

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

SISTEMA DE COMPRA, VENTA DE PRODUCTOS Y
CONTROL DE STOCK PARA BUSANDWICHERÍA

Trabajo de Titulación para optar al Título
de Técnico universitario en INFORMÁTICA

Alumnos:

Nicolás Antonio Huerta Riova

Diego Andrés Jara Estivill

Profesor Guía:

Ricardo Cahe Cabach

Dedicatoria y agradecimientos

A mi madre Cecilia: Por siempre haberme apoyado y estar preocupada por mí en todo sentido, mostrándome con su ejemplo que con esfuerzo y dedicación se puede llegar muy lejos.

A mi padre Carlos: Por el apoyo, motivación y ejemplo durante este tiempo. Agradezco todos los sacrificios hechos para que lograra completar esta etapa de mi vida.

A mis amigos: Por todas las palabras de apoyo y el ánimo que me daban, a pesar de no estar presentes. Siempre podía contar con sus fuerzas empujándome desde mi espalda.

A mi perro (Q.E.P.D): Por su amor incondicional, siempre podía contar con sus orejas para escucharme y su cariño para apoyarme; fue vital durante largos años, y en principal mi vida universitaria.

A mis profesores: Por su dedicación, paciencia y motivación al momento de enseñarme, atender mis dudas y siempre dispuestos en momentos de querer aprender aún más.

Nicolás Huerta

A mi madre Eliana: Por haberme apoyado tanto monetaria como psicológicamente, siendo un pilar fundamental en el éxito de mi carrera.

A mi padre Jorge: Por el apoyo y la paciencia que me otorgó en este duro proceso, al igual que me mi madre es un factor fundamental en el éxito de mi carrera.

A mis amigos: Por el apoyo que me dieron y motivándome a seguir adelante con este duro andar. Ellos tendrán mi apoyo cuando lo necesiten como ellos dieron el suyo.

A mis mascotas: Por su amor incondicional, fueron un factor importante en este viaje, sobre todo mi perro desaparecido Perla, quien siempre estuvo allí para apoyarme antes de que desapareciera de mi vida.

A mis profesores: Por su dedicación y paciencia en responder todas mis dudas y apoyarme incluso cuando estaban pasando ellos mismos por momentos difíciles y de estrés.

Diego Jara

RESUMEN

Keywords: Sistema de compra, venta y control de stock – El Privilegio del Sabor.

El Privilegio del Sabor es un restaurant que brinda variada comida rápida y al paso. Como todo establecimiento, necesita de diferentes sistemas y servicios que los apoyen en el día a día con las diferentes funciones que deben desarrollar.

Debido a ésto se realizó un sistema informático stand-alone, con el objetivo de apoyar al establecimiento con la funcionalidad de agilizar las transacciones de compra, venta y control del inventario.

Este proyecto se basa en los requerimientos solicitados por el cliente y, principalmente, en problemas detectados en una primera etapa de análisis y diseño, procurando cumplir con todos los objetivos propuestos.

Para la creación de este trabajo se utilizaron las siguientes herramientas: Java como lenguaje de programación, Netbeans IDE como software de desarrollo y MariaDB como motor de base de datos.

En el primer capítulo, denominado “Aspectos Relevantes del diseño lógico”, se detallará la descripción de la organización, la descripción de la situación actual, se listarán los problemas detectados y se realizará una descripción del sistema propuesto.

Para el segundo capítulo, denominado “Medio Ambiente Computacional y Descripción de Archivos”, se especificarán los datos técnicos, tanto a nivel de hardware como de software, utilizados para desarrollar el sistema propuesto. Además, se detallarán todas las tablas utilizadas en la base de datos.

Para el tercer capítulo, denominado “Descripción de Programas”, se detallará la arquitectura del sistema, el diagrama de menú, programas, incluyendo diagrama de bloques y las pantallas que en conjunto a las reglas de proceso, representan las funciones que conforman el sistema.

Finalmente, se desarrolla la conclusión del trabajo, junto con un anexo con los códigos de los programas fuentes seleccionados.

ÍNDICE

RESUMEN

INTRODUCCIÓN 1

CAPÍTULO 1: ASPECTOS RELEVANTES DEL DISEÑO LÓGICO 2

1.1 DESCRIPCIÓN DE LA ORGANIZACIÓN..... 3

1.2 SITUACIÓN ACTUAL 3

1.3 PROBLEMAS DETECTADOS..... 9

1.4 DESCRIPCIÓN DEL SISTEMA PROPUESTO 9

1.4.1 Objetivos y beneficios del sistema propuesto..... 10

1.4.2 Descripción general de la solución propuesta 11

1.4.3 Funcionalidades..... 12

1.4.4 Descripción de fórmulas y cálculos realizados..... 13

1.4.5 Información a manejar..... 14

1.4.6 Modelo de datos 17

1.4.7 Estructura del código 17

1.4.8 Condicionantes del diseño 18

CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS 19

2.1 DESCRIPCIÓN DEL RECURSO COMPUTACIONAL 20

2.1.1 Configuración del sistema 20

2.1.2 Software utilizado..... 20

2.1.3 Herramientas de desarrollo..... 21

2.2 DESCRIPCIÓN DE ARCHIVOS 21

2.2.1 Proveedor..... 21

2.2.2 Registro-compra 21

2.2.3 Detalle- Registro-Compra 22

2.2.4 Insumos 22

2.2.5 Producto..... 24

2.2.6 Detalle-producto 24

2.2.7 Categoría 24

2.2.8 Detalle-pedido 25

2.2.9	Pedido.....	25
2.2.10	Mesas.....	26
2.2.11	Usuario	27
2.2.12	Deuda-cliente.....	27
2.2.13	Cliente-frecuente	28
2.2.14	Registro-pago-deuda	28
2.2.15	Ajuste-stock.....	29
2.2.16	Llevar.....	29
CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS		31
3.1	Diagrama Modular	32
3.2	Diagrama de menús	33
3.3	Descripción de programas	34
3.4	Descripción detalle de programas	35
3.4.1	Ventas, Menú.....	35
3.4.2	Ventas, Menú, Pedido	37
3.4.3	Ventas, Pagar deuda	40
3.4.4	Productos, Mantenedores, Insumos	41
3.4.5	Productos, Mantenedores, Productos	43
3.4.6	Productos, Control Stock.....	45
3.4.7	Informe de ganancia	47
CONCLUSIONES		50
BIBLIOGRAFÍA.....		51
ANEXOS.....		52

ÍNDICE DE FIGURAS

Figura 1-1 – Organigrama de la empresa.	3
Figura 1-2 – Ejemplo hoja de toma de pedidos.	4
Figura 1-3 – Página de libro de ventas.	6
Figura 1-4 – Modelo de datos.....	17
Figura 3-1 – Pantalla Menú	36
Figura 3-2 – Pantalla Pedido	39
Figura 3-3 – Pantalla Pago deuda.....	41
Figura 3-4 – Pantalla Mantenedor Insumos.....	42
Figura 3-5 – Pantalla Mantenedor Productos	45
Figura 3-6 – Pantalla Control stock.....	47
Figura 3-7 – Pantalla Informe ganancias.....	49

ÍNDICE DE DIAGRAMAS

Diagrama. 1-1 – Diagrama de flujo administrativo del proceso “Venta”.....	7
Diagrama. 1-2 – Diagrama de flujo administrativo del proceso “Compra”	8
Diagrama. 3-1 – Diagrama Modular	32
Diagrama. 3-2 – Diagrama de menús	33
Diagrama. 3-3 – Diagrama de bloque de Menú	35
Diagrama. 3-4 – Diagrama de bloque de pedido.....	37
Diagrama. 3-5 – Diagrama de bloque de Pagar deuda	40
Diagrama. 3-6 – Diagrama de bloque mantenedor insumos	41
Diagrama. 3-7 – Diagrama de bloque mantenedor productos	43
Diagrama. 3-8 – Diagrama de bloque Control stock.....	45
Diagrama. 3-9 – Diagrama de bloque informe ganancias	48

ÍNDICE DE TABLAS

Tabla. 2-1 – Tabla Proveedor 21

Tabla. 2-2 – Tabla Reg-compra..... 22

Tabla. 2-3 – Tabla Detalle-Reg-Compra 22

Tabla. 2-4 – Tabla Insumos 23

Tabla. 2-5 – Tabla Producto 24

Tabla. 2-6 – Tabla Detalle-producto 24

Tabla. 2-7 – Tabla Categoría..... 25

Tabla. 2-8 – Tabla Detalle-pedido..... 25

Tabla. 2-9 – Tabla Pedido 26

Tabla. 2-10 – Tabla Mesas 27

Tabla. 2-11 – Tabla Usuario..... 27

Tabla. 2-12 – Tabla Deuda-Cliente 28

Tabla. 2-13 – Tabla Cliente-frecuente..... 28

Tabla. 2-14 – Tabla Registro-pago-deuda 28

Tabla. 2-15 – Tabla Ajuste-stock 29

Tabla. 2-16 – Tabla Llevar 29

Tabla. 3-1 – Tabla Descripción de programas..... 34

INTRODUCCIÓN

El desarrollo de la tecnología ha ayudado en casi todos los aspectos del quehacer humano a lo largo del tiempo, mejorando las actividades que éste realiza y la comunicación dentro del globo. De la misma manera el desarrollo informático ha ido creciendo de manera exponencial los últimos años, abarcando casi cada aspecto de la vida diaria. Por lo tanto, es necesario para los profesionales la creación de nuevas tecnologías y desarrollo de nuevos programas, además de mejorar y optimizar las ya existentes; bajo esta misma premisa es que surgió la necesidad de realizar el siguiente trabajo, el cual será detallado paso a paso en este informe.

Este trabajo de título se realizará en el restaurant Busandwichería “El Privilegio del Sabor”, el cual ofrece diversos platos caseros y comida rápida al paso; ubicado en Las Araucarias #1390 La Ligua.

El sistema a implementar trabajará de manera independiente, sin la interacción de algún otro sistema. La necesidad de este sistema nace de optimizar y facilitar las actividades diarias y tener un control absoluto de los insumos que entran y salen del establecimiento, ya que la ausencia de éste produce pérdidas de tiempo, falta de información y errores en los gastos.

Lo que se pretende lograr con el sistema es automatizar los registros de entradas y salidas de los productos/insumos, además de generar un control de stock automático que permita saber la cantidad actual de activos.

CAPÍTULO 1: ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1. ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1.1 DESCRIPCIÓN DE LA ORGANIZACIÓN

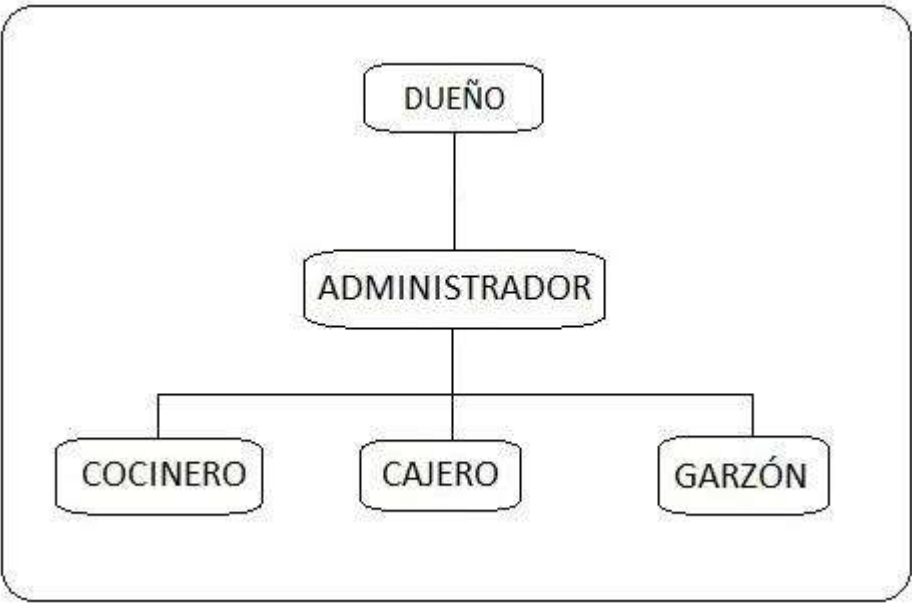
El restaurante “El Privilegio del sabor”, fundado en enero del 2015, corresponde a una sociedad con fines de lucro y a plazo indefinido, cuyo objetivo es ofrecer los más deliciosos platos y comida al paso, al menor precio.

Es una sociedad de responsabilidad limitada. Esto quiere decir que el capital está dividido entre las cuotas sociales según el capital aportado por los socios. Además, en este tipo de sociedad, se brinda mayor importancia al grupo de personas que comparten el bloque inversor al momento de la toma de decisiones.

Su actividad se basa en elaborar y proporcionar diversos platos y/o comida al paso, tales como churrascos, lomitos, completos, papas fritas, empanadas además de diversos platos caseros; mientras que para beber se cuenta con diversas bebidas, néctares y jugos naturales, además de ofrecer diversos helados.

Los insumos utilizados para preparar y ofrecer estos artículos, son adquiridos mediante proveedores o puntos de ventas especializados.

El organigrama de la empresa es el siguiente:



Fuente: Lucy Riova T., Jefa del Negocio.
Figura. 1-1 – Organigrama de la empresa.

1.2 SITUACIÓN ACTUAL

El establecimiento ofrece una diversidad de platos y comida al paso, por lo cual trabaja con una gran diversidad de productos perecibles para el servicio del consumidor. Esto implica que sea necesario reabastecerse constantemente de insumos, para lo cual cuenta con diversos proveedores y, además, se realizan compras a puntos de ventas especializados para la reposición de los elementos que se necesiten con urgencia.

Las actividades que se realizan dentro del establecimiento giran en torno a la toma de pedidos, la cual se realiza de dos maneras; la primera cuando el cliente se acerca a la caja y solicita lo deseado (generalmente esto sucede cuando el individuo desea su pedido para llevar) y, segunda, sucede cuando el garzón se acerca a la mesa del cliente para tomar la orden. Ambas acciones son registradas en papel manualmente, para posteriormente en otra hoja registrar la orden, la cual sirve para que el cocinero se entere de qué fue lo que se solicitó. Los pedidos para servir tienen la posibilidad de añadir productos adicionales a su pedido, aparte de los ordenados anteriormente; estos pueden ser solicitados en cualquier momento, siempre y cuando, la “cuenta” aún no haya sido solicitada, generando ocasionalmente interposición de pedidos.

El establecimiento cuenta con un número determinado de mesas, la identificación de cada de éstas se establece con una placa metálica que indica mediante un número único, el número de dicha mesa.

A continuación, en la figura 1-2 se muestra un ejemplo del documento toma de pedido.

FECHA		CAJA	
2	con I		
1	pepsi wts	2800	
3	con con	4900	
1	EXV		
1	B l u c o	2800	
1	p o p o	3800	
2	con I	4800	P
1	B l u c o		
1	Supremo	3900	P
1	CH I		
1	con I	1000	P
1	con fante mayo	1000	
1	B l u c o	2800	6200
1	p o p o	2400	

Fuente: Lucy Riova T., Jefa del Negocio.

Figura. 1-2 – Ejemplo hoja de toma de pedidos.

Al momento que el cliente desea pagar su orden, se calcula manualmente el total según lo detallado en el pedido, para que finalmente se realice el pago, siendo éste en efectivo, tarjeta o si desea anotar

el total en una cuenta “privada” (siempre y cuando, éste cuente con una), la cual se cancela a fin de mes.

Como se mencionó anteriormente, otra de las funciones que se realiza dentro del recinto corresponde a la adquisición de nuevos productos para insumos y la venta. A continuación, se procederá a describir como se realiza dicha actividad actualmente:

El proceso general de compras consiste en que, cada proveedor envía a un vendedor de terreno, el cual toma contacto con el jefe en turno (Administrador), quien debe solicitar los insumos faltantes que se estimen convenientes, para lo cual necesita haber revisado anteriormente en bodega, los productos que se encontraban en stock crítico. Tras escoger los productos y las cantidades necesarias de cada uno, valorizarlos e informar las condiciones de pago; dependiendo del distribuidor, se recibe el pedido en el instante, al día siguiente o subsiguiente.

Una vez lleguen los productos con su respectiva factura y guía de despacho, éstos serán aprobados por el administrador, siempre que se encuentren en su correcto estado, y el tipo y la cantidad sean conforme a lo solicitado; finalmente, la factura es archivada. Ocasionalmente, proveedores o personas conocidas por los dueños, realizan donaciones al establecimiento, como recompensa por haber sido buenos clientes o por una amistad establecida.

El listado de insumos existentes en el establecimiento creado no posee una estructura definida, ni tampoco existe una forma de categorizar cada insumo existente, por lo que solo se limita a especificar a nombrar el insumo y su cantidad correspondiente.

Al finalizar la jornada, la cajera en compañía del jefe de turno realizan el arqueo de caja, donde calculan el total obtenido durante la jornada gracias a las hojas de pedido (presentada anteriormente), donde además separan los ingresos entre el monto inicial en caja, monto generado en efectivo, monto generado por pago con tarjeta, el monto generado de las cuentas de los clientes y se calcula el total del día. Finalmente, el jefe de turno registra en un libro de ventas las compras del día (si es que hubo alguna) y el total de ventas realizadas en el día (la suma del total de ambos turnos). A continuación, en la figura 1-3 se muestra un ejemplo del registro de ventas y compras.

ABRIL 2017			Costo	venta	Costo
01/4/17	Pan completo		4200		
	Pan churrosco		4800		
	Pan masa		1700		
	CCU		31000		
	Detergente		3000		
	Mayonesa		13400		
			44200	121200	165400
2/4/17	ago.				41800
3/4/17 Lunas	pan dulce		4380		
	pan churrosco 16		3500		
	pan masa		1800		
	Perdurar		1500		
	Wewerla		118300		
	traste raga 12000 Lechuga chola		18000		
			147480	17520	165000
4/4/17 marzo	puera		1500		208200
	Huevos 2 Boleyns		6400		
	Agro Super canda		25200		
	1 Saco papa		8000		
	GAS		13500		
	pan completo		2800		
	" masa		36000		
	Super		93400	114800	208200
5/4/17 marzo	CCU		54600		184400
	pan completo		4940		
	" churrosco		5000		
	papas		8000		
	pan masa + 1000		2800		
			75340	109660	184400

Fuente: Lucy Riova T., Jefa del Negocio.
Figura. 1-3 – Página de libro de ventas.

1.2.1 Diagrama de flujo administrativo

1.2.1.1 D.F.A del proceso “Venta”:

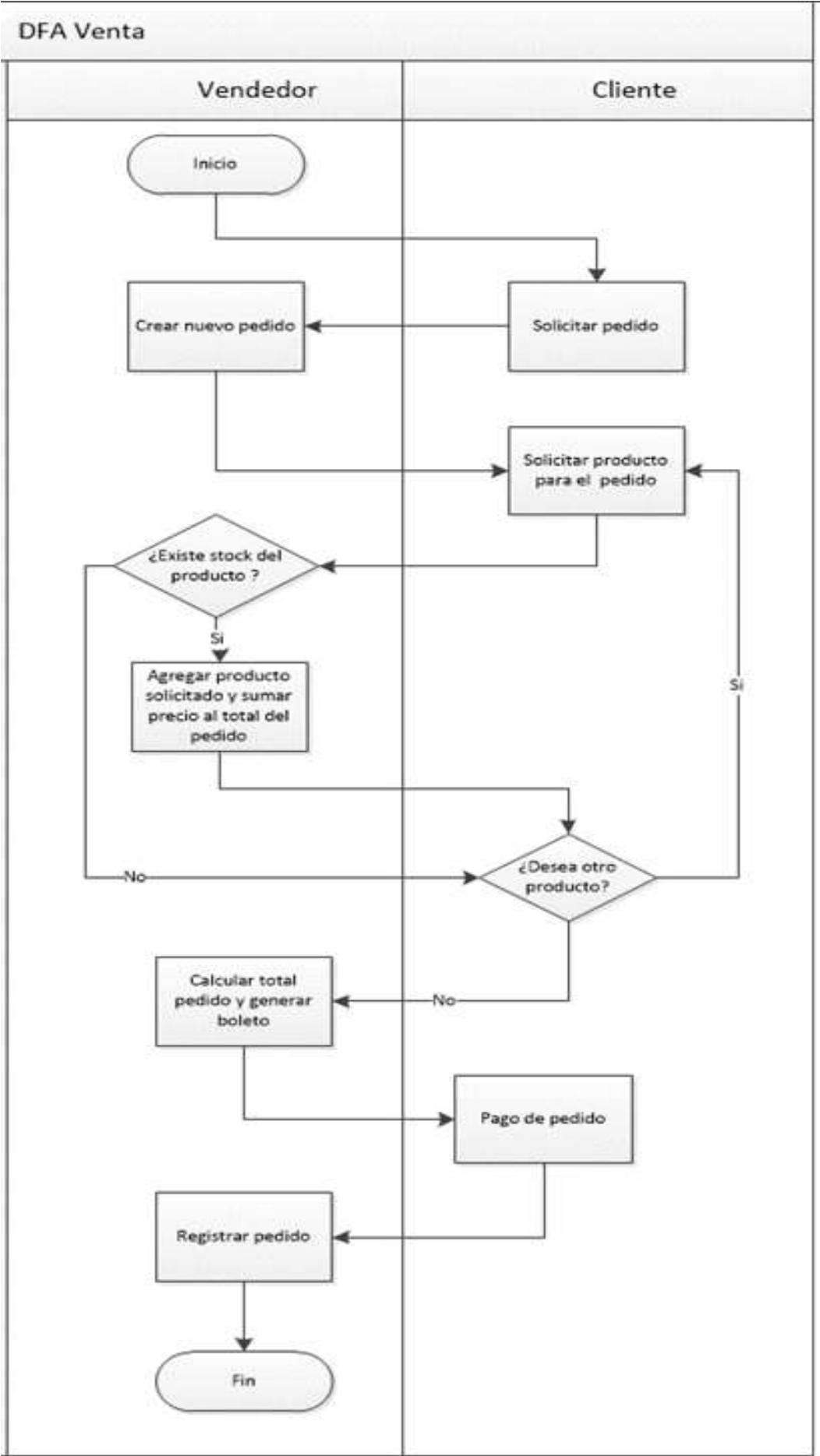


Diagrama. 1-1 – Diagrama de flujo administrativo del proceso “Venta”.

1.2.1.2 D.F.A del proceso “Comprar insumos”:

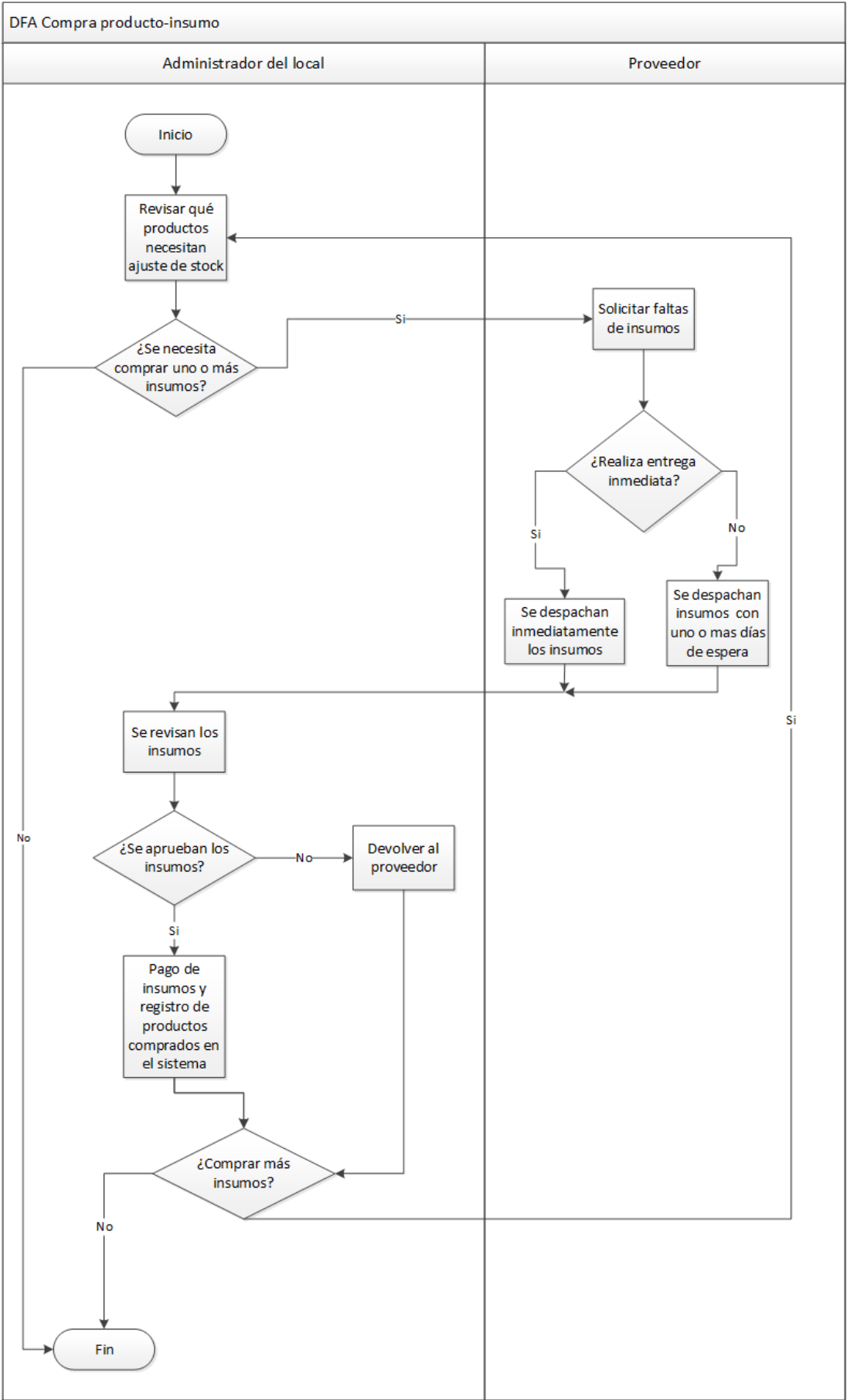


Diagrama. 1-2 - Diagrama de flujo administrativo del proceso “Compra”.

1.3 PROBLEMAS DETECTADOS

En resumen, los problemas apreciables dentro de la organización apuntan principalmente a la incomodidad, “esfuerzo humano” y al excesivo tiempo requerido para efectuar las labores, dada la existencia de un sistema manual. Todos estos problemas serán descritos a continuación:

- Carencia de un control de stock real: Provoca que el usuario deba revisar constantemente en “bodega” el stock disponible, y determinar si es necesario solicitar insumos a proveedores.
- Desconocimiento de stock crítico: En conjunto al problema anterior, no se puede determinar qué producto se encuentra con poca existencia.
- Errores al anotar pedido: Como los pedidos son anotados manualmente, se puede dificultar entender la letra, provocando errores al calcular el total de la cuenta y, además, el cocinero prepararía el producto erróneo, generando pérdida de insumos en merma.
- Demora en arqueo de caja: Pérdida excesiva de tiempo al calcular y comprobar el total de ventas realizadas en la jornada.
- Problemas al agregar mas productos a un pedido: Cuando un cliente ha ordenado un pedido para servir, ya lo ha recibido y desea agregar más productos a dicho pedido; ocasionalmente se genera un problema al momento de registrarlos. Debido a que el registro de pedidos de distintas mesas se realiza sucesivamente uno después de otro en una libreta, provoca que el listado total de productos correspondientes sea desorganizado. Como se aprecia en la figura 1-2, donde los pedidos anotados se encuentran separados con un espacio en blanco.
- Demora en calcular el total de boleta: Ligera pérdida de tiempo al calcular el total de la boleta, además, en conjunto al problema anterior, implica demorar más tiempo en obtener el cálculo.
- Lentitud al calcular el total de la deuda para un cliente frecuente.
- Dificultades al calcular las ventas mensuales realizadas.
- No se posee un registro de todas la mermas y/o pérdidas producidas.
- No se posee un registro de todas las donaciones realizadas al establecimiento.

1.4 DESCRIPCIÓN DEL SISTEMA PROPUESTO

La solución consiste en un sistema, que será capaz de registrar las ventas de los productos, registrar el control de stock, registrar las compras de productos, registrar a los proveedores del local y los clientes frecuentes (aquellos que poseen una cuenta personal). Es importante explicar la clasificación y diferencia entre “insumos” y “productos” que el sistema implementará. El término “insumo” corresponderá a las partes que conforman un “producto”, es decir, por ejemplo palta, tomate, vienesa, son “insumos” que conforman un producto denominado “completo italiano”.

1.4.1 Objetivos y beneficios del sistema propuesto

1.4.1.1 Objetivo General

Construir un sistema que sea capaz de automatizar las labores de compra y venta de productos y control de stock dentro del establecimiento busandwichería.

1.4.1.2 Objetivos específicos

- Registrar las ventas de productos en el local.
- Crear informes del listado de las ventas de los productos, los cuales estarán ordenados por el código de los productos vendidos. Además dichos informes podrán ser filtrados por un rango de fechas específico.
- Crear informes del listado del stock de los productos/insumos, los cuales estarán ordenados por el código individual del producto. Además se permitirá realizar búsquedas según código, nombre o categoría según corresponda.
- Crear informes del listado de los proveedores, los cuales estarán ordenados de menor a mayor según el Rut del proveedor. Además de poseer un registro histórico de cada compra realizada a un proveedor en específico.
- Poseer un registro de los usuarios del sistema, con un grado de acceso de cada uno, “Común” para el cajero, “Administrador” para el jefe y “Maestro” para los usuarios de mantención del sistema.
- Poseer un registro de las mesas del local.
- Representar el “estado” del pedido.
- Almacenar un registro de los clientes frecuentes y de confianza al establecimiento, los cuales disponen de una “cuenta” que permite almacenar el monto de una deuda a pagar en un futuro cercano (cada fin de mes o quincena, generalmente).
- Realizar un control de clientes frecuentes y sus respectivos pagos.
- Registrar cada aprovisionamiento realizado al establecimiento, indicando el proveedor y los insumos adquiridos, además, ingresando automáticamente estos últimos al stock del sistema. Considerando además, los insumos adquiridos mediante donaciones realizadas al establecimiento.
- Manejar un registro de los insumos adquiridos y los productos ofrecidos en el local.
- Apoyar y facilitar el proceso de arqueo de caja.
- Realizar un registro de cada merma producida.

1.4.1.3 Beneficios

- Se contará con un control de stock real y cómodo para el usuario.
- Disminución de errores al anotar un pedido.
- Rapidez en calcular arqueo caja.

- Disminución considerable del margen de error con el orden de pedidos.
- Rapidez y orden al calcular el total de boleta.
- Facilitar ampliamente, la labor de calcular total de las ventas semanales, mensuales o anuales, así como el total de ganancias y pérdidas de dinero que se efectúen en el local.
- Mayor rapidez y comodidad al calcular y registrar los pagos correspondientes a deudas de los clientes.
- Conocimiento de toda la entrada y salida de insumos, estas sean por compras, donaciones, ventas y/o pérdidas.
- Tener conocimiento del historial de pago de un cliente frecuente.
- Tener conocimiento del historial de compras a un proveedor específico.
- Generar una mejor imagen hacia sus clientes.

