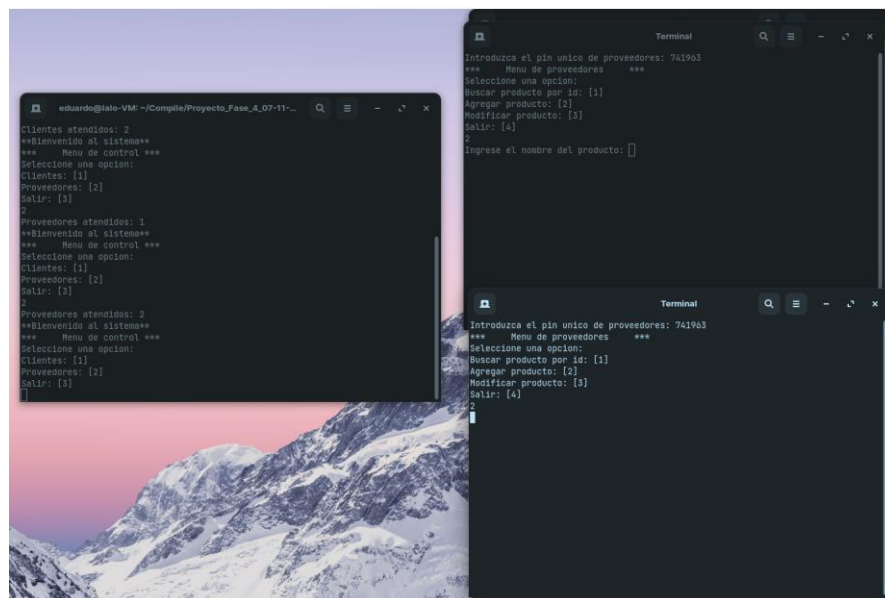
*Figura 28. Sistema atendiendo el registro de dos clientes.*

La figura 29 muestra que una vez que terminó de registrarse el primer usuario, el segundo puede terminar con su registro.



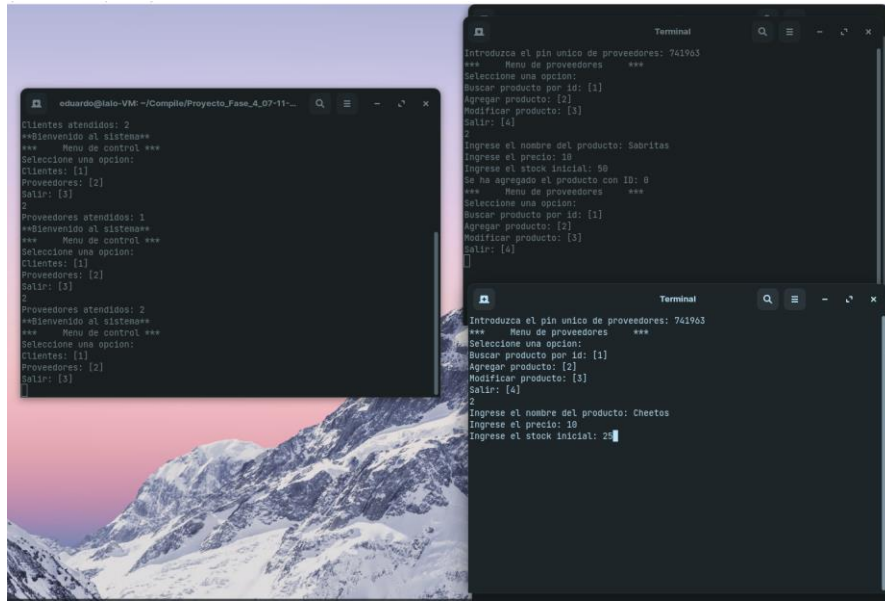
*Figura 29. Segundo cliente terminando su registro.*

Antes de continuar, se quieren agregar productos al catálogo, por tanto, desde el control se atiende a dos proveedores a los cuales se les pide la clave de acceso y una vez ingresada se les muestra el menú principal de proveedores, el primer proveedor decide registrar un producto por tanto el segundo debe esperar a que el primero termine (figura 30).



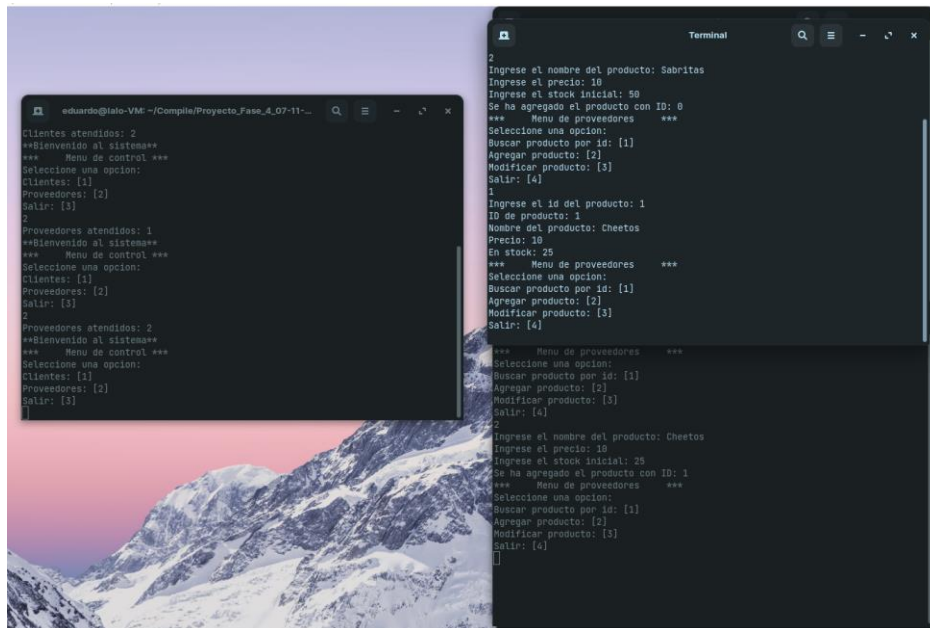
*Figura 30. Dos proveedores siendo atendidos.*

Una vez que el primer proveedor termina de registrar el producto, el segundo puede registrar el otro producto (figura 31).



*Figura 31. Segundo proveedor registrando su producto.*

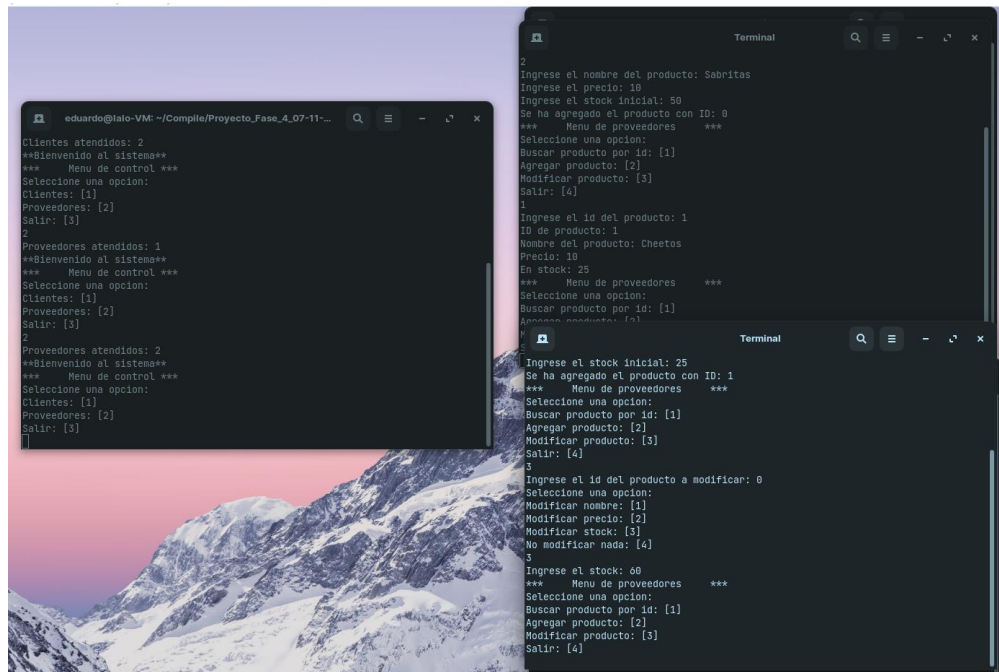
La figura 32 muestra que el primer proveedor puede consultar el producto que agregó el segundo mediante su id.



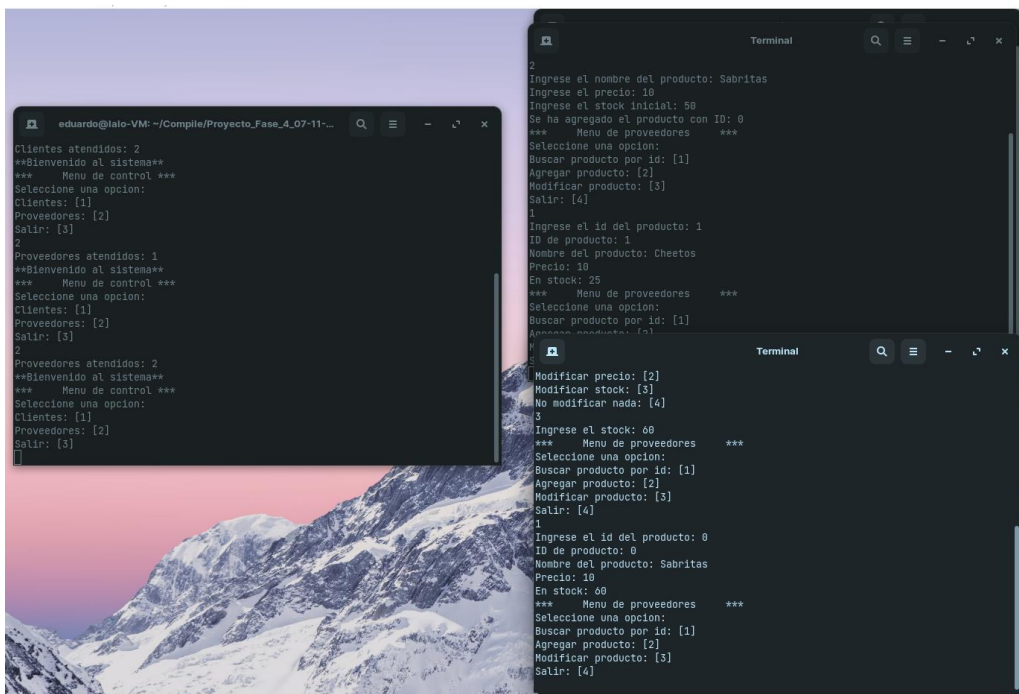
*Figura 32. Primer proveedor consultando el producto con id: 1.*

Las figuras 33 y 34 muestran que el segundo proveedor puede editar el stock disponible del producto con id 0.