1.4.2 Descripción general de la solución propuesta

La solución que se implementará para solucionar todos los problemas anteriormente mencionados, producto de una falta de documentación y datos, es proporcionar un sistema informático con una base de datos, en la cual registrar todos los datos necesarios de los distintos productos e insumos que existen en el establecimiento.

Dicha solución consiste en instalar un sistema en un solo equipo stand-alone, el cual optimizará las labores del local asociadas al control de stock y la compra y venta de productos, automatizando labores que ocupan mucho tiempo, tales como, generar listados de productos, proveedores, compras y ventas o calcular el total de ventas de la jornada. Labores que un software realiza en cuestión de segundos. Además de que dicho sistema debe de apoyar al registro histórico de ventas y compras del local.

El sistema podrá ser operado por distintos usuarios, los cuales tendrán acceso a las funciones dependiendo de su nivel de usuario. Los distintos niveles de usuarios se dividen entre:

- Usuario Maestro: Posee acceso completo al sistema, tiene la capacidad de agregar, modificar y eliminar usuarios de cualquier nivel; es usado como usuario de respaldo para realizar mantención al sistema.
- Usuario Administrador: Al igual que el usuario “Maestro” posee acceso completo a las funciones sistema, así como agregar y eliminar usuarios (excepto a usuarios de nivel “Maestro”). Este nivel de usuario es asignado solo a los posibles administradores del local.
- Usuario Común: Posee el menor acceso, solo posee acceso al módulo de ventas. Este tipo de usuario es asignado a los cajeros del establecimiento.

Para que una venta sea realizada mediante el sistema, el Usuario Común (cajero) recibirá el listado de productos solicitados por el cliente por medio del garzón por escrito. En caso de un pedido para llevar, el cliente podrá acercarse directamente al cajero para solicitar su orden. El cajero digita el

listado de productos en el sistema, y éste último se encarga de calcular automáticamente el valor total del pedido y una vez pagado el monto, lo registra y finaliza el pedido.

1.4.3 Funcionalidades

- **Mantenedor de proveedores:** Esta funcionalidad permite mantener actualizados los datos de los proveedores del local. Permite el ingreso, modificación y eliminación de los datos de un proveedor. Esta funcionalidad sólo está permitida a usuarios de acceso “Maestro” y/o “Administrador”.
- **Mantenedor de productos:** Esta funcionalidad permite mantener actualizados el registro de productos asociados al local. Permite el ingreso, modificación y eliminación de los datos de un producto, esto incluye el listado de insumos utilizados para su confección. Esta funcionalidad sólo está permitida a un usuario de acceso maestro.
- **Mantenedor insumos:** Esta funcionalidad permite mantener actualizados el registro de insumos asociados al local. Permite el ingreso, modificación y eliminación de los datos de un insumo determinado. Esta funcionalidad sólo está permitida a un usuario de acceso maestro.
- **Mantenedor de categorías:** Esta funcionalidad permite mantener actualizados el registro de las categorías asociados al local. Permite el ingreso, modificación y eliminación de los datos de una categoría determinada. Esta funcionalidad sólo está permitida a un usuario de acceso maestro.
- **Mantenedor de mesas:** Mantenedor cuya función se basa en modificar el número de mesas del local, ingresando una o más mesas o, eliminando una o más mesas.
- **Mantenedor de clientes frecuentes:** Esta funcionalidad permite mantener actualizados el registro de los clientes frecuentes asociados al local. Permite el ingreso, modificación y eliminación de los datos de un cliente frecuente del local. Esta funcionalidad sólo está permitida a un usuario de acceso maestro.
- **Mantenedor de usuarios:** Permite modificar el nivel de acceso de un usuario, además de crear o eliminar un usuario. Esta funcionalidad sólo está permitida a un usuario de acceso maestro.
- **Menú de pedidos:** Funcionalidad que consiste en desplegar en pantalla la lista de productos, para así realizar una venta de algún producto a algún cliente. Esta funcionalidad está permitida para un “Usuario Común”.
- **Registro compra:** Funcionalidad que permite registrar las compras realizadas a proveedores.
- **Control-stock:** Esta función se caracteriza en actualizar automáticamente los cambios realizados en el stock; esto hace referencia a, por ejemplo, cuando se realiza la compra de un determinado insumo, la cantidad adquirida se añade de manera automática al registro de stock; y a su vez, cuando se realiza una venta de ciertos productos, se reduce la cantidad de insumos utilizados en la preparación de estos productos automáticamente del stock actual, según corresponda.

- Ajuste Stock: Permite modificar el stock manualmente, con la finalidad de registrar donaciones que se hagan al local o, pérdidas producidas por “mermas”.
- Generar listado de proveedores: Funcionalidad que permite crear listados ordenados de los proveedores del local. El criterio para ordenar los proveedores es según el Rut de menor a mayor.
- Generar listado de ventas: Funcionalidad que permite crear listados ordenados y/o filtrados de las ventas del local. El criterio para filtrar el listado sería por una fecha específica.
- Generar listado de compras: Funcionalidad que permite crear listados ordenados y/o filtrados de las compras del local. El criterio para filtrar el listado sería por una fecha específica.
- Generar listado de insumos: Funcionalidad que permite crear listados ordenados y/o filtrados de los insumos del local. El criterio de búsqueda permite filtrar los insumos según su código, nombre, tipo de insumo y categoría.
- Generar listado de productos: Funcionalidad que permite crear listados ordenados y/o filtrados de los productos del local. El criterio de búsqueda permite filtrar los productos según su código, y nombre.
- Generar listado de clientes frecuentes: Funcionalidad que permite crear listados ordenados y/o filtrados de los clientes frecuentes del local.
- Generar listado de usuarios: Funcionalidad que permite crear listados ordenados y/o filtrados de los usuarios del sistema.
- Generar listado de mesas: Funcionalidad que permite crear listados ordenados de las mesas del local.
- Conexión (Login): Permite a un usuario identificarse en el sistema ingresando su código de usuario y contraseña.
- Registrar pago de deuda: Función que registra el pago del monto cancelado por el cliente. Además, descuenta el monto cancelado del monto total de la deuda.
- Registrar venta: Esta función se encarga de, al momento de finalizar un pedido, registrar una copia de éste y su detalle en la base de datos. Además, según la fecha, se almacena en un registro las ventas totales por jornada;
- Registrar Deuda: Función que permite adicionar el total de un pedido al monto actual registrado de un cliente frecuente.

1.4.4 Descripción de fórmulas y cálculos realizados

Los cálculos a realizar serán expresados dependiendo del proceso que correspondan, y se detallan las entidades que los conforman.

- Generar informe de ventas turno:
Se usará la siguiente fórmula:
 - Monto efectivo: Corresponde al total de ventas pagadas en efectivo durante la jornada.

- Monto tarjeta: Corresponde al total de ventas pagadas vía tarjeta durante la jornada.
- Monto crédito: Corresponde al total de ventas pagadas registradas a crédito durante la jornada.
- Total efectivo en caja: Corresponde al dinero total existente en caja; éste es la suma entre el monto inicial de caja y la ganancia en efectivo de la jornada.

1.4.5 Información a manejar

Entradas del sistema: Todas las entradas del sistema son por teclado.

- Datos de los proveedores: Se ingresan los datos correspondientes del proveedor, tales como: rut, nombre, apellidos, dirección, fono y e-mail.
- Datos de compra: Se ingresan los datos correspondientes a las compras de insumos/productos hechas a proveedores, tales como rut proveedor y, código, precio y cantidad del objeto adquirido.
- Datos de los productos: Se ingresan los datos correspondientes del producto, tales como nombre y precio. El código del producto lo genera el usuario administrador.
- Datos de los insumos: Se ingresan los datos correspondientes del insumo, tales como código, nombre, descripción, tipo de insumo, unidad de medida, stock actual, stock crítico y categoría.
- Datos de las categorías: Se ingresan los datos correspondientes de la categoría, tal como su nombre y código.
- Datos de las ventas: Al realizar un pedido, se ingresarán los datos del pedido en el sistema, los cuales son tipo de pedido y, la cantidad y precio de los productos. El número de pedido lo genera automáticamente el sistema.
- Datos de las mesas: Al momento de registrar cada mesa, es necesario ingresar además la capacidad de cada una de éstas.
- Datos de usuario: Se ingresan los datos de uno o más usuarios los cuales son: id de usuario, contraseña del usuario y el nivel de acceso de dicho usuario.
- Datos de inicio de sesión: Al iniciar el sistema, el usuario deberá ingresar al sistema ingresando su nombre de usuario y contraseña.
- Datos de clientes frecuentes: Se ingresan los datos de uno o más clientes, tales como: rut, nombre, apellido paterno, apellido materno y fono que lo identifican.

Salidas: Todas las salidas serán por pantalla.

- Listado del stock de los insumos: Se muestra en pantalla un listado del stock de insumos. Este listado estará ordenado según el código del elemento correspondiente. Además dicho listado poseerá búsquedas filtradas por: código, nombre, y categoría.

- Listado del stock de los productos: Se muestra por pantalla un listado de todos los productos registrados, junto a su stock virtual (cantidad máxima que se puede elaborar dicho producto dependiendo del stock de insumos que lo confoman).
- Listado de productos: Muestra el listado de los productos registrados en el sistema. Además dicho listado poseerán búsquedas filtradas por: código, nombre, y precio.
- Listado de detalle del producto: Se muestra el listado de cada insumo necesario para conformar un producto específico.
- Listado de los proveedores: Se muestra por pantalla un listado de los proveedores del local. Este listado estará ordenado según el Rut del proveedor.
- Listado de detalle del pedido: Presenta un listado de todos los productos asociados y registrados a un pedido en particular.
- Listado de las categorías: Se muestra en pantalla un listado de las categorías de los productos del local. Este listado estará ordenado de menor a mayor según el código.
- Listado de usuarios: Se muestra un listado de los usuarios del local. Este listado estará ordenado alfabéticamente según el “Id” del usuario.
- Listado de detalle de compras: Se muestra un listado de los insumos adquiridos y registrados a una compra en particular.
- Listado de mesas: Presenta un listado de todas las mesas registradas, así como la capacidad y el estado de disponibilidad de cada una de éstas.
- Listado de ventas: Presenta un listado de todas las ventas producidas en un período determinado.
- Listado de compras: Presenta un listado de todas las compras realizadas en un período determinado o a un proveedor en específico.
- Informe de las ventas: Este informe detalla el total de dinero obtenido por venta de productos.
- Listado de clientes frecuentes: Se muestra un listado ordenado por el “Rut” de todos los clientes frecuentes registrados en el sistema junto al total de su deuda.
- Informe de pérdidas/donaciones: Presenta un listado indicando todos los productos o insumos que fueron mermados y/o donados.
- Listado de pagos: Presenta un listado con la fecha y el monto de cada pago realizado por un cliente frecuente.
- Listado de crédito: Presenta un listado con la fecha, monto y el número de cada pedido realizado por un cliente frecuente.

Entidades de información.

- Proveedores: Esta entidad contiene datos acerca de los proveedores del local y datos que ayudarán al dueño del local contactarse con el proveedor (tales como dirección, fono y e-mail del proveedor).

- Registro-compra: Entidad que contiene el registro de cada compra realizada a uno de los distribuidores del establecimiento. Ésta cuenta con el código identificador de la compra, el código del proveedor a quién se realizó la compra y la fecha en la que se realizó.
- Detalle-registro-compra: Entidad que almacena el detalle de los productos adquiridos en el registro de las compras, y a su vez, la cantidad de éstos.
- Insumos: Esta entidad contiene los datos de las materias primas necesarias para conformar un producto, las cuales poseen nombre, stock actual y stock crítico.
- Producto: Entidad que almacena los datos de los productos que el local ofrece a la venta; es decir, el nombre y precio de éstos.
- Detalle-producto: Esta entidad almacena el detalle las materias primas de las que son necesarias para preparar un “producto” y la cantidad requerida.
- Categoría: Entidad que almacena los datos de las categorías que permiten otorgar clasificación a los productos e insumos.
- Pedidos: Esta entidad contiene los datos asociados a un pedido, es decir, el monto total del pedido de un cliente, ya sea para servir o llevar. Además, contiene el medio de pago, estado del pedido, descuento si es que corresponde y en qué jornada se realiza el pedido.
- Detalle-pedido: Entidad que contiene la descripción de la venta de un producto específico de un pedido, indicando la cantidad y precio que se vendió de dicho producto.
- Mesas: Esta entidad contiene los datos de las mesas del local, los cuales son, el número de la mesa, la capacidad de éstas, su estado de disponibilidad y además, el numero del pedido asociado.
- Llevar: Entidad que contiene los datos que representan el número y el estado de los pedidos para llevar del local.
- Deuda cliente: Entidad que almacena el monto de un pedido específico para un cliente frecuente; almacenado según la fecha.
- Cliente frecuente: Entidad que almacena el nombre, fono y el total de la deuda de un cliente de confianza, identificándolo según un código único.
- Registro-pago-deuda: Esta entidad contiene como dato cada monto pagado por un cliente frecuente para cancelar su deuda, sea parcial o totalmente.
- Usuario: Entidad que almacena los usuarios anexados al sistema y que trabajarán con éste. Se distinguen tres tipos de usuarios: Maestro, Administrador y Común.
- Ajuste_stock: Contiene los datos relacionados con el ajustes de stock, es decir, mermas o donaciones de productos.

1.4.6 Modelo de datos

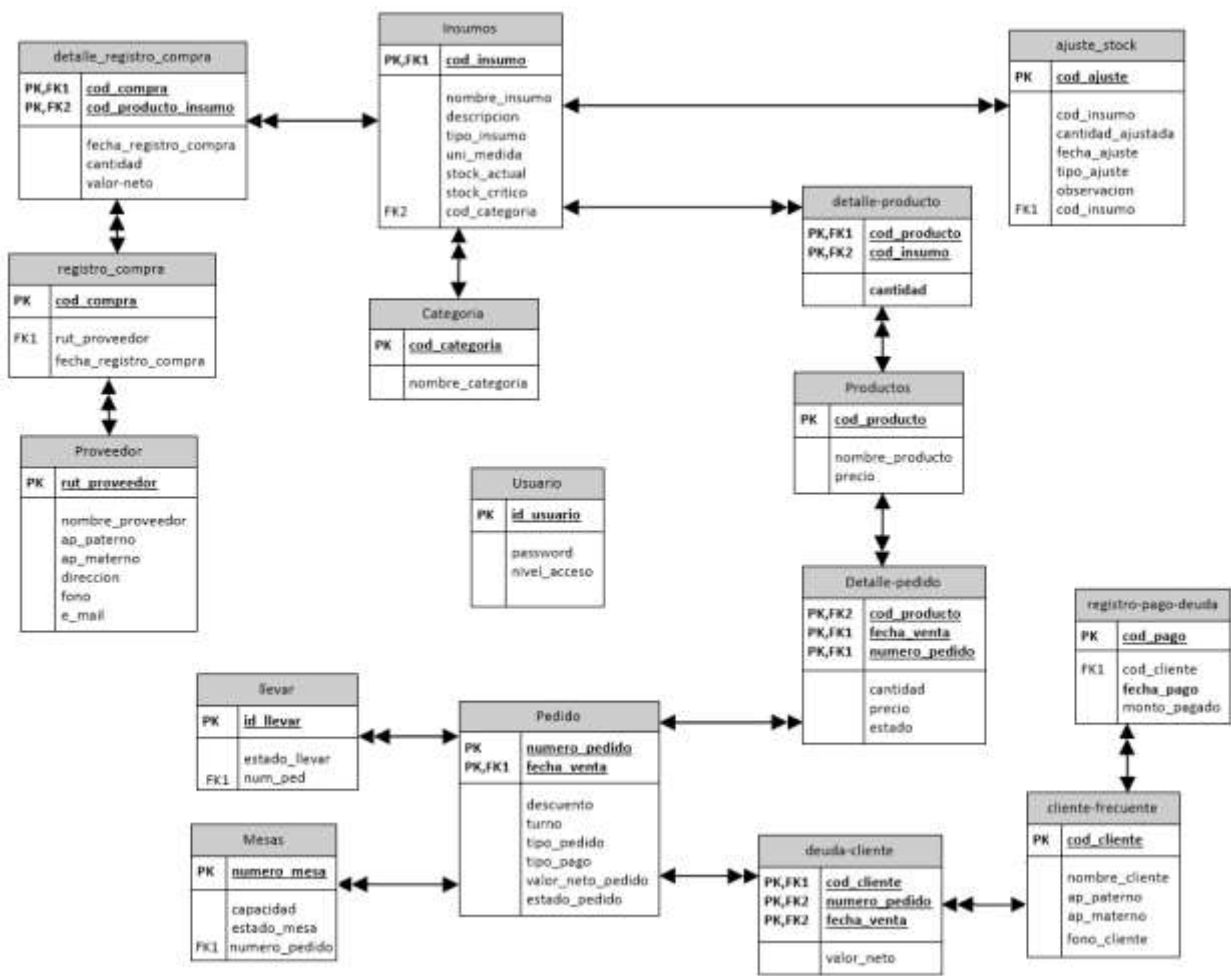


Figura. 1-4 – Modelo de datos.

1.4.7 Estructura de código

Para este sistema, se utilizarán tres tipos de códigos diferentes para identificar las diferentes entidades, los cuales serán detallados a continuación:

- **Numéricos Correlativos:** Como bien lo indica el nombre, estos códigos constarán de números sucesivos. Dentro de los atributos que serán utilizados por las entidades como claves irrepetibles se encuentran “cod_compra”, “cod_ajuste”, “numero_mesa”, “id_llevar”, “cod_pago” y “numero_pedido”. Cabe destacar el caso del atributo “numero_pedido”, el cual, a diferencia de los anteriores si posee números repetidos entre si. Esto se debe a que tras cada día, el valor de los pedidos es reiniciado, es decir, vuelve al valor “0”; es por esto que las entidades que utilizan dicho atributo, hacen uso además de “fecha_venta” como clave compuesta.
- **Numéricos:** Estos códigos corresponden a valores numéricos que representan una entidad o cualidad. Dentro de los atributos que serán utilizados por las entidades como claves irrepetibles se encuentran, “rut_proveedor” y “rut_cliente”, almacenando el rut de la persona correspondiente y, “cod_categoria”, “cod_insumo” y “cod_producto” como claves numéricas

representativas. Por otro lado existen atributos que no serán utilizados como claves, tales como “nivel” y “estado_llevar”.

- **Carácter:** Los códigos que pertenecen a esta categoría son: “tipo_insumo”, “uni_medida”, “estado”, “tipo_pedido”, “tipo_pago”, “estado_pedido”, “estado_mesa” y “tipo_ajuste”. Estos códigos corresponden a un campo de largo uno que almacena la letra inicial de lo que representan, por ejemplo “estado_pedido” con los valores “N” (Nuevo), “A” (Activo), “P” (Pagado) y “F” (Finalizado). Existe solamente un caso de un campo de este tipo que es utilizado como clave, cuyo caso es “turno”, donde sus valores son “D” (proveniente de “Diurno”) y “V” (proveniente de “Vespertino”).
- **Alfanumérico:** Según indica el nombre, corresponden a códigos compuestos por cadena de caracteres y/o números. El único caso corresponde al código “id_usuario”, esto se debe a que otorga la libertad al usuario de crear su identificación de acceso.

1.4.8 Condicionantes del diseño

Dado que, idealmente se busca disminuir el costo del proyecto para el usuario, se planea utilizar un software gratuito o, en un peor caso, de bajo costo.

CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE
ARCHIVOS

2. MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

2.1 DESCRIPCIÓN DEL RECURSO COMPUTACIONAL

A continuación se detallarán los aspectos de hardware y software que se utilizarán para diseñar y modelar el sistema. Cabe destacar que, como ya se ha mencionado, el proceso actual se desarrolla de manera completamente manual, por lo que los recursos de hardware y software a utilizar nacen del análisis de las herramientas idóneas para el negocio, las cuales buscan principalmente evitar gastos adicionales, de manera que la puesta en marcha del sistema sea accesible económicamente.

Para la configuración del sistema se detallarán las características más importantes del recurso de hardware a adquirir para la puesta en marcha del sistema.

2.1.1 Configuración del sistema

El sistema trabaja con una única estación de trabajo, la cual se encuentra ubicada en el sector de la caja. A continuación, se describirán las características del equipo y los periféricos a utilizar:

- o Estación de trabajo
 - Procesador : Intel® Celeron® N3050
 - Disco Duro : 500 GB SATA III (7200 RPM)
 - Memoria : 4GB DDR3L 1600 MHz PC3-12800
- o Periféricos
 - Pantallas : El sistema trabaja haciendo uso de dos monitores, uno establecido en la estación de trabajo (donde se ubica el cajero) y un anexo extendido hasta la cocina desde el equipo.
 - Monitor Samsung 24"
 - Monitor Samsung 14"

2.1.2 Software utilizado

El recurso de software a utilizar pretende, tal como se ha mencionado, evitar en mayor medida posible que el cliente incurra en gastos adicionales, por concepto de licencias. También se requiere de un sistema de trabajo, el cual el usuario pueda aprender a manejar fácilmente.

- MariaDB: MariaDB es uno de los Sistemas Gestores de bases de Datos (SQL) más populares desarrollados bajo la licencia de público general (GPL, General Public License). Este software lo desarrolla y mantiene la empresa MariaDB, pero puede utilizarse gratuitamente y su código fuente está disponible para las principales plataformas de mercado, MySQL se compone básicamente de un programa que actúa como servidor y atiende las peticiones de los clientes.
- Windows 7: Este sistema alojará el software cliente, que es la interfaz de usuario del sistema. Se escogió Windows 7 puesto que, a pesar de que no es la versión más reciente del sistema

operativo de Microsoft, ofrece una interfaz simple de utilizar y posee un bajo uso de recursos.

2.1.3 Herramientas de desarrollo

- NetBeans IDE: Es un entorno de desarrollo gratuito, de código abierto y profesional. Ofrece una plataforma que permite optimizar programas e intenta que se ejecuten más rápido y con el mínimo uso de memoria.

2.2 DESCRIPCIÓN DE ARCHIVOS

2.2.1 Proveedor

- Nombre de la tabla: Proveedores.
- Descripción: Almacena los datos de los proveedores asociados al local.
- Clave primaria: Cod_proveedor.
- Clave foránea: No posee.

Nombre	Tipo	Largo	Descripción
rut_proveedor	Integer	10	Código que identifica de forma única al proveedor.
nombre_proveedor	Varchar	20	Nombre del proveedor.
ap_materno	Varchar	15	Apellido paterno del proveedor.
ap_materno	Varchar	15	Apellido materno del proveedor.
Dirección	Varchar	60	Dirección del proveedor.
Fono	Varchar	12	Número telefónico o de celular para contactar con el proveedor.
Email	Varchar	30	Dirección de correo electrónico del proveedor.

Tabla. 2-1 – Tabla proveedor

2.2.2 Registro-compra

- Nombre de la tabla: Registro-compra.
- Descripción: Almacena datos de las compras realizadas en el local, asociando la compra con el proveedor correspondiente.
- Clave primaria: cod_compra.
- Clave foránea: rut_proveedor (referencia a archivo Proveedor).

Nombre	Tipo	Largo	Descripción
cod_compra	Integer	6	Código que identifica de forma única la compra.
rut_proveedor	Integer	10	Código que identifica al proveedor asociado a la compra.
fecha_registro_compra	Date	10	Fecha en que se realizó la compra.

Tabla. 2-2 – Tabla Reg-compra

2.2.3 Detalle- Registro-Compra

- Nombre de la tabla: Registro-Compra-Insumos.
- Descripción: Contiene los datos de un insumo asociado a una compra.
- Clave primaria: Cod_producto_insumo + cod_compra.
- Clave foránea: Cod_compra (referencia a archivo Compra), Cod_producto_insumo (referencia a archivo Insumos).

Nombre	Tipo	Largo	Descripción
cod_producto_insumo	Integer	6	Identifica al insumo que se adquirió en una compra específica.
cod_compra	Integer	6	Identifica a la compra asociada.
fecha_registro_compra	Date	10	Fecha en la cual se realizó la compra.
cantidad	Short Int	4	Cantidad de insumo.
valor_neto	Integer	6	Valor total de un insumo específico.

Tabla. 2-3 – Tabla Detalle-Reg-Compra

2.2.4 Insumos

- Nombre de la tabla: Insumos.
- Descripción: Contiene datos de los ingredientes que se utilizan como materia prima para la elaboración de productos.
- Clave primaria: cod_insumo.
- Clave foránea: cod_categoria (referencia a archivo Categoría).

Nombre	Tipo	Largo	Descripción
cod_insumo	Integer	6	Código que identifica de forma única al insumo.
nombre_insumo	Varchar	30	Nombre del insumo.
descripcion	Varchar	40	Detalla una breve descripción del insumo.
tipo_insumo	Char	1	Indica si el insumo es una materia prima o un producto ya manufacturado. Los valores posibles son: • M = Materia Prima. • P = Producto.
uni_medida	Varchar	2	Indica la unidad de medida del insumo, por ejemplo: kilogramos, unidad o litros. • Un =Unidad. • Kl = Kilo. • Lt = Litro.
stock_actual	Short	4	Indica la cantidad de stock-actual que posee el insumo.
stock_crítico	Byte	2	Indica el valor mínimo que debe poseer el campo “stock-actual” para ser considerado como “stock-critico”.
cod_categoria	tinyint	2	Representa la categoría asociada a este producto. Los valores de este campo dependen de los datos almacenados en la entidad “Categoría”, los valores por establecidos por defectos son los siguientes: • 1 = Vegetales. • 2 = Carnes. • 3 = Fiambreria. • 4 = Lacteos. • 5 = Bebidas. • 6 = Abarrotes. • 7 = Masas.

Tabla. 2-4 – Tabla Insumos

2.2.5 Producto

- Nombre de la tabla: Tabla producto.
- Descripción: Contiene los datos de cada producto que vende el local.
- Clave primaria: cod-producto.
- Clave foránea: No posee.

Nombre	Tipo	Largo	Descripción
cod_producto	Integer	6	Código que identifica de forma única el producto.
nombre_producto	Varchar	30	Nombre del producto.
Precio	Short	5	Precio en el que se vende el producto en el local.

Tabla. 2-5 – Tabla Producto

2.2.6 Detalle-producto

- Nombre de la tabla: Detalle-Producto.
- Descripción: Contiene los datos de la cantidad de insumos necesarios para elaborar un producto.
- Clave primaria: cod_producto + cod_insumo.
- Clave foránea: cod_producto (referencia a archivo Productos), cod_insumo (referencia a archivo Insumos).

Nombre	Tipo	Largo	Descripción
cod_producto	Integer	6	Código que identifica al producto al cual se le asocian uno o más insumos.
cod_insumo	Integer	6	Código que identifica al insumo que está asociado a un producto.
Cantidad	Float	4	Cantidad de insumos necesarios para elaborar un producto.

Tabla. 2-6 – Tabla Detalle-producto

2.2.7 Categoría

- Nombre de la tabla: Categoría.
- Descripción: Almacena los datos de las categorías en las que se clasifican los insumos.
- Clave primaria: cod_categoria.
- Clave foránea: No posee.

Nombre	Tipo	Largo	Descripción
cod_categoria	Tinyint	2	Código que identifica de forma única la categoría.
nombre_categoria	Varchar	20	Nombre de la categoría.

Tabla. 2-7 – Tabla Categoría

2.2.8 Detalle-pedido

- Nombre de la tabla: detalle_pedido.
- Descripción: Contiene los datos de los productos específicos que contiene un pedido.
- Clave primaria: Cod_producto + cod_pedido + fecha_venta.
- Clave foránea: Cod_producto (referencia archivo Productos), cod_pedido (referencia a archivo Pedido).

Nombre	Tipo	Largo	Descripción
cod_producto	Integer	6	Código que identifica el producto asociado al pedido.
numero_pedido	Short	3	Código que identifica el pedido al cual se le asocian uno o más productos.
fecha_venta	Date	10	Fecha en la cual se realizó la venta.
cantidad	Short	4	Cantidad del producto solicitado.
Precio	Short	5	Precio en el que se vende el producto.
Estado	Char	1	Indica el estado del producto asociado al pedido. - “P”: Pendiente. - “R”: Registrado.

Tabla. 2-8 – Tabla Detalle-pedido

2.2.9 Pedido

- Nombre de la tabla: pedido.
- Descripción: Almacena datos de los pedidos realizados en el local.
- Clave primaria: numero_pedido + fecha_venta.
- Clave foránea: No posee.

Nombre	Tipo	Largo	Descripción
numero_pedido	Short	3	Código que junto a “fecha-venta” identifican de forma única el pedido.
fecha-venta	Date	10	Fecha que junto a “numero-pedido” identifican de forma única el pedido.
Descuento	Short	4	Descuento que se le hace al pedido.
Turno	Char	1	Turno en el que se realiza el pedido, representado por: • “D” = Diurno. • “V” = Vespertino.
tipo_pedido	Char	1	Tipo de pedido que se realiza, representado por: • “L” = Para llevar. • “S” = Para servir.
tipo_pago	Char	1	Tipo de pago del pedido, representado por: • “E” = Efectivo. • “C” = Crédito. • “T” = Tarjeta.
valor_neto_pedido	Integer	6	Valor total del pedido, el cual es la suma de todos los “detalle-pedido” asociados al pedido.
estado_pedido	Char	1	Estado del pedido, representado por: • “N” = Nuevo. • “A” = Activo. • “P” = Pagado. • “F” = Finalizado.

Tabla. 2-9 – Tabla Pedido

2.2.10 Mesas

- Nombre de la tabla: mesas.
- Descripción: Contiene un registro por cada mesa existente.
- Clave primaria: numero_mesa.
- Clave foránea: No posee.

Nombre	Tipo	Largo	Descripción
numero_mesa	Byte	2	Código que identifica de forma única la entidad mesa.
capacidad	Byte	2	Cantidad de personas que pueden ocupar la mesa al mismo tiempo.
estado_mesa	Char	1	Representa si la mesa está ocupada o no. • ‘O’ = Ocupado. • ‘D’= Desocupado.
numero_pedido	Short	3	Código que representa el pedido asociado.

Tabla. 2-10 – Tabla Mesas

2.2.11 Usuario

- Nombre de la tabla: usuario.
- Descripción: Contiene los datos de cada uno de los usuarios dentro del sistema.
- Clave primaria: Id_usuario.
- Clave foránea: No posee.

Nombre	Tipo	Largo	Descripción
id_usuario	Varchar	20	Código alfanumérico que identifica de forma única el usuario.
Password	Varchar	20	Contraseña encriptada que utilizará un usuario para ingresar al sistema.
nivel	Char	1	Nivel de acceso del usuario: • ‘0’ = Maestro. • ‘1’ = Administrador. • ‘2’ = Común.

Tabla. 2-11 – Tabla Usuario

2.2.12 Deuda-cliente

- Nombre de la tabla: deuda-cliente.
- Descripción: Contiene un registro por cada una de las deudas asociadas a un cliente frecuente.
- Clave primaria: rut_cliente + numero_pedido + fecha_venta.
- Clave foránea: rut_cliente (referencia a archivo Cliente frecuente), numero_pedido + fecha_venta (referencia a archivo Pedido).

Nombre	Tipo	Largo	Descripción
rut_cliente	Integer	10	Código que identifica el cliente asociado a la deuda.
numero_pedido	Short	3	Identifica, junto al campo “fecha-venta” el pedido asociado a la deuda.
fecha_venta	Date	10	Identifica, junto al campo “numero-pedido” el pedido asociado a la deuda.
valor_net	Integer	6	Valor registrado de la deuda total.

Tabla. 2-12 – Tabla Deuda-Cliente

2.2.13 Cliente-frecuente

- Nombre de la tabla: cliente_frecuente.
- Descripción: Contiene un registro de los clientes frecuentes del local, los cuales tienen permitido “fiar” en el establecimiento.
- Clave primaria: rut_cliente.
- Clave foránea: No posee.

Nombre	Tipo	Largo	Descripción
rut_cliente	Integer	10	Código que identifica de forma única el cliente frecuente.
nombre_cliente	Varchar	30	Nombre del cliente frecuente.
ap_paterno	Varchar	15	Apellido paterno del cliente.
ap_materno	Varchar	15	Apellido materno del cliente.
fono_cliente	Varchar	12	Número telefónico del cliente.

Tabla. 2-13 – Tabla Cliente-frecuente

2.2.14 Registro-pago-deuda

- Nombre de la tabla: registro_pago_deuda.
- Descripción: Contiene un registro de los pagos de una deuda de un “cliente-frecuente”.
- Clave primaria: cod_pago.
- Clave foránea: rut_cliente (referencia a archivo Cliente-frecuente).

Nombre	Tipo	Largo	Descripción
cod_pago	Integer	6	Código que identifica de forma única el el pago de una deuda.

fecha_pago	Date	10	Fecha en la que se realizó un pago.
rut_cliente	Integer	10	Código que asocia el pago a un cliente específico.
monto_pagado	Integer	6	Monto pagado de una deuda, el cual puede cubrir una parte de la deuda o bien la deuda completa.

Tabla. 2-14 – Registro-pago-deuda

2.2.15 Ajuste-stock

- Nombre de la tabla: Ajuste.
- Descripción: Contiene un registro de los ajustes de stock realizado.
- Clave primaria: cod_ajuste.
- Clave foránea: cod_insumo (referencia a archivo insumos).

Nombre	Tipo	Largo	Descripción
cod_ajuste	Integer	6	Código que identifica de forma única al ajuste.
cod_insumo	Integer	6	Código que asocia el ajuste stock a un producto o insumo según corresponda.
cantidad_ajustada	Integer	3	Cantidad de productos mermados o, que fueron regalados por proveedores.
fecha_ajuste	Date	10	Fecha en que se realizó el ajuste.
tipo_ajuste	Char	1	Tipo de ajuste que se realizará: • ‘M’= Merma. • ‘D’ = Donación.
Observación	Varchar	50	Observaciones acerca del ajuste del stock

Tabla. 2-15 – Tabla Ajuste-stock

2.2.16 Llevar

- Nombre de la tabla: Llevar.
- Descripción: Almacena datos por cada representación de un pedido para llevar en la interfaz de “Layout de Mesas”.
- Clave primaria: id_llevar.
- Clave foránea: numero-pedido (referencia a archivo Pedido).

Nombre	Tipo	Largo	Descripción
id_llevar	Tinyint	1	Código que identifica de forma única la entidad llevar.

estado_llevar	Bit	1	Representa el estado del pedido para llevar: • 0 = Entregado. • 1 = No Entregado.
num_ped	Short	3	Identifica al número de pedido asociado.

Tabla. 2-16 –Tabla Llevar

CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS

3. DESCRIPCIÓN DE PROGRAMAS

A continuación se muestra el diagrama modular y el de menús del sistema.

3.1 DIAGRAMA MODULAR

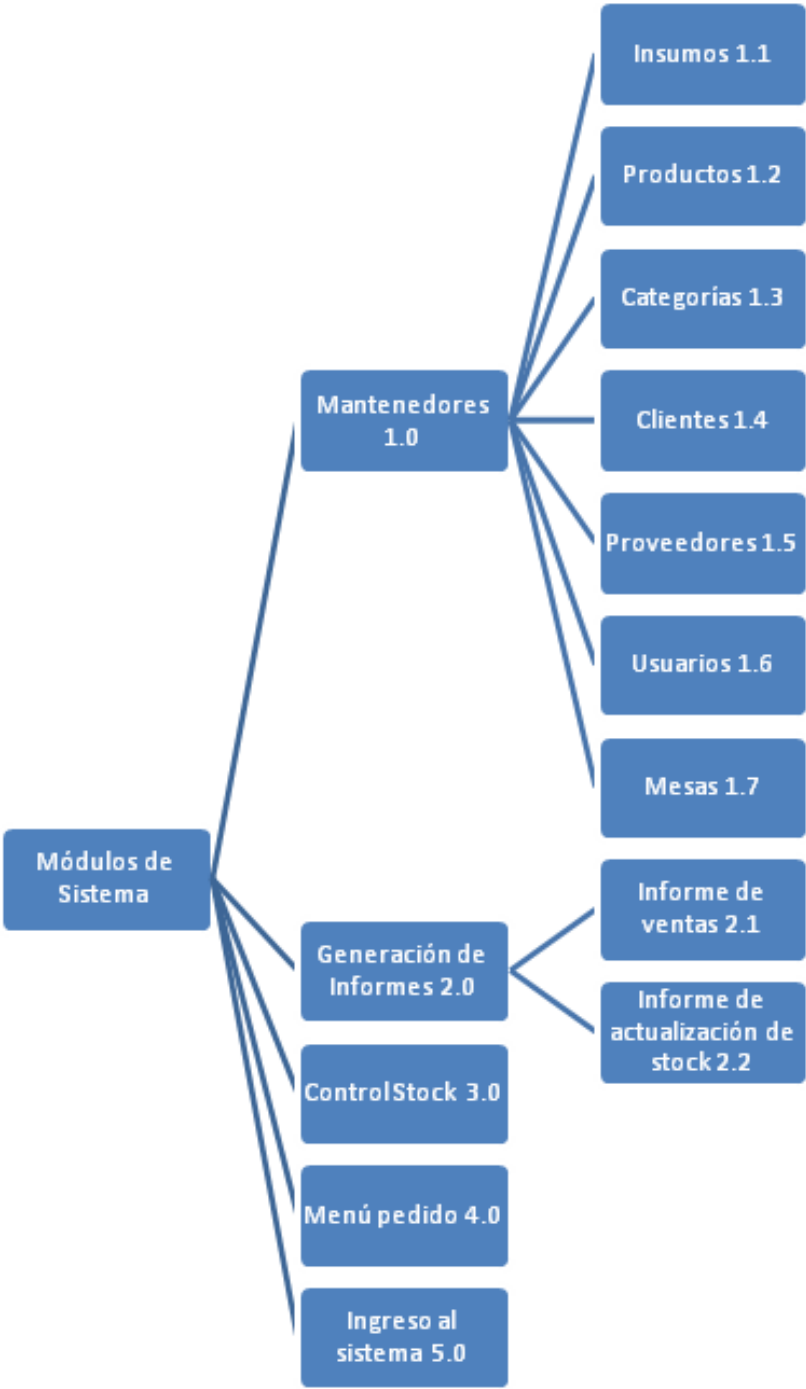


Diagrama. 3-1 – Diagrama Modular

3.2 DIAGRAMA DE MENÚS

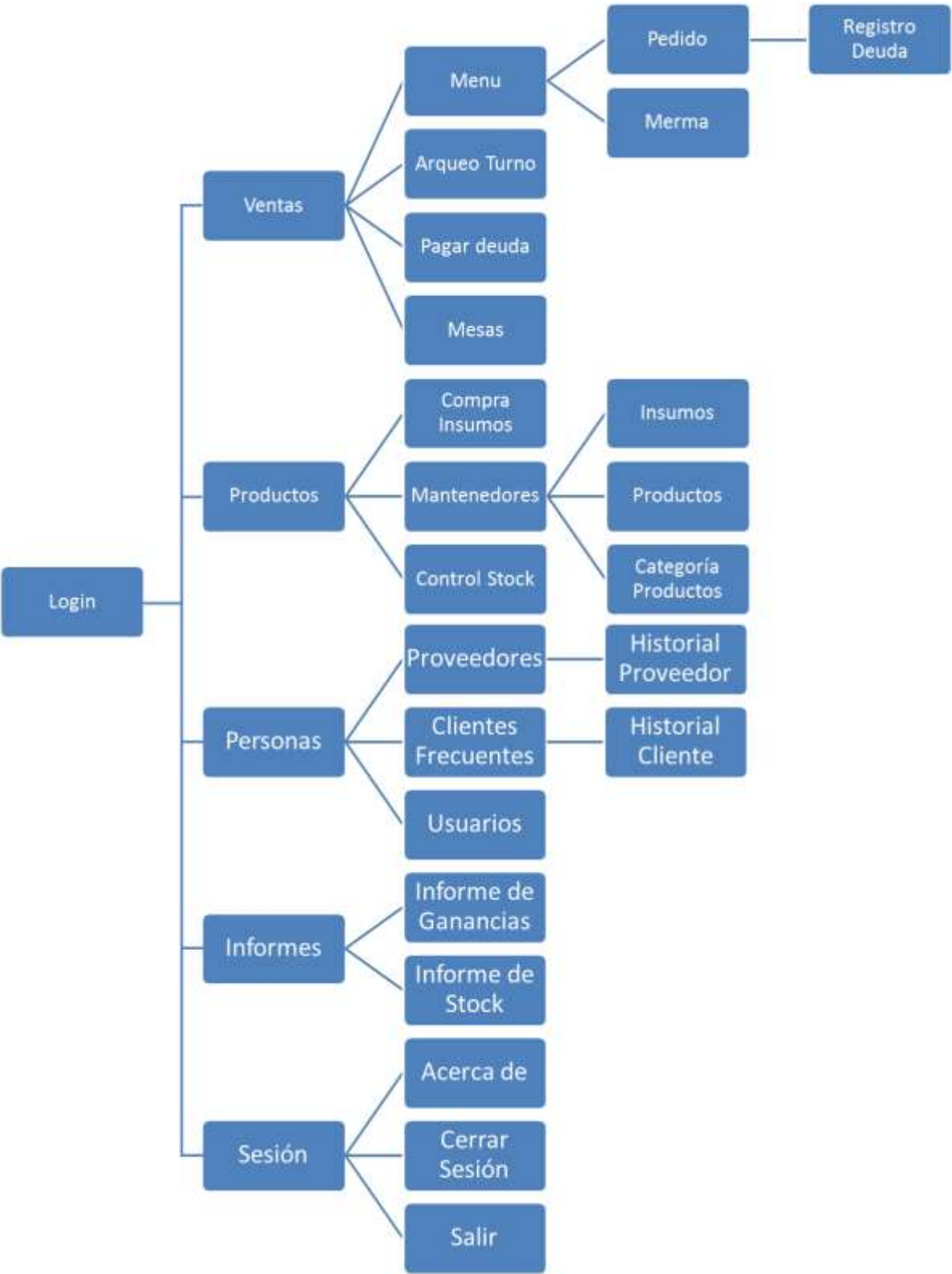


Diagrama. 3-2 – Diagrama de menús

3.3 DESCRIPCIÓN DE PROGRAMAS

Programa	Objetivo
formLogin	Registro de usuarios en el sistema y, según el nivel de acceso de éstos, el poder acceder a las funciones permitidas
formMenu *	Mostrar una representación de las mesas disponibles en el local, así como los pedidos en cola.
formPedido *	Ofrecer una interfaz que permita al usuario añadir todos los productos deseados por un cliente a un pedido.
formDeuda	Registrar cualquier pedido realizado a crédito.
formMerma	Registrar un ajuste de un producto terminado.
formATurno	Registrar el ingreso de dinero adquirido por la venta de pedidos durante la jornada.
formPagoDeuda *	Registrar el monto pagado por un cliente frecuente a la deuda correspondiente.
formMesa	Consultar, Agregar, Modificar o Eliminar mesas del local.
formRCompra	Registrar todas las compras hechas a proveedores y/o establecimientos específicos.
formInsumos *	Consultar, Agregar, Modificar o Eliminar Insumos.
formProductos *	Consultar, Agregar, Modificar o Eliminar productos ofrecidos en el local.
formCategoria	Consultar, Agregar, Modificar o Eliminar Categorías.
formPerdida *	Registrar cualquier cambio del stock, que no corresponda a una venta o comprs.
formProveedores	Consultar, Agregar, Modificar o Eliminar Proveedores.
formClientes	Consultar, Agregar, Modificar o Eliminar Clientes.
formUsuarios	Consultar, Agregar, Modificar o Eliminar Usuarios.
formInRangoVenta *	Mostrar un informe de los ingresos y egresos del establecimiento.
formInMerma	Mostrar un informe de los insumos mermados del establecimiento.
formHistorialCliente	Mostrar un listado del crédito y los pagos realizados de un cliente frecuente específico.
formHistorialProveedor	Mostrar un listado de las compras realizadas a un proveedor específico.
formAcerca	Mostrar información sobre la creación del programa.

Tabla. 3-1 – Descripción de programas

3.4 DESCRIPCIÓN DETALLE DE PROGRAMAS

3.4.1 Ventas, Menú

Nombre lógico: formMenu

Nombre físico: formMenu.java

Objetivo: Mostrar una representación de las mesas disponibles en el local, así como los pedidos en cola.

Descripción: Se muestran distintos botones que representan a las mesas del establecimiento y los posibles pedidos para llevar.

Diagrama de bloques:

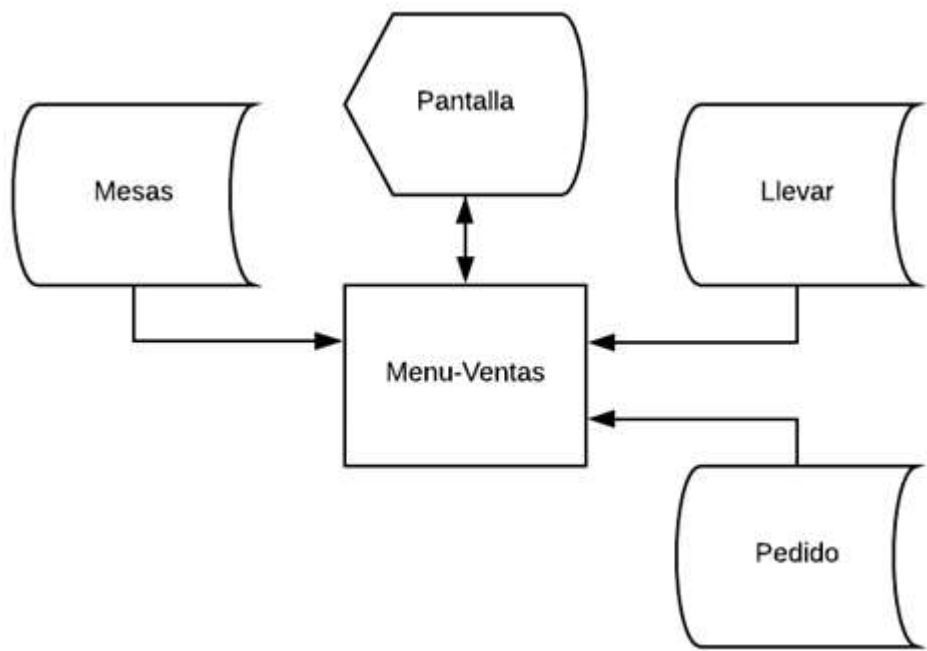


Diagrama. 3.3 – Diagrama de bloque de Menú

Reglas del proceso: Luego de seleccionar la opción Ventas/Menú en la barra de menú de la pantalla Login, el programa desplegará el formulario formMenu (ver figura 3-1). Al iniciar se realizará una consulta a la tabla mesas lo que permitirá que se muestren, según la cantidad de mesas registradas en la base de datos, botones que representarán las mesas en el establecimiento, así como los de los posibles pedidos para llevar. Al seleccionar un botón de “Mesa”, se consultará la tabla mesas con la finalidad de revisar el estado de la mesa (Ocupada o libre). En caso de que ésta se encuentre disponible, generará un nuevo número de pedido con el siguiente valor correlativo del último de éstos registrados en la tabla pedido y desplegará un formulario de pedido (ver figura 3-2) completamente nuevo asignando dicho valor generado; en caso contrario, consultará el número de pedido asignado a la mesa seleccionada en la tabla mesas, para finalmente entregarlo como parámetro al formulario pedido (ver figura 3-2) y desplegar dicho formulario. En caso de que se

presione uno de los botones de “Llevar”, se realizará una consulta en la tabla llevar para asignar el pedido con la id del botón presionado, en caso de que el pedido registrado se encuentre activo y no finalizado (esto quiere decir que aún no es entregado al cliente), se podrá consultar su detalle; por el contrario, en caso de no poseer pedido registrado, abrirá un formulario de pedido (ver figura 3-2) completamente nuevo.

Cuando se oprima el botón “Ingresar Merma”, se desplegará el formulario formMerma. El cual, permite al usuario “Común” sólo registrar la merma de productos ofrecidos, debido a que éste no posee acceso a la función

Al presionar el botón “Atrás” se regresará al menú principal (formLogin).

Pantalla:



Figura 3-1 – Pantalla Menú

3.4.2 Ventas, Menú, Pedido

Nombre lógico: formPedido

Nombre físico: formPedido.java

Objetivo: Ofrecer una interfaz que permita al usuario añadir todos los productos deseados por un cliente a un pedido.

Descripción: Presenta dos listados, el primero los productos que el establecimiento ofrece, mientras que el segundo sirve para almacenar los productos que el cliente desee.

Diagrama de bloques:

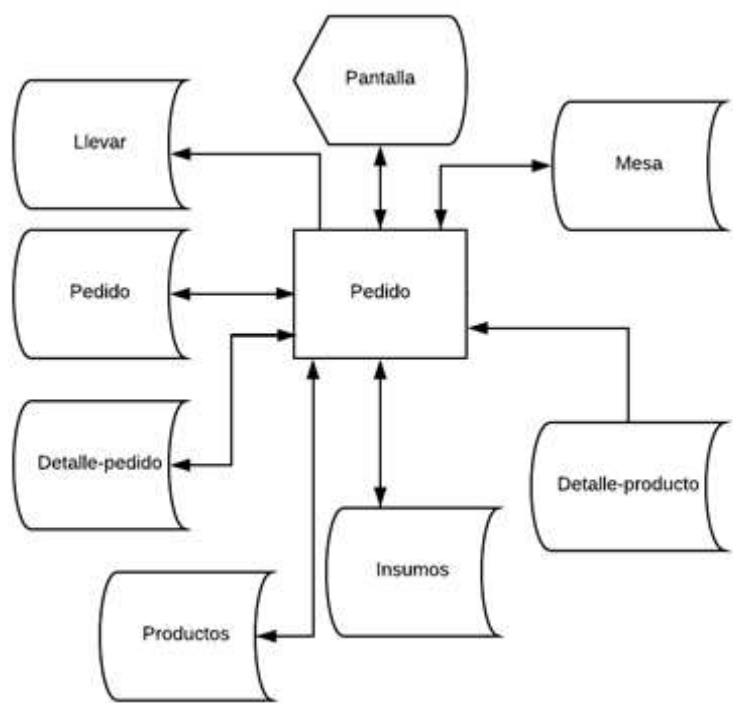


Diagrama. 3-4 – Diagrama de bloque de pedido.

Reglas de proceso: Al haber presionado sobre un botón representativo en la pantalla de menú (ver figura 3-1) se desplegará un formulario de pedido (ver figura 3-2), el cual, recibe como parámetro el número de la mesa (en caso de haber seleccionado un botón representativo de mesa) o el número de llevar representativo (en caso de haber presionado un botón de “llevar”) y el número de pedido desde la pantalla anterior (ver figura 3-1). El sistema se encarga de determinar el turno actual mediante la hora local. Adicionalmente actualizará el estado de pedido a “activo” (en la tabla pedido) y, modificará el estado de la mesa seleccionada a “ocupada” (en la tabla “mesas”) y el estado de la entidad llevar (en la tabla “llevar”) según corresponda.

Mediante una consulta a la tabla productos se logra llenar el listado de productos ofrecidos en el establecimiento, además de informar cuanto stock disponible de cada producto hay actualmente, en base a la cantidad de insumos que utiliza cada uno y la cantidad restante en stock; realizando

consultas a la tabla detalle_producto para obtener el detalle/listado de los insumos utilizados para fabricar dicho producto y, otra consulta realizada a la tabla insumos para obtener el stock actual, para finalmente la cantidad máxima de productos a preparar según la cantidad necesitada y la disponible.

En caso de que el pedido sea nuevo (en caso de los pedidos para servir, que la mesa este desocupada, mientras que en los pedidos para llevar, que el campo representativo esté disponible), se desplegará el formulario (ver figura 3-2) solo con el listado de los productos a ofrecer y los parámetros recibidos mencionados anteriormente, mientras que, si el pedido no es nuevo y no se encuentra finalizado (es decir posee estado activo o pagado), desplegará, además, en el listado de detalle, los productos que ya posea registrados, realizando una consulta a la tabla detalle_pedido.

Al presionar el combobox “Buscar por” (ver figura 3-2), se podrá seleccionar el criterio de búsqueda deseado para filtrar el listado de productos; esto permite buscar según código, nombre y precio. El listado se actualizará según la coincidencia del campo de ingreso de parámetros y el criterio de búsqueda seleccionado.

Al seleccionar un producto de la lista “Listado de productos” (ver figura 3-2), se traspasarán los datos de estos a unos campos de texto, donde el usuario, con ayuda de un “spinner” podrá ingresar la cantidad de producto solicitada por el cliente, para luego estos datos sean traspasados al listado de detalle del pedido.

Cuando se oprima el botón “Registrar detalle” (ver figura 3-2), se almacenará todos los productos del listado “Detalle de pedido” (ver figura 3-2) en la tabla detalle_pedido. Además descontará automáticamente la cantidad proporcional de insumos utilizados del stock actual (en la tabla insumos).

El botón “Eliminar Detalle” (ver figura 3-2), permite borrar un elemento seleccionado del listado “Detalle de pedido” (ver figura 3-2), lo que provoca que los insumos utilizados en la preparación del producto eliminado se añadan nuevamente al stock actual.

Al presionar el botón “Confirmar Pago” (ver figura 3-2), se validará que el monto de pago ingresado no esté en blanco o, sea menor al monto total obtenido al presionar el botón “Calcular total” (ver figura 3-2). Adicionalmente, en caso de que el pago sea en efectivo, se validará que el pago sea mayor o igual al monto total calculado, y en el caso de que el monto pagado sea mayor que el monto a pagar, al presionar el botón “Vuelto”, se calculará el vuelto que deberá ser entregado al cliente. Por otra parte, en el caso de que el pago se realice con tarjeta o a crédito, se exigirá que se ingrese el monto exacto a pagar. Una vez realizadas las validaciones correspondientes, se procederá a registrar el pago del pedido en el sistema.

Finalmente al presionar “Finalizar Pedido” (ver figura 3-2), primero se validará que el pedido tenga productos asociados, en caso de no tenerlos finalizará el pedido (como cancelación de pedido), en caso contrario se validará que el pedido se encuentre pagado (es decir que “estado_pedido” tenga el valor “P”), de no ser así, desplegará un mensaje informando que el pedido aún no se ha pagado, de otro modo se actualizarán los datos en pedido, cambiando su estado a “finalizado”; actualizando a “disponible” en el campo estado_mesa, en caso de que fuera un pedido para servir o el campo estado_llevar, en caso de tratarse de un pedido para llevar; cambios realizados en las tablas “mesas” y “llevar” respectivamente y, a continuación, el campo “estado_pedido” cambiará a su valor inicial en ambas tablas ya mencionadas.

Pantalla:

Fecha: 2018/06/21

Atrás

Turno: V

Pedido

N° Pedido: 0Mesa: 0

Listado de productos

Código	Nombre	Precio	Stock Disponible
1	Churrasco Solo	2500	45
2	Churrasco Mayo	2600	45
3	Churrasco Tomate	2600	45
4	Churrasco Tomate ...	2600	45
5	Churrasco Palta To...	2700	45

Buscar por:

Código

1

Agregar

Cancelar

Detalle de pedido

Código	Nombre	Precio	Cantidad	Estado
1	Churras...	2500	4	P
4	Churras...	2600	4	P
13	Comple...	1000	3	P
20	Bebida ...	500	6	P

Eliminar Detalle

Registrar Detalle

Seleccione medio de pago:

☐ Efectivo

☒ Tarjeta

☐ Crédito

Calcular total

26400

Descuento:

0

Pago:

26400

Vuelto

Finalizar Pedido

Confirmar Pago

Figura 3-2 – Pantalla Pedido

3.4.3 Ventas, Pagar deuda

Nombre lógico: formPagoDeuda

Nombre físico: formPagoDeuda.java

Objetivo: Registrar el monto pagado por un cliente frecuente a la deuda correspondiente.

Descripción: Muestra una lista de los clientes asociados, permite al usuario seleccionar el cliente adecuado e ingresar el monto.

Diagrama de bloques:

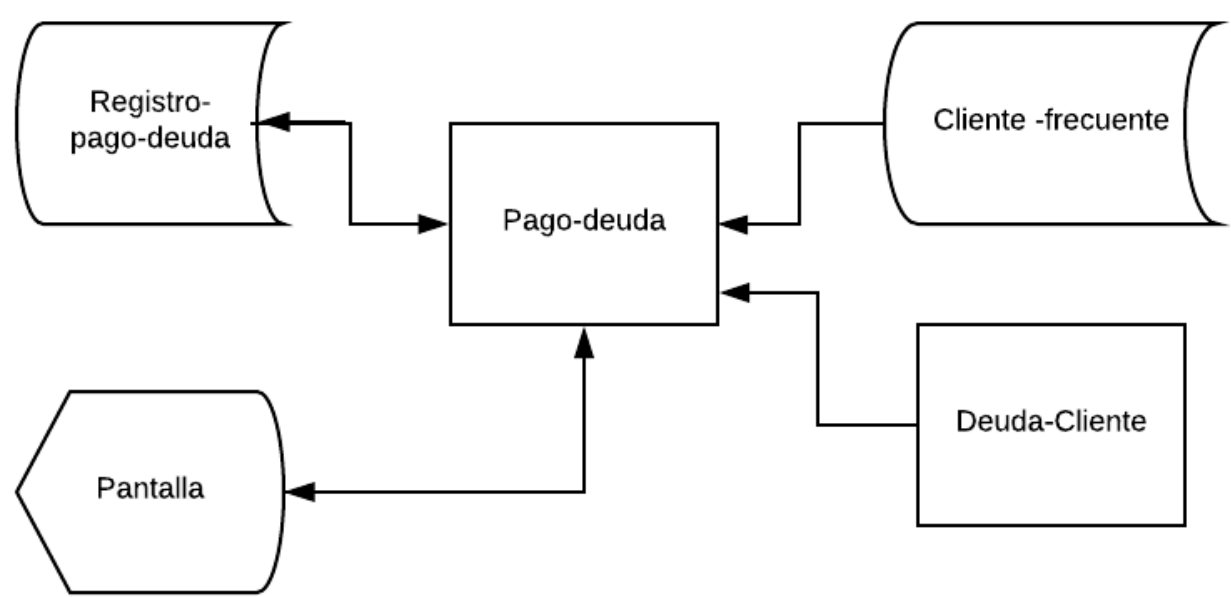


Diagrama. 3-5 – Diagrama de bloque de Pagar deuda

Reglas de proceso: Luego de seleccionar la opción Ventas/Pagar deuda en la barra de menú de la pantalla Login, el programa desplegará el formulario formDeuda (ver figura 3-3). Al iniciar el formulario se obtienen los datos: “rut cliente”, “nombre cliente” y el “apellido paterno” desde la tabla “cliente-frecuente”, además de calcular el monto total de la deuda del cliente, (el cual se obtiene mediante la diferencia del total de deudas y el total de pagos del cliente seleccionado), para posteriormente, llenar el listado “Listado de clientes” (ver figura 3-3). Al hacer doble click sobre un cliente en el listado anteriormente mencionado, se cargarán los datos de éste, en los textbox de “Rut Cliente” y “Monto a Pagar” respectivamente (ver figura 3-3). También existe la opción de ingresar manualmente los datos en dichos textboxes. En tal caso el sistema validará que el rut posea un largo no mayor a 8 dígitos y que el monto no sea mayor a 6 dígitos, validando también que ambos sean numéricos. Luego, al hacer click sobre el botón “Aceptar” (ver figura 3-3), se corrobora que el monto ingresado no sea mayor al monto de la deuda, de ser así el sistema registrará el pago de la deuda en la tabla “registro-pago-deuda”.

Pantalla:

Fecha: 2018/06/21 Atrás

Pago Deuda

Listado de Clientes

Rut	Nombre	Ap. Paterno	Monto Deuda
18534772	Nicolas	Luna	0
18705922	Heptor	Sandoval	34300

Ingreso de datos

Rut Cliente: 18705922 Monto a Pagar: 34300

Aceptar Cancelar

Figura 3-3 – Pantalla Pago deuda

3.4.4 Productos, Mantenedores, Insumos

Nombre lógico: formInsumos

Nombre físico: formInsumos.java

Objetivo: Consultar, Agregar, Modificar o Eliminar Insumos.

Descripción: Muestra una lista donde se ven los insumos registrados, además de ofrecer la opción de agregar uno nuevo o, modificar o eliminar uno existente.

Diagrama de bloques:

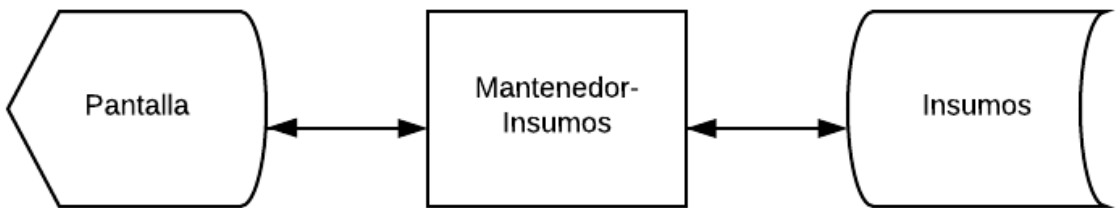


Diagrama. 3-6 – Diagrama de bloque mantenedor insumos

Reglas de proceso: Luego de seleccionar la opción Productos/Insumos en la barra de menú de la pantalla Login, el programa desplegará el formulario formInsumo (ver figura 3-4). Al iniciar, se llena la lista “Listado de Insumos” con los insumos registrados en el local (ver figura 3-4), cargando

los datos desde la tabla “insumos”. Al hacer click sobre un elemento del listado ya mencionado, se cargarán los datos del insumo en los textbox respectivos, cargando así su “código”, “nombre”, “descripción”, “tipo_insumo”, “unidad_de_medida”, “stock-actual”, “stock-critico” y “categoría” (ver figura 3-4). El usuario tiene también la opción de ingresar manualmente dichos datos. El sistema validará que los datos ingresados posean el formato adecuado. Al presionar el botón “Agregar” (ver figura 3-4), el sistema validará que el código del insumo ingresado no exista en la tabla “Insumos”, para luego añadir el nuevo insumo a dicha tabla. Cuando se oprima el botón “Modificar” (ver figura 3-4) el sistema validará que el campo código exista en la tabla “Insumos”, para luego modificar los datos del insumo asociado en la tabla ya mencionada anteriormente. En ambos casos se validará que los campos no estén vacíos. Si se da click sobre el botón “Eliminar” (ver figura 3-4) el sistema validará que el campo “Código” no esté vacío (ver figura 3-4) y que el código ingresado exista, para luego eliminar el insumo que posea dicho código en la tabla Insumos. Al presionar el combobox “Buscar Por” (ver figura 3-4) se podrá seleccionar el criterio de búsqueda, según “código”, “nombre”, “tipo de insumo” y “categoria”. El listado se filtrará según la coincidencia del campo de ingreso de parámetros y el criterio de búsqueda seleccionado.