*Figura 33. Segundo proveedor modificando el stock del producto con id: 0.*



*Figura 34. Segundo proveedor consultando el producto con id: 0.*

Los proveedores dejan de ser atendidos y ahora los clientes inician sesión, la figura 35 muestra que el primer cliente inicia sesión a la primera sin errores y el segunda falla dos

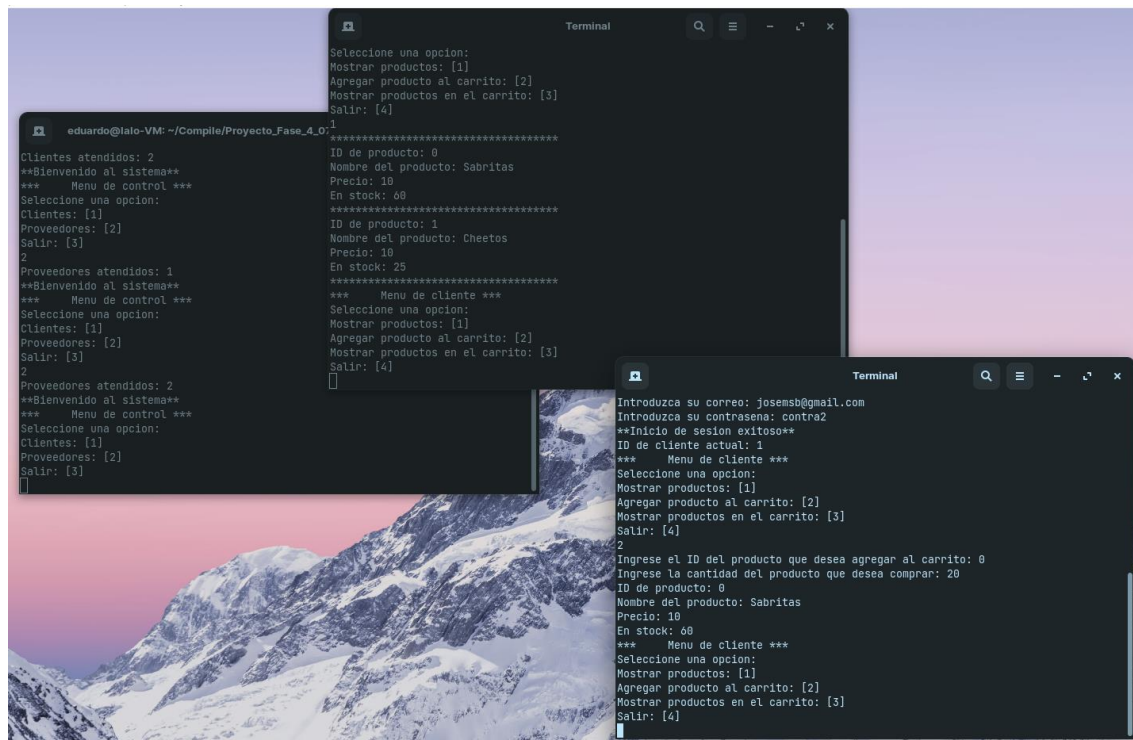


veces en iniciar sesión, cuando ambos logran iniciar sesión el programa les muestra el menú para los clientes en sesión.

[illegible]

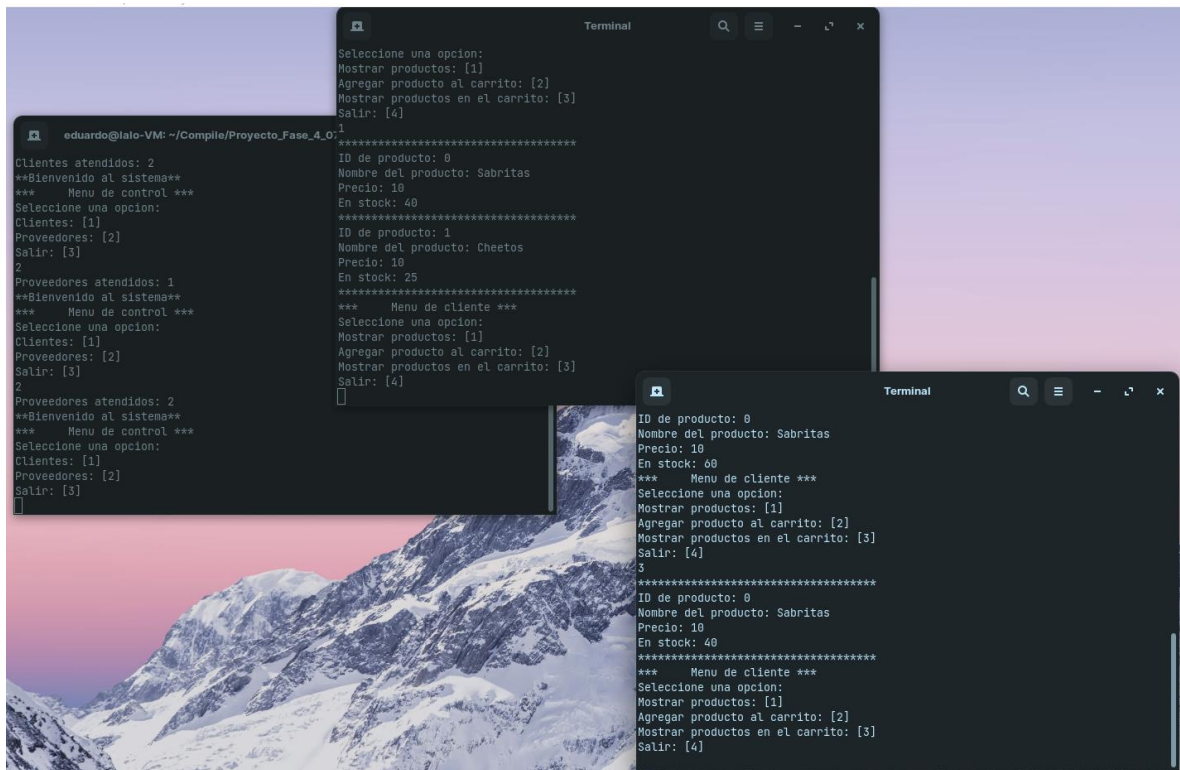
*Figura 35. Los dos clientes iniciando sesión.*

En la figura 36 se puede apreciar que el primer cliente consulta los productos disponibles en el catálogo mientras el segunda compra directamente el producto con id: 0 en una cantidad de 20.



*Figura 36. Dos clientes interactuando con el sistema.*

La figura 37 muestra que el primer cliente decide consultar de nuevo los productos y muestra que el stock disponible para el producto con id: 0 ha disminuido.

The image shows three overlapping terminal windows against a background of a snowy mountain range. The top-left window shows a menu with options for clients, providers, and products. The top-right window shows the details for a product named 'Sabritas'. The bottom-right window shows the details for a product named 'Cheetos'.

```
Terminal
Seleccione una opcion:
Mostrar productos: [1]
Agregar producto al carrito: [2]
Mostrar productos en el carrito: [3]
Salir: [4]

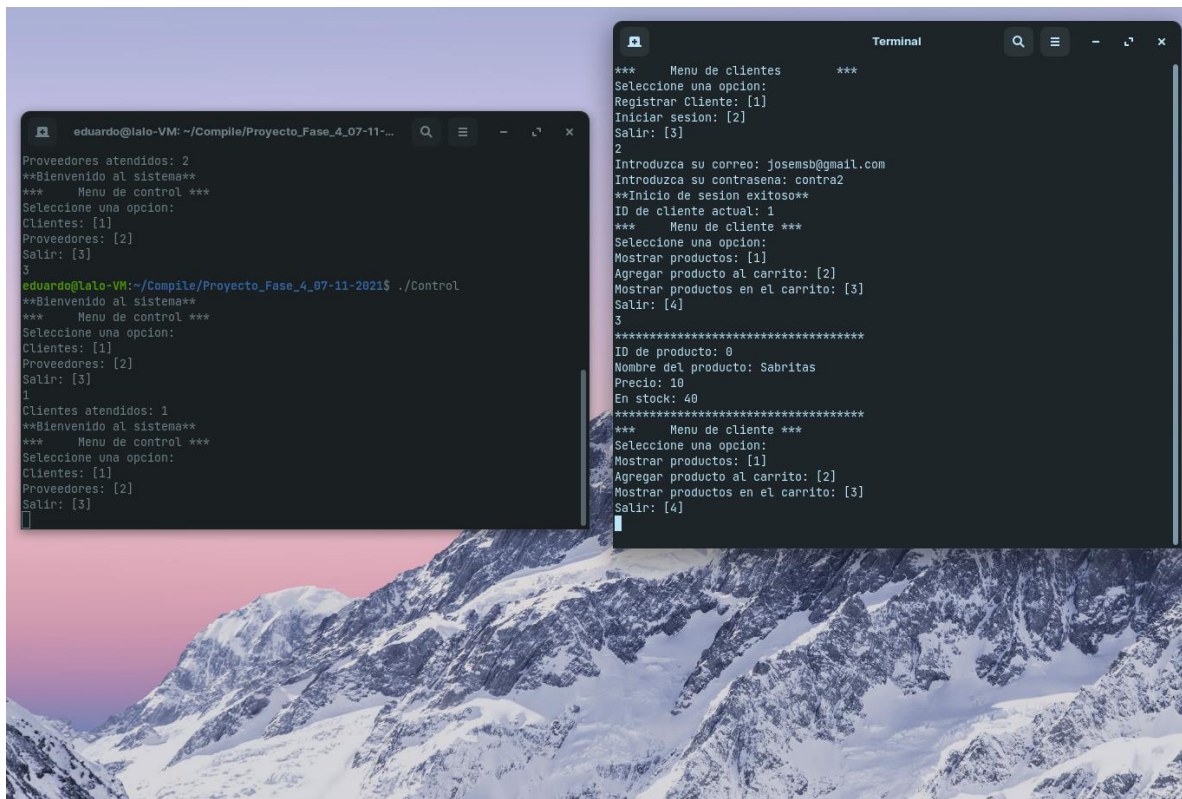
eduardo@lalo-VM: ~/Complle/Proyecto_Fase_4_0:1
*****
ID de producto: 0
Nombre del producto: Sabritas
Precio: 10
En stock: 40
*****
*****
ID de producto: 1
Nombre del producto: Cheetos
Precio: 10
En stock: 25
*****
*****
Menu de cliente ***
Seleccione una opcion:
Mostrar productos: [1]
Agregar producto al carrito: [2]
Mostrar productos en el carrito: [3]
Salir: [4]

Terminal
ID de producto: 0
Nombre del producto: Sabritas
Precio: 10
En stock: 00
*** Menu de cliente ***
Seleccione una opcion:
Mostrar productos: [1]
Agregar producto al carrito: [2]
Mostrar productos en el carrito: [3]
Salir: [4]

3
*****
ID de producto: 0
Nombre del producto: Sabritas
Precio: 10
En stock: 40
*****
*** Menu de cliente ***
Seleccione una opcion:
Mostrar productos: [1]
Agregar producto al carrito: [2]
Mostrar productos en el carrito: [3]
Salir: [4]
```

*Figura 37. Primer cliente consultando por segunda vez el catálogo.*

Posteriormente los dos clientes se retiran y se elige finalizar la ejecución del sistema, se vuelve a iniciar el proceso Control, se atiende a un cliente el cual inicia sesión y consulta su carrito de compras comprobando que en el archivo de clientes se almacenaron sus datos, en el archivo de carritos se almacenó su elección y por tanto en el archivo de catalogo también se almacenaron los productos (figura 38). Aquí al consultar los productos en el carrito se muestra el stock disponible en el catálogo, para esta implementación se decidió que se mostrara así para comprobar la escritura de todos los archivos.