Pantalla:

Fecha: 2018/06/21

Atrás

INGRESO INSUMO

Ingreso de datos

Código: 1

Nombre: Tomate

Descripción: tomatito

Tipo Insumo: Materia prima

Unidad de medida: Kilo

Stock Actual: 39.9

Stock Crítico: 2.5

Categoría: Vegetales

Agregar

Modificar

Eliminar

Cancelar

Listado de Insumos

Buscar Por: Código

Código	Nombre	Descripción	Tipo Insumo	U. Medi...	Stock Act.	Stock Cr.	Categoría
1	Tomate	tomatito	Materia prima	Kilo	39.9	2.5	Vegetales
2	Palta	Hass	Materia prima	Kilo	40.4	5	Vegetales
3	Papa	Papita	Materia prima	Kilo	110.5	6	Vegetales
4	Zanahoria	Carrot	Materia prima	Kilo	40	2	Vegetales
5	Limon	Limoncito	Materia prima	Kilo	40	2	Vegetales
9	Pan Completo	Pan para completos	Materia prima	Unidad	2	10	Abarrotes

Figura 3-4 – Pantalla Mantenedor Insumos

3.4.5 Productos, Mantenedores, Productos

Nombre lógico: formProductos

Nombre físico: formProductos.java

Objetivo: Consultar, Agregar, Modificar o Eliminar productos ofrecidos en el local.

Descripción: Presenta una lista que muestra los productos registrados, además de ofrecer la opción de agregar uno nuevo o, modificar o eliminar uno existente.

Diagrama de bloques:

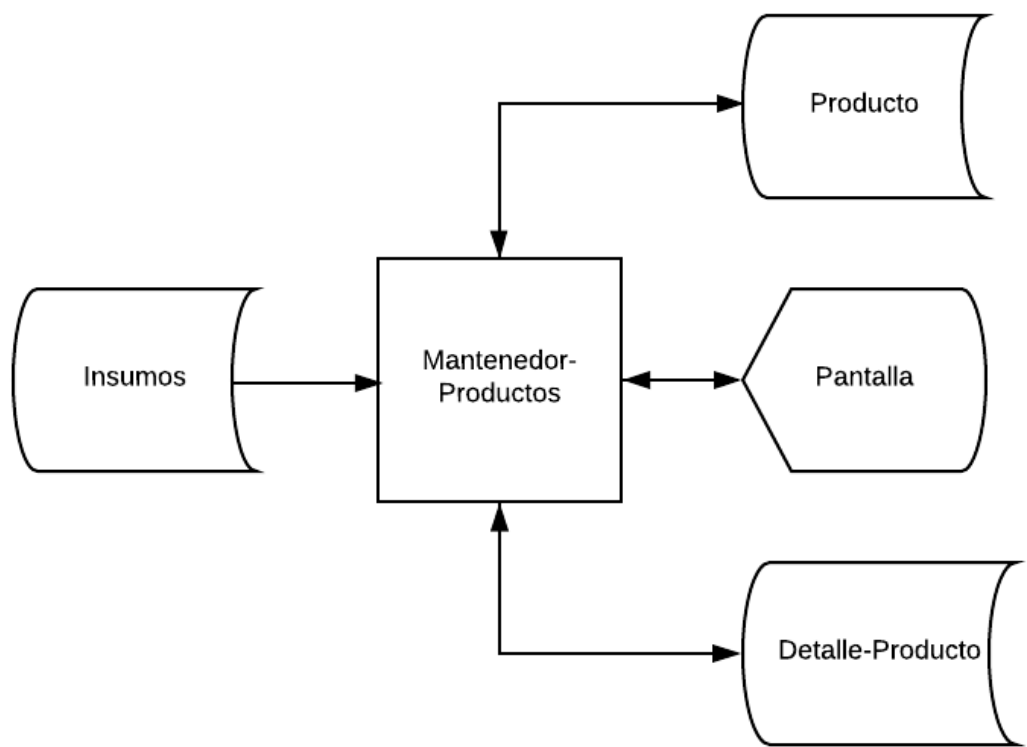


Diagrama. 3-7 – Diagrama de bloque mantenedor productos

Reglas de proceso: Luego de seleccionar la opción Menú/Productos en la barra de menú de la pantalla Login, el programa desplegará el formulario formProducto (ver figura 3-5). Al ingresar al mantenedor, se completa el “Listado de Productos” con los datos recibidos desde la tabla productos, y el “Listado de Insumos” consultando a datos de la tabla insumos (ver figura 3-5). Al seleccionar un producto disponible desde el listado de éstos, se llenan los campos con los datos del producto seleccionado, además se cargará el detalle de los insumos utilizados para su confección en el listado “Detalle Producto” (ver figura 3-5), realizando una consulta a la tabla detalle_producto en la base de datos.

Al presionar el combobox “Buscar por” bajo el “Listado de Productos” (ver figura 3-5), se podrá seleccionar el criterio de búsqueda deseado para filtrar el listado de productos, permitiendo buscar por coincidencia de caracteres según el código o el nombre del producto. El ingresar caracteres en el textbox ubicado junto al combobox mencionado, provoca que se actualice el “Listado de

Productos” (ver figura 3-5) según el criterio de búsqueda y la coincidencia de los parámetros ingresados.

Al presionar el combobox “Buscar por” bajo el “Listado de Insumos” (ver figura 3-5), se podrá seleccionar el criterio de búsqueda deseado para filtrar el listado de insumos según “código”, “nombre”, “tipo de insumo” y “categoria”. Posteriormente al ingresar caracteres en el textbox ubicado junto al combobox mencionado, se actualizará el “Listado de Insumos” (ver figura 3-5) según el criterio de búsqueda y la coincidencia de los parámetros ingresados.

Para poder crear un nuevo producto es necesario completar los datos del producto vía teclado luego, se procederá a detallar los insumos que componen al producto, para esto se ingresa el código de insumo seleccionando un insumo desde la lista “Listado de Insumos” o, se llena el campo “código” vía teclado, y la cantidad deseada; guardando los datos en el listado “Detalle producto” (ver figura 3-5). Finalmente se oprime el botón “Agregar” (ver figura 3-5), lo que valida que los campos no se encuentren vacíos y que el código del producto nuevo no se encuentre en la base de datos. Una vez aprobadas todas las validaciones, almacena los datos del producto en la tabla “producto” y su detalle en la tabla “detalle_producto” de la base de datos.

Al oprimir el botón “Modificar” (ver figura 3-5), se validan que los campos no se encuentren vacíos, de no ser así, se valida que el código del producto a modificar exista en la base de datos, una vez aprobadas todas las validaciones, almacena los datos del producto en la tabla “producto” y su detalle en la tabla “detalle_producto” de la base de datos. Al presionar el botón “Eliminar” (ver figura 3-5), se validan que los campos no se encuentren vacíos, de no ser así, se valida que el código del producto a eliminar exista en la base de datos, una vez aprobadas todas las validaciones, se elimina el producto seleccionado de la tabla producto, junto con su detalle. Al presionar el botón “Cancelar” (ver figura 3-5) se limpiarán todos los campos. El botón “Atrás” (ver figura 3-5) permite volver a la ventana anterior. Al ingresar, modificar, eliminar un producto se registran los cambios en la tabla producto.

Pantalla:

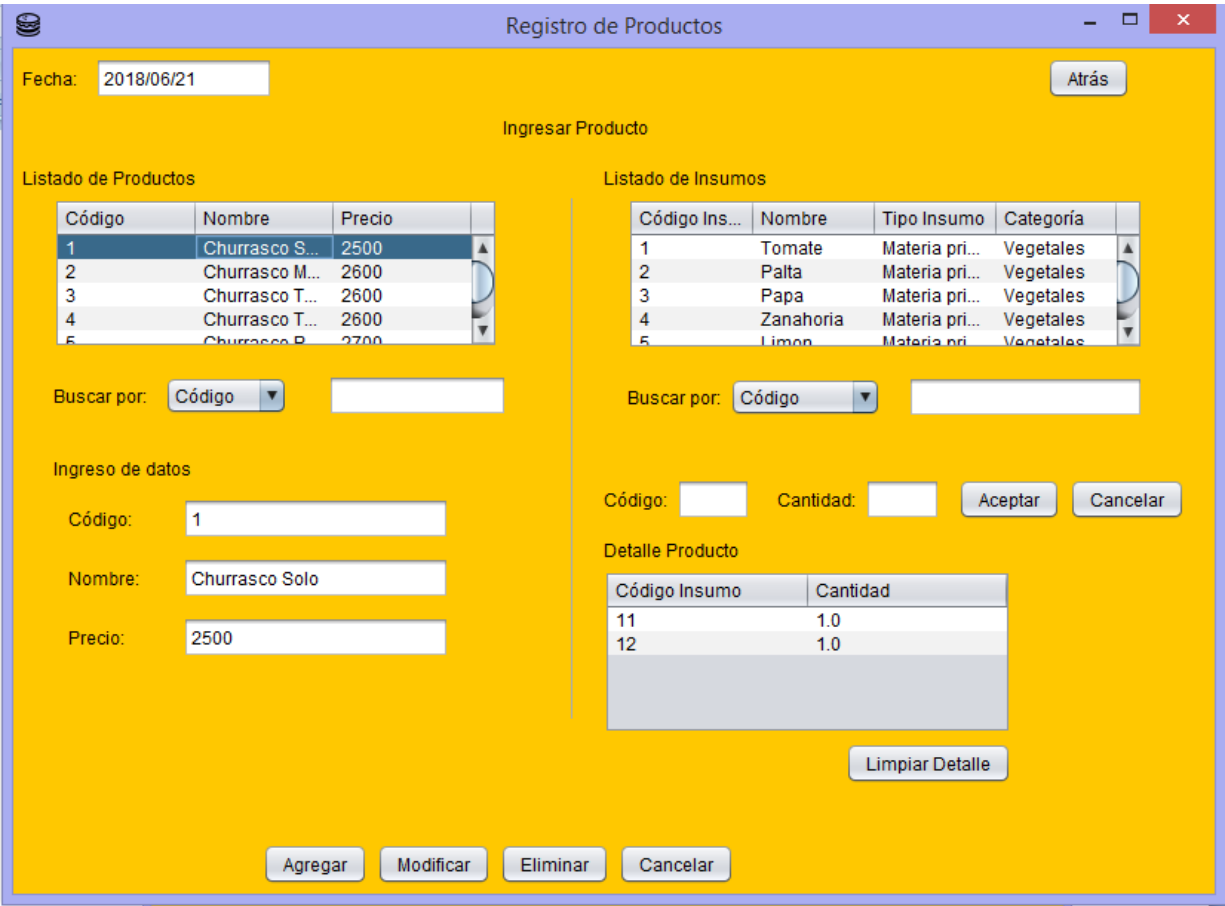


Figura 3-5 – Pantalla Mantenedor Productos

3.4.6 Productos, Control Stock

Nombre lógico: formPerdida

Nombre físico: formPerdida.java

Objetivo: Registrar cualquier cambio del stock, que no corresponda a una venta o compra.

Descripción: Permite escoger el tipo de ajuste y campos de llenado para el usuario.

Diagrama de bloques:

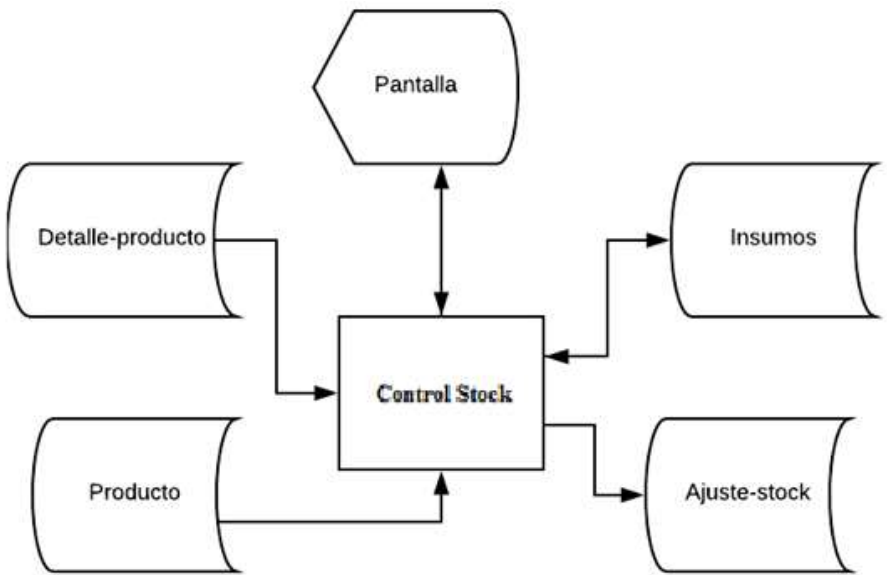


Diagrama. 3-8 – Diagrama de bloque Control stock

Reglas de proceso: Luego de seleccionar la opción Productos/Control Stock en la barra de menú de la pantalla Login, el sistema desplegará el formulario formPerdida (ver figura 3-6). El programa inicia cargando datos desde la tabla “insumos” para llenar la lista de insumos del local (ver figura 3-6). En caso de que el usuario realice un click sobre el checkbox “Producto” se cargarán los datos en el listado, consultando la tabla “producto”, además, obtendrá el “stock virtual” (es decir, cuantos productos se pueden confeccionar teniendo en cuenta cuanta materia prima de éstos se posee) de cada uno de éstos, consultando la tabla “detalle_producto” y la tabla “insumos” para obtener el listado de insumos necesarios para la confección del producto y la cantidad de insumos disponibles, respectivamente. Si se realiza un click sobre el checkbox “Insumo” el sistema generará el listado de insumos del local de la misma forma que se realizó al iniciar el programa (ver figura 3-6).

Al ingresar por teclado el código del producto/insumo y una cantidad a descontar o aumentar, y luego se presiona el botón “Aceptar”, posteriormente, si el checkbox “Insumo” (ver figura 3-6) está activado se leerá el stock actual de dicho insumo para luego descontar (si el checkbox “Perdida” está activado) o aumentar (si el checkbox “Ingreso” está activado) stock a dicho elemento y de esta forma calcular su nuevo stock. En caso de que el checkbox “Producto” esté activado (ver figura 3-6), el sistema consultará el “detalle-producto” de dicho producto y por cada insumo que lo componga, para así calcular la cantidad de insumos necesarios para elaborar el producto seleccionado, ésta se multiplicará por la cantidad del producto que se necesite agregar o eliminar. Posteriormente se actualizará el “stock” de los insumos asociados al producto seleccionado en la tabla “insumos” y se registrará el ajuste en la tabla de “ajuste-stock”. Finalmente se limpiarán los campos de “código” y “cantidad” y, se actualizará el listado de productos o insumos según corresponda (ver figura 3-6).

Al presionar el “combobox” “Filtro” (ver figura 3-6), se podrá seleccionar entre “código”, “nombre” y “stock” para escoger un criterio de búsqueda. El listado se actualizará según el criterio de búsqueda seleccionado y la coincidencia de los caracteres ingresados en el “textbox” “Filtro” (ver figura 3-6).

Pantalla:

Fecha: Control Stock

Atrás

Seleccione

☒ Ingreso / Donación

☐ Pérdida / Merma

Insumo:

☐ Tabla Producto

☒ Tabla Insumo

Código: Cantidad:

Código	Nombre	Stock Act.
1	Tomate	39.9
2	Palta	40.4
3	Papa	110.5
4	Zanahoria	40
5	Limon	40
9	Pan Completo	2
10	Vienesas	30
11	Churrasco	69.89
12	Pan Churrasco	45

Buscar por:

Código

Aceptar

Cancelar

Figura 3-6 – Pantalla Control stock

3.4.7 Informe de ganancia

Nombre lógico: formInRangoVenta

Nombre Físico: formInRangoVenta.java

Objetivo: Mostrar un informe de los ingresos y egresos del establecimiento.

Descripción: Presenta dos campos para seleccionar un rango de fecha a examinar, permite filtrar según el turno y, dos listas que muestran los datos por pantalla.

Diagrama de bloques:

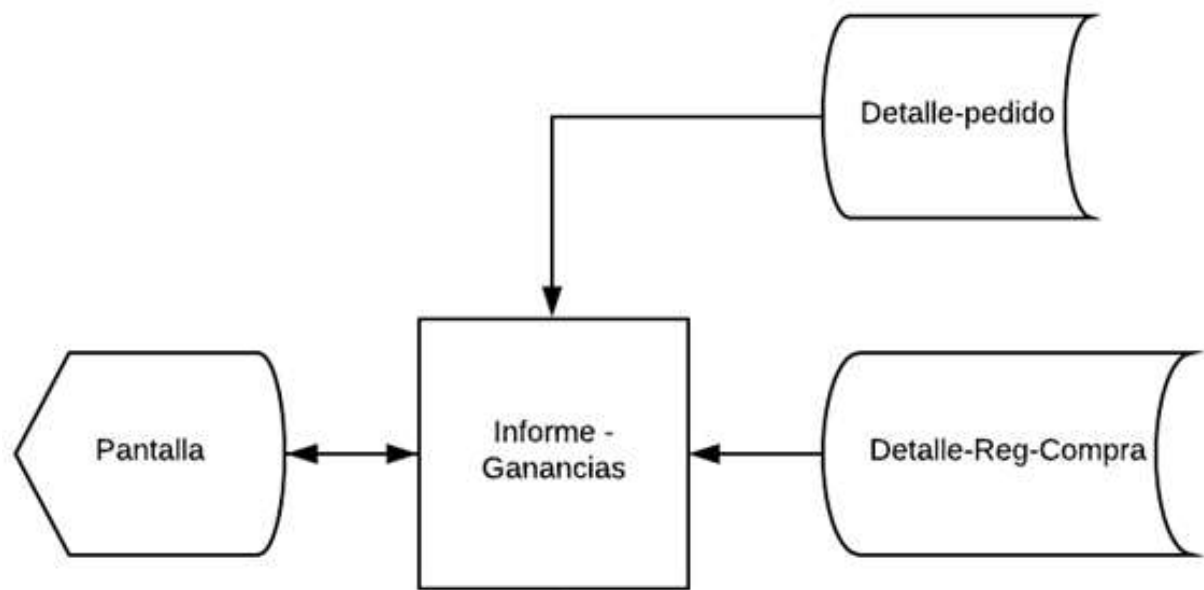


Diagrama. 3-9 – Diagrama de bloque informe ganancias

Reglas de proceso: Luego de seleccionar la opción Productos/Productos en la barra de menú de la pantalla Login, el sistema desplegará el formulario formInRangoVenta (ver figura 3-7). Al seleccionar en el combo box la opción “Mes Actual” (ver figura 3-7) se tomarán los datos que estén en el rango de fecha desde el primero del mes hasta la fecha en la que se consulta. Si se selecciona la opción “Rango de fechas”, se desplegarán botones que permitirán seleccionar un rango de fechas desde un calendario que se mostrará en pantalla (ver figura 3-7); pero al seleccionar la fecha específica se validará que el rango final de fecha sea mayor.

Una vez validadas las fechas, se realizarán las consultas deseadas, para llenar el listado de ingresos (ver figura 3-7) se consultará a la tabla “detalle_pedido”, obteniendo de esta forma todos los productos que fueron vendidos en el rango de fecha seleccionado. Para llenar el listado de egresos (ver figura 3-7), se consulta a la tabla “detalle_registro_compra”, obteniendo así, todos los insumos adquiridos en el rango de fecha.

Al oprimir el botón “Total Neto” se obtiene el total neto adquirido en cada listado (ver figura 3-7). Finalmente al presionar “Ganancia” (ver figura 3-7) se calculará la diferencia entre el total neto de ingresos y el de egresos, obteniendo un valor positivo en caso de haber ganancia o negativo en caso de pérdida.

Pantalla:

Informe de Ganancias

Fecha: 2018/06/21

Atrás

Informe de Ganancia

Seleccione fecha

☐ Mes Actual

☒ Rango de fechas

Desde2017/12/01

Hasta2018/06/13

Aceptar

Cancelar

Detalle

Listado de Ingresos

Código	Nombre	Precio	Cantid...
1	Churra...	2500	18
2	Churra...	2600	11
3	Churra...	2600	8
4	Churra...	2600	6
5	Churra...	2700	9
6	Churra...	2700	7
8	Churra...	3400	1

Total Neto:

1592980

Listado de Egresos

Código	Nombre	Precio	Cantidad
1	Tomate	500	30
2	Palta	16	2009
2	Palta	500	2009
4	Zanaho...	500	10
5	Limon	36	6
13	Masa Pi...	5100	510

Total Neto:

3657860

Ganancia:

Figura 3-7 – Pantalla Informe ganancias

CONCLUSIONES

El desarrollo del presente Sistema de Información dejó diversas apreciaciones en las distintas etapas que lo componen, las cuales se procede a mencionar a continuación.

En la etapa de análisis se logró concluir que es la etapa más compleja de la construcción de un sistema, pues toca enfrentar los requerimientos del cliente, y se debe revisar constantemente lo que se necesita, ya que generalmente no es conciso en sus requerimientos, además de que las ideas de optimización no siempre coincidirán con la perspectiva del cliente.

En la etapa de diseño se pudo concluir que la concepción de la Base de Datos es un proceso sumamente crítico en el desarrollo, puesto que siempre está susceptible a cambios provocados por nuevos requerimientos y modificación por parte del cliente, o bien por la aparición de nuevas tablas durante el proceso de normalización.

En la etapa de construcción de las interfaces de usuario, se logró concluir que si bien lo más difícil fue lograr una interfaz atractiva y cómoda para el cliente, es una etapa relativamente más fácil que las anteriores. No obstante, en situaciones donde se requieren nuevos aspectos a ser implementados, implica volver a etapas anteriores, por lo que se requiere reevaluar el proyecto desde análisis, tomando bastante tiempo y generalmente viene con una mayor carga de trabajo.

Tras la finalización de este trabajo se concluye que se pudo cumplir con los objetivos señalados con anterioridad. Los resultados del sistema propuesto cumplen con los objetivos señalados con anterioridad y las expectativas tanto nuestras como las del cliente. También cabe señalar que todo el proceso de desarrollo es arduo y bastante complejo y, para desarrollarlo de la mejor manera es vital la comunicación con el cliente y un profundo análisis de todos los requerimientos solicitados.

Toda la construcción de este sistema implicó adquirir conocimientos importantes para nuestra formación como profesionales del área de informática.

Como futura proyección del sistema se visualiza agregar un mayor detalle de datos en la generación de informes, además de permitir que éstos puedan ser exportados en formato Excel. Como segunda proyección se planea implementar la impresión de boletas junto con su detalle correspondiente.

Evolucionar el funcionamiento de la base de datos migrándola a un servidor en red local, otorgando el acceso de los distintos dispositivos conectados a la misma red. Ésto permite, además, poder implementar una aplicación que posibilite a los garzones tomar pedidos desde dispositivos móviles (Tablets), eliminando así el uso de papel para registrar los productos pedidos.

BIBLIOGRAFÍA

<https://www.mysql.com/about/>

https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

<https://netbeans.org/features/index.html>

<https://mariadb.org/>

[http://www.w3api.com/wiki/Java:String.equals\(\)](http://www.w3api.com/wiki/Java:String.equals())

ANEXOS

Anexos

Anexo 1

*formMenu.java

```

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Toolkit;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.table.DefaultTableModel;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;
import javax.swing.JButton;
import javax.swing.JOptionPane;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.ArrayList;
import java.util.List;
import javax.swing.BoxLayout;
import javax.swing.JPanel;
import javax.swing.JToggleButton;
public class formMenu extends javax.swing.JFrame implements KeyListener, ActionListener,
ItemListener{
private List<JButton> botonesMesas;
private Integer indice ;
    private Statement sentencia; /*puedo ocupar sentencias de control sql*/
    private Connection conexion; /*objeto que permite conectar mi base datos con programa*/
/* private String nomBD="bussandwicheria";
private String usuario="root";
private String password="informatica"; */
private String msj;
private int cantidad_mesas;
private int limite_llevar;
private int contLlevarN;
public formMenu() {
    initComponents();
    limite_llevar = 20;
    contLlevarN = 0;
    botonesMesas = new ArrayList<>();
    indice = 0 ;
    conectar();
    this.setLocationRelativeTo(null);
    setIcon();
    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
    LocalDate localDate = LocalDate.now();
    txtFecha.setText(dtf.format(localDate));
    GetTurno();

```

```

        cantidad_mesas = contarMesas();
        mostrarMesasN();
        mostrarLlevarN();
        this.getContentPane().setBackground(Color.orange);
    }

    public formMenu(String text, String text0) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    public void conectar() {
        try{
            Class.forName("com.mysql.jdbc.Driver");
            String url ="jdbc:mysql://localhost:3306/"+this.nomBD;
            this.conexion=(Connection)DriverManager.getConnection(url,this.usuario,this.password);
            this.sentencia=(Statement)this.conexion.createStatement();
        }
        catch(Exception e){

        }
    }

    public void insertar(String sql){
        try{
            sentencia.executeUpdate(sql);
            msj="Datos guardados";
        }
        catch(SQLException e){
            System.out.print(e.getMessage());
            msj="no ingreso";
        }
    }

    private void setIcon() {

setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/imagen/icono.png"
)));
    }

    public int GetPedidoNuevo(){
        String query="";
        ResultSet existe;
        int num=0;
        int numped = 0;
        try {
            sentencia = (Statement) conexion.createStatement();
            query = "SELECT * FROM pedido where fecha_venta = '" + txtFecha.getText() + "'";
            existe = sentencia.executeQuery(query);
            if (existe.next() == false) {
                numped = 1;
            } else {
                try {
                    sentencia = (Statement) conexion.createStatement();
                    ResultSet lista = sentencia.executeQuery("SELECT * FROM pedido where
fecha_venta = '" + txtFecha.getText() + "'");
                    int cont = 0;

```

```

        while (lista.next()) {
            num = (lista.getInt("numero_pedido"));
            numped = num + 1;
        }
    } catch (SQLException e) {
    }

    }
} catch (SQLException e) {

}
return numped;
}

public String GetEstadoPedido(int pedido){
    String state = "";
    String query = "";
    ResultSet existe;
    String valFecha = txtFecha.getText();
    try {
        sentencia = (Statement) conexion.createStatement();
        query = "SELECT * from pedido where numero_pedido = " + pedido + " and fecha_venta"
= "" + valFecha + """;
        existe = sentencia.executeQuery(query);
        if (existe.next() == false){
            state = "N";
        } else {
            try {
                sentencia = (Statement) conexion.createStatement();
                ResultSet lista = sentencia.executeQuery("SELECT estado_pedido FROM pedido
where numero_pedido = " + pedido + " and fecha_venta = "" + valFecha + """);
                while (lista.next()) {
                    state = (lista.getString("estado_pedido"));
                }
            } catch (SQLException e) {
                state = "N";
            }
        }

    }

} catch (SQLException e) {
    state = "N";
}

return state;
}

public int ConsultarEstadoMesa(int codmesa){
    boolean ocupada = false;
    int cod_ped = 0;
    String estado = "";
    try{
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT estado FROM mesas where
numero_mesa = " + codmesa);

```

```

while (lista.next()) {
    estado = (lista.getString("estado"));

    if (estado.equals("L")){
        ocupada = false;
    }else{
        ocupada = true;
    }
}
}catch (SQLException e) {

}

if (ocupada == false){
    cod_ped = GetPedidoNuevo();
}else{
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT numero_pedido FROM mesas
where numero_mesa = " + codmesa);
        int cont = 0;
        while (lista.next()) {
            cod_ped = (lista.getInt("numero_pedido"));
        }
    }catch (SQLException e) {
    }
}
return cod_ped;
}

public int consultar_estado_llevar(int v_idllevar){
    int estado = 0;
    try{
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT estado_llevar FROM llevar where
id_llevar = " + v_idllevar);
        while (lista.next()) {
            estado = (lista.getInt("estado_llevar"));
        }
    }catch (SQLException e) {
    }

    return estado;
}

public int consultar_pedido_llevar(int v_idllevar){
    boolean ocupada = false;
    int cod_ped = 0;
    int estado = 0;
    try{
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT estado_llevar FROM llevar where
id_llevar = " + v_idllevar);
        while (lista.next()) {
            estado = (lista.getInt("estado_llevar"));
            if (estado == (0)){

```

```

        ocupada = false;
    }else{
        ocupada = true;
    }
}
}catch (SQLException e) {
}
if (ocupada == false){
    cod_ped = GetPedidoNuevo();
}else{
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT numero_ped FROM llevar where
id_llevar = " + v_idllevar);
        int cont = 0;
        while (lista.next()) {
            cod_ped = (lista.getInt("numero_ped"));
        }
    }catch (SQLException e) {
    }
}
return cod_ped;
}

```

```

public String GetTurno(){
    Calendar calendario = Calendar.getInstance();
    String watashinoturn = "";
    int hora, minutos;
    hora =calendario.get(Calendar.HOUR_OF_DAY);
    minutos = calendario.get(Calendar.MINUTE);
    if (hora < 17){
        watashinoturn = "D";
    }else{
        watashinoturn = "V";
    }
    return watashinoturn;
}

```

```

public int contarMesas(){
    int cant_mesa = 12;
    try{
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM mesas");
        cant_mesa = 0;
        while (lista.next()) {
            cant_mesa = cant_mesa + 1;
            System.out.println("cant mesa" + cant_mesa);
        }
    }catch (SQLException e) {
        e.printStackTrace();
    }

    return cant_mesa;
}
public void  modificarJscrollMesasLayout(){

```

```

}

public void mostrarMesas() {
    Integer contMesas = 0 ;
    Integer sumarX = 0 ;
    Integer sumarY = 0 ;
    Integer cantASumarX = 85;
    Integer cantASumarY = 50;

    while (contMesas < cantidad_mesas){
        JButton auxNomBtn =new JButton("1");
        auxNomBtn .setText(String.valueOf("Mesa " + (contMesas + 1)));
        auxNomBtn .reshape(20 + sumarX,120 + sumarY,80,40);// x,y,ancho,largo
        auxNomBtn.addActionListener(this);
        auxNomBtn.setActionCommand(String.valueOf(contMesas));
        auxNomBtn .setName(String.valueOf(contMesas));
        if ( ((contMesas+1) % 4 == 0)) {
            sumarY = sumarY + cantASumarY;
            sumarX = sumarX - cantASumarX*4;
        }
        contMesas = 1 + contMesas ;
        sumarX = sumarX + cantASumarX;
    }

}

public void mostrarLlevarN(){
    while ( contLlevarN < limite_llevar){
        JButton auxNomBtn =new JButton("Llevar " + (contLlevarN+1) );
        panelLlevar.add(auxNomBtn);
        panelLlevar.updateUI();
        auxNomBtn.addActionListener(this);
        auxNomBtn.setActionCommand("L"+String.valueOf(contLlevarN ));

        auxNomBtn.setName("L"+String.valueOf(contLlevarN));

        contLlevarN = contLlevarN + 1;
    }
}

public void mostrarMesasN(){
    Integer contN = 0;
    while (contN < cantidad_mesas){
        JButton auxNomBtn =new JButton("Mesa " + (contN+1) );
        panelMesas.add(auxNomBtn);
        panelMesas.updateUI();
        auxNomBtn.addActionListener(this);
        auxNomBtn.setActionCommand("M"+String.valueOf(contN ));

        auxNomBtn.setName("M"+String.valueOf(contN ));

        contN = contN + 1;
    }
}

public void ingresarAFormMesas(Integer auxNumBotonMesa ){

```

```

        int num_pedido = ConsultarEstadoMesa(auxNumBotonMesa);
        String turno_act = GetTurno();
        String stat = GetEstadoPedido(num_pedido);
        formPedido pedi = new formPedido(txtFecha.getText(), auxNumBotonMesa, num_pedido,
"S", stat, turno_act);
        System.out.println("menu, estado: " + stat);
        pedi.setVisible(true);
    }

    public void ingresarAFormLlevar(Integer auxNumBotonLlevar ){
        int num_pedido = consultar_pedido_llevar(1);
        String turno_act = GetTurno();
        String stat = GetEstadoPedido(num_pedido);

        formPedido pedi = new formPedido(txtFecha.getText(), num_pedido, "L", stat, turno_act,
auxNumBotonLlevar);
        pedi.setVisible(true);

    }

    public void actionPerformed(ActionEvent e) {
Integer contAbstracto =0;
        while (contAbstracto < cantidad_mesas){
if(e.getActionCommand().equals("M"+(String.valueOf(contAbstracto)))){
            ingresarAFormMesas(contAbstracto);
        }
if(e.getActionCommand().equals("L"+(String.valueOf(contAbstracto)))){
            ingresarAFormLlevar(contAbstracto);
        }
        contAbstracto = contAbstracto + 1 ;
    }
}

////////// Botones //////////
@SuppressWarnings("unchecked")
private void initComponents() {

    scrollPane1 = new java.awt.ScrollPane();
    jLabel1 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();
    jLabel2 = new javax.swing.JLabel();
    jSeparator2 = new javax.swing.JSeparator();
    btnRegresar = new javax.swing.JButton();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jButton11 = new javax.swing.JButton();
    jSeparator3 = new javax.swing.JSeparator();
    txtFecha = new javax.swing.JTextField();
    scrollPanelMesas = new javax.swing.JScrollPane();
    panelMesas = new javax.swing.JPanel();
    scrollPaneLlevar = new javax.swing.JScrollPane();
    panelLlevar = new javax.swing.JPanel();
    btnAumentoLlevar = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setTitle("Layout de Mesas");
    setResizable(false);

```

```

jLabel1.setText("Mesas");

jLabel2.setText("Pedidos");

btnRegresar.setText("Atrás");
btnRegresar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnRegresarActionPerformed(evt);
    }
});

jLabel3.setText("Fecha");

jLabel4.setText("\nEL PRIVILEGIO DEL SABOR\n");

jButton11.setText("Ingresar Merma");
jButton11.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton11ActionPerformed(evt);
    }
});

txtFecha.setEditable(false);

panelMesas.setLayout(new java.awt.GridLayout(0, 4));
scrollPanelMesas.setViewportView(panelMesas);

panelLlevar.setLayout(new java.awt.GridLayout(0, 4));
scrollPaneLlevar.setViewportView(panelLlevar);

btnAumentoLlevar.setText("Aumentar Llevar");
btnAumentoLlevar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnAumentoLlevarActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(scrollPanelMesas)
        .addComponent(scrollPaneLlevar)
        .addGroup(layout.createSequentialGroup()
            .addGap(11, 11, 11)
            .addComponent(jLabel3)
            .addGap(4, 4, 4)
            .addComponent(txtFecha, javax.swing.GroupLayout.PREFERRED_SIZE,
75, javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

        .addGap(222, 222, 222)
        .addComponent(btnRegresar))
    .addGroup(layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addComponent(jSeparator3, javax.swing.GroupLayout.PREFERRED_SIZE,
390, javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createSequentialGroup()
        .addGap(130, 130, 130)
        .addComponent(jLabel4))
    .addGroup(layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addComponent(jLabel1)
        .addGap(10, 10, 10)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE,
326, javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addComponent(jLabel2)
        .addGap(10, 10, 10)
        .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE,
1, javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGap(0, 0, Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(btnAumentoLlevar)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jButton11)))
    .addContainerGap())
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(11, 11, 11)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(3, 3, 3)
                .addComponent(jLabel3))
                .addComponent(txtFecha, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(btnRegresar))
            .addGap(6, 6, 6)
            .addComponent(jLabel4)
            .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1)
            .addGroup(layout.createSequentialGroup()
                .addGap(4, 4, 4)
                .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(scrollPanelMesas, javax.swing.GroupLayout.DEFAULT_SIZE, 155,
Short.MAX_VALUE)

```

```

        .addGap(18, 18, 18)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel2)
        .addGroup(layout.createSequentialGroup()
            .addGap(4, 4, 4)
            .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(scrollPaneLlevar, javax.swing.GroupLayout.PREFERRED_SIZE, 127,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(jSeparator3, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jButton11)
        .addComponent(btnAumentoLlevar))
        .addContainerGap()
    );

    pack();
}

private void btnRegresarActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    this.dispose();
}

private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {
    formMerma merm = new formMerma();
    merm.setVisible(true);
}

private void btnAumentoLlevarActionPerformed(java.awt.event.ActionEvent evt) {
    limite_llevar = limite_llevar + 20;
    mostrarLlevarN();
}

public static void main(String args[]) {
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(formMenu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

```

```

java.util.logging.Logger.getLogger(formMenu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(formMenu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(formMenu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new formMenu().setVisible(true);
        }
    });
}
private javax.swing.JButton btnAumentoLlevar;
private javax.swing.JButton btnRegresar;
private javax.swing.JButton jButton11;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JSeparator jSeparator3;
private javax.swing.JPanel panelLlevar;
private javax.swing.JPanel panelMesas;
private java.awt.ScrollPane scrollPane1;
private javax.swing.JScrollPane scrollPaneLlevar;
private javax.swing.JScrollPane scrollPanelMesas;
private javax.swing.JTextField txtFecha;
public void keyTyped(KeyEvent e) {
    throw new UnsupportedOperationException("Not supported yet.");
}
public void keyPressed(KeyEvent e) {
    throw new UnsupportedOperationException("Not supported yet.");
}
public void keyReleased(KeyEvent e) {
    throw new UnsupportedOperationException("Not supported yet.");
}
public void itemStateChanged(ItemEvent e) {
    throw new UnsupportedOperationException("Not supported yet.");
}
}

```

Anexo 2

***formPedido.java**

```

import java.awt.Color;
import java.awt.Toolkit;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

```

```

import java.sql.Connection;
import java.sql.Statement;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import javax.swing.JOptionPane;
import javax.swing.table.*;
import java.util.logging.*;
import javax.swing.RowFilter;

```

```

public class formPedido extends javax.swing.JFrame {

```

```

    private Statement sentencia;
    private Statement sentencia2;
    private Connection conexion;
    private String nomBD = "bussandwicheria";
    private String usuario = "root";
    private String password = "informatica";
    private String msj;
    private TableRowSorter trsFiltro;
    private int SelectDetalle = -1;
    private int swStockInsuficiente = 0;
    private int id_llevar = 0;
    private boolean pedido_type = true;
    private String mesa;

```

```

    public formPedido() {
        initComponents();
        conectar();
        this.setLocationRelativeTo(null);
        DefaultTableModel modelo = (DefaultTableModel) tblproductos.getModel();
        DefaultTableModel modeloI = (DefaultTableModel) tbldetalle.getModel();
    }

```

```

    public formPedido(String fecha, int numMesa, int numPedido, String tipPedi, String stat,
String turn) {
        initComponents();
        conectar();
        this.setLocationRelativeTo(null);
        llenarTablaP();
        String states = "stat " + stat;
        lblMesa.setVisible(true);
        txtMesa.setVisible(true);
        txtMesa.setText(String.valueOf(numMesa+1));
        mesa = String.valueOf(numMesa);
        pedido_type = true;
        if (stat.equals("N")) {
            RegistrarPedido(fecha, numPedido, tipPedi, turn);
            OcuparMesa(numMesa);
            RegistrarDetalleMesa(numMesa, numPedido, fecha);
        }

        if (stat.equals("A") || stat.equals("P")) {
            LlenarDetalle(numPedido, fecha);
        }
    }

```

```

    btnVuelto.setEnabled(false);
    txtVuelto.setEnabled(false);
    txtNum.setText(Integer.toString(numPedido));
    txtTurno.setText(turn);
    txtFecha.setText(fecha);
    System.out.println(tipPedi);
    this.getContentPane().setBackground(Color.orange);
    setIcon();
}

```

```

public formPedido(String fecha, int numPedido, String tipPedi, String stat, String turn, int
idLLevar) {
    initComponents();
    conectar();
    lblMesa.setVisible(false);
    txtMesa.setVisible(false);
    txtMesa.setText("");
    id_llevar = idLLevar;
    pedido_type = false;
    this.setLocationRelativeTo(null);
    llenarTablaP();
    if (stat.equals("N")) {
        registrarPedido_llevar(idLLevar, numPedido);
        RegistrarPedido(fecha, numPedido, tipPedi, turn);
        registrarEstado_llevar(idLLevar);
    }

    if (stat.equals("A") || stat.equals("P")) {
        LlenarDetalle(numPedido, fecha);
    }
    btnVuelto.setEnabled(false);
    txtVuelto.setEnabled(false);
    txtNum.setText(Integer.toString(numPedido));
    txtTurno.setText(turn);
    txtFecha.setText(fecha);
    System.out.println(tipPedi);
    System.out.println("pedido llevar, stat" + stat);
    this.getContentPane().setBackground(Color.orange);
    setIcon();
}

```

```

public Integer comprobarExistencia(String auxCodProd ){
    Integer existex = (-1);
    Integer nuevaCantidad = 0 ;
    int largo = tbldetalle.getRowCount();
    for (int i = 0; i < largo; i++) {
        Object codp = tbldetalle.getValueAt(i, 0);
        String auxCodP = codp.toString();
        if (txtcod.getText().equals(auxCodP)){
            existex = i ;
        }
    }
    return existex ;
}

```

```

public void registrarPedido_llevar(int v_numLLevar, int v_numped){

```



```

    }
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
}

public void conectar() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/" + this.nomBD;
        this.conexion = (Connection) DriverManager.getConnection(url, this.usuario,
this.password);
        this.sentencia = (Statement) this.conexion.createStatement();
    } catch (Exception e) {
        msj = "error al conectar";
    }
}

public void insertar(String sql) {
    try {
        sentencia.executeUpdate(sql);
        msj = "Datos guardados";
    } catch (SQLException e) {
        System.out.print(e.getMessage());
        msj = "no ingreso";
    }
}

private void setIcon() {

setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/imagen/icono.png"
)));
}

public void llenarTablaP() {
    String auxCod = "";
    String auxNom = "";
    String auxPre = "";
    String auxStock = "";
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM producto");
        int cont = 0;
        while (lista.next()) {
            auxCod = (lista.getString("cod_producto")); //System.out.println(auxCod);
            auxNom = (lista.getString("nombre_producto")); // System.out.println(auxNom);
            auxPre = (lista.getString("precio")); // System.out.println(auxPre);
            auxStock = calcularStockVirtual(auxCod);
            DefaultTableModel modelo = (DefaultTableModel) tblproductos.getModel();
            modelo.addRow(new Object[]{auxCod, auxNom, auxPre,auxStock});
            cont = cont + 1;
        }
    } catch (SQLException e) {
        msj = "no se pudo seleccionar";
    }
}

```

```

}

public void limpiarTablaP() {
    try {
        DefaultTableModel modelo = (DefaultTableModel) tblproductos.getModel();
        int filas = tblproductos.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
}

public void llenarTablaDP(){
    String auxCod = "";
    String auxCant = "";
    String auxPre = "";
    String auxState = "";
    String auxNom = "";
    String auxFec = txtFecha.getText();
    String auxNum = txtNum.getText();
    int auxiNum = Integer.parseInt(auxNum);
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM detalle_pedido where
fecha_venta = " + auxFec + " and numero_pedido = " + auxiNum + "");
        int cont = 0;
        while (lista.next()) {
            auxCod = (lista.getString("cod_producto"));
            auxCant = (lista.getString("cantidad"));
            auxPre = (lista.getString("precio"));
            auxState = (lista.getString("precio"));
            DefaultTableModel modelo = (DefaultTableModel) tbldetalle.getModel();
            modelo.addRow(new Object[]{auxCod, auxCant, auxPre,auxState});
            cont = cont + 1;
        }
    } catch (SQLException e) {
        msj = "no se pudo seleccionar";
    }
}

public void limpiarTablaDP(){
    try {
        DefaultTableModel modelo = (DefaultTableModel) tbldetalle.getModel();
        int filas = tbldetalle.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
}

public void CalcularTotal() {
    int total = 0;

```

```

int largo = tbldetalle.getRowCount();
System.out.println("el largo es: " + largo);
for (int i = 0; i < largo; i++) {
    Object valor = tbldetalle.getValueAt(i, 3);
    int auxValor = Integer.parseInt(valor.toString());
    total = total + auxValor;
}
}

public void OcuparMesa(int valMesa) {
    String sql = "UPDATE mesas set estado='O' WHERE numero_mesa=" + valMesa;
    insertar(sql);
}

public void limpiarCampos() {
    txtcod.setText("");
    txtnom.setText("");
    txtprec.setText("");
    spinCant.setValue(1);
}

public String calcularStockVirtual(String paramCodProd) {
    String returnStockVirtual = "000";
    String auxCodInsumox = "111";
    String auxCodProd = paramCodProd;
    float valorMinimo = 0;
    float auxCantDetalle = 0;
    Integer auxCantIns = 0;
    float auxStockVirtual = 0;
    float auxStockInsumo = 0;
    Integer swValorMin = 1;
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM detalle_producto where
cod_producto ='" + auxCodProd + "'");
        int cont = 0;
        while (lista.next()) {
            auxCodInsumox = (lista.getString("cod_insumo"));
            auxCantDetalle = Float.parseFloat((lista.getString("cantidad")));
            sentencia2 = (Statement) conexion.createStatement();
            ResultSet lista2 = sentencia2.executeQuery("SELECT * FROM insumos where
cod_insumo ='" + auxCodInsumox + "'");
            while (lista2.next()) {
                auxStockInsumo = Float.parseFloat((lista2.getString("stock_actual")));
            }
            auxStockVirtual = auxStockInsumo / auxCantDetalle;
            if (swValorMin == 1) {
                valorMinimo = auxStockVirtual;
                swValorMin = 2;
            }
            if (swValorMin == 2) {
                if (auxStockVirtual < valorMinimo) {
                    valorMinimo = auxStockVirtual;
                }
            }
        }
    }
}

```

```

    }
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
int retorno = (int)valorMinimo ;
returnStockVirtual = Integer.toString(retorno);
return returnStockVirtual;
}

public void ActualizarStockAuto(int v_codProd, int v_cant, int v_tipoAct) {
    String auxCodIns = "";
    float auxCant = 0;
    try {
        System.out.println("actualizar stock");
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM detalle_producto WHERE
cod_producto= " + v_codProd + "");
        int cont = 0;
        while (lista.next()) {
            auxCodIns = lista.getString("cod_insumo");
            auxCant = (Float.parseFloat(lista.getString("cantidad"))) * v_cant;

            if (v_tipoAct == 1) {
                descontarAumentarInsumo(auxCodIns, v_tipoAct, auxCant);
            }
            if (v_tipoAct == 2) {
                descontarAumentarInsumo(auxCodIns, v_tipoAct, auxCant);
            }
        }
    } catch (SQLException e) {
        msj = "no se pudo seleccionar";
    }
}

public void descontarAumentarInsumo(String paramCodIns, Integer accion, float cantidad) {
    int switchNoNegativo = 0;
    String auxCodIns = paramCodIns;
    float auxMontoBD = 0;
    float auxMontoForm = cantidad;
    float auxMontoNuevo = 0;
    String query = "";
    String auxTipoAjuste = "X";
    String auxObs = "Nada";
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM insumos where cod_insumo
=" + auxCodIns + "");
        int cont = 0;
        while (lista.next()) {
            auxMontoBD = Float.parseFloat((lista.getString("stock_actual")));

            if (accion == 1) {

                auxMontoNuevo = auxMontoBD + auxMontoForm;
            }
            if (accion == 2) {

```

```

        auxMontoNuevo = auxMontoBD - auxMontoForm;
        System.out.println(auxMontoNuevo + "=" + auxMontoBD + "-" + auxMontoForm);
    }
    if (auxMontoNuevo < 0) {
        switchNoNegativo = 1;
        swStockInsuficiente = 1;
        JOptionPane.showMessageDialog(null, "Error! Cantidad de insumos solicitados NO
disponible", "ERROR", JOptionPane.INFORMATION_MESSAGE);
    }
    cont = cont + 1;
}
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
if (switchNoNegativo == 0) {
    ResultSet existe;
    try {
        query = "SELECT * FROM insumos where cod_insumo =" + auxCodIns + """;
        existe = sentencia.executeQuery(query);
        if (existe.next() == true) {
            String sql = "UPDATE insumos SET stock_actual=" + auxMontoNuevo +
""WHERE cod_insumo =" + auxCodIns + """;
            try {
                sentencia.executeUpdate(sql);
            } catch (SQLException e) {
                msj = "no ingreso";
                JOptionPane.showMessageDialog(null, "Datos no actualizados", "Error",
JOptionPane.INFORMATION_MESSAGE);
            }
            msj = "Modificado";
        } else {
            msj = "Insumo seleccionado no existe";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        msj = "No existe persona, no se puede modificar";
    }
}
}
}

```

@SuppressWarnings("unchecked")

```

private void initComponents() {
    buttonGroup2 = new javax.swing.ButtonGroup();
    jScrollPane1 = new javax.swing.JScrollPane();
    tblproductos = new javax.swing.JTable();
    jScrollPane2 = new javax.swing.JScrollPane();
    tbldetalle = new javax.swing.JTable();
    btnAgregar = new javax.swing.JButton();
    txtcod = new javax.swing.JTextField();
    txtnom = new javax.swing.JTextField();
    rbtnEfectivo = new javax.swing.JRadioButton();
    rbtnTarjeta = new javax.swing.JRadioButton();
    rbtnCredito = new javax.swing.JRadioButton();
    spinCant = new javax.swing.JSpinner();
    jSeparator1 = new javax.swing.JSeparator();
}

```

```

txtTotal = new javax.swing.JTextField();
jLabel2 = new javax.swing.JLabel();
txtPago = new javax.swing.JTextField();
btnVuelto = new javax.swing.JButton();
txtVuelto = new javax.swing.JTextField();
btnAtras = new javax.swing.JButton();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jSeparator2 = new javax.swing.JSeparator();
btnCancelar = new javax.swing.JButton();
jSeparator3 = new javax.swing.JSeparator();
btnPago = new javax.swing.JButton();
btnFinalizar = new javax.swing.JButton();
txtDescuento = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
jSeparator4 = new javax.swing.JSeparator();
jLabel8 = new javax.swing.JLabel();
jSeparator5 = new javax.swing.JSeparator();
jLabel9 = new javax.swing.JLabel();
txtTurno = new javax.swing.JTextField();
jLabel10 = new javax.swing.JLabel();
txtNum = new javax.swing.JTextField();
txtprec = new javax.swing.JTextField();
btnTotal = new javax.swing.JButton();
txtFecha = new javax.swing.JTextField();
btnRegDetalle = new javax.swing.JButton();
lblMesa = new javax.swing.JLabel();
txtMesa = new javax.swing.JTextField();
jLabel1 = new javax.swing.JLabel();
cmbFiltro = new javax.swing.JComboBox<>();
txtFiltro = new javax.swing.JTextField();
jButton1 = new javax.swing.JButton();
setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
tblproductos.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {
        "Código", "Nombre", "Precio", "Stock Disponible"
    }
) {
    boolean[] canEdit = new boolean [] {
        false, false, false, false
    };

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
    }
});
tblproductos.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        tblproductosMouseClicked(evt);
    }
    public void mouseEntered(java.awt.event.MouseEvent evt) {

```

```

        tblproductosMouseClicked(evt);
    }
});
jScrollPane1.setViewportViewView(tblproductos);
if (tblproductos.getColumnModel().getColumnCount() > 0) {
    tblproductos.getColumnModel().getColumn(0).setResizable(false);
    tblproductos.getColumnModel().getColumn(0).setPreferredWidth(50);
    tblproductos.getColumnModel().getColumn(1).setResizable(false);
    tblproductos.getColumnModel().getColumn(1).setPreferredWidth(120);
    tblproductos.getColumnModel().getColumn(2).setResizable(false);
    tblproductos.getColumnModel().getColumn(2).setPreferredWidth(40);
    tblproductos.getColumnModel().getColumn(3).setResizable(false);
    tblproductos.getColumnModel().getColumn(3).setPreferredWidth(120);
}

tbldetalle.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

        },
    new String [] {
        "Código", "Nombre", "Precio", "Cantidad", "Estado"
    }
));
tbldetalle.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        tbldetalleMouseClicked(evt);
    }
});
jScrollPane2.setViewportViewView(tbldetalle);

btnAgregar.setText("Agregar");
btnAgregar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnAgregarActionPerformed(evt);
    }
});

buttonGroup2.add(rbtnEfectivo);
rbtnEfectivo.setText("Efectivo");
rbtnEfectivo.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        rbtnEfectivoActionPerformed(evt);
    }
});

buttonGroup2.add(rbtnTarjeta);
rbtnTarjeta.setText("Tarjeta");
rbtnTarjeta.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        rbtnTarjetaActionPerformed(evt);
    }
});

buttonGroup2.add(rbtnCredito);
rbtnCredito.setText("Crédito");
rbtnCredito.addActionListener(new java.awt.event.ActionListener() {

```

```

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            rbtnCreditoActionPerformed(evt);
        }
    });

    spinCant.setModel(new javax.swing.SpinnerNumberModel(1, 1, 50, 1));
    spinCant.setToolTipText("");

    txtTotal.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            txtTotalActionPerformed(evt);
        }
    });

    jLabel2.setText("Pago:");

    txtPago.addKeyListener(new java.awt.event.KeyAdapter() {
        public void keyTyped(java.awt.event.KeyEvent evt) {
            txtPagoKeyTyped(evt);
        }
    });

    btnVuelto.setText("Vuelto");
    btnVuelto.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnVueltoActionPerformed(evt);
        }
    });

    txtVuelto.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            txtVueltoActionPerformed(evt);
        }
    });

    btnAtras.setText("Atrás");
    btnAtras.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnAtrasActionPerformed(evt);
        }
    });

    jLabel3.setText("Pedido");

    jLabel4.setText("Fecha:");

    jLabel5.setText("Seleccione medio de pago:");

    btnCancelar.setText("Cancelar");
    btnCancelar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnCancelarActionPerformed(evt);
        }
    });

    btnPago.setText("Confirmar Pago");

```

```

btnPago.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPagoActionPerformed(evt);
    }
});

btnFinalizar.setText("Finalizar Pedido");
btnFinalizar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnFinalizarActionPerformed(evt);
    }
});

txtDescuento.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        txtDescuentoActionPerformed(evt);
    }
});
txtDescuento.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyTyped(java.awt.event.KeyEvent evt) {
        txtDescuentoKeyTyped(evt);
    }
});

jLabel6.setText("Descuento:");

jLabel7.setText("Listado de productos");

jLabel8.setText("Detalle de pedido");

jLabel9.setText("Turno:");

txtTurno.setEditable(false);

jLabel10.setText("N° Pedido:");

txtNum.setEditable(false);

btnTotal.setText("Calcular total");
btnTotal.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnTotalActionPerformed(evt);
    }
});

txtFecha.setEditable(false);

btnRegDetalle.setText("Registrar Detalle");
btnRegDetalle.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnRegDetalleActionPerformed(evt);
    }
});

lblMesa.setText("Mesa:");

```

```
txtMesa.setEditable(false);

jLabel1.setText("Buscar por:");

cmbFiltro.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Código",
"Nombre", "Precio" }));

txtFiltro.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        txtFiltroActionPerformed(evt);
    }
});
txtFiltro.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyTyped(java.awt.event.KeyEvent evt) {
        txtFiltroKeyTyped(evt);
    }
});

jButton1.setText("Eliminar Detalle");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(jSeparator3)
                .add(jSeparator1))
            .addContainerGap())
        .addGroup(layout.createSequentialGroup()
            .add(btnAgregar)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 57, Short.MAX_VALUE)
            .add(btnCancelar)
            .addContainerGap())
        .addGroup(layout.createSequentialGroup()
            .add(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 336, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap())
        .addGroup(layout.createSequentialGroup()
            .add(jButton1)
            .addContainerGap())
);
```

```

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(0, 0, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
    .addComponent(jLabel6)
    .addComponent(jLabel2)
    .addComponent(btnVuelto)
    .addComponent(btnTotal))
    .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
    .addComponent(txtPago,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
85, Short.MAX_VALUE)
    .addComponent(txtDescuento,
javax.swing.GroupLayout.Alignment.TRAILING)
    .addComponent(txtTotal,
javax.swing.GroupLayout.Alignment.TRAILING)
    .addComponent(txtVuelto)))
    .addGroup(layout.createSequentialGroup()
        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(13, 13, 13)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(rbtnTarjeta)
    .addComponent(rbtnEfectivo)
    .addComponent(rbtnCredito)))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
        .addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jSeparator2,
javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGap(0, 46, Short.MAX_VALUE))))
    .addGroup(layout.createSequentialGroup()
        .addComponent(btnRegDetalle)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(btnFinalizar)
    .addGap(18, 18, 18)
    .addComponent(btnPago))
    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel9)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

        .addComponent(txtTurno,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel7)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jSeparator4,
javax.swing.GroupLayout.PREFERRED_SIZE, 192,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel10)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(txtNum,
javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(27, 27, 27)
        .addComponent(lblMesa)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(txtMesa,
javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup()
        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 396,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel1)
        .addGroup(layout.createSequentialGroup()
        .addGap(10, 10, 10)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(cmbFiltro,
javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(txtFiltro,
javax.swing.GroupLayout.PREFERRED_SIZE, 146,
javax.swing.GroupLayout.PREFERRED_SIZE))))))
        .addGap(0, 0, Short.MAX_VALUE))
        .addGroup(layout.createSequentialGroup()
        .addComponent(txtcod, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(30, 30, 30)
        .addComponent(txtnom)
        .addGap(18, 18, 18)
        .addComponent(txtprec, javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(spinCant, javax.swing.GroupLayout.PREFERRED_SIZE,
49, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(238, 238, 238)))

```

```

        .addContainerGap())
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel4)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(txtFecha, javax.swing.GroupLayout.PREFERRED_SIZE, 123,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel3)
        .addGap(176, 176, 176)
        .addComponent(btnAtras))))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addGap(20, 20, 20)
            .addComponent(jLabel3)
            .addGap(51, 51, 51))
        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(btnAtras)
        .addComponent(jLabel4)
        .addComponent(txtFecha, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel9)
        .addComponent(txtTurno, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(5, 5, 5)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel10)
        .addComponent(txtNum, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(lblMesa)
        .addComponent(txtMesa, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED))))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jLabel7)
        .addComponent(jSeparator4, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 115,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGroup(layout.createSequentialGroup())
        .addComponent(jLabel1)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(cmbFiltro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(14, 14, 14)
        .addComponent(txtFiltro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtcod, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(txtnom, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(spinCant, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(txtprec, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(11, 11, 11)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(btnAgregar, javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(btnCancelar))
        .addGap(17, 17, 17)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup())

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jLabel5)
        .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE,
10, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(rbtnEfectivo)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(rbtnTarjeta)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(rbtnCredito)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtTotal, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(btnTotal))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtDescuento, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel6))

```

```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtPago, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel2)))
    .addGroup(layout.createSequentialGroup())

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jLabel8)
        .addComponent(jSeparator5, javax.swing.GroupLayout.PREFERRED_SIZE,
10, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE,
152, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtVuelto, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(btnVuelto))
        .addComponent(jButton1))
        .addGap(20, 20, 20)
        .addComponent(jSeparator3, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(1, 1, 1)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(btnFinalizar, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(btnPago, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(btnRegDetalle)))
    );

    pack();
}

private void rbtnTarjetaActionPerformed(java.awt.event.ActionEvent evt) {
    btnVuelto.setEnabled(false);
    txtVuelto.setEnabled(false);
}

private void btnVueltoActionPerformed(java.awt.event.ActionEvent evt) {
    int tot = Integer.parseInt(txtTotal.getText());
    int des = 0;
    if (txtDescuento.getText().equals("")) {
        des = 0;
    } else {
        des = Integer.parseInt(txtDescuento.getText());
    }

    int vuelto = 0;
    int pag = Integer.parseInt(txtPago.getText());

```

```

        if ((pag - tot - des) > 0) {
            vuelto = pag - tot - des;
        }

        txtVuelto.setText(Integer.toString(vuelto));
    }
    private void txtVueltoActionPerformed(java.awt.event.ActionEvent evt) {

    }
    private void btnAtrasActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.dispose();
    }

    private void rbtnCreditoActionPerformed(java.awt.event.ActionEvent evt) {
        btnVuelto.setEnabled(false);
        txtVuelto.setEnabled(false);
    }

    private void btnPagoActionPerformed(java.awt.event.ActionEvent evt) {
        String numPedido = txtNum.getText();
        String numTotal = txtTotal.getText();
        String fech = txtFecha.getText();
        int desc = Integer.parseInt(txtDescuento.getText());
        int total_pagar = Integer.parseInt(numTotal) - desc;
        if (numTotal.equals("")) {
            JOptionPane.showMessageDialog(null, "ERROR! Campo -Total- vacio ",
"INFORMACION", JOptionPane.INFORMATION_MESSAGE);
        } else {
            if (txtPago.getText().equals("")) {
                JOptionPane.showMessageDialog(null, "ERROR! Campo -Pago- vacio ",
"INFORMACION", JOptionPane.INFORMATION_MESSAGE);
            } else {
                if (Integer.parseInt(txtPago.getText()) < total_pagar) {
                    JOptionPane.showMessageDialog(null, "ERROR! Pago Insuficiente ",
"INFORMACION", JOptionPane.INFORMATION_MESSAGE);
                } else {
                    byte sw = 0;

                    if (rbtnEfectivo.isSelected()) {
                        String sql1 = "UPDATE pedido set tipo_pago='E' WHERE numero_pedido = " +
Integer.parseInt(numPedido) + " and fecha_venta = '" + fech + "'";
                        insertar(sql1);
                        sw = 1;
                    }
                    if (rbtnCredito.isSelected()) {
                        if (Integer.parseInt(txtPago.getText()) > total_pagar) {
                            JOptionPane.showMessageDialog(null, "Pago Excesivo \n Esta opción no
permite vuelto", "INFORMACION", JOptionPane.INFORMATION_MESSAGE);
                            sw = 0;
                        } else {
                            String sql1 = "UPDATE pedido set tipo_pago='C' WHERE numero_pedido = "
+ Integer.parseInt(numPedido) + " and fecha_venta = '" + fech + "'";
                            insertar(sql1);
                            sw = 1;
                        }
                    }
                }
            }
        }
    }