The image shows two terminal windows overlaid on a background of a snowy mountain range. The left terminal window, titled 'eduardo@lalo-VM: ~/Compile/Proyecto\_Fase\_4\_07-11-...', displays a system control menu. It shows a sequence of menu selections: first, option 2 for 'Proveedores atendidos', then option 1 for 'Clientes', and finally option 3 to 'Salir'. After exiting, it shows 'Clientes atendidos: 1' and returns to the main menu. The right terminal window, titled 'Terminal', shows a client session menu. It starts with option 1 to 'Registrar Cliente', followed by entering the email 'josemsb@gmail.com' and password 'contra2', resulting in a successful login. Then, option 2 is selected to 'Mostrar productos', displaying details for 'Sabritas' (ID: 0, Price: 10, Stock: 40). Finally, option 3 is selected to 'Agregar producto al carrito', and option 4 is selected to 'Mostrar productos en el carrito'.

*Figura 38. Reinicio del sistema e inicio de sesión de un cliente.*

## Conclusiones

### **Suárez Bautista José Manuel**

Hasta este punto del curso, se ha obtenido un panorama más amplio de cómo operan los sistemas operativos, recapitulando, en la unidad 1 se vieron conceptos generales de los sistemas operativos, en la unidad 2 nos adentramos un poco más a los procesos e hilos, en la unidad 3 se vio como se administra la memoria y hasta el momento en la unidad 5 se vio cómo funcionan los dispositivos de entrada y salida. Como consecuencia de esta amplitud de conocimientos en esta fase se pudo detallar mucho más a fondo el contenido tanto de nuestro reporte como de nuestro sistema "Mi Tiendita", se obtuvo un avance en el análisis y diseño de nuestro sistema, de igual forma se realizó la implementación de nuestro programa así obteniendo la primera versión de nuestro sistema "Mi Tiendita", para posteriormente en la siguiente fase, realizar cambios a esa primera versión de nuestro programa y entregar la segunda versión de nuestro programa.

Es claro que aún quedan unidades por ver no contempladas en esta fase por lo que este trabajo seguirá modificándose y detallando conforme se termine de ver el temario.

### **Sandoval Hernández Eduardo**

Los procesos funcionan como la base de los sistemas operativos, juegan un papel importante en el ordenador al gestionar el hardware, el uso del CPU, el uso de la memoria, etc. En fases anteriores aprendimos una parte de cómo podemos gestionar los procesos, así como el uso e implementación de los hilos, además, aprendimos a comunicar procesos mediante mecanismos como los semáforos, las colas de mensajes y la memoria compartida, de igual manera comprendimos la importancia que tienen los archivos en el almacenamiento de información necesaria para la ejecución de los programas y la importancia de los dispositivos de entrada y salida para interactuar con el ordenador.

En esta fase se pudo hacer una implementación preliminar del sistema, pudimos aplicar los conocimientos adquiridos para realizar esta implementación y conseguir que fuera funcional para la estructura presentada en el análisis la cual fue modificada en esta fase para que coincidiera con la idea que se tuvo sobre cómo construir el sistema y aunque aún puede mejorar en varios aspectos ya contamos con la experiencia para seguir optimizando el

sistema, naturalmente nunca llegará a ser perfecto pues siempre pueden ocurrir imprevistos o podemos tener ideas nuevas y diferentes sobre cómo quedaría mejor implementado.

### **Renteria Arriaga Josue**

Los procesos y la administración de estos son muy importantes para llevar a cabo una tarea con ayuda de otros conceptos que ya hemos estudiado posteriormente. Con lo que estudiamos en la fase anterior del proyecto podemos decir que los semáforos, procesos, memoria virtual, hilos, etc., son parte muy importante para la realización del proyecto. Con los semáforos podemos realizar diferentes operaciones como lo vimos en el marco teórico, son muy importantes para administrar los procesos, al igual que los semáforos las Colas de mensajes son muy importantes para administrar la llegada de mensajes y estos no se interrumpan, pero, en la actualidad no son tan utilizadas ya que hay diferentes formas de administrar los mensajes.

En esta fase del proyecto empezaremos a llevar a cabo todo lo que hemos aprendido hasta ahora y empezaremos a implementar código para tener nuestra primera versión del proyecto final (Nuestro sistema para una tienda de conveniencia). En esta fase se empezarán a diseñar las partes principales de nuestro sistema “La Tienda de Conveniencia” para poder empezar a implementarla y codificarla, así obteniendo una primera versión del proyecto final.

En esta fase del proyecto nos ayudara a comprender más acerca de los conceptos investigados y entregaremos una primera versión del proyecto final, así en fases posteriores del proyecto podremos hacer más cambios en nuestro proyecto.

### **Montenegro Cervantes Tania Viridiana**

Conforme aumenta más el conocimiento respecto a cómo funcionan los sistemas operativos, se tienen más herramientas para desarrollar puntos que quedaron pendientes cuando se planteó desarrollar un sistema para una pequeña tienda de conveniencia, durante la primera fase del proyecto se abordaron los aspectos más importantes para la implementación del proyecto, posteriormente en la segunda fase al tener un primer acercamiento con los procesos se mejoró el enfoque del proyecto y en esta tercera fase se conocen los aspectos más importantes de la memoria virtual y segmentación, tomando en cuenta lo mencionado

anteriormente, se aplicará para desarrollar un prototipo mejorado para el funcionamiento del sistema.

Implementar todos los conceptos relacionados con el correcto manejo de la memoria compartida, el uso de semáforos, la utilización de hilos que son más fáciles de manejar y la planificación, facilitan mucho las tareas relacionadas con el manejo de las operaciones y son de gran ayuda para gestionar correctamente un sistema operativo. Además, se considerará la memoria virtual para que se haga un adecuado manejo del almacenamiento.

Ahora al ocupar los conceptos de los dispositivos de entrada y salida, junto con la biblioteca ncurses que pertenece al lenguaje C y que permite al programador escribir interfaces basadas en texto, además puede actualizar la pantalla óptimamente, esto es necesario para las rutinas, ya que se conocer que aspecto tiene la pantalla actual y que aspecto quiere el programador que tenga después. Con todo lo mencionado anteriormente se propuso una implementación de prueba para nuestro proyecto final.



## Referencias

- [1]. What is Computational System | IGI Global (s, f). [Internet] Disponible en: <https://www.igi-global.com/dictionary/computational-system/4998#:~:text=1.,computations%20is%20called%20computational%20system>
- [2]. Secretaría de Economía – Microempresas (2021). [Internet] Disponible en: <http://www.2006-2012.economia.gob.mx/mexico-emprende/empresas/microempresario>
- [3]. Hayes, A. How Point of Sale (POS) Works. Investopedia. (26 de octubre de 2020) [Internet] Disponible en: <https://www.investopedia.com/terms/p/point-of-sale.asp>
- [4]. "Top Retail POS Systems - 2021 Reviews, Pricing & Demos", Softwareadvice.com (2021). [Internet]. Disponible en: <https://www.softwareadvice.com/retail/>.
- [5]. MILENIO. (30 de enero de 2020). Ventas por internet: cómo digitalizar tu negocio con el menor riesgo posible. [Internet]. Disponible en: <https://www.milenio.com/negocios/ventas-por-internet>
- [6]. Sistemas operativos (2014). [Internet]. Disponible en <https://www.fing.edu.uy/inco/cursos/sistoper/recursosTeoricos/5-SO-Teo-Procesos.pdf>
- [7]. Llamadas al sistema para la gestión de procesos (s.f.). [Internet]. Disponible en <https://w3.ual.es/~jjfdez/SOA/pract6.html>
- [8]. J. Hernández Ramírez (s.f.). “Fork”. [Internet]. Disponible en <http://sopa.dis.ulpgc.es/iidso/leclinux/procesos/fork/fork.pdf>
- [9]. Gómez, R (2019). ¿Qué es el hilo en términos de programación? [Internet]. Disponible en: <https://es.quora.com/Qu%C3%A9-es-el-hilo-en-t%C3%A9rminos-de-programaci%C3%B3n>
- [10]. BUAP (s.f.). Hilos [Internet]. Disponible en: [https://www.cs.buap.mx/~iolmos/redes/3\\_Hilos.pdf](https://www.cs.buap.mx/~iolmos/redes/3_Hilos.pdf)
- [11]. Memoria compartida - Sección BD/Programación (s.f) [Internet]. Disponible en: [https://www.glosarioit.com/Memoria\\_compartida](https://www.glosarioit.com/Memoria_compartida)
- [12]. Sonzogni, V (s.f) Programación de Computadoras de Memoria Compartida [Internet]. Disponible en: [http://venus.ceride.gov.ar/cursos/moodledata/29/Transparencias/Transp\\_PCMC-1x2.pdf](http://venus.ceride.gov.ar/cursos/moodledata/29/Transparencias/Transp_PCMC-1x2.pdf)
- [13]. Díaz, N (s.f). Programación de SM de Memoria Compartida [Internet]. Disponible en: [http://www.atc.uniovi.es/inf\\_superior/4atc/trabajos/paralelas/5-Programacion%20de%20SM%20de%20Memoria%20Compartida-memoria.pdf](http://www.atc.uniovi.es/inf_superior/4atc/trabajos/paralelas/5-Programacion%20de%20SM%20de%20Memoria%20Compartida-memoria.pdf)



- [14]. Shared Memory (s.f) [Internet]. Disponible en: [https://www.tutorialspoint.com/inter\\_process\\_communication/inter\\_process\\_communication\\_shared\\_memory.htm](https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_shared_memory.htm)
- [15]. Quesignificado (s.f.) “En programación, ¿qué es un semáforo?”. [Internet]. Disponible en <https://quesignificado.org/en-programacion-que-es-un-semaforo/>
- [16]. “Semáforo (informática)”. (s.f.) [Internet]. Disponible en [https://es.wikipedia.org/wiki/Sem%C3%A1foro\\_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Sem%C3%A1foro_(inform%C3%A1tica))
- [17]. Webdiis (s.f.) “Programación concurrente con semáforos”. [Internet]. Disponible en <http://webdiis.unizar.es/~ezpeleta/lib/exe/fetch.php?media=misdatos:pc:09.pdf>
- [18]. Amazon (s.f.). “Colas de mensajes”. [Internet]. Disponible en <https://aws.amazon.com/es/message-queue/#:~:text=Una%20cola%20de%20mensajes%20es%20una%20forma%20de,procesa%20una%20vez%20sola%2C%20por%20un%20solo%20consumidor>
- [19]. M. Ángel. “Colas de mensajes: RabbitMQ”. MagMax Blog. (s.f.). [Internet]. Disponible en <https://magmax.org/blog/colas-de-mensajes-rabbitmq/#programando>

## Anexo 1. Códigos completos

### Archivos.h

```
#include "Catalogo.h"
#include "Carrito.h"
#include "Cliente.h"

// Archivo de clientes
int escribir_clientes(){
    FILE *archivo;
    archivo = fopen("Archivo_Clientes","wb");
    if(archivo){
        fwrite(Aux,sizeof(struct NodoCliente),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int leer_clientes(){
    FILE *archivo;
    archivo = fopen("Archivo_Clientes","rb");
    if(archivo){
        fread(Aux,sizeof(struct NodoCliente),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int reset_archivo_clientes(){
    FILE *archivo;
    archivo = fopen("Archivo_Clientes","wb");
    if(archivo){
        fwrite(Reset,sizeof(struct NodoCliente),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int reset_clientes(){
    reset_archivo_clientes();
    leer_clientes();
    return 1;
}

//*****
```

```

//Archivo de catalogo (productos)
int escribir_catalogo(){
    FILE *archivo;
    archivo = fopen("Archivo_Catalogo","wb");
    if(archivo){
        fwrite(Aux_Productos,sizeof(struct NodoProducto),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int leer_catalogo(){
    FILE *archivo;
    archivo = fopen("Archivo_Catalogo","rb");
    if(archivo){
        fread(Aux_Productos,sizeof(struct NodoProducto),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int reset_archivo_catalogo(){
    FILE *archivo;
    archivo = fopen("Archivo_Catalogo","wb");
    if(archivo){
        fwrite(Reset_Productos,sizeof(struct NodoProducto),64,archivo);
        fclose(archivo);
    }
    return 1;
}

int reset_catalogo(){
    reset_archivo_catalogo();
    leer_catalogo();
    return 1;
}

//*****

//Archivo de carritos

int escribir_carritos(){
    FILE *archivo;
    archivo = fopen("Archivo_Carritos","wb");
    if(archivo){

```

```

        fwrite(Aux_Carrito, sizeof(struct NodoCarrito), 64, archivo);
        fclose(archivo);
    }
    return 1;
}

int leer_carritos(){
    FILE *archivo;
    archivo = fopen("Archivo_Carritos", "rb");
    if(archivo){
        fread(Aux_Carrito, sizeof(struct NodoCarrito), 64, archivo);
        fclose(archivo);
    }
    return 1;
}

int reset_archivo_carritos(){
    FILE *archivo;
    archivo = fopen("Archivo_Carritos", "wb");
    if(archivo){
        fwrite(Reset_Carrito, sizeof(struct NodoCarrito), 64, archivo);
        fclose(archivo);
    }
    return 1;
}

int reset_carritos(){
    reset_archivo_carritos();
    leer_carritos();
    return 1;
}

int set_archivo_carritos(){
    for(int i=0 ; i<get_NumClientes() ; i++)
        Set_Carrito[i].id_cliente = i;
    FILE *archivo;
    archivo = fopen("Archivo_Carritos", "wb");
    if(archivo){
        fwrite(Set_Carrito, sizeof(struct NodoCarrito), 64, archivo);
        fclose(archivo);
    }
    return 1;
}

```

```

int set_carritos(){
    set_archivo_carritos();
    leer_carritos();
    return 1;
}
//*****

int reset_all(){
    reset_archivo_clientes();
    leer_clientes();
    reset_archivo_catalogo();
    leer_catalogo();
    reset_archivo_carritos();
    leer_carritos();
    return 1;
}

```

## Carrito.h

```

typedef struct NodoCarrito{
    int id_cliente;
    int Num_Productos_Carrito;
    int id_productos[64];
} Carrito;

Carrito Aux_Carrito[64];
Carrito Reset_Carrito[64];
Carrito Set_Carrito[64];

void add_carrito(int id){
    Aux_Carrito[id].id_cliente = id;
}

int get_Num_Productos_Carrito(int id){
    return Aux_Carrito[id].Num_Productos_Carrito;
}

void add_producto_carrito(int id_cliente){
    int stock_disponible, id_producto, cantidad;
    printf("Ingrese el ID del producto que desea agregar al carrito: ");
    id_producto = leer_int();
    printf("Ingrese la cantidad del producto que desea comprar: ");
    cantidad = leer_int();
    stock_disponible = consultar_stock(id_producto);
}

```

```

    if(stock_disponible<cantidad)
        printf("**Stock insuficiente**\n");
    else if(NumProductos>id_producto)
        printf("**Articulo inexistente**\n");
    else{
        int id_aux = Aux_Carrito[id_cliente].Num_Productos_Carrito;
        Aux_Carrito[id_cliente].id_productos[id_aux] = id_producto;
        Aux_Carrito[id_cliente].Num_Productos_Carrito++;
        imp_producto(Aux_Carrito[id_cliente].id_productos[id_aux]);
        decrementar_stock(id_producto,cantidad);
    }
}

```

## Catalogo.h

```

typedef struct NodoProducto{
    int id_producto;
    char producto_nombre[64];
    int precio;
    int stock;
} Producto;

Producto Aux_Productos[64];
Producto Reset_Productos[64];

int NumProductos = 0;

int get_NumProductos(){
    int n = 0;
    for(int i=0 ; i<64 ; i++){
        if(i==0 && i!=Aux_Productos[i].precio)
            n++;
        if(i==Aux_Productos[i].id_producto && i!=0)
            n++;
    }

    return n;
}

void add_producto(int id, char const *nombre, int price, int stock){
    Aux_Productos[id].id_producto = id;
    Aux_Productos[id].precio = price;
    Aux_Productos[id].stock = stock;
}

```

```

        strcpy(Aux_Productos[id].producto_nombre,nombre);
    }

void agregar_producto(){
    int id = get_NumProductos();
    printf("Ingrese el nombre del producto: ");
    char nombre[64];
    fgets(nombre,64,stdin);
    nombre[strlen(nombre)-1] = '\0';
    printf("Ingrese el precio: ");
    int price = leer_int();
    printf("Ingrese el stock inicial: ");
    int stock = leer_int();
    add_producto(id,nombre,price,stock);
    NumProductos = get_NumProductos();
}

int consultar_precio(int id){
    return Aux_Productos[id].precio;
}

int consultar_stock(int id){
    return Aux_Productos[id].stock;
}

void modificar_nombre(int id){
    printf("Ingrese el nuevo nombre del producto: ");
    char nombre_nuevo[64];
    fgets(nombre_nuevo,64,stdin);
    nombre_nuevo[strlen(nombre_nuevo)-1] = '\0';
    strcpy(Aux_Productos[id].producto_nombre,nombre_nuevo);
}

void modificar_precio(int id){
    printf("Ingrese el nuevo precio: ");
    int price = leer_int();
    Aux_Productos[id].precio = price;
}

void modificar_stock(int id){
    printf("Ingrese el stock: ");
    int stock = leer_int();
    Aux_Productos[id].stock = stock;
}

void incrementar_stock(int id, int cantidad){

```

```

        Aux_Productos[id].stock += cantidad;
    }

void decrementar_stock(int id, int cantidad){
    Aux_Productos[id].stock -= cantidad;
}

void imp_producto(int id){
    printf("ID de producto: %d\n",id);
    printf("Nombre del producto: %s\n",Aux_Productos[id].producto_nombre);
    printf("Precio: %d\n",consultar_precio(id));
    printf("En stock: %d\n",consultar_stock(id));
}

void imp_todos_productos(){
    printf("*****\n");
    for(int i=0 ; i<get_NumProductos() ; i++){
        imp_producto(i);
        printf("*****\n");
    }
}

void imp_nombres_productos(){
    for(int i=0 ; i<64 ; i++){
        if(i==0 && i!=Aux_Productos[i].precio)
            printf("[%s] ",Aux_Productos[i].producto_nombre);
        if(i==Aux_Productos[i].id_producto && i!=0)
            printf("[%s] ",Aux_Productos[i].producto_nombre);
    }
    printf("\n");
}

```

## Cliente.c

```

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include"Semaforos.h"
#include"Util.h"
#include"Archivos.h"

int imp_menu_cliente();
void imp_productos_carrito(int);

```



```

int imp_segundo_menu_cliente();
void seleccion_accion_cliente(int s);
int seleccion_cliente(int);

int sem_id_cliente;
int sem_id_carrito;
int sem_id_catalogo;

int main(int argc, char *argv[]){
    int menu = 0;
    leer_clientes();
    leer_carritos();
    leer_catalogo();
    NumClientes = get_NumClientes();
    //sem_id_cliente = createsem(0x1234,1);
    sem_id_cliente = atoi(argv[1]);
    sem_id_carrito = atoi(argv[2]);
    sem_id_catalogo = atoi(argv[3]);
    //printf("Sem id: %d\n",sem_id_cliente);
    do{
        menu = imp_menu_cliente();
        seleccion_cliente(menu);
    }while(menu!=3);

    return 0;
}

int imp_menu_cliente(){
    printf("***\tMenu de clientes\t***\n");
    printf("Seleccione una opcion:\n");
    printf("Registrar Cliente: [1]\n");
    printf("Iniciar sesion: [2]\n");
    printf("Salir: [3]\n");
    return leer_int();
}

void imp_productos_carrito(int id_cliente){
    int n = Aux_Carrito[id_cliente].Num_Productos_Carrito;
    if(n>0)
        printf("*****\n");
    for(int i=0 ; i<n ; i++){
        imp_producto(Aux_Carrito[id_cliente].id_productos[i]);
        printf("*****\n");
    }
}

```

```

int imp_segundo_menu_cliente(){
    printf("***\tMenu de cliente\t***\n");
    printf("Seleccione una opcion:\n");
    printf("Mostrar productos: [1]\n");
    printf("Agregar producto al carrito: [2]\n");
    printf("Mostrar productos en el carrito: [3]\n");
    printf("Salir: [4]\n");
    return leer_int();
}

void seleccion_accion_cliente(int s){
    switch(s){
        case 1:
            semwait(sem_id_catalogo);
            leer_catalogo();
            imp_todos_productos();
            semsignal(sem_id_catalogo);
            break;
        case 2:
            semwait(sem_id_carrito);
            leer_carritos();
            semwait(sem_id_catalogo);
            leer_catalogo();
            add_producto_carrito(Cliente_Actual.id_cliente);
            escribir_catalogo();
            semsignal(sem_id_catalogo);
            escribir_carritos();
            semsignal(sem_id_carrito);
            break;
        case 3:
            semwait(sem_id_carrito);
            leer_carritos();
            imp_productos_carrito(Cliente_Actual.id_cliente);
            semsignal(sem_id_carrito);
            break;
        case 4:
            exit(0);
            break;
        case 59:
            leer_carritos();
            escribir_carritos();
        case 60:
            leer_carritos();
            break;
    }
}