```

```

        formDeuda deud = new formDeuda(numPedido, txtPago.getText());
        deud.setVisible(true);
    }
}

if (rbtnTarjeta.isSelected()) {
    if (Integer.parseInt(txtPago.getText()) > Integer.parseInt(numTotal)) {
        JOptionPane.showMessageDialog(null, "Pago Excesivo \n Esta opción no
permite vuelto", "INFORMACION", JOptionPane.INFORMATION_MESSAGE);
        sw = 0;
    } else {
        String sql1 = "UPDATE pedido set tipo_pago='T' WHERE numero_pedido = "
+ Integer.parseInt(numPedido) + " and fecha_venta = '" + fech + "'";
        insertar(sql1);
        sw = 1;
    }
}

int pago = Integer.parseInt(txtTotal.getText()) -
Integer.parseInt(txtDescuento.getText());

if (sw == 1) {
    String sql = "UPDATE pedido set estado_pedido='P' WHERE numero_pedido = "
+ Integer.parseInt(numPedido) + " and fecha_venta = '" + fech + "'";
    insertar(sql);
    if (rbtnCredito.isSelected() == false) {
        JOptionPane.showMessageDialog(null, "Pago Registrado", "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
    }
    String sqlX = "UPDATE pedido set valor_netto_pedido= " + pago + " WHERE
numero_pedido = " + Integer.parseInt(numPedido) + " and fecha_venta = '" + fech + "'";
    insertar(sqlX);
}
}
}

private void tblproductosMouseClicked(java.awt.event.MouseEvent evt) {

}

private void tblproductosMouseClicked(java.awt.event.MouseEvent evt) {

try{
if (tblproductos.getRowCount() > 0) {
    int sel = tblproductos.getSelectedRow();
    Object cod = tblproductos.getValueAt(sel, 0);
    String auxCode = cod.toString();
    Object nom = tblproductos.getValueAt(sel, 1);
    String auxNom = nom.toString();

    Object pre = tblproductos.getValueAt(sel, 2);
    String auxPre = pre.toString();

```

```

        txtcod.setText(auxCode);
        txtnom.setText(auxNom);
        txtprec.setText(auxPre);
    }}
    catch(Exception e){
    }
}

private void rbtnEfectivoActionPerformed(java.awt.event.ActionEvent evt) {
    btnVuelto.setEnabled(true);
    txtVuelto.setEnabled(true);
}

private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    String code = txtcod.getText();
    String nom = txtnom.getText();
    String pre = txtprec.getText();
    String cant = spinCant.getValue().toString();
    Integer existencia =comprobarExistencia(code);
    if (existencia == (-1) ){
        DefaultTableModel modelodet = (DefaultTableModel) tbldetalle.getModel();
        modelodet.addRow(new Object[]{code, nom, pre, cant, "P"});
    }else{
        Integer nuevaCantidad = 0 ;
        Object codC = tbldetalle.getValueAt(existencia, 3);
        nuevaCantidad = Integer.parseInt(codC.toString()) +
Integer.parseInt(spinCant.getValue().toString());

        DefaultTableModel modelo = (DefaultTableModel) tbldetalle.getModel();
        modelo.removeRow(existencia);
        modelo.addRow(new Object[]{code, nom, pre, nuevaCantidad, "P"});

    }
    limpiarCampos();
}

private void txtTotalActionPerformed(java.awt.event.ActionEvent evt) {
}

private void btnTotalActionPerformed(java.awt.event.ActionEvent evt) {
    int largo = tbldetalle.getRowCount();
    System.out.println("el largo es: " + largo);

    int total = 0;
    txtDescuento.setText("0");

    for (int i = 0; i < largo; i++) {
        Object pre = tbldetalle.getValueAt(i, 2);
        int calPrecio = Integer.parseInt(pre.toString());

        Object cant = tbldetalle.getValueAt(i, 3);
        int calCant = Integer.parseInt(cant.toString());

        int subtotal = calPrecio * calCant;
    }
}

```

```

        total = total + subtotal;
    }
    txtTotal.setText(Integer.toString(total));
}

private void txtPagoKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 5;
    if (txtPago.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if (c < '0' || c > '9') {
        evt.consume();
    }
}

private void txtDescuentoKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 4;

    if (txtDescuento.getText().length() > lar) {
        evt.consume();
    }

    char c = evt.getKeyChar();

    if (c < '0' || c > '9') {
        evt.consume();
    }
}

private void btnRegDetalleActionPerformed(java.awt.event.ActionEvent evt) {
    int NumPed = Integer.parseInt(txtNum.getText());
    String Fech = txtFecha.getText();
    String query = "";
    ResultSet existe;
    System.out.println("Fecha: " + Fech);
    swStockInsuficiente = 0;

    try {
        sentencia = (Statement) conexion.createStatement();
        query = "SELECT * FROM detalle_pedido WHERE numero_pedido = " + NumPed + "
and fecha_venta = " + Fech + """;
        existe = sentencia.executeQuery(query);
        if (existe.next() == true) {
            String sql = ("Delete from detalle_pedido WHERE numero_pedido = " + NumPed + "
and fecha_venta = " + Fech + """);
            insertar(sql);
        }

    } catch (SQLException e) {
        String sql = ("Delete from detalle_pedido WHERE numero_pedido = " + NumPed + " and
fecha_venta = " + Fech + """);
        insertar(sql);
    }
    int largo = tbldetalle.getRowCount();
    System.out.println("el largo es: " + largo);
}

```

```

for (int i = 0; i < largo; i++) {
    String auxFec = txtFecha.getText();
    System.out.println(auxFec);
    int auxNumM = Integer.parseInt(txtNum.getText());
    Object codp = tbldetalle.getValueAt(i, 0);
    int auxCodP = Integer.parseInt(codp.toString());
    Object pre = tbldetalle.getValueAt(i, 2);
    int auxPre = Integer.parseInt(pre.toString());
    Object cant = tbldetalle.getValueAt(i, 3);
    int auxCant = Integer.parseInt(cant.toString());
    Object est = tbldetalle.getValueAt(i, 4);
    String auxEst = est.toString();
    if (auxEst.equals("P")){
        ActualizarStockAuto(auxCodP, auxCant, 2);
        auxEst = "R";
    }

    String sql2 = "INSERT INTO detalle_pedido VALUES (" + auxFec + "," + auxNumM
+ "," + auxCodP + "," + auxCant + "," + auxPre + "," + auxEst + ")";
    insertar(sql2);

}
JOptionPane.showMessageDialog(null, "Detalle registrado", "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
limpiarTablaP();
llenarTablaP();
limpiarTablaDP();
LlenarDetalle(NumPed, Fech);
}

private void btnFinalizarActionPerformed(java.awt.event.ActionEvent evt) {
    int num_pedido = Integer.parseInt(txtNum.getText());
    String fech = txtFecha.getText();
    String query = "";
    ResultSet existe;
    boolean existe_detalle = false;

    try {
        sentencia = (Statement) conexion.createStatement();
        query = "SELECT * FROM detalle_pedido WHERE numero_pedido = " + num_pedido +
" and fecha_venta = " + fech + "";
        existe = sentencia.executeQuery(query);
        if (existe.next() == true) {
            existe_detalle = true;
        }
    } catch (SQLException e) {

    }

    String query_estPed;
    String ped_stat = "";
    try {
        sentencia = (Statement) conexion.createStatement();
        query_estPed = "SELECT estado_pedido FROM pedido WHERE numero_pedido = " +
num_pedido + " and fecha_venta = " + fech + "";
        existe = sentencia.executeQuery(query_estPed);
    }

```

```

        if (existe.next() == true) {
            ped_stat = (existe.getString("estado_pedido"));
        }
    } catch (SQLException e) {

    }

    if (existe_detalle == true) {
        if (ped_stat.equals("P")) {
            String sql = "UPDATE pedido set estado_pedido='F' WHERE numero_pedido= " +
num_pedido + " and fecha_venta= '" + fech + "'";
            System.out.println(sql);
            insertar(sql);
            if (mesa.equals("")) {
                System.out.println("Pedido para llevar");
                String sqlestado = "UPDATE llevar set estado_llevar= 0 WHERE id_llevar=" +
id_llevar;
                insertar(sqlestado);
                String sqlpedido = "UPDATE llevar set num_ped = NULL WHERE id_llevar=" +
id_llevar;
                insertar(sqlpedido);
            } else {
                int mesita = Integer.parseInt(mesa);
                String sqlx = "UPDATE mesas set estado='L' WHERE numero_mesa=" + mesita;
                insertar(sqlx);
                String sql_pedidoMesa = "UPDATE mesas set numero_pedido= NULL WHERE
numero_mesa=" + mesita;
                insertar(sql_pedidoMesa);
            }
            JOptionPane.showMessageDialog(null, "Pedido finalizado", "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "Pedido No Pagado", "¡ADVERTENCIA!",
JOptionPane.ERROR_MESSAGE);
        }

    } else {
        String sql = "UPDATE pedido set estado_pedido='F' WHERE numero_pedido= " +
num_pedido + " and fecha_venta= '" + fech + "'";
        System.out.println(sql);
        insertar(sql);
        if (mesa.equals("")) {
            System.out.println("Pedido para llevar");
            String sqlestado = "UPDATE llevar set estado_llevar= 0 WHERE id_llevar=" +
id_llevar;
            insertar(sqlestado);
            String sqlpedido = "UPDATE llevar set num_ped = NULL WHERE id_llevar=" +
id_llevar;
            insertar(sqlpedido);
        } else {
            int mesita = Integer.parseInt(mesa);
            String sqlx = "UPDATE mesas set estado='L' WHERE numero_mesa=" + mesita;
            insertar(sqlx);
            String sql_pedidoMesa = "UPDATE mesas set numero_pedido= NULL WHERE
numero_mesa=" + mesita;

```

```

        insertar(sql_pedidoMesa);
    }
    JOptionPane.showMessageDialog(null, "Pedido finalizado", "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
    }

}

private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
    limpiarCampos();
}

private void txtDescuentoActionPerformed(java.awt.event.ActionEvent evt) {
}

public void filtro() {
    int columnaABuscar = 0;
    if (cmbFiltro.getSelectedItem() == "Código") {
        columnaABuscar = 0;
    }
    if (cmbFiltro.getSelectedItem().toString() == "Nombre") {
        columnaABuscar = 1;
    }
    if (cmbFiltro.getSelectedItem() == "Precio") {
        columnaABuscar = 2;
    }
    trsFiltro.setRowFilter(RowFilter.regexFilter(txtFiltro.getText(), columnaABuscar));
}

private void txtFiltroKeyTyped(java.awt.event.KeyEvent evt) {
    txtFiltro.addKeyListener(new KeyAdapter() {
        public void keyReleased(final KeyEvent e) {
            String cadena = (txtFiltro.getText());
            txtFiltro.setText(cadena);
            repaint();
            filtro();
        }
    });
    trsFiltro = new TableRowSorter(tblproductos.getModel());
    tblproductos.setRowSorter(trsFiltro);
}

private void tbldetalleMouseClicked(java.awt.event.MouseEvent evt) {
    if (tbldetalle.getRowCount() > 0) {
        SelectDetalle = tbldetalle.getSelectedRow();
    }
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { //GEN-
FIRST:event_jButton1ActionPerformed
    DefaultTableModel ModeloDet = (DefaultTableModel) tbldetalle.getModel();
    int auxNumPed = 0;

    if (SelectDetalle < 0) {

```

```

        JOptionPane.showMessageDialog(null, "Debe seleccionar un elemento de la lista.",
        "ADVERTECIA!", JOptionPane.INFORMATION_MESSAGE);
    } else {
        try {
            Object cod = tbldetalle.getValueAt(SelectDetalle, 0);
            int auxCodProd = Integer.parseInt(cod.toString());
            Object cant = tbldetalle.getValueAt(SelectDetalle, 3);
            int auxCant = Integer.parseInt(cant.toString());
            String auxFec = txtFecha.getText();
            auxNumPed = Integer.parseInt(txtNum.getText());
            ActualizarStockAuto(auxCodProd, auxCant, 1);
            String sqlDD = "Delete from detalle_pedido WHERE fecha_venta = '" + auxFec + "'
and numero_pedido = " + auxNumPed + " and cod_producto = " + auxCodProd + "'";
            System.out.println(sqlDD);
            sentencia.executeUpdate(sqlDD);
            ModeloDet.removeRow(SelectDetalle);
        } catch (SQLException ex) {
            Logger.getLogger(formPedido.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    limpiarTablaDP();
    LlenarDetalle(auxNumPed, txtFecha.getText());
    limpiarTablaP();
    llenarTablaP();
}

private void txtFiltroActionPerformed(java.awt.event.ActionEvent evt) {
}

public static void main(String args[]) {
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
        javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(formPedido.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(formPedido.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(formPedido.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(formPedido.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    java.awt.EventQueue.invokeLater(new Runnable() {

```

```

        public void run() {
            new formPedido().setVisible(true);
        }
    });
}

private javax.swing.JButton btnAgregar;
private javax.swing.JButton btnAtras;
private javax.swing.JButton btnCancelar;
private javax.swing.JButton btnFinalizar;
private javax.swing.JButton btnPago;
private javax.swing.JButton btnRegDetalle;
private javax.swing.JButton btnTotal;
private javax.swing.JButton btnVuelto;
private javax.swing.ButtonGroup buttonGroup2;
private javax.swing.JComboBox<String> cmbFiltro;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JSeparator jSeparator3;
private javax.swing.JSeparator jSeparator4;
private javax.swing.JSeparator jSeparator5;
private javax.swing.JLabel lblMesa;
private javax.swing.JRadioButton rbtnCredito;
private javax.swing.JRadioButton rbtnEfectivo;
private javax.swing.JRadioButton rbtnTarjeta;
private javax.swing.JSpinner spinCant;
private javax.swing.JTable tbldetalle;
private javax.swing.JTable tblproductos;
private javax.swing.JTextField txtDescuento;
private javax.swing.JTextField txtFecha;
private javax.swing.JTextField txtFiltro;
private javax.swing.JTextField txtMesa;
private javax.swing.JTextField txtNum;
private javax.swing.JTextField txtPago;
private javax.swing.JTextField txtTotal;
private javax.swing.JTextField txtTurno;
private javax.swing.JTextField txtVuelto;
private javax.swing.JTextField txtcod;
private javax.swing.JTextField txtnom;
private javax.swing.JTextField txtprec;
}

```


Anexo 3***formPagoDeuda.java**

```

import java.awt.Color;
import java.awt.Toolkit;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import javax.swing.JOptionPane;
import javax.swing.table.*;
import java.util.logging.*;

public class formPagoDeuda extends javax.swing.JFrame {

    private Statement sentencia; /*puedo ocupar sentencias de control sql*/
    private Connection conexion; /*objeto que permite conectar mi base datos con programa*/
    private String msj; /*objeto que permite conectar mi base datos con programa*/
    private String nomBD = "bussandwicheria";
    private String usuario = "root";
    private String password = "informatica";

    public formPagoDeuda() {
        initComponents();
        conectar();
        llenarTabla();
        asignarFecha();
        setIcon();
        this.setLocationRelativeTo(null);
        this.getContentPane().setBackground(Color.orange);
    }

    public void asignarFecha(){
        this.setLocationRelativeTo(null);
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
        LocalDate localDate = LocalDate.now();
        txtFecha.setText(dtf.format(localDate));
        setIcon();
    }

    private void setIcon() {

setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/imagen/icono.png"
)));
    }

    public void conectar() {
        try{
            Class.forName("com.mysql.jdbc.Driver");
            String url ="jdbc:mysql://localhost:3306/"+this.nomBD;
            this.conexion=(Connection)DriverManager.getConnection(url,this.usuario,this.password);
            this.sentencia=(Statement)this.conexion.createStatement();

```

```

    }
    catch(Exception e){
        msj="error al conectar";
    }
}

public Integer validarBtnAgregar(){
    Integer swAceptar = 0 ;
    Integer sw=0;
    if (txtRut.getText().isEmpty() ) {
        JOptionPane.showMessageDialog(null, "ERROR! Campo rut vacio.",
        "¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    } else {
    }
    if (txtMonto.getText().isEmpty() ) {
        JOptionPane.showMessageDialog(null, "ERROR! Campo monto vacio.",
        "¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    } else {
    }

    if (sw==0 ){
        Integer auxRutProveedor = Integer.parseInt(txtRut.getText());
        String query = "";
        ResultSet existe;
        if (auxRutProveedor == 0) {
            JOptionPane.showMessageDialog(null, "ERROR!Rut no puede ser 0.",
            "¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
            sw = 1;
        }

        try {
            sentencia = (Statement) conexion.createStatement();
            query = "SELECT * FROM `cliente_frecuente` WHERE rut_cliente = " +
auxRutProveedor;
            existe = sentencia.executeQuery(query);
            if (existe.next() == false) {
                JOptionPane.showMessageDialog(null, " Error NO EXISTE el cliente .",
                "¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
                sw = 1;
            } else {
            }
        } catch (SQLException e) {
            sw = 1;
        }
    }
    swAceptar = sw;
    return swAceptar;
};

public void llenarTabla(){
    Integer auxRut = 0;
    String auxNom = "";
    String auxAp = "";
    Integer auxMonto = 0 ;

```

```

String query = "";
try{
    sentencia=(Statement)conexion.createStatement();
    ResultSet lista=sentencia.executeQuery("SELECT * FROM cliente_frecuente");
    int cont=0;
    while(lista.next()){
        auxRut = Integer.parseInt((lista.getString("rut_cliente")));
        auxNom = (lista.getString("nombre_cliente"));
        auxAp = (lista.getString("ap_paterno"));
        auxMonto = Integer.parseInt((lista.getString("monto_deuda")));
        DefaultTableModel modelo = (DefaultTableModel) tblClientes.getModel();
        modelo.addRow(new Object[]{auxRut,auxNom,auxAp,auxMonto});
        cont = cont + 1;
    }
}
catch(SQLException e){
    msj="no se pudo seleccionar";
}

}

public void limpiarcampos(){
    txtRut.setText("");
    txtMonto.setText("");
}

public void limpiarTabla(){
    try {
        DefaultTableModel modelo=(DefaultTableModel) tblClientes.getModel();
        int filas=tblClientes.getRowCount();
        for (int i = 0;filas>i; i++) {
            modelo.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
}

private void initComponents() {
    jLabel5 = new javax.swing.JLabel();
    jLabel1 = new javax.swing.JLabel();
    btnRegresar = new javax.swing.JButton();
    lblFecha = new javax.swing.JLabel();
    txtRut = new javax.swing.JTextField();
    txtMonto = new javax.swing.JTextField();
    lblRut = new javax.swing.JLabel();
    lblMonto = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();
    btnAceptar = new javax.swing.JButton();
    btnCancelar = new javax.swing.JButton();
    jLabel2 = new javax.swing.JLabel();
    jSeparator2 = new javax.swing.JSeparator();
    jScrollPane2 = new javax.swing.JScrollPane();
    tblClientes = new javax.swing.JTable()

```

```

{
    public boolean isCellEditable(int rowIndex, int colIndex) {
        return false;
    }
}
;
txtFecha = new javax.swing.JTextField();
jLabel5.setText("jLabel5");
setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
jLabel1.setText("Pago Deuda");
btnRegresar.setText("Atrás");
btnRegresar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnRegresarActionPerformed(evt);
    }
});
lblFecha.setText("Fecha:");
txtRut.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyTyped(java.awt.event.KeyEvent evt) {
        txtRutKeyTyped(evt);
    }
});
txtMonto.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyTyped(java.awt.event.KeyEvent evt) {
        txtMontoKeyTyped(evt);
    }
});
lblRut.setText("Rut Cliente:");
lblMonto.setText("Monto a Pagar:");
jLabel6.setText("Listado de Clientes");
btnAceptar.setText("Aceptar");
btnAceptar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnAceptarActionPerformed(evt);
    }
});
btnCancelar.setText("Cancelar");
btnCancelar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCancelarActionPerformed(evt);
    }
});
jLabel2.setText("Ingreso de datos");
tblClientes.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        },
    new String [] {
        "Rut", "Nombre", "Ap. Paterno", "Monto Deuda"
    }
) {
    boolean[] canEdit = new boolean [] {
        true, false, false, true
    };
    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
    }
});

```



```

        .addGap(24, 24, 24))
        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(lblRut)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(txtRut, javax.swing.GroupLayout.PREFERRED_SIZE, 82,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(70, 70, 70)
        .addComponent(lblMonto)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(txtMonto, javax.swing.GroupLayout.PREFERRED_SIZE,
59, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(57, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(btnRegresar)
        .addComponent(lblFecha)
        .addComponent(txtFecha, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(3, 3, 3)
        .addComponent(jLabel1)
        .addGap(11, 11, 11)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jLabel6)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jSeparator2, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtRut, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(lblRut)
        .addComponent(txtMonto, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(lblMonto))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 36,
Short.MAX_VALUE)

```

```

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(btnAceptar)
    .addComponent(btnCancelar))
    .addContainerGap())
);

pack();
}

private void btnRegresarActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    this.dispose();
}

private void tblClientesMouseClicked(java.awt.event.MouseEvent evt) {
    if(tblClientes.getRowCount() > 0 ){
        int sel = tblClientes.getSelectedRow();
        Object cod = tblClientes.getValueAt(sel, 0);
        String auxRut = cod.toString();
        Object cat = tblClientes.getValueAt(sel, 3);
        String auxMonto = cat.toString();
        txtRut.setText(auxRut);
        txtMonto.setText(auxMonto);
    }
}

private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    int sw = validarBtnAgregar();
    int switchNoNegativo = 0;
    if (sw == 0){
        String auxRut = txtRut.getText();
        Integer auxMontoBD = 0 ;
        Integer auxMontoForm = Integer.parseInt(txtMonto.getText());
        Integer auxMontoNuevo = 0 ;
        String query = "";
        try{
            sentencia=(Statement)conexion.createStatement();
            ResultSet lista=sentencia.executeQuery("SELECT * FROM
cliente_frecuente where rut_cliente ='"+ auxRut + "'");
            int cont=0;
            while(lista.next()){
                auxMontoBD = Integer.parseInt((lista.getString("monto_deuda")));
                auxMontoNuevo = auxMontoBD - auxMontoForm;
                if (auxMontoBD == 0 ){
                    switchNoNegativo = 1;
                    JOptionPane.showMessageDialog(null,"Error el cliente no posee
deuda","ERROR",JOptionPane.INFORMATION_MESSAGE);
                }

                if ((auxMontoNuevo < 0) && (switchNoNegativo == 0) ){

                    switchNoNegativo = 1;

```

```
JOptionPane.showMessageDialog(null,"Error, la deuda que desea reducir es menor al
monto que esta reduciendo","ERROR",JOptionPane.INFORMATION_MESSAGE);
```

```

    }
        cont = cont + 1;
    }
}
catch(SQLException e){
    msj="no se pudo seleccionar";
}

    ResultSet existe;
    if (switchNoNegativo == 0){
        try{
            sentencia=(Statement)conexion.createStatement();
            query="SELECT * from cliente_frecuente WHERE rut_cliente='" + auxRut + "'";
            existe=sentencia.executeQuery(query);
            if(existe.next()==true){
                try{
                    String sql="INSERT INTO registro_pago_deuda VALUES
(" + txtRut.getText() + "','" + txtFecha.getText() + "','" + txtMonto.getText() + "');" ;
                    sentencia.executeUpdate(sql);
                    msj="Datos guardados";
                }
                catch(SQLException e){
                    msj="no ingreso";
                    JOptionPane.showMessageDialog(null,msj,"Error al
actualizar",JOptionPane.INFORMATION_MESSAGE);
                }
                msj="Modificado";

JOptionPane.showMessageDialog(null,msj,"INFORMACION",JOptionPane.INFORMATION_
MESSAGE);
            }else{
                msj="No existe la persona no se puede modificar";

JOptionPane.showMessageDialog(null,msj,"INFORMACION",JOptionPane.INFORMATION_
MESSAGE);
            }
        }
        catch(SQLException e){
            msj="No existe persona, no se puede modificar";
        }
        limpiarTabla();
        llenarTabla();
        limpiarcampos();
    }
}

private void txtRutKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 9;
    if (txtRut.getText().length() >= lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();

```

```

        if (c < '0' || c > '9') {
            evt.consume();
        }
    }

    private void txtMontoKeyTyped(java.awt.event.KeyEvent evt) {
        int lar = 6;
        if (txtMonto.getText().length() >= lar) {
            evt.consume();
        }

        char c = evt.getKeyChar();
        if (c < '0' || c > '9') {
            evt.consume();
        }
    }

    private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
        limpiarcampos();
    }

    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
                javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

            java.util.logging.Logger.getLogger(formPagoDeuda.class.getName()).log(java.util.logging.Level.
                SEVERE, null, ex);
        } catch (InstantiationException ex) {

            java.util.logging.Logger.getLogger(formPagoDeuda.class.getName()).log(java.util.logging.Level.
                SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

            java.util.logging.Logger.getLogger(formPagoDeuda.class.getName()).log(java.util.logging.Level.
                SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

            java.util.logging.Logger.getLogger(formPagoDeuda.class.getName()).log(java.util.logging.Level.
                SEVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new formPagoDeuda().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btnAceptar;
    private javax.swing.JButton btnCancelar;
    private javax.swing.JButton btnRegresar;
    private javax.swing.JLabel jLabel1;

```

```

private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JLabel lblFecha;
private javax.swing.JLabel lblMonto;
private javax.swing.JLabel lblRut;
private javax.swing.JTable tblClientes;
private javax.swing.JTextField txtFecha;
private javax.swing.JTextField txtMonto;
private javax.swing.JTextField txtRut;
}

```

Anexo 4

***formInsumos.java**

```

import java.awt.Color;
import java.awt.Toolkit;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import javax.swing.JOptionPane;
import javax.swing.table.*;
import java.util.logging.*;
import javax.swing.RowFilter;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class formInsumos extends javax.swing.JFrame {
    private Statement sentencia;
    /*puedo ocupar sentencias de control sql*/
    private Connection conexion;
    /*objeto que permite conectar mi base datos con programa*/
    private String nomBD = "bussandwicheria";
    private String usuario = "root";
    private String password = "informatica";
    private String msj;
    private TableRowSorter trsFiltro;

    public formInsumos() {
        initComponents();
        conectar();
        this.setLocationRelativeTo(null);
        DefaultTableModel modelo = (DefaultTableModel) tblInsumo.getModel();
        this.getContentPane().setBackground(Color.orange);
    }

    public formInsumos(String ID, String Pass) {
        initComponents();
    }

```

```

    conectar();
    llenarTabla();
    this.setLocationRelativeTo(null);
    llenarCombo();
    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
    LocalDate localDate = LocalDate.now();
    txtFecha.setText(dtf.format(localDate));
    this.getContentPane().setBackground(Color.orange);
    setIcon();
}

private void setIcon() {
setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/imagen/icono.png"
)));
}

//CONECTAR
public void conectar() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/" + this.nomBD;
        this.conexion = (Connection) DriverManager.getConnection(url, this.usuario,
this.password);
        this.sentencia = (Statement) this.conexion.createStatement();
    } catch (Exception e) {
        msj = "error al conectar";
    }
}

//INSERTAR
public void insertar(String sql) {
    try {
        sentencia.executeUpdate(sql);
        msj = "Datos guardados";
        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
    } catch (SQLException e) {
        msj = "no ingreso";
    }
}

public void llenarTabla() {
    String auxCod = "";
    String auxStoA = "";
    String auxStoC = "";
    String auxTipo = "";
    String auxNom = "";
    String auxCat = "";
    String auxDes = "";
    String auxUni = "";
    String query = "";
    String uni = "";
    String tipo = "";
    int cat;

    try {

```

```

sentencia = (Statement) conexion.createStatement();
ResultSet lista = sentencia.executeQuery("SELECT * FROM insumos");
int cont = 0;
while (lista.next()) {
    auxCod = (lista.getString("cod_insumo"));
    auxStoA = (lista.getString("stock_actual"));
    auxStoC = (lista.getString("stock_critico"));
    tipo = (lista.getString("tipo_insumo"));
    auxNom = (lista.getString("nombre_insumo"));
    cat = (lista.getInt("cod_categoria"));
    auxDes = (lista.getString("descripcion"));
    uni = (lista.getString("uni_medida"));

    switch (tipo) {
        case "M":
            auxTipo = "Materia prima";
            break;
        case "P":
            auxTipo = "Producto";
            break;
    }

    switch (uni) {
        case "Un":
            auxUni = "Unidad";
            break;
        case "Kl":
            auxUni = "Kilo";
            break;
        case "Lt":
            auxUni = "Litro";
            break;
    }

    try {
        sentencia = (Statement) conexion.createStatement();
        String queryz = "SELECT * from categoria";
        ResultSet listaz = sentencia.executeQuery(queryz);
        while (listaz.next() == true) {
            String nomCat = (listaz.getString("nombre_categoria"));
            int idCat = (listaz.getInt("cod_categoria"));
            if (cat == idCat) {
                auxCat = nomCat;
            }
        }
    } catch (Exception e) {
    }

    DefaultTableModel modelo = (DefaultTableModel) tblInsumo.getModel();
    modelo.addRow(new Object[]{auxCod, auxNom, auxDes, auxTipo, auxUni, auxStoA,
auxStoC, auxCat});
    cont = cont + 1;
}
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
}

```

```

}

public void limpiarTabla() {
    try {
        DefaultTableModel modelo = (DefaultTableModel) tblInsumo.getModel();
        int filas = tblInsumo.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
}

public void llenarCombo() {
    cboxCategoria.removeAllItems();
    String query = "";
    ResultSet lista;
    String nomCat;
    try {
        sentencia = (Statement) conexion.createStatement();
        query = "SELECT * from categoria";
        lista = sentencia.executeQuery(query);
        while (lista.next() == true) {
            nomCat = (lista.getString("nombre_categoria"));
            cboxCategoria.addItem(nomCat);
        }
    } catch (SQLException e) {
    }
}

public void limpiarcampos() {
    txtCodigo.setText("");
    txtNombre.setText("");
    txtDescripcion.setText("");
    txtStockA.setText("");
    txtStockC.setText("");
    cboxUni.setSelectedIndex(0);
    cboxCategoria.setSelectedIndex(0);
    cboxTipo.setSelectedIndex(0);
}

public void filtro() {
    int columnaABuscar = 0;
    if (cmbFiltro.getSelectedItem() == "Código") {
        columnaABuscar = 0;
    }
    if (cmbFiltro.getSelectedItem().toString() == "Nombre") {
        columnaABuscar = 1;
    }
    if (cmbFiltro.getSelectedItem() == "Tipo Insumo") {
        columnaABuscar = 3;
    }
    if (cmbFiltro.getSelectedItem().toString() == "Categoria") {
        columnaABuscar = 7;
    }
}