```

```

        case 61:
            escribir_carritos();
            break;
        case 62:
            set_carritos();
            break;
        case 63:
            leer_carritos();
            imp_productos_carrito(Cliente_Actual.id_cliente);
            break;
        case 64:
            reset_carritos();
            break;
        default:
            puts("***Opcion no valida***");
            break;
    }
}

int seleccion_cliente(int n){
    switch(n){
        case 1:
            semwait(sem_id_cliente);
            leer_clientes();
            agregar_cliente();
            escribir_clientes();
            semsignal(sem_id_cliente);
            semwait(sem_id_carrito);
            leer_carritos();
            add_carrito(NumClientes-1);
            escribir_carritos();
            semsignal(sem_id_carrito);
            printf("Se ha agregado al cliente con correo:
[%s]\n",get_correo(NumClientes-1));
            break;
        case 2:
            leer_clientes();
            iniciar_sesion();
            int sub_menu = 0;
            do{
                sub_menu = imp_segundo_menu_cliente();
                seleccion_accion_cliente(sub_menu);
            }while(sub_menu!=4);
        case 3:

```

```

        exit(0);
        break;
    case 59:
        leer_clientes();
        escribir_clientes();
    case 60:
        leer_clientes();
        break;
    case 61:
        escribir_clientes();
        break;
    case 62:
        printf("Numero de clientes: %d\n", NumClientes);
        break;
    case 63:
        leer_clientes();
        imp_all_correos();
        break;
    case 64:
        reset_clientes();
        break;
    case 100:
        reset_all();
        break;
    default:
        puts("***Opcion no valida***");
        break;
}

return 1;
}

```

## Cliente.h

```

typedef struct NodoCliente{
    int id_cliente;
    int edad;
    char correo[64];
    char password[64];
    char nombre_s[64];
    char apellido_paterno[64];
    char apellido_materno[64];
} Cliente;

```

```

Cliente Aux[64];
Cliente Reset[64];
Cliente Cliente_Actual;

int NumClientes = 0;

int get_NumClientes(){
    int n = 0;
    for(int i=0 ; i<64 ; i++){
        if(i==0 && i!=Aux[i].edad)
            n++;
        if(i==Aux[i].id_cliente && i!=0)
            n++;
    }

    return n;
}

void add(int id, int ed, char const *email, char const *pass, char const
*name_s, char const *ap, char const *am){
    Aux[id].id_cliente = id;
    Aux[id].edad = ed;
    strcpy(Aux[id].correo,email);
    strcpy(Aux[id].password,pass);
    strcpy(Aux[id].nombre_s,name_s);
    strcpy(Aux[id].apellido_paterno,ap);
    strcpy(Aux[id].apellido_materno,am);
}

const char* get_correo(int id){return Aux[id].correo;}

const char* get_nombre(int id){return Aux[id].nombre_s;}

const char* get_apellidop(int id){return Aux[id].apellido_paterno;}

const char* get_apellidom(int id){return Aux[id].apellido_materno;}

int coincide_password(int id, char const *pass){return
strcmp(pass,Aux[id].password)==0;}

int esta_cliente_correo(char const *correo){
    for(int i=0 ; i<64 ; i++){
        if(strcmp(correo,Aux[i].correo)==0)
            return i;
    }
}

```

```

    }
    return -1;
}

void agregar_cliente(){
    int id = get_NumClientes();
    printf("Ingrese su edad: ");
    int edad = leer_int();
    printf("Ingrese su nombre(s): ");
    char nombre_s[64];
    fgets(nombre_s,64,stdin);
    nombre_s[strlen(nombre_s)-1] = '\0';
    printf("Ingrese su apellido paterno: ");
    char a_p[64];
    fgets(a_p,64,stdin);
    a_p[strlen(a_p)-1] = '\0';
    printf("Ingrese su apellido materno: ");
    char a_m[64];
    fgets(a_m,64,stdin);
    a_m[strlen(a_m)-1] = '\0';
    printf("Ingrese su correo: ");
    char email[64];
    fgets(email,64,stdin);
    email[strlen(email)-1] = '\0';
    printf("Ingrese su contrasena: ");
    char password[64];
    fgets(password,64,stdin);
    password[strlen(password)-1] = '\0';
    add(id,edad,email,password,nombre_s,a_p,a_m);
    NumClientes = get_NumClientes();
}

void imp_all_ids(){
    for(int i=0 ; i<64 ; i++){
        if(i==0 && i!=Aux[i].edad)
            printf("[%d] ",Aux[i].id_cliente);
        if(i==Aux[i].id_cliente && i!=0)
            printf("[%d] ",Aux[i].id_cliente);
    }
    printf("\n");
}

void imp_all_nombres(){
    for(int i=0 ; i<64 ; i++){
        if(i==0 && i!=Aux[i].edad)

```

```

        printf("[%s] ",Aux[i].nombre_s);
        if(i==Aux[i].id_cliente && i!=0)
            printf("[%s] ",Aux[i].nombre_s);
    }
    printf("\n");
}

void imp_all_correos(){
    for(int i=0 ; i<64 ; i++){
        if(i==0 && i!=Aux[i].edad)
            printf("[%s] ",Aux[i].correo);
        if(i==Aux[i].id_cliente && i!=0)
            printf("[%s] ",Aux[i].correo);
    }
    printf("\n");
}

int iniciar_sesion(){
    char email[64];
    char password[64];
    do{
        printf("Introduzca su correo: ");
        fgets(email,64,stdin);
        email[strlen(email)-1] = '\0';
        printf("Introduzca su contraseña: ");
        fgets(password,64,stdin);
        password[strlen(password)-1] = '\0';
        if(esta_cliente_correo(email)<0)
            printf("***Correo no registrado**\n");
        else if(!coincide_password(esta_cliente_correo(email),password))
            printf("***Contraseña incorrecta**\n");
        else
            printf("***Inicio de sesion exitoso**\n");
    } while(esta_cliente_correo(email)<0
|| !coincide_password(esta_cliente_correo(email),password));
    Cliente_Actual = Aux[esta_cliente_correo(email)];
    printf("ID de cliente actual: %d\n",Cliente_Actual.id_cliente);
    return 1;
}

```

## Control.c

```

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

```

```

#include<unistd.h>
#include"Semaforos.h"
#include"Control.h"

int main(){
    id_sem_cliente = createsem(0x1234,1);
    id_sem_proveedor = createsem(0x0234,1);
    id_sem_carrito = createsem(0x0034,1);
    id_sem_catalogo = createsem(0x0004,1);
    int menu;
    do{
        menu = imp_menu_control();
        seleccion(menu);
    }while(menu!=3);

    return 0;
}

```

## Control.h

```

int NumClientesAtendidos;
int NumProveedoresAtendidos;
int id_sem_cliente;
char string_sem_cliente[20];
int id_sem_carrito;
char string_sem_carrito[20];
int id_sem_catalogo;
char string_sem_catalogo[20];
int id_sem_proveedor;
char string_sem_proveedor[20];

int imp_menu_control(){
    printf("***Bienvenido al sistema**\n");
    printf("***\tMenu de control\t***\n");
    printf("Seleccione una opcion:\n");
    printf("Clientes: [1]\n");
    printf("Proveedores: [2]\n");
    printf("Salir: [3]\n");
    int s;
    scanf("%d",&s);
    return s;
}

int atender_cliente(){

```



```

    NumClientesAtendidos++;
    printf("Clientes atendidos: %d\n", NumClientesAtendidos);
    if(fork() == 0){
        // Child process will return 0 from fork()
        //printf("I'm the child process.\n");
        sprintf(string_sem_cliente, "%d", id_sem_cliente);
        sprintf(string_sem_carrito, "%d", id_sem_carrito);
        sprintf(string_sem_catalogo, "%d", id_sem_catalogo);
        //printf("Sem ID cliente: %d\n", id_sem_cliente);
        int err = execlp("gnome-terminal", "gnome-terminal", "--
", "./Cliente", string_sem_cliente, string_sem_carrito, string_sem_catalogo, (char*)NULL);
    }
}

int atender_proveedor(){
    NumProveedoresAtendidos++;
    printf("Proveedores atendidos: %d\n", NumProveedoresAtendidos);
    if(fork() == 0){
        // Child process will return 0 from fork()
        //printf("I'm the child process.\n");
        sprintf(string_sem_proveedor, "%d", id_sem_proveedor);
        //printf("Sem ID proveedor: %d\n", id_sem_proveedor);
        int err = execlp("gnome-terminal", "gnome-terminal", "--
", "./Proveedor", string_sem_proveedor, (char*)NULL);
    }
}

int seleccion(int n){
    switch(n){
        case 1:
            atender_cliente();
            break;
        case 2:
            atender_proveedor();
            break;
        case 3:
            ereasesem(id_sem_proveedor);
            ereasesem(id_sem_cliente);
            exit(0);
            break;
        default:
            puts("***Opcion no valida***");
            break;
    }
}

```

```
    return 1;
}
```

## Proveedor.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include"Semaforos.h"
#include"Util.h"
#include"Proveedor.h"
#include"Archivos.h"

void modificar_producto();
int seleccion_proveedores(int);

int sem_id_proveedor;

int main(int argc, char *argv[]){
    int menu = 0;
    leer_catalogo();
    NumProductos = get_NumProductos();
    //sem_id_proveedor = createsem(0x0234,1);
    sem_id_proveedor = atoi(argv[1]);
    //printf("Sem id: %d\n",sem_id_proveedor);
    login_proveedor();
    do{
        menu = imp_menu_proveedor();
        seleccion_proveedores(menu);
    }while(menu!=4);

    return 0;
}

void modificar_producto(){
    printf("Ingrese el id del producto a modificar: ");
    int pm = leer_int();
    printf("Seleccione una opcion:\n");
    printf("Modificar nombre: [1]\n");
    printf("Modificar precio: [2]\n");
    printf("Modificar stock: [3]\n");
    printf("No modificar nada: [4]\n");
```

```

int opcion = leer_int();
switch(opcion){
    case 1:
        semwait(sem_id_proveedor);
        leer_catalogo();
        modificar_nombre(pm);
        escribir_catalogo();
        semsignal(sem_id_proveedor);
        break;
    case 2:
        semwait(sem_id_proveedor);
        leer_catalogo();
        modificar_precio(pm);
        escribir_catalogo();
        semsignal(sem_id_proveedor);
        break;
    case 3:
        semwait(sem_id_proveedor);
        leer_catalogo();
        modificar_stock(pm);
        escribir_catalogo();
        semsignal(sem_id_proveedor);
        break;
    case 4:
        break;
    default:
        puts("***Opcion no valida***");
        break;
}

}

int seleccion_proveedores(int n){
    switch(n){
        case 1:
            semwait(sem_id_proveedor);
            leer_catalogo();
            semsignal(sem_id_proveedor);
            printf("Ingrese el id del producto: ");
            int a_id = leer_int();
            imp_producto(a_id);
            break;
        case 2:
            semwait(sem_id_proveedor);
            leer_catalogo();

```