```

```

        trsFiltro.setRowFilter(RowFilter.regexFilter(txtFiltro.getText(), columnaABuscar));
    }

    /*-----AQUI TERMINAN LAS FUNCIONES Y SUPROCESOS DE APOYO-----
    */

    private void btnVolverActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.dispose();
    }

    private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
        String cod = txtCodigo.getText();
        String stocka = txtStockA.getText();
        String stockc = txtStockC.getText();
        String unit = (String) cboxUni.getSelectedItem();
        String nom = txtNombre.getText();
        String cate = (String) cboxCategoria.getSelectedItem();
        String des = txtDescripcion.getText();
        String tipe = (String) cboxTipo.getSelectedItem();

        int cat = 0;
        try {
            sentencia = (Statement) conexion.createStatement();
            String query = "SELECT * from categoria";
            ResultSet lista = sentencia.executeQuery(query);
            while (lista.next() == true) {
                String nomCat = (lista.getString("nombre_categoria"));
                int idCat = (lista.getInt("cod_categoria"));

                if (cate.equals(nomCat)) {
                    cat = idCat;
                }
            }
        } catch (Exception e) {
        }

        String tipo = "";
        switch (tipe) {
            case "Materia prima":
                tipo = "M";
                break;
            case "Producto":
                tipo = "P";
                break;
        }

        String uni = "";
        switch (unit) {
            case "Unidad":
                uni = "Un";
                break;
            case "Kilo":
                uni = "Kl";
                break;
            case "Litro":
                uni = "Lt";

```

```

        break;
    }
    int sw = 0;
    if (txtNombre.getText().isEmpty() || txtCodigo.getText().isEmpty() ||
txtStockA.getText().isEmpty() || txtStockA.getText().isEmpty()) {
        System.out.println("Una de las casillas vacías error ");
        JOptionPane.showMessageDialog(null, "ERROR! Campos vacios.", "¡ADVERTENCIA!",
JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    } else {
        if (Integer.parseInt(txtCodigo.getText()) <= 0 || Float.parseFloat(txtStockA.getText()) <=
0 || Float.parseFloat(txtStockC.getText()) <= 0) {
            JOptionPane.showMessageDialog(null, "ERROR! Campo invalido.",
"¡ADVERTENCIA", JOptionPane.INFORMATION_MESSAGE);
            sw = 1;
        }
    }

    if (sw == 0) {
        int codE = Integer.parseInt(cod);
        String query = "";
        ResultSet existe;

        try {
            sentencia = (Statement) conexion.createStatement();
            query = "SELECT * from insumos WHERE cod_insumo=" + codE;
            existe = sentencia.executeQuery(query);
            if (existe.next() == false) {
                String sql = "INSERT INTO insumos VALUES (" + codE + "," + nom + "," + des
+ "," + tipo + "," + uni + "," + Float.parseFloat(stocka) + "," + Float.parseFloat(stockc) + "," +
cat + ")";
                System.out.println(sql);
                insertar(sql);
            } else {
                msj = "Ya existe insumo";
                JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
            }
        } catch (SQLException e) {
            msj = "No existe persona, no se puede modificar";
        }
    }
    limpiarTabla();
    llenarTabla();
    limpiarcampos();
}

private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
    limpiarcampos();
}

private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
    int sw = 0;
    if (txtNombre.getText().isEmpty() || txtCodigo.getText().isEmpty() ||
txtStockA.getText().isEmpty() || txtStockA.getText().isEmpty()) {
        System.out.println("Una de las casillas vacías error ");
    }
}

```

```

        JOptionPane.showMessageDialog(null, "ERROR! Casillas vacias.", "¡ADVERTENCIA",
JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    } else {
        if (Integer.parseInt(txtCodigo.getText()) <= 0 || Float.parseFloat(txtStockA.getText()) <=
0 || Float.parseFloat(txtStockC.getText()) <= 0) {
            JOptionPane.showMessageDialog(null, "ERROR! Campo invalido.",
"¡ADVERTENCIA", JOptionPane.INFORMATION_MESSAGE);
            sw = 1;
        }
    }
}

if (sw == 0) {
    String cod = txtCodigo.getText();
    String stocka = txtStockA.getText();
    String stockc = txtStockC.getText();
    String unit = (String) cboxUni.getSelectedItem();
    String nom = txtNombre.getText();
    String cate = (String) cboxCategoria.getSelectedItem();
    String des = txtDescripcion.getText();
    String tipe = (String) cboxTipo.getSelectedItem();

    int codE = Integer.parseInt(cod);
    String query = "";
    ResultSet existe;
    int cat = 0;
    try {
        sentencia = (Statement) conexion.createStatement();
        String queryx = "SELECT * from categoria";
        ResultSet lista = sentencia.executeQuery(queryx);
        while (lista.next() == true) {
            String nomCat = (lista.getString("nombre_categoria"));
            int idCat = (lista.getInt("cod_categoria"));

            if (cate.equals(nomCat)) {
                cat = idCat;
            }
        }
    } catch (Exception e) {
    }

    String tipo = "";
    switch (tipe) {
        case "Materia prima":
            tipo = "M";
            break;
        case "Producto":
            tipo = "P";
            break;
    }
    String uni = "";
    switch (unit) {
        case "Unidad":
            uni = "Un";
            break;
        case "Kilo":

```

```

        uni = "Kl";
        break;
    case "Litro":
        uni = "Lt";
        break;
    }

    try {
        sentencia = (Statement) conexion.createStatement();
        query = "SELECT * from insumos WHERE cod_insumo=" + codE;
        existe = sentencia.executeQuery(query);
        if (existe.next() == true) {
            sentencia.executeUpdate("Delete from insumos WHERE cod_insumo =" + codE);

            String sql = "INSERT INTO insumos VALUES (" + codE + "," + nom + "," + des
+ "," + tipo + "," + uni + "," + Float.parseFloat(stocka) + "," + Float.parseFloat(stockc) + "," +
cat + ")";
            insertar(sql);
            msj = "Modificado";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            msj = "No existe insumo";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        msj = "No existe persona, no se puede modificar";
    }
    limpiarTabla();
    llenarTabla();
    limpiarcampos();
}
}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int sw = 0;
    if (txtNombre.getText().isEmpty() || txtCodigo.getText().isEmpty() ||
txtStockA.getText().isEmpty() || txtStockC.getText().isEmpty()) {
        System.out.println("Una de las casillas vacías error ");
        JOptionPane.showMessageDialog(null, "ERROR! Casillas vacias.", "¡ADVERTENCIA",
JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    } else {
        if (Integer.parseInt(txtCodigo.getText()) <= 0 || Float.parseFloat(txtStockA.getText()) <=
0 || Float.parseFloat(txtStockC.getText()) <= 0) {
            JOptionPane.showMessageDialog(null, "ERROR! Codigo invalido.",
"¡ADVERTENCIA", JOptionPane.INFORMATION_MESSAGE);
            sw = 1;
        }
    }
}

if (sw == 0) {
    String codi = txtCodigo.getText();
    String query = "";

```

```

ResultSet existe;
int codE = Integer.parseInt(codi);

try {
    sentencia = (Statement) conexion.createStatement();
    query = "SELECT * from insumos WHERE cod_insumo =" + codE;
    existe = sentencia.executeQuery(query);
    if (existe.next() == true) {
        sentencia.executeUpdate("Delete from insumos WHERE cod_insumo =" + codE);
        msj = "ELIMINADO";
        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
    } else {
        msj = "No existe insumo";
        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
    }
} catch (SQLException e) {
    msj = "No existe persona, no se puede eliminar";
}
limpiarTabla();
llenarTabla();
limpiarcampos();
}
}

private void tblInsumoMouseClicked(java.awt.event.MouseEvent evt) {
    if (tblInsumo.getRowCount() > 0) {
        int sel = tblInsumo.getSelectedRow();
        Object cod = tblInsumo.getValueAt(sel, 0);
        String auxCode = cod.toString();

        Object nom = tblInsumo.getValueAt(sel, 1);
        String auxNom = nom.toString();

        Object des = tblInsumo.getValueAt(sel, 2);
        String auxDes = des.toString();
        Object tipoS = tblInsumo.getValueAt(sel, 3);
        String auxTipo = tipoS.toString();
        Object uni = tblInsumo.getValueAt(sel, 4);
        String auxUni = uni.toString();
        Object stoA = tblInsumo.getValueAt(sel, 5);
        String auxAct = stoA.toString();
        Object stoC = tblInsumo.getValueAt(sel, 6);
        String auxCri = stoC.toString();
        Object cat = tblInsumo.getValueAt(sel, 7);
        String auxCat = cat.toString();

        txtCodigo.setText(auxCode);
        txtNombre.setText(auxNom);
        txtStockA.setText(auxAct);
        txtStockC.setText(auxCri);
        txtDescripcion.setText(auxDes);

        int cate = 0;
        try {

```

```

    sentencia = (Statement) conexion.createStatement();
    String query = "SELECT * from categoria";
    ResultSet lista = sentencia.executeQuery(query);
    while (lista.next() == true) {
        String nomCat = (lista.getString("nombre_categoria"));
        int idCat = (lista.getInt("cod_categoria"));
        if (auxCat.equals(nomCat)) {
            cate = idCat;
        }
    }
} catch (Exception e) {
}
cboxCategoria.setSelectedIndex(cate-1);

switch (auxUni) {
    case "Unidad":
        cboxUni.setSelectedIndex(0);
        break;
    case "Kilo":
        cboxUni.setSelectedIndex(1);
        break;
    case "Litro":
        cboxUni.setSelectedIndex(2);
        break;
}

switch (auxTipo) {
    case "Materia Prima":
        cboxTipo.setSelectedIndex(0);
        break;
    case "Producto":
        cboxTipo.setSelectedIndex(1);
        break;
}
}
}

private void txtCodigoKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 4;
    if (txtCodigo.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if (c < '0' || c > '9') {
        evt.consume();
    }
}

private void txtNombreKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 29;
    if (txtNombre.getText().length() > lar) {
        evt.consume();
    }
}

private void txtStockAKeyTyped(java.awt.event.KeyEvent evt) {

```

```

    int lar = 5;
    if (txtStockA.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if ((c < '0' || c > '9') && (c != '.')) {
        evt.consume();
    }
}

private void txtStockCKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 3;
    if (txtStockC.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if ((c < '0' || c > '9') && (c != '.')) {
        evt.consume();
    }
}

private void txtFiltroKeyTyped(java.awt.event.KeyEvent evt) {
    txtFiltro.addKeyListener(new KeyAdapter() {
        public void keyReleased(final KeyEvent e) {
            String cadena = (txtFiltro.getText());
            txtFiltro.setText(cadena);
            repaint();
            filtro();
        }
    });
    trsFiltro = new TableRowSorter(tblInsumo.getModel());
    tblInsumo.setRowSorter(trsFiltro);
}

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
    * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
    javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

    java.util.logging.Logger.getLogger(formInsumos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

    java.util.logging.Logger.getLogger(formInsumos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

```

```

    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(formInsumos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(formInsumos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new formInsumos().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton btnAgregar;
private javax.swing.JButton btnCancelar;
private javax.swing.JButton btnEliminar;
private javax.swing.JButton btnModificar;
private javax.swing.JButton btnVolver;
private javax.swing.JComboBox<String> cboxCategoria;
private javax.swing.JComboBox<String> cboxTipo;
private javax.swing.JComboBox<String> cboxUni;
private javax.swing.JComboBox<String> cmbFiltro;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JTable jTable1;
private javax.swing.JLabel lblCategoria;
private javax.swing.JLabel lblCodigo;
private javax.swing.JLabel lblNombre;
private javax.swing.JLabel lblStockA;
private javax.swing.JLabel lblStockC;
private javax.swing.JLabel lblTipoSto;
private javax.swing.JTable tblInsumo;
private javax.swing.JTextField txtCodigo;
private javax.swing.JTextField txtDescripcion;
private javax.swing.JTextField txtFecha;
private javax.swing.JTextField txtFiltro;
private javax.swing.JTextField txtNombre;
private javax.swing.JTextField txtStockA;
private javax.swing.JTextField txtStockC;
// End of variables declaration

```

}

Anexo 5***formProductos.java**

```

import java.awt.Color;
import java.awt.Toolkit;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import javax.swing.JOptionPane;
import javax.swing.table.*;
import java.util.logging.*;
import javax.swing.RowFilter;
public class formProductos extends javax.swing.JFrame {
    private Statement sentencia;
    /*puedo ocupar sentencias de control sql*/
    private Connection conexion;
    /*objeto que permite conectar mi base datos con programa*/
    private String nomBD = "bussandwicheria";
    private String usuario = "root";
    private String password = "informatica";

    private String msj;
    private TableRowSorter trsFiltro;
    private TableRowSorter trsFiltro2;

    public formProductos() {
        initComponents();
        conectar();
        this.setLocationRelativeTo(null);
        DefaultTableModel modelo = (DefaultTableModel) tblProd.getModel();
        DefaultTableModel modeloI = (DefaultTableModel) tblIns.getModel();
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
        LocalDate localDate = LocalDate.now();
        txtFecha.setText(dtf.format(localDate));
        this.getContentPane().setBackground(Color.orange);
        setIcon();
    }

    public formProductos(String ID) {
        initComponents();
        conectar();
        llenarTablaP();
    }

```

```

llenarTablaI();
this.setLocationRelativeTo(null);
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
LocalDate localDate = LocalDate.now();
txtFecha.setText(dtf.format(localDate));
this.getContentPane().setBackground(Color.orange);
setIcon();
}

private void setIcon() {
setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/imagen/icono.png"
)));
}

//conectar
public void conectar() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/" + this.nomBD;
        this.conexion = (Connection) DriverManager.getConnection(url, this.usuario,
this.password);
        this.sentencia = (Statement) this.conexion.createStatement();
    } catch (Exception e) {
        msj = "error al conectar";
    }
}

//INSERTAR
public void insertar(String sql) {
    try {
        sentencia.executeUpdate(sql);
        msj = "Datos guardados";
        System.out.print(msj);
    } catch (SQLException e) {
        System.out.print(e.getMessage());
        msj = "no ingreso";
    }
}

public void llenarTablaP() {
    String auxCod = "";
    String auxNom = "";
    String auxPre = "";
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM producto");
        int cont = 0;
        while (lista.next()) {
            auxCod = (lista.getString("cod_producto"));

```

```

        auxNom = (lista.getString("nombre_producto"));
        auxPre = (lista.getString("precio"));
        DefaultTableModel modelo = (DefaultTableModel) tblProd.getModel();
        modelo.addRow(new Object[]{auxCod, auxNom, auxPre});
        cont = cont + 1;
    }
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
}

```

```

public void limpiarTablaP() {
    try {
        DefaultTableModel modelo = (DefaultTableModel) tblProd.getModel();
        int filas = tblProd.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
}

```

```

//llenar tabla insumos
public void llenarTablaI() {
    String auxCod = "";
    String auxNom = "";
    String tipo = "";
    String auxTipo = "";
    String auxCat = "";
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM insumos");
        int cont = 0;
        while (lista.next()) {
            auxCod = (lista.getString("cod_insumo"));
            auxNom = (lista.getString("nombre_insumo"));
            tipo = (lista.getString("tipo_insumo"));
            String cate = (lista.getString("cod_categoria"));
            switch (tipo) {
                case "M":
                    auxTipo = "Materia prima";
                    break;
                case "P":
                    auxTipo = "Producto";
                    break;
            }
        }

        int cat = Integer.parseInt(cate);
    }
}

```

```

try {
    sentencia = (Statement) conexion.createStatement();
    String queryz = "SELECT * from categoria";
    ResultSet listaz = sentencia.executeQuery(queryz);
    while (listaz.next() == true) {
        String nomCat = (listaz.getString("nombre_categoria"));
        //int cate = Integer.parseInt(cat);
        int idCat = (listaz.getInt("cod_categoria"));

        if (cat == idCat) {
            auxCat = nomCat;
        }
    }
} catch (Exception e) {
}
DefaultTableModel modeloI = (DefaultTableModel) tblIns.getModel();
modeloI.addRow(new Object[]{auxCod, auxNom, auxTipo, auxCat});
cont = cont + 1;
}
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
}

public void vaciarTablaPI() {
    try {
        DefaultTableModel modelo = (DefaultTableModel) tblPI.getModel();
        int filas = tblPI.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
}

public void limpiarCampos1() {
    txtCodP.setText("");
    txtNomP.setText("");
    txtPreP.setText("");
}

public void limpiarCampos2() {
    txtCodI.setText("");
    txtCantI.setText("");
}

public void filtro() {
    int columnaABuscar = 0;

```

```

    if (cmbFiltro1.getSelectedItem() == "Código") {
        columnaABuscar = 0;
    }
    if (cmbFiltro1.getSelectedItem().toString() == "Nombre") {
        columnaABuscar = 1;
    }
    trsFiltro.setRowFilter(RowFilter.regexFilter(txtFiltro1.getText(), columnaABuscar));
}

public void filtro2() {
    int columnaABuscar = 0;
    if (cmbFiltro2.getSelectedItem() == "Código") {
        columnaABuscar = 0;
    }
    if (cmbFiltro2.getSelectedItem().toString() == "Nombre") {
        columnaABuscar = 1;
    }
    if (cmbFiltro2.getSelectedItem() == "Tipo Insumo") {
        columnaABuscar = 2;
    }
    if (cmbFiltro2.getSelectedItem().toString() == "Categoria") {
        columnaABuscar = 3;
    }
    trsFiltro2.setRowFilter(RowFilter.regexFilter(txtFiltro2.getText(), columnaABuscar));
}

////////// eventos //////////
private void btnRegresarActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    this.dispose();
}

private void tblInsMouseClicked(java.awt.event.MouseEvent evt) {
    if (tblIns.getRowCount() > 0) {
        int sel = tblIns.getSelectedRow();
        Object cod = tblIns.getValueAt(sel, 0);
        String auxCode = cod.toString();

        Object nom = tblIns.getValueAt(sel, 1);
        String auxNom = nom.toString();

        Object tipoS = tblIns.getValueAt(sel, 2);
        String auxTipo = tipoS.toString();

        Object cat = tblIns.getValueAt(sel, 3);
        String auxCat = cat.toString();

        txtCodI.setText(auxCode);
    }
}

```

```

private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    Integer sw = validarBtnAceptar();

    if (sw==0){
        String code = txtCodI.getText();
        String cant = txtCantI.getText();

        DefaultTableModel modeloPI = (DefaultTableModel) tblPI.getModel();
        modeloPI.addRow(new Object[]{code, cant});
        limpiarCampos2();
    }

private Integer  validarBtnAceptar(){

    Integer sw = 0 ;

    if (txtCodI.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "ERROR! Campo codigo Insumo vacio.",
";ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    }
    if (txtCantI.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "ERROR! Campo cantidad insumo.",
";ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    }

    return sw;
}

private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
    limpiarCampos2();
}

private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
    limpiarCampos1();
    limpiarCampos2();
    vaciarTablaPI();
}

private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    String codp = txtCodP.getText();
    String nomp = txtNomP.getText();
    String prep = txtPreP.getText();
    int sw = 0;

```

```

sw = validarBtnAgregar();
//ingreso a tabla productos
if (sw == 0) {
    int codE = Integer.parseInt(codp);
    String query = "";
    ResultSet existe;

    try {
        sentencia = (Statement) conexion.createStatement();
        query = "SELECT * from producto WHERE cod_producto=" + codE;
        existe = sentencia.executeQuery(query);
        if (existe.next() == false) {
            String sql = "INSERT INTO producto VALUES (" + codE + "," + nomp + "," +
Integer.parseInt(prepare) + ")";
            insertar(sql);
            int largo = tblPI.getRowCount();
            System.out.println("el largo es: " + largo);
            for (int i = 0; i < largo; i++) {
                Object cod = tblPI.getValueAt(i, 0);
                int auxCode = Integer.parseInt(cod.toString());
                Object cant = tblPI.getValueAt(i, 1);
                float auxCant = Float.parseFloat(cant.toString());
                int codpr = Integer.parseInt(txtCodP.getText());
                String sql1 = "INSERT INTO detalle_producto VALUES (" + codpr + "," +
auxCode + "," + auxCant + ")";
                System.out.println(sql1);
                insertar(sql1);
            }
            msj = "Producto Registrado";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            msj = "Ya existe producto";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        msj = "No existe persona, no se puede modificar";
    }
    limpiarTablaP();
    llenarTablaP();
    limpiarCampos1();
    limpiarCampos2();
    vaciarTablaPI();
}
}

public Integer validarBtnAgregar(){
Integer swAgregar = 0 ;

```

```

Integer sw = 0;
DefaultTableModel modelo = (DefaultTableModel) tblPI.getModel();
int largo = modelo.getRowCount();
if (largo <= 0) {
    JOptionPane.showMessageDialog(null, "Tabla detalle vacia .", "¡ERROR!",
JOptionPane.INFORMATION_MESSAGE);
    sw = 1;
};
if (txtCodP.getText().isEmpty()) {
    JOptionPane.showMessageDialog(null, "ERROR! Campo codigo vacio.",
"¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
    sw = 1;
}
if (txtNomP.getText().isEmpty()) {
    JOptionPane.showMessageDialog(null, "ERROR! Campo nombre vacio.",
"¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
    sw = 1;
}
if (txtPreP.getText().isEmpty()) {
    JOptionPane.showMessageDialog(null, "ERROR! Campo precio vacio.",
"¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
    sw = 1;
}
return sw;
};

```

```

private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
    int sw = 0;
    sw = validarBtnAgregar();

    // borrar datos primero
    if (sw == 0) {
        String codp = txtCodP.getText();
        String nomp = txtNomP.getText();
        String prep = txtPreP.getText();
        int codE = Integer.parseInt(codp);

```

```

        String query = "";
        String query2 = "";
        ResultSet existe;
        ResultSet existe2;

```

```

        try {
            sentencia = (Statement) conexion.createStatement();
            query = "SELECT * from producto WHERE cod_producto =" + codE;
            existe = sentencia.executeQuery(query);
            if (existe.next() == true) {

```

```

        query2 = "SELECT * from detalle_producto WHERE cod_producto =" + codE;
        existe2 = sentencia.executeQuery(query2);
        if (existe2.next() == true) {
            sentencia.executeUpdate("Delete from detalle_producto WHERE cod_producto ="
+ codE);
            System.out.println("Detalle eliminado");
        }
        } else {
            msj = "No existe detalle de producto";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        msj = "No existe producto, no se puede eliminar";
    }

    // REINGRESAR VALORES////////
    String query3 = "";
    ResultSet existe3;

    try {
        sentencia = (Statement) conexion.createStatement();
        query3 = "SELECT * from producto WHERE cod_producto=" + codE;
        existe3 = sentencia.executeQuery(query3);
        if (existe3.next() == true) {
            String sql3 = "UPDATE producto SET nombre_producto=" + nombp + "WHERE
cod_producto =" + codE + """;
            sentencia.executeUpdate(sql3);
            sql3 = "UPDATE producto SET precio= " + Integer.parseInt(prepare) + " WHERE
cod_producto =" + codE + """;
            sentencia.executeUpdate(sql3);
            int largo = tblPI.getRowCount();
            for (int i = 0; i < largo; i++) {
                Object cod = tblPI.getValueAt(i, 0);
                int auxCode = Integer.parseInt(cod.toString());
                Object cant = tblPI.getValueAt(i, 1);
                float auxCant = Float.parseFloat(cant.toString());
                int codpr = Integer.parseInt(txtCodP.getText());
                String sql4 = "INSERT INTO detalle_producto VALUES (" + codpr + "," +
auxCode + "," + auxCant + ")";
                sentencia.executeUpdate(sql4);
            }
            msj = "Producto Registrado";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            msj = "Ya existe producto";

```

```

        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
    }
    } catch (SQLException e) {
        msj = "Error de entidad referencial";

    }
}
limpiarTablaP();
llenarTablaP();
limpiarCampos1();
limpiarCampos2();
vaciarTablaPI();
}

```

```

////////// BOTON ELIMINAR //////////
private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int sw = 0;
    sw= validarBtnEliminar();
    if (sw == 0) {
        String codi = txtCodP.getText();
        String query = "";
        String query2 = "";
        ResultSet existe;
        ResultSet existe2;
        int codE = Integer.parseInt(codi);
        try {
            sentencia = (Statement) conexion.createStatement();
            query = "SELECT * from producto WHERE cod_producto =" + codE;
            existe = sentencia.executeQuery(query);
            if (existe.next() == true) {

                query2 = "SELECT * from detalle_producto WHERE cod_producto =" + codE;
                existe2 = sentencia.executeQuery(query2);
                if (existe2.next() == true) {
                    sentencia.executeUpdate("Delete from detalle_producto WHERE cod_producto ="
+ codE);
                    System.out.println("Detalle eliminado");
                }

                sentencia.executeUpdate("Delete from producto WHERE cod_producto =" + codE);

                msj = "ELIMINADO";
                JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
            } else {
                msj = "No existe detalle de producto";
            }
        } catch (SQLException e) {
            msj = "Error de entidad referencial";
        }
    }
}

```

```

        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
    }
    } catch (SQLException e) {
        msj = "No se puede eliminar producto por actividad referencial";
        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
    }

    limpiarTablaP();
    llenarTablaP();
    limpiarCampos1();
    limpiarCampos2();
    vaciarTablaPI();
}

}

private Integer validarBtnEliminar(){
    Integer swAgregar = 0;
    Integer sw = 0;
    if (txtCodP.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "ERROR! Campo rut vacio.",
";¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    }
    return 0;
};

private void tblProdMouseClicked(java.awt.event.MouseEvent evt) {
    String codEX = "";

    if (tblProd.getRowCount() > 0) {
        int sel = tblProd.getSelectedRow();
        Object cod = tblProd.getValueAt(sel, 0);
        String auxCode = cod.toString();
        codEX = auxCode;

        Object nom = tblProd.getValueAt(sel, 1);
        String auxNom = nom.toString();

        Object pre = tblProd.getValueAt(sel, 2);
        String auxPre = pre.toString();

        txtCodP.setText(auxCode);
        txtNomP.setText(auxNom);
        txtPreP.setText(auxPre);
    }
}

```

```

    }

    /////
    int auxCodPI = 0;
    float auxCant = 0;
    vaciarTablaPI();
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM detalle_producto where
cod_producto =" + codEX);
        int cont = 0;
        while (lista.next()) {
            auxCodPI = (lista.getInt("cod_insumo"));
            auxCant = (lista.getFloat("cantidad"));
            DefaultTableModel modeloPI = (DefaultTableModel) tblPI.getModel();
            modeloPI.addRow(new Object[]{auxCodPI, auxCant});
            cont = cont + 1;
        }
    } catch (SQLException e) {
        msj = "no se pudo seleccionar";
    }
}

private void btnEliminarTodosActionPerformed(java.awt.event.ActionEvent evt) {
    vaciarTablaPI();
}

private void txtCodPKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 4;
    if (txtCodP.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if (c < '0' || c > '9') {
        evt.consume();
    }
}

private void txtCantIKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 3;
    if (txtCantI.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if ((c < '0' || c > '9') && (c != '.')) {
        evt.consume();
    }
}

```

```

private void txtPrePKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 4;
    if (txtPreP.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if (c < '0' || c > '9'){
        evt.consume();
    }
}

private void txtCodIKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 4;
    if (txtCodI.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if (c < '0' || c > '9'){
        evt.consume();
    }
}

private void txtNomPKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 29;
    if (txtNomP.getText().length() > lar) {
        evt.consume();
    }
}

private void txtFiltro1KeyTyped(java.awt.event.KeyEvent evt) {
    txtFiltro1.addKeyListener(new KeyAdapter() {
        public void keyReleased(final KeyEvent e) {
            String cadena = (txtFiltro1.getText());
            txtFiltro1.setText(cadena);
            repaint();
            filtro();
        }
    });
    trsFiltro = new TableRowSorter(tblProd.getModel());
    tblProd.setRowSorter(trsFiltro);
}

private void txtFiltro2KeyTyped(java.awt.event.KeyEvent evt) {
    txtFiltro2.addKeyListener(new KeyAdapter() {
        public void keyReleased(final KeyEvent e) {
            String cadena = (txtFiltro2.getText());
            txtFiltro2.setText(cadena);
            repaint();
            filtro2();
        }
    });
}

```

```

    }
});
trsFiltro2 = new TableRowSorter(tblIns.getModel());
tblIns.setRowSorter(trsFiltro2);
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(formProductos.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(formProductos.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(formProductos.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(formProductos.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
    }
}
//</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new formProductos().setVisible(true);
        }
    });
}

```

```

// Variables declaration - do not modify
private javax.swing.JButton btnAceptarI;
private javax.swing.JButton btnAgrega;
private javax.swing.JButton btnCancelar;
private javax.swing.JButton btnCancelarI;
private javax.swing.JButton btnEliminar;
private javax.swing.JButton btnEliminarTodos;
private javax.swing.JButton btnModificar;
private javax.swing.JButton btnRegresar;
private javax.swing.JComboBox<String> cmbFiltro1;
private javax.swing.JComboBox<String> cmbFiltro2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTable tblIns;
private javax.swing.JTable tblPI;
private javax.swing.JTable tblProd;
private javax.swing.JTextField txtCantI;
private javax.swing.JTextField txtCodI;
private javax.swing.JTextField txtCodP;
private javax.swing.JTextField txtFecha;
private javax.swing.JTextField txtFiltro1;
private javax.swing.JTextField txtFiltro2;
private javax.swing.JTextField txtNomP;
private javax.swing.JTextField txtPreP;
// End of variables declaration
}

```

Anexo 6***formPerdida.java**

```

import java.awt.Color;
import java.awt.Toolkit;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import javax.swing.JOptionPane;
import javax.swing.RowFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;
import javax.swing.table.TableColumnModel;
import javax.swing.table.TableRowSorter;
public class formPerdida extends javax.swing.JFrame {
    private Statement sentencia;
    private Statement sentencia2;
    private Connection conexion;
    private String nomBD = "bussandwicheria";
    private String usuario = "root";
    private String password = "informatica";
    private String msj;
    private TableRowSorter trsFiltro;
    private int swAntiRepet = 0; /*con este switch no se repite el mensj de datos act*/
    private int globalError = 0 ; /*Sw para mostrar una sola vez un error*/
    public formPerdida() {
        initComponents();
        conectar();
        chkInsumo.setSelected(true);
        chkIngreso.setSelected(true);
        limpiarTabla();
        llenarColI();
        llenarTablaI();
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
        LocalDate localDate = LocalDate.now();
        lblFecha2.setText(dtf.format(localDate));
        this.setLocationRelativeTo(null);
        setIcon();
        this.getContentPane().setBackground(Color.orange);
    }

```

```

private void setIcon() {
setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/imagen/icono.png"
)));
}

```

```

public void filtro() {
int columnaABuscar = 0;
if (cmbFiltro.getSelectedItem() == "Código") {
columnaABuscar = 0;
}
if (cmbFiltro.getSelectedItem().toString() == "Nombre") {
columnaABuscar = 1;
}
if (cmbFiltro.getSelectedItem() == "Stock Actual") {
columnaABuscar = 2;
}
trsFiltro.setRowFilter(RowFilter.regexFilter(txtFiltro.getText(), columnaABuscar));
}

```

```

public Integer validarProdNegativo(Integer paramCodProdix) {
/*En esta parte del codigo ,se realiza en el caso de que el chk del producto este activado */
int swMaestro = 0;
/*valida que el prod sea positivo*/
int sw = 0;
if (chkProducto.isSelected()) {
if (sw == 0) {
if (chkIngreso.isSelected() || chkPerdida.isSelected()) {
Integer auxCodProd = Integer.parseInt(txtCodIns.getText());
String auxCodIns = "";
float auxCant = 0;
try {
sentencia = (Statement) conexion.createStatement();
ResultSet lista = sentencia.executeQuery("SELECT * FROM detalle_producto
WHERE cod_producto='" + auxCodProd + "'");
int cont = 0;
while (lista.next()) {
auxCodIns = lista.getString("cod_insumo");
auxCant = (Float.parseFloat(lista.getString("cantidad"))) *
(Float.parseFloat(txtCantidad.getText()));
int switchNoNegativo = 0;
float auxMontoBD = 0;
float auxMontoForm = auxCant;
float auxMontoNuevo = 0;
String query = "";
String auxTipoAjuste = "X";
String auxObs = "Nada";
try {
sentencia = (Statement) conexion.createStatement();

```

```

        ResultSet lista2 = sentencia.executeQuery("SELECT * FROM insumos
where cod_insumo ='" + auxCodIns + "'");
        cont = 0;
        while (lista2.next()) {
            auxMontoBD = Float.parseFloat((lista2.getString("stock_actual")));
            auxMontoNuevo = auxMontoBD - auxMontoForm;
            if (auxMontoNuevo < 0 && chkPerdida.isSelected() ) {
                swMaestro = 1;
                switchNoNegativo = 1;
                if (globalError == 0 ){
                    globalError = 1;
                    JOptionPane.showMessageDialog(null, "Error no se puede eliminar una
cantidad que no dispone", "ERROR", JOptionPane.INFORMATION_MESSAGE);
                }
            }
            cont = cont + 1;
        }
    } catch (SQLException e) {
        msj = "no se pudo seleccionar";
    }
}
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
} else {
    JOptionPane.showMessageDialog(null, msj, "Error no hay checkbox activados",
JOptionPane.INFORMATION_MESSAGE);
}
}
}
return swMaestro;
}

public Integer calcCodAjuste() {
    int cod_final = 0;
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM ajuste_stock ");
        while (lista.next()) {
            cod_final = Integer.parseInt((lista.getString("cod_ajuste")));
            cod_final = cod_final + 1;
        }
    } catch (SQLException e) {
    }
    if (cod_final == 0) {
        cod_final = 1;
    }
    return cod_final;
}
}

```

```

public String calcularStockVirtual(String paramCodProd) {
    String returnStockVirtual = "000";
    String auxCodInsumox = "111";
    String auxCodProd = paramCodProd;
    float valorMinimo = 0;
    float auxCantDetalle = 0;
    float auxCantIns = 0;
    float auxStockVirtual = 0;
    float auxStockInsumo = 0;
    Integer swValorMin = 1;
    /*switch para determinar el valor min*/
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM detalle_producto where
cod_producto ='" + auxCodProd + "'");
        int cont = 0;
        while (lista.next()) {
            auxCodInsumox = (lista.getString("cod_insumo"));
            auxCantDetalle = Float.parseFloat((lista.getString("cantidad")));
            sentencia2 = (Statement) conexion.createStatement();
            ResultSet lista2 = sentencia2.executeQuery("SELECT * FROM insumos where
cod_insumo ='" + auxCodInsumox + "'");
            while (lista2.next()) {
                auxStockInsumo = Float.parseFloat((lista2.getString("stock_actual")));
            }
            auxStockVirtual = auxStockInsumo / auxCantDetalle;
            if (swValorMin == 1) {
                valorMinimo = auxStockVirtual;
                swValorMin = 2;
            }
            if (swValorMin == 2) {
                if (auxStockVirtual < valorMinimo) {
                    valorMinimo = auxStockVirtual;
                }
            }
        }
    } catch (SQLException e) {
        msj = "no se pudo seleccionar";
    }
    returnStockVirtual = Float.toString(valorMinimo);
    return returnStockVirtual;
}

public void llenarColI() {
    /*esto va aqui por que si lo pongo en la funcion
    , recibiria un valor nulo al realizar calculos*/
    TableColumn tcCod = new TableColumn(0);
    TableColumn tcNom = new TableColumn(1);
}

```

```

    TableColumn tcStockA = new TableColumn(2);
    tblInsProd.addColumn(tcCod);
    tblInsProd.addColumn(tcNom);
    tblInsProd.addColumn(tcStockA);
}
public void llenarColP() {
    TableColumn tcCod = new TableColumn(0);
    TableColumn tcNom = new TableColumn(1);
    TableColumn tcStockVirtual = new TableColumn(3);
    tblInsProd.addColumn(tcCod);
    tblInsProd.addColumn(tcNom);
    tblInsProd.addColumn(tcStockVirtual);
}
public void llenarTablaI() {
    String auxCod = "";
    String auxStoA = "";
    String auxNom = "";
    String query = "";
    String uni = "";
    String tipo = "";
    String cat = "";
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM insumos");
        int cont = 0;
        while (lista.next()) {
            auxCod = (lista.getString("cod_insumo"));
            auxStoA = (lista.getString("stock_actual"));
            auxNom = (lista.getString("nombre_insumo"));
            DefaultTableModel modelo = (DefaultTableModel) tblInsProd.getModel();
            modelo.addRow(new Object[]{auxCod, auxNom, auxStoA});
            cont = cont + 1;
        }
    } catch (SQLException e) {
        msj = "no se pudo seleccionar";
    }
}
public void llenarTablaP() {
    String auxCod = "";
    String auxNom = "";
    String auxPre = "";
    String auxStockVirtual = "";
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM producto");
        int cont = 0;
        while (lista.next()) {
            auxCod = (lista.getString("cod_producto"));
            auxNom = (lista.getString("nombre_producto"));

```

```

        auxPre = (lista.getString("precio"));
        auxStockVirtual = calcularStockVirtual(auxCod);
        DefaultTableModel modelo = (DefaultTableModel) tblInsProd.getModel();
        modelo.addRow(new Object[]{auxCod, auxNom, "vaciox", auxStockVirtual});
        cont = cont + 1;
    }
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
}

public void conectar() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/" + this.nomBD;
        this.conexion = (Connection) DriverManager.getConnection(url, this.usuario,
this.password);
        this.sentencia = (Statement) this.conexion.createStatement();
    } catch (Exception e) {
        msj = "error al conectar";
    }
}

//INSERTAR
public void insertar(String sql) {
    try {
        sentencia.executeUpdate(sql);
    } catch (SQLException e) {
        msj = "ERROR numero demasiado grande para el programa";
        JOptionPane.showMessageDialog(null, "Numero demasiado grande para el programa",
"ERROR", JOptionPane.INFORMATION_MESSAGE);
    }
}

private void btnRegresarActionPerformed(java.awt.event.ActionEvent evt) {
    //new fLogin().setVisible(true);
    this.setVisible(false);
    this.dispose();
}

public Integer validarBtnAceptarIns() {
    Integer swAcepta = 0;
    Integer sw = 0;
    if (txtCodIns.getText().isEmpty() ) {
        JOptionPane.showMessageDialog(null, "ERROR! Campo codigo vacio.",
";ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    }
    if (txtCantidad.getText().isEmpty() ) {
        JOptionPane.showMessageDialog(null, "ERROR! Campo cantidad vacio.",
";ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);

```

```

        sw = 1;
    }
    if (sw == 0) {
        /*en esta segunda validacion necesitamos que exista el rutprov por lo que debe eser 0*/
        /*creamos variables necesarias para el prog y resto de validaciones*/
        Integer auxCodIns = Integer.parseInt(txtCodIns.getText());
        String query = "";
        ResultSet existe;
        try {
            sentencia = (Statement) conexion.createStatement();
            query = "SELECT * FROM insumos WHERE cod_insumo = " + auxCodIns;
            existe = sentencia.executeQuery(query);
            if (existe.next() == false) {
                JOptionPane.showMessageDialog(null, "ERROR!NO EXISTE insumo.",
                "¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
                sw = 1;
            } else {
                }
            } catch (SQLException e) {
                sw = 1;
            }
        }
        swAceptar = sw;
        return swAceptar;
    }
    public Integer validarBtnAceptarPro() {
        Integer swAceptar = 0;
        Integer sw = 0;
        if (txtCodIns.getText().isEmpty() || txtCantidad.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "ERROR! Campos vacios.", "¡ADVERTENCIA!",
            JOptionPane.INFORMATION_MESSAGE);
            sw = 1;
        } else {
        }
        if (sw == 0) {
            /*en esta segunda validacion necesitamos que exista el rutprov por lo que debe eser 0*/
            /*creamos variables necesarias para el prog y resto de validaciones*/
            Integer auxCodIns = Integer.parseInt(txtCodIns.getText());
            String query = "";
            ResultSet existe;
            try {
                sentencia = (Statement) conexion.createStatement();
                query = "SELECT * FROM producto WHERE cod_producto = " + auxCodIns;
                existe = sentencia.executeQuery(query);
                if (existe.next() == false) {
                    JOptionPane.showMessageDialog(null, "ERROR!NO EXISTE producto.",
                    "¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
                    sw = 1;
                } else {

```

```

        JOptionPane.showMessageDialog(null, "FELICIDADES EXISTE el producto .",
        "¡ADVERTENCIA!", JOptionPane.INFORMATION_MESSAGE);
    }
    } catch (SQLException e) {
        sw = 1;
    }
}
swAceptar = sw;
return swAceptar;
}
;
private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    Integer sw = 0;
    globalError = 0;
    swAntiRepet = 0;
    if (chkInsumo.isSelected()) {
        sw = validarBtnAceptarIns();
        if (sw == 0) {
            if (chkIngreso.isSelected()) {
                descontarAumentarInsumo(txtCodIns.getText(), 1,
Float.parseFloat(txtCantidad.getText()));
            }
            if (chkPerdida.isSelected()) {
                descontarAumentarInsumo(txtCodIns.getText(), 2,
Float.parseFloat(txtCantidad.getText()));
            }
            if (chkIngreso.isSelected() || chkPerdida.isSelected()) {
            } else {
            }
            limpiarCampos();
            limpiarTabla();
            llenarColI();
            llenarTablaI();

        }
    }
    if (chkProducto.isSelected()) {
        Integer auxCodProd2 = Integer.parseInt(txtCodIns.getText());
        sw = validarProdNegativo(auxCodProd2);

        if (sw == 0) {
            if (chkIngreso.isSelected() || chkPerdida.isSelected()) {
                Integer auxCodProd = Integer.parseInt(txtCodIns.getText());
                String auxCodIns = "";
                float auxCant = 0;
                if (chkIngreso.isSelected()) {
                    descontarAumentarInsumosAsociadoAProducto(auxCodIns, 1, auxCant);
                }
                if (chkPerdida.isSelected()) {

```

```

        descontarAumentarInsumosAsociadoAProducto(auxCodIns, 2, auxCant);
    }
} else {
    JOptionPane.showMessageDialog(null, msj, "ERROR NO HAY CHK
ACTIVADOS", JOptionPane.INFORMATION_MESSAGE);
}
limpiarTabla();
llenarColP();
llenarTablaP();
limpiarCampos();
}
}
// TODO add your handling code here:      // TODO add your handling code here:
}
public void limpiarTabla() {
    try {
        DefaultTableModel modelo = (DefaultTableModel) tblInsProd.getModel();
        int filas = tblInsProd.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
    TableColumnModel tcm = tblInsProd.getColumnModel();
    int filas = tblInsProd.getColumnCount();
    for (int i = 0; filas > i; i++) {
        TableColumn columnaABorrar = tcm.getColumn(0);
        tblInsProd.removeColumn(columnaABorrar);
    }
}
public void limpiarCampos() {
    txtCodIns.setText("");
    txtFiltro.setText("");
    txtCantidad.setText("");
}
public void descontarAumentarInsumosAsociadoAProducto(String paramCodPro, Integer
accionx, float cantidadx){
    Integer auxCodProd = Integer.parseInt(txtCodIns.getText());
    String auxCodIns = "";
    float auxCant = cantidadx;
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM detalle_producto
WHERE cod_producto='" + auxCodProd + "'");
        int cont = 0;
        while (lista.next()) {
            auxCodIns = lista.getString("cod_insumo");

```

```

        auxCant = (Float.parseFloat(lista.getString("cantidad"))) *
(Float.parseFloat(txtCantidad.getText()));
        descontarAumentarInsumo(auxCodIns, accionx, auxCant);
    }
} catch (SQLException e) {
    if (swAntiRepet == 0){
        msj = "ERROR numero demasiado alto para ser ingresado en el programa";
        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        swAntiRepet = 1;
    }
}
}

public void descontarAumentarInsumo(String paramCodIns, Integer accion, float cantidad) {
    int switchNoNegativo = 0;
    String auxCodIns = paramCodIns;
    float auxMontoBD = 0;
    float auxMontoForm = cantidad;
    float auxMontoNuevo = 0;
    String query = "";
    String auxTipoAjuste = "X";
    String auxObs = "Nada";
    try {
        sentencia = (Statement) conexion.createStatement();
        ResultSet lista = sentencia.executeQuery("SELECT * FROM insumos where cod_insumo
=" + auxCodIns + "");
        int cont = 0;
        while (lista.next()) {
            auxMontoBD = Float.parseFloat((lista.getString("stock_actual")));
            if (accion == 1) {
                auxMontoNuevo = auxMontoBD + auxMontoForm;
                auxTipoAjuste = "D";
                auxObs = "Donacion";
            }
            if (accion == 2) {
                auxMontoNuevo = auxMontoBD - auxMontoForm;
                auxTipoAjuste = "M";
                auxObs = "Merma";
            }
            if (auxMontoNuevo <= 0 && chkPerdida.isSelected()) {
                if (chkInsumo.isSelected()){
                    switchNoNegativo = 1;
                    if (globalError == 0){
                        JOptionPane.showMessageDialog(null, "Error no se puede eliminar una cantidad que
no dispone", "ERROR", JOptionPane.INFORMATION_MESSAGE);
                        globalError = 1;
                    }
                }
            }
        }
    }
}

```

```

        cont = cont + 1;
    }
} catch (SQLException e) {
    if (swAntiRepet == 0){
        msj = "ERROR numero demasiado alto para el programa";
        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        swAntiRepet = 1;
    }
}
if (switchNoNegativo == 0) {
    ResultSet existe;
    try {
        query = "SELECT * FROM insumos where cod_insumo =" + auxCodIns + "";
        existe = sentencia.executeQuery(query);
        if (existe.next() == true) {
            String sql = "UPDATE insumos SET stock_actual=" + auxMontoNuevo + "WHERE
cod_insumo =" + auxCodIns + "";
            try {
                sentencia.executeUpdate(sql);
            } catch (SQLException e) {
                if (swAntiRepet == 0){
                    msj = "ERROR numero demasiado alto para el programa";
                    JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
                    swAntiRepet = 1;
                }
            }
            msj = "Modificado";
        } else {
            msj = "Error , ingreso una cantidad que no puede existir";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (SQLException e) {
        if (swAntiRepet == 0){
            msj = "ERROR numero demasiado alto para el programa";
            JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
            swAntiRepet = 1;
        }
    }
    Integer auxCodAjuste = calcCodAjuste();
    String auxFechaAjuste = lblFecha2.getText();
    String sql = "INSERT INTO ajuste_stock VALUES (" + auxCodAjuste + "," + auxCodIns
+ "," + auxMontoForm + "," + auxFechaAjuste + "," + auxTipoAjuste + "," + auxObs + ")";
    insertar(sql);
    limpiarTabla();
    if (swAntiRepet == 0){
        msj = "Datos guardados";
    }
}

```

```

        JOptionPane.showMessageDialog(null, msj, "INFORMACION",
JOptionPane.INFORMATION_MESSAGE);
        swAntiRepet = 1;
    }
}
}
private void tblInsProdMouseClicked(java.awt.event.MouseEvent evt) {
    if (tblInsProd.getRowCount() > 0) {
        int sel = tblInsProd.getSelectedRow();
        Object cod = tblInsProd.getValueAt(sel, 0);
        String auxCode = cod.toString();
        txtCodIns.setText(auxCode);
    }
}
private void txtCodInsActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void chkProductoActionPerformed(java.awt.event.ActionEvent evt) {
    if (chkInsumo.isSelected()) {
        chkInsumo.setSelected(false);
    }
    if (chkProducto.isSelected()) {
        chkProducto.setSelected(true);
    }
    else{
        chkProducto.setSelected(true);
    }
    limpiarTabla();
    llenarColP();
    llenarTablaP();
    // TODO add your handling code here:
}
private void chkInsumoActionPerformed(java.awt.event.ActionEvent evt) {
    if (chkProducto.isSelected()) {
        chkProducto.setSelected(false);
    }
    if (chkInsumo.isSelected()) {
        chkInsumo.setSelected(true);
    }
    else{
        chkInsumo.setSelected(true);
    }
    limpiarTabla();
    TableColumn tcCod = new TableColumn(0);
    TableColumn tcNom = new TableColumn(1);
    TableColumn tcStockA = new TableColumn(2);
    tblInsProd.addColumn(tcCod);
    tblInsProd.addColumn(tcNom);
    tblInsProd.addColumn(tcStockA);
}

```

```

llenarTablaI(); // TODO add your handling code here:
}
private void chkIngresoActionPerformed(java.awt.event.ActionEvent evt) {
    if (chkPerdida.isSelected()) {
        chkPerdida.setSelected(false);
    } // TODO add your handling code here:
    if (chkIngreso.isSelected()) {
        chkIngreso.setSelected(true);
    }
    else{
        chkIngreso.setSelected(true);
    }
}
private void chkPerdidaActionPerformed(java.awt.event.ActionEvent evt) {
    if (chkIngreso.isSelected()) {
        chkIngreso.setSelected(false);
    }
    if (chkPerdida.isSelected()) {
        chkPerdida.setSelected(true);
    }
    else{
        chkPerdida.setSelected(true);
    }
}

private void txtCodInsKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 5;
    if (txtCodIns.getText().length() >= lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if (c < '0' || c > '9') {
        evt.consume();
    }
}
private void txtCantidadKeyTyped(java.awt.event.KeyEvent evt) {
    int lar = 4;
    if (txtCantidad.getText().length() > lar) {
        evt.consume();
    }
    char c = evt.getKeyChar();
    if ((c < '0' || c > '9') && (c != '.')) {
        evt.consume();
    }
}
private void txtFiltroKeyTyped(java.awt.event.KeyEvent evt) {
    if (chkInsumo.isSelected()) {
        txtFiltro.addKeyListener(new KeyAdapter() {
            public void keyReleased(final KeyEvent e) {

```

```

        String cadena = (txtFiltro.getText());
        txtFiltro.setText(cadena);
        repaint();
        filtro();
    }
});
    trsFiltro = new TableRowSorter(tblInsProd.getModel());
    tblInsProd.setRowSorter(trsFiltro);
}
}
private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
limpiarCampos();
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(formPerdida.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(formPerdida.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(formPerdida.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(formPerdida.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new formPerdida().setVisible(true);

```

```

    }
    });
}
// Variables declaration - do not modify
private javax.swing.JButton btnAceptar;
private javax.swing.JButton btnCancelar;
private javax.swing.JButton btnRegresar;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JCheckBox chkIngreso;
private javax.swing.JCheckBox chkInsumo;
private javax.swing.JCheckBox chkPerdida;
private javax.swing.JCheckBox chkProducto;
private javax.swing.JComboBox<String> cmbFiltro;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JLabel lblBuscarPor;
private javax.swing.JLabel lblCantidad;
private javax.swing.JLabel lblCodigoIns;
private javax.swing.JLabel lblFecha;
private javax.swing.JTextField lblFecha2;
private javax.swing.JTable tblInsProd;
private javax.swing.JTextField txtCantidad;
private javax.swing.JTextField txtCodIns;
private javax.swing.JTextField txtFiltro;
// End of variables declaration
}

```

Anexo 7***formInRangoVenta.java**

```

import java.awt.Color;
import java.awt.Toolkit;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
public class formInRangoVenta extends javax.swing.JFrame {
    private Statement sentencia;
    private Connection conexion;
    private String msj;
    private String nomBD = "bussandwicheria";
    private String usuario = "root";
    private String password = "informatica";
    public formInRangoVenta() {
        initComponents();
        this.setLocationRelativeTo(null);
        setIcon();
        conectar();
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
        LocalDate localDate = LocalDate.now();
        txtFecha.setText(dtf.format(localDate));
        lblDesde.setVisible(false);
        lblHasta.setVisible(false);
        dcDesde.setVisible(false);
        dcHasta.setVisible(false);
        btnAceptar.setVisible(false);
        btnCancelar.setVisible(false);
        jLabel12.setVisible(false);
        this.getContentPane().setBackground(Color.orange);
        rbtnDiurno.setVisible(false);
        rbtnVespertino.setVisible(false);
        rbtnAmbos.setVisible(false);
    }
    public void conectar() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            String url = "jdbc:mysql://localhost:3306/" + this.nomBD;
            this.conexion = (Connection) DriverManager.getConnection(url, this.usuario,
this.password);
            this.sentencia = (Statement) this.conexion.createStatement();
        } catch (Exception e) {
            msj = "error al conectar";
        }
    }
}

```

```

private void setIcon() {
setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/imagen/icono.png"
)));
}
public void vaciarListadoI() {
try {
DefaultTableModel modelo = (DefaultTableModel) tblIng.getModel();
int filas = tblIng.getRowCount();
for (int i = 0; filas > i; i++) {
modelo.removeRow(0);
}
} catch (Exception e) {
JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
}
}
public void vaciarListadoE() {
try {
DefaultTableModel modelo = (DefaultTableModel) tblEg.getModel();
int filas = tblEg.getRowCount();
for (int i = 0; filas > i; i++) {
modelo.removeRow(0);
}
} catch (Exception e) {
JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
}
}
private String CalcularPrimero() {
String primero = "";
Calendar c1 = Calendar.getInstance();
String mes = Integer.toString(c1.get(Calendar.MONTH));
String anno = Integer.toString(c1.get(Calendar.YEAR));
System.out.println("mes " + mes + " agno " + anno);
primero = anno + "/" + mes + "/" + "01";
System.out.println(primero);
return primero;
}
private void llenarEgresos(String fechaA, String fechaB) {
try {
sentencia = (Statement) conexion.createStatement();
ResultSet lista = sentencia.executeQuery("SELECT
DISTINCT(cod_producto_insumo),valor_neto FROM detalle_registro_compra WHERE
fecha_registro_compra BETWEEN '" + fechaA + "' AND '" + fechaB + "'");
int cont = 0;
while (lista.next()) {
int auxCodPro = (lista.getInt("cod_producto_insumo"));
int auxPre = (lista.getInt("valor_neto"));
int auxCant = 0;
String auxNom = "";
try {
sentencia = (Statement) conexion.createStatement();
String queryz = "SELECT * from insumos";
ResultSet listaz = sentencia.executeQuery(queryz);
while (listaz.next() == true) {
String nomIns = (listaz.getString("nombre_insumo"));
int idIns = (listaz.getInt("cod_insumo"));

```

```

        if (auxCodPro == idIns) {
            auxNom = nomIns;
        }
    }
} catch (Exception e) {
}
try {
    sentencia = (Statement) conexion.createStatement();
    String queryx = "SELECT * FROM detalle_registro_compra WHERE
cod_producto_insumo =" + auxCodPro;
    int cantidad = 0;
    ResultSet listax = sentencia.executeQuery(queryx);
    while (listax.next()) {
        int CI = (listax.getInt("cod_producto_insumo"));
        int canti = (listax.getInt("cantidad"));
        if (auxCodPro == CI) {
            cantidad = cantidad + canti;
        }
    }
    auxCant = cantidad;
} catch (Exception e) {
}
DefaultTableModel modelo = (DefaultTableModel) tblEg.getModel();
modelo.addRow(new Object[]{auxCodPro, auxNom, auxPre, auxCant});
cont = cont + 1;
}
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
}
private void llenarIngresos(String fechaA, String fechaB, String jornada) {
    if (jornada.equals("A")) {
        try {
            sentencia = (Statement) conexion.createStatement();
            ResultSet lista = sentencia.executeQuery("SELECT DISTINCT(cod_producto),precio
FROM detalle_pedido WHERE fecha_venta BETWEEN '" + fechaA + "' AND '" + fechaB + "'");
            int cont = 0;
            while (lista.next()) {
                int auxCodPro = (lista.getInt("cod_producto"));
                int auxPre = (lista.getInt("precio"));
                int auxCant = 0;
                String auxNom = "";
                try {
                    sentencia = (Statement) conexion.createStatement();
                    String queryz = "SELECT * from producto";
                    ResultSet listaz = sentencia.executeQuery(queryz);
                    while (listaz.next() == true) {
                        String nomIns = (listaz.getString("nombre_producto"));
                        int idIns = (listaz.getInt("cod_producto"));
                        if (auxCodPro == idIns) {
                            auxNom = nomIns;
                        }
                    }
                }
            } catch (Exception e) {
            }
        }
    }
}

```

```

        sentencia = (Statement) conexion.createStatement();
        String queryx = "SELECT * FROM detalle_pedido WHERE cod_producto =" +
auxCodPro;
        int cantidad = 0;
        ResultSet listax = sentencia.executeQuery(queryx);
        while (listax.next()) {
            int CI = (listax.getInt("cod_producto"));
            int canti = (listax.getInt("cantidad"));
            if (auxCodPro == CI) {
                cantidad = cantidad + canti;
            }
        }
        auxCant = cantidad;
    } catch (Exception e) {
    }
    DefaultTableModel modelo = (DefaultTableModel) tblIng.getModel();
    modelo.addRow(new Object[]{auxCodPro, auxNom, auxPre, auxCant});
    cont = cont + 1;
}
} catch (SQLException e) {
    msj = "no se pudo seleccionar";
}
}
}
}
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */

```

```

private void btnAtrasActionPerformed(java.awt.event.ActionEvent evt) {
    //new fLogin().setVisible(true);
    this.setVisible(false);
    this.dispose();
}
private void rbtnMesActionPerformed(java.awt.event.ActionEvent evt) {
    lblDesde.setVisible(false);
    lblHasta.setVisible(false);
    dcDesde.setVisible(false);
    dcHasta.setVisible(false);
    btnAceptar.setVisible(false);
    btnCancelar.setVisible(false);
    vaciarListadoI();
    vaciarListadoE();
    String jor = "";
    String Afecha = CalcularPrimero();
    String BFecha = txtFecha.getText();
    if (rbtnDiurno.isSelected()) {
        jor = "D";
    }
    if (rbtnVespertino.isSelected()) {
        jor = "V";
    }
    if (rbtnAmbos.isSelected()) {
        jor = "A";
    }
}

```

```

    llenarIngresos(Afecha, BFecha, "A");
    llenarEgresos(Afecha, BFecha);
}
private void rbtnRangoActionPerformed(java.awt.event.ActionEvent evt) {
    lblDesde.setVisible(true);
    lblHasta.setVisible(true);
    dcDesde.setVisible(true);
    dcHasta.setVisible(true);
    btnAceptar.setVisible(true);
    btnCancelar.setVisible(true);
}
private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    String hoy = txtFecha.getText();
    String fecha1 = "";
    int sw = 0;
    String jor = "";
    if (rbtnDiurno.isSelected()) {
        jor = "D";
    }
    if (rbtnVespertino.isSelected()) {
        jor = "V";
    }
    if (rbtnAmbos.isSelected()) {
        jor = "A";
    }
    vaciarListadoI();
    vaciarListadoE();
    try {
        String formato = "yyyy/MM/dd";
        //Formato
        Date date = dcDesde.getDate();
        SimpleDateFormat sdf = new SimpleDateFormat(formato);
        fecha1 = (sdf.format(date));
    } catch (NullPointerException ex) {
        JOptionPane.showMessageDialog(this, "Al menos selecciona una fecha válida!", "Error!",
JOptionPane.INFORMATION_MESSAGE);
    }
    String fecha2 = "";
    try {
        String formato = "yyyy/MM/dd";
        //Formato
        Date date = dcHasta.getDate();
        SimpleDateFormat sdf = new SimpleDateFormat(formato);
        fecha2 = (sdf.format(date));
    } catch (NullPointerException ex) {
        JOptionPane.showMessageDialog(this, "Al menos selecciona una fecha válida!", "Error!",
JOptionPane.INFORMATION_MESSAGE);
    }
    int mayor = (fecha1.compareTo(fecha2));
    if (mayor > 0) {
        JOptionPane.showMessageDialog(null, "'Fecha inicial' supuera a 'Fecha final'",
"ADVERTENCIA", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    }
    int hoyM = (fecha1.compareTo(hoy));
    if (hoyM > 0) {

```

```

        JOptionPane.showMessageDialog(null, "Fecha inicial' supuera a Fecha actual",
"ADVERTENCIA", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    }
    int hoyMF = (fecha2.compareTo(hoy));
    if (hoyMF > 0) {
        JOptionPane.showMessageDialog(null, "Fecha final' supuera a Fecha actual",
"ADVERTENCIA", JOptionPane.INFORMATION_MESSAGE);
        sw = 1;
    }
    llenarIngresos(fecha1, fecha2, "A");
    llenarEgresos(fecha1, fecha2);
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int largo = tblIng.getRowCount();
    int total = 0;
    for (int i = 0; i < largo; i++) {
        Object pre = tblIng.getValueAt(i, 2);
        int calPrecio = Integer.parseInt(pre.toString());

        Object cant = tblIng.getValueAt(i, 3);
        int calCant = Integer.parseInt(cant.toString());
        int subtotal = calPrecio * calCant;
        total = total + subtotal;
    }
    txtTotalIng.setText(Integer.toString(total));
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    int largo = tblEg.getRowCount();
    int total = 0;
    for (int i = 0; i < largo; i++) {
        Object pre = tblEg.getValueAt(i, 2);
        int calPrecio = Integer.parseInt(pre.toString());
        Object cant = tblEg.getValueAt(i, 3);
        int calCant = Integer.parseInt(cant.toString());
        int subtotal = calPrecio * calCant;
        total = total + subtotal;
    }
    txtTotalEg.setText(Integer.toString(total));
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    String ingre = txtTotalIng.getText();
    String egre = txtTotalEg.getText();
    if (ingre.equals("") || egre.equals("")) {
        if (ingre.equals("")) {
            JOptionPane.showMessageDialog(null, "Campo 'Total Ingresos' Vacio",
"ADVERTENCIA", JOptionPane.INFORMATION_MESSAGE);
        }
        if (egre.equals("")) {
            JOptionPane.showMessageDialog(null, "Campo 'Total Egresos' Vacio",
"ADVERTENCIA", JOptionPane.INFORMATION_MESSAGE);
        }
    } else {
        int ing = Integer.parseInt(txtTotalIng.getText());

```

```

        int eg = Integer.parseInt(txtTotalEg.getText());
        int ganancia = ing - eg;
        txtGanancia.setText(