```

        agregar_producto();
        escribir_catalogo();
        semsignal(sem_id_proveedor);
        printf("Se ha agregado el producto con ID: %d\n", NumProductos-
1);
        break;
    case 3:
        modificar_producto();
        break;
    case 4:
        exit(0);
        break;
    case 59:
        leer_catalogo();
        escribir_catalogo();
    case 60:
        leer_catalogo();
        break;
    case 61:
        escribir_catalogo();
        break;
    case 62:
        printf("Numero de productos: %d\n", NumProductos);
        break;
    case 63:
        leer_catalogo();
        imp_nombres_productos();
        break;
    case 64:
        reset_catalogo();
        break;
    default:
        puts("***Opcion no valida***");
        break;
    }
    return 1;
}

```

## Proveedor.h

```

#define PIN_Proveedores 741963

int login_proveedor(){

```

```

printf("Introduzca el pin unico de proveedores: ");
int p = 0;
while(p!=PIN_Proveedores){
    p = leer_int();
    if(p!=PIN_Proveedores){
        printf("**PIN Incorrecto**\n");
        printf("Introduzca de nuevo el pin: ");
    }
    if(p==3)
        exit(0);
}
return 1;
}

int imp_menu_proveedor(){
printf("***\tMenu de proveedores\t***\n");
printf("Seleccione una opcion:\n");
printf("Buscar producto por id: [1]\n");
printf("Agregar producto: [2]\n");
printf("Modificar producto: [3]\n");
printf("Salir: [4]\n");
return leer_int();
}

```

## Semáforos.h

```

#include<sys/sem.h>
#include<sys/types.h>
#include<sys/ipc.h>

#define SEM_ID int

void semwait(int semid){
    struct sembuf s;
    s.sem_num=0;
    s.sem_op=-1;
    s.sem_flg=SEM_UNDO;

    semop(semid,&s,1);
    return;
}

void semsignal(int semid){
    struct sembuf s;

```

```

        s.sem_num=0;
        s.sem_op=1;
        s.sem_flg=SEM_UNDO;

        semop(semid,&s,1);
        return;
    }

int createsem(int key, int value){
    int semid;
    semid = semget(key,1,0666|IPC_CREAT);
    semctl(semid,0,SETVAL,value);
    return(semid);
}

void ereasesem(int semid){
    semctl(semid,0,IPC_RMID,0);
    return;
}

```

## Util.h

```

int leer_int(){
    int t;
    scanf("%d",&t);
    getchar();
    return t;
}

const char* leer_string(){
    fflush(stdin);
    char *temp;
    fgets(temp,64,stdin);
    temp[strlen(temp)-1] = '\0';
    return temp;
}

int int_equals(int a, int b){return a==b;}

```

## Anexo 2. Protocolo

### **Sistema Para la Gestión de “Mi Tiendita” Proyecto Sistemas Operativos (Protocolo).**

Alumnos: Montenegro Cervantes Tania Viridiana, Rentería Arriaga Josué, Sandoval

Hernández Eduardo, Suárez Bautista José Manuel.

Docente: Jiménez Benítez José Alfredo

### **Resumen**

Se presenta la propuesta de desarrollo de un sistema de software de compras electrónicas cuyo objetivo principal es el de agilizar la compra y venta de productos entre minoristas y usuarios que accedan con una cuenta registrada, los objetivos específicos se centran en mejorar la experiencia del usuario con ayuda de un buscador de productos, una interfaz agradable y sencilla de usar, etc. Un sistema como este es muy importante hoy en día para cualquier comercio electrónico que se quiera implementar, ya que permitirá llamar la atención de una mayor cantidad de usuarios, mejora la organización de los comercios, se tiene un seguimiento de los productos lo cual lleva a un mejor manejo del inventario y también les permite a los usuarios tener seguridad en sus compras.

Al final se espera un que el sistema pueda cumplir con las exigencias mencionadas anteriormente de la manera más eficiente posible. Para cumplir el objetivo se pretende utilizar una metodología ágil para el desarrollo del sistema, esta será la metodología SCRUM, la cual tiene entre sus ventajas una mejor calidad del producto final, experiencia satisfactoria en los usuarios finales, mayor motivación de los miembros del equipo de trabajo y optimización de los recursos. Para ello implementaremos un cronograma de actividades para concluir exitosamente dicho Sistema.

### **Palabras clave**

Gestión de usuarios, procesos, control, archivos, semáforos, memoria, búsqueda.

### **Introducción**

Actualmente las actividades cotidianas se vuelven cada vez más sencillas de realizar con ayuda de la tecnología, existen un sinnúmero de aplicaciones y sistemas informáticos que se especializan en un sector en específico y son ideales para cumplir con una tarea en particular. Es necesario definir qué es un sistema informático para explicar mejor algunos puntos que se tratarán a continuación, de forma general es aquel que puede procesar cualquier tipo de información que se pueda representar de forma matemática, ya sean cálculos básicos hasta tratar de analizar el pensamiento humano [1], por esta razón son utilizados por grandes empresas de todo tipo para mejorar su organización. Existen diferentes tipos de empresas que gozan de los beneficios de un sistema informático, pero en este caso el enfoque son los

sistemas informáticos que se basan en la gestión de ventas en línea para un pequeño establecimiento (tienda de conveniencia) cuyo principal problema se refiere a mejorar las búsquedas al catálogo de productos al que acceden los clientes, que también permita agregar artículos en existencia y mejore el proceso de añadir productos al carrito principalmente, posteriormente se explicará el funcionamiento en el apartado “Productos o Resultados esperados”.

Las necesidades que necesita cumplir un sistema varían dependiendo de quién lo va a ocupar, se requiere diferenciar entre los tipos de clientes que pueden solicitar este tipo de software para sus negocios como se muestra en la Tabla 1.

| Tipo de cliente   | Cantidad de empleados | Volumen de ventas  |
|-------------------|-----------------------|--|
| Grandes empresas  | 101 hasta 251         | Superiores a los 250 millones de pesos                                     |
| Medianas empresas | 31 hasta 100          | Varían desde los 100 millones y pueden superar hasta 250 millones de pesos |
| Pequeñas empresas | 11 hasta 30           | Varían entre los 4 hasta los 100 millones de pesos                         |
| Microempresas     | Menos de 10           | Menos de 4 millones de pesos   |

***Tabla 1. Tipos de empresas en México***

Una vez se conocen los tipos de empresas que existen en México, es importante resaltar que con base a la problemática de este proyecto referente a la creación de un sistema informático de ventas para una tienda de conveniencia, esta pertenece al sector de las microempresas, mismas que son fundamentales para la economía del país, ya que, según datos presentados por el último Censo Económico publicado por el Instituto Nacional de Estadística y Geografía (INEGI), del total de las empresas de México, el 95.2% son microempresas, a su vez generan el 45.6% del empleo y contribuyen con 15% del valor agregado de la economía [2], por ello es fundamental que cuenten con un sistema informático de ventas adecuado para su correcta administración.

Después de analizar qué tipo de clientes pueden solicitar un sistema informático, es importante explicar en qué consiste, se puede decir que es un lugar virtual donde un cliente ejecuta el pago de bienes o servicios y donde los impuestos sobre las ventas pueden llegar a ser pagaderos, se realiza principalmente utilizando una computadora o dispositivo electrónico móvil [3], además pueden brindar a los minoristas más oportunidades de micro mercado de categorías de productos específicas e influir en los consumidores para hacer más atractivos los productos que ofrecen.



Los sistemas de software de ventas electrónicos agilizan las operaciones minoristas al automatizar el proceso de compra [4], realizar un seguimiento de los datos de ventas importantes, coordinar los datos recopilados de las compras diarias, generar recibos de forma electrónica, administración de inventario, informes y análisis de ventas, entre otros.

Por ejemplo, el concepto de tienda de conveniencia de Amazon, Amazon Go, que implementa tecnologías que permiten a los compradores entrar, tomar artículos y salir sin pasar por una caja registradora, podría revolucionar los sistemas de venta tal y como se conocen al aumentar la comodidad, esto podría permitir que los sistemas de venta, la lealtad y los pagos se conviertan en una única experiencia centrada en el cliente.

## **Objetivos**

Desarrollar un sistema informático sobre la correcta administración y gestión de los procedimientos relacionados con las ventas para una pequeña tienda de conveniencia.

### **Objetivos específicos:**

- Desarrollar un algoritmo eficaz para la búsqueda por producto.
- Utilizar las herramientas proporcionadas por el sistema operativo para gestionar procesos.
- Diseñar una interfaz agradable, intuitiva e interactiva para los usuarios.
- Implementar el uso de semáforo, archivos y memoria compartida dentro del sistema de ventas para facilitar su accesibilidad.
- Sincronizar de manera eficiente y correcta la información mientras se interactúa con ella desde diferentes usuarios como clientes o proveedores.
- Implementar un sistema multiusuario, que permita múltiples interacciones en momentos simultáneos.

## **Justificación**

En el sistema por desarrollar se propone resolver la correcta administración y gestión de los procedimientos relacionados con las ventas para una pequeña tienda de conveniencia. Principalmente con la digitalización de una interfaz agradable e interactiva, el desarrollo de un algoritmo eficaz para la búsqueda por productos, el uso de semáforos, la utilización de archivos y memoria compartida dentro del sistema de ventas para facilitar su accesibilidad; además de utilizar las herramientas proporcionadas por el Sistema Operativo para gestionar los procesos. Por lo mismo, se implementarán algunas características de un Sistema Seguro.

Con lo anterior, podremos resolver las siguientes dificultades que se presentan en una Tienda de conveniencia:

- **Comunicación:** En gran parte de los casos, cuando se tiene una tienda (pequeña o mediana) los gerentes son los encargados de gestionar la información de la tienda (inventarios, ventas de productos y empleados) con ayuda de sus empleados que registran cada una de sus ventas, en ocasiones este responsable no se encuentra en su área de trabajo, así que la información llega a acumularse o perderse. Esto es un problema para el responsable de gestión, pues no tiene un canal dedicado para gestionar dicha información.
- **Organización:** En gran parte de los casos, en las pequeñas y medianas tiendas no se lleva un gran control de información acerca de los productos, personal de trabajo y proveedores. Esto es un problema para los responsables de gestionar dicha información, pues no cuentan con un canal dedicado para consultar la información de toda la tienda.
- **Trazabilidad:** Una vez que los proveedores dejan su mercancía correspondiente con el encargado de administrarla y gestionarla, se pierde el seguimiento de cada uno de los productos puestos a disposición del encargado de la tienda, ya que no cuentan con su propio canal para administrar la información de productos recibidos.
- **Seguridad:** Al no contar con un Sistema eficiente para administrar el inventario de entradas y salidas de productos y personal en la Tienda, esto puede llegar a ser un gran problema, ya que dichos productos pueden ser robados sin que nadie se dé cuenta (ya sea por los clientes o por el personal).
- **Recursos:** Para guardar todos los datos de nuestra tienda sin un Sistema que nos ayude a hacerlo sería muy complicado, muy cansado y nos llevaría mucho tiempo hacerlo. Además de no administrar correctamente nuestra información.
- **Tiempo:** Para guardar los datos de los clientes, proveedores y empleado se lleva un gran tiempo en guardar y recuperar la información de toda la tienda sin un Sistema que lo haga por nosotros. El realizar el guardado de información de la manera tradicional solo nos aumenta el tiempo que nos llevara realizar dicha tarea, por ende, una pérdida de dinero.

Todo lo anterior genera una pérdida de información importante que es difícil de recuperar, además de que genera un punto de contacto de múltiples personas.

Siendo conscientes de la situación que estamos viviendo (sociedad tecnológica), los sistemas deberán permitir establecer distintas reglas de negocios adaptadas a nuestro nuevo entorno de trabajo. Así podremos establecer un nuevo Sistema de trabajo que nos ayudara a administrar mejor la información de una Tienda de conveniencia.

## Productos o Resultados esperados

En el siguiente diagrama se muestra el flujo del proceso usando el producto esperado.

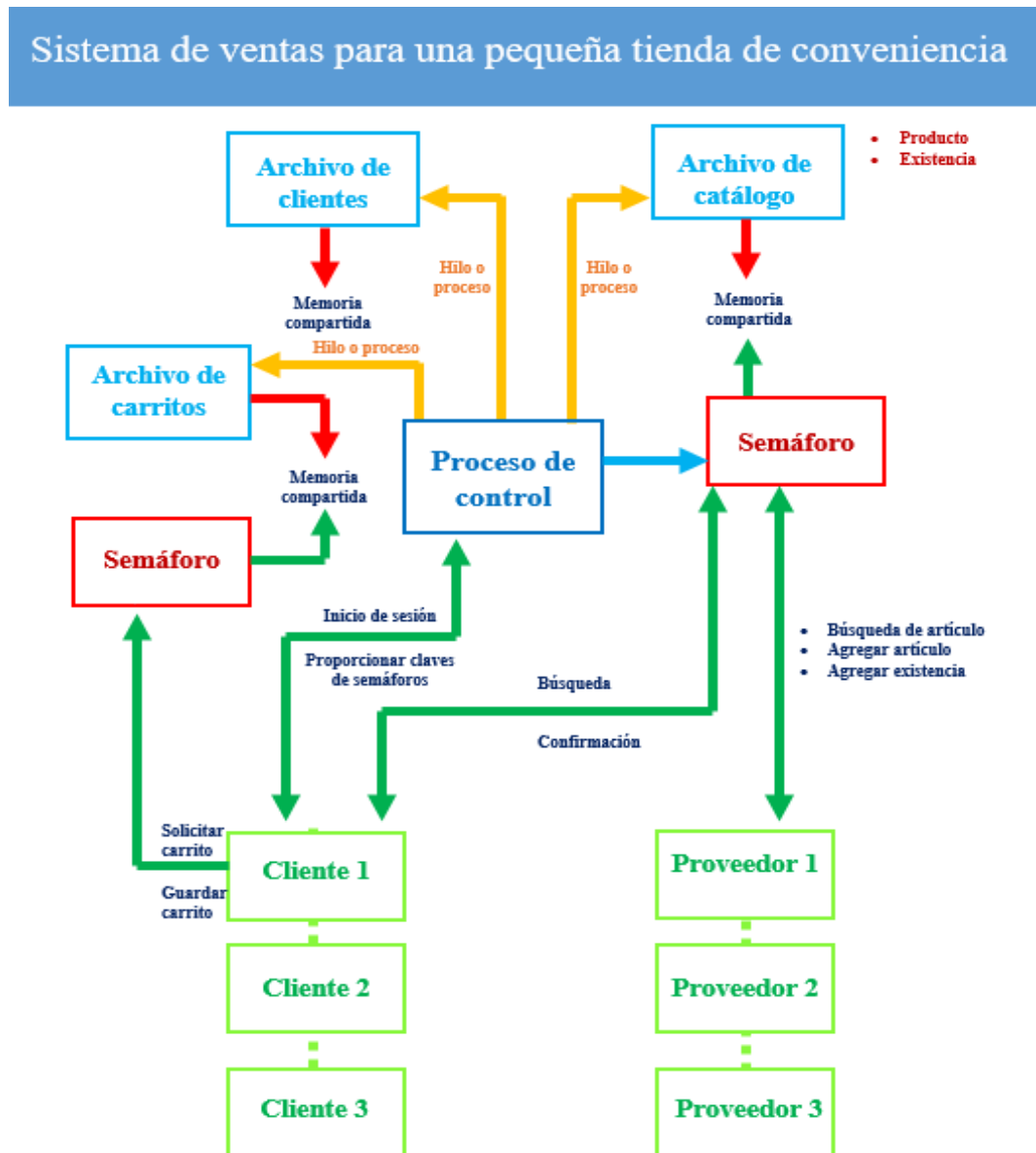


Figura 1. Diagrama del producto esperado.

A continuación, expondremos los resultados que se esperan obtener de manera exitosa en este proyecto:

1. Un sistema funcional:

Se busca realizar un sistema de ventas para una tienda el cual permite archivar información del catálogo de venta, carrito de compras y de los clientes.

El sistema deberá tener la capacidad de realizar varias acciones de manera simultánea, así como de sincronizar los resultados de cada acción de así requerirse. Además de lo ya mencionado se espera tener un sistema estable que permite realizar lo que se requiere sin ningún problema como pérdida de información o de arrojar datos erróneos.

Se permitirá iniciar sesión a través de un cliente, éste tendrá la posibilidad de solicitar un carrito de compras, consultar en el catálogo de venta y guardar productos en el carrito.

También se espera permitir que proveedores interactúen con el sistema, buscando y agregando artículos.

2. Reporte de proyecto:

Se espera generar un documento en el que se documente todo lo realizado a lo largo del desarrollo del proyecto, a través de este se buscará dar a conocer cada detalle.

3. Videos explicativos:

Se espera que a través de videos se simule el uso del sistema y se explique paso a paso la interacción con el producto final.

## Metodología

Se optó por investigar sobre una metodología ágil ya que se encontró que estas permiten adaptar la forma de trabajar según las condiciones del proyecto, de la manera en que existe una flexibilidad y rápida respuesta para el desarrollo del proyecto ante las circunstancias del entorno. Se encontraron las siguientes ventajas del uso del Agile Project Management:

- **Mejor calidad de producto:** fomentan un enfoque “proactivo”, existe una integración, comprobación y mejora continua que mejora el resultado del proyecto.
- **Satisfacción del cliente:** el cliente está en constante contacto con los avances del proyecto por lo que se siente involucrado a lo largo del proceso del proyecto.
- **Mayor motivación en los miembros:** cuando los equipos toman las riendas de manera autogestionada, se facilita la capacidad creativa y de innovación de los involucrados.

- **Trabajo colaborativo:** permite una mejor organización por equipos, ya que el trabajo es dividido por distintas colaboraciones y roles.
- **Métricas más relevantes:** se es más consciente de lo que está sucediendo ya que los equipos son más pequeños.
- **Reducción de costes:** ya que los errores se identifican en el momento y no al final, se elimina prácticamente la probabilidad de fracaso.

Después de optar e investigar sobre las metodologías ágiles se encontró que SCRUM es de las más utilizadas y se obtuvo lo siguiente:

Proceso donde se aplican buenas prácticas colaborativamente, con el objetivo de obtener el mejor resultado posible de un proyecto. Se busca trabajar en equipos potencialmente productivos. Aunque esta metodología está basada en el caos "controlado" tiene la capacidad de enfrentar lo indeterminado, lo impredecible y tener control sobre la flexibilidad que permite.

Se caracteriza por entregas parciales y regulares del producto: esto nos dice que está orientado para proyectos donde se requiere ver y obtener resultados lo más pronto posible, donde no hay requisitos exactamente definidos, es decir no hay un camino exacto al cual seguir.

#### **Fundamentos:**

1. Cada requisito es realizado en espacios temporales cortos y fijos, es decir se tiene una fecha destinada para la cual debe ser entregada una parte del proyecto.
2. Por cada espacio temporal se priorizan tareas por valor para el cliente (Producto Backlog).
3. Sincronización constante y realización de adaptaciones en forma.
4. Los equipos cuentan con la suficiente autoridad requerida para entregar los requisitos sin complicaciones.
5. Técnica del timebox para conseguir resultados en un tiempo fijo.

#### **Proceso de aplicación:**

“El proyecto se ejecuta en bloques temporales cortos y fijos”. [6]

##### **1. Sprint planning:**

Planificación de tareas divididas en dos partes. Timebox de 1 a dos horas, Se presenta la Producto Backlog por parte del cliente y se resuelven dudas y detalles por parte del equipo. De nuevo se realiza un Timebox donde se planifica y se elabora una táctica beneficiaria.

##### **2. Sprint:**

Es la ejecución de la iteración o el bloque de un requisito, diariamente se realiza un Daily meeting por equipos, donde se sincroniza el proyecto y se actualiza la lista de tareas. En este punto el Scrum Master se encarga de eliminar cualquier obstáculo que reduzca la productividad de los equipos.

### **3. Daily meeting:**

Nuevamente se realiza una reunión donde se expone que se ha hecho y que es lo que sigue por hacer.

### **4. Sprint review:**

Reunión informal con el cliente donde se presentan requisitos completados.

### **5. Sprint Retrospective:**

Se analiza la manera de trabajar reflexionando si se está cumpliendo o no con los objetivos.

### **6. Product Backlog Refinement:**

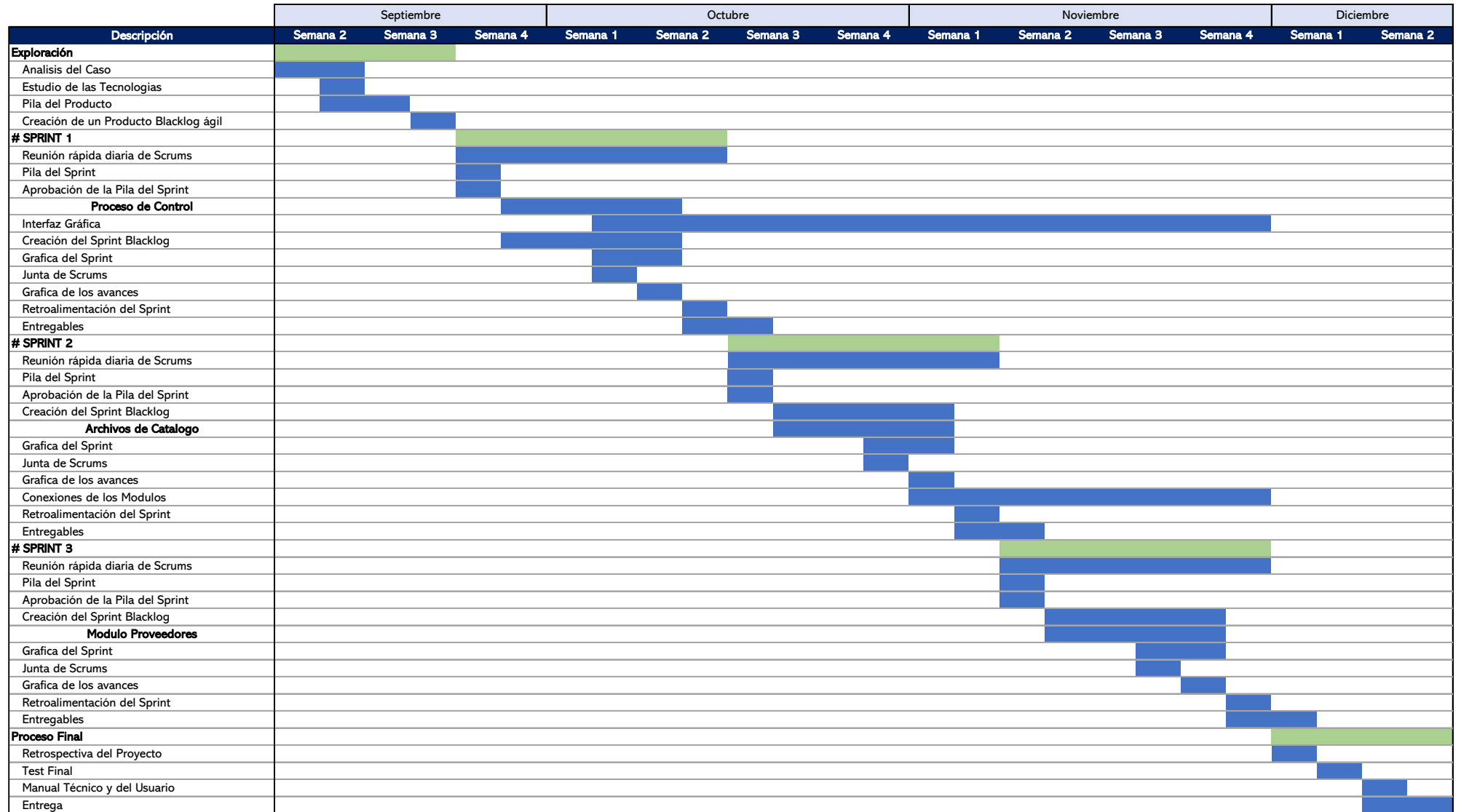
El cliente trabaja con la lista de requisitos, añadiendo, modificando o eliminando, priorizando y definiendo un calendario de entrega que se ajuste a nuevas necesidades.

### **Beneficios:**

- Se entregan avances a corto plazo.
- Se basa en resultados tangibles.
- Se anticipan resultados.
- Es flexible y adaptable ante nuevas necesidades del cliente.
- Productividad – calidad.
- Comunicación cliente – equipo de desarrollo.
- Los riesgos se mitigan.

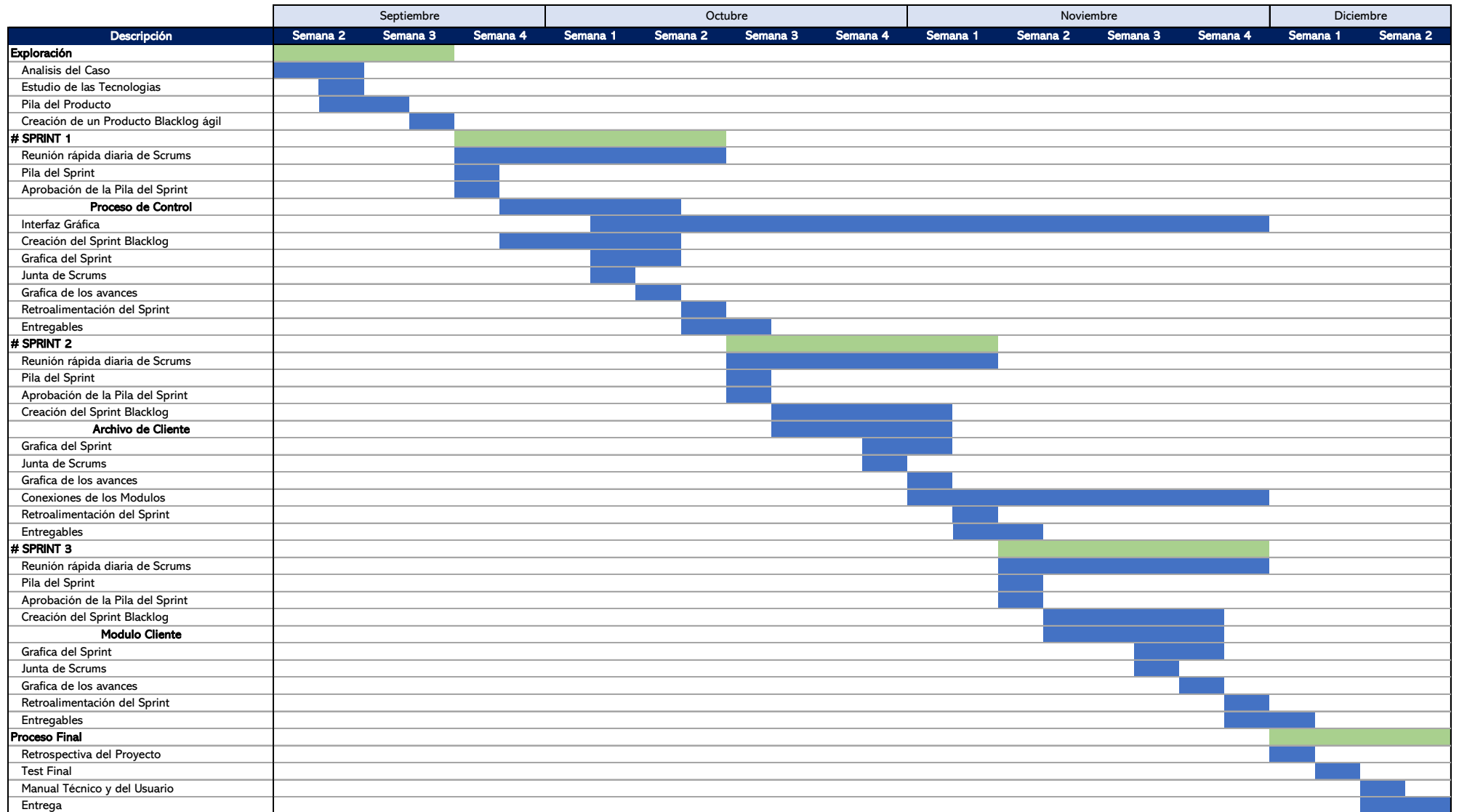
# Sistema Para la Gestión de “Mi Tiendita”

Título: Sistema Para la Gestión de “Mi Tiendita”



Alumno: Rentería Arriaga Josué.

Título: Sistema Para la Gestión de “Mi Tiendita”





**Título:** Sistema Para la Gestión de “Mi Tiendita”

|                                      | Septiembre |          |          | Octubre  |          |          |          | Noviembre |          |          |          | Diciembre |          |  |
|--------------------------------------|------------|----------|----------|----------|----------|----------|----------|-----------|----------|----------|----------|-----------|----------|--|
| Descripción                          | Semana 2   | Semana 3 | Semana 4 | Semana 1 | Semana 2 | Semana 3 | Semana 4 | Semana 1  | Semana 2 | Semana 3 | Semana 4 | Semana 1  | Semana 2 |  |
| Exploración                          |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Análisis del Caso                    |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Estudio de las Tecnologías           |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Pila del Producto                    |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Creación de un Producto Backlog ágil |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| # SPRINT 1                           |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Reunión rápida diaria de Scrums      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Pila del Sprint                      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Aprobación de la Pila del Sprint     |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Proceso de Control                   |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Interfaz Gráfica                     |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Creación del Sprint Backlog          |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Grafica del Sprint                   |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Junta de Scrums                      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Grafica de los avances               |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Retroalimentación del Sprint         |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Entregables                          |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| # SPRINT 2                           |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Reunión rápida diaria de Scrums      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Pila del Sprint                      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Aprobación de la Pila del Sprint     |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Creación del Sprint Backlog          |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Archivos del Carrito                 |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Grafica del Sprint                   |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Junta de Scrums                      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Grafica de los avances               |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Conexiones de los Modulos            |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Retroalimentación del Sprint         |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Entregables                          |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| # SPRINT 3                           |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Reunión rápida diaria de Scrums      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Pila del Sprint                      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Aprobación de la Pila del Sprint     |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Creación del Sprint Backlog          |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Uso de Memorias Compartidas          |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Grafica del Sprint                   |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Junta de Scrums                      |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Grafica de los avances               |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Retroalimentación del Sprint         |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Entregables                          |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Proceso Final                        |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Retrospectiva del Proyecto           |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Test Final                           |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Manual Técnico y del Usuario         |            |          |          |          |          |          |          |           |          |          |          |           |          |  |
| Entrega                              |            |          |          |          |          |          |          |           |          |          |          |           |          |  |

Título: Sistema Para la Gestión de “Mi Tiendita”

|                                      | Septiembre |          |          | Octubre  |          |          |          | Noviembre |          |          |          | Diciembre |          |
|--------------------------------------|------------|----------|----------|----------|----------|----------|----------|-----------|----------|----------|----------|-----------|----------|
| Descripción                          | Semana 2   | Semana 3 | Semana 4 | Semana 1 | Semana 2 | Semana 3 | Semana 4 | Semana 1  | Semana 2 | Semana 3 | Semana 4 | Semana 1  | Semana 2 |
| Exploración                          |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Análisis del Caso                    |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Estudio de las Tecnologías           |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Pila del Producto                    |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Creación de un Producto Backlog ágil |            |          |          |          |          |          |          |           |          |          |          |           |          |
| # SPRINT 1                           |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Reunión rápida diaria de Scrums      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Pila del Sprint                      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Aprobación de la Pila del Sprint     |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Proceso de Control                   |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Interfaz Gráfica                     |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Creación del Sprint Backlog          |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Gráfica del Sprint                   |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Junta de Scrums                      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Gráfica de los avances               |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Retroalimentación del Sprint         |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Entregables                          |            |          |          |          |          |          |          |           |          |          |          |           |          |
| # SPRINT 2                           |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Reunión rápida diaria de Scrums      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Pila del Sprint                      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Aprobación de la Pila del Sprint     |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Creación del Sprint Backlog          |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Módulo de Inicio Sesión              |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Gráfica del Sprint                   |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Junta de Scrums                      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Gráfica de los avances               |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Conexiones de los Módulos            |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Retroalimentación del Sprint         |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Entregables                          |            |          |          |          |          |          |          |           |          |          |          |           |          |
| # SPRINT 3                           |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Reunión rápida diaria de Scrums      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Pila del Sprint                      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Aprobación de la Pila del Sprint     |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Creación del Sprint Backlog          |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Implementación de Semáforos          |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Gráfica del Sprint                   |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Junta de Scrums                      |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Gráfica de los avances               |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Retroalimentación del Sprint         |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Entregables                          |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Proceso Final                        |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Retrospectiva del Proyecto           |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Test Final                           |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Manual Técnico y del Usuario         |            |          |          |          |          |          |          |           |          |          |          |           |          |
| Entrega                              |            |          |          |          |          |          |          |           |          |          |          |           |          |

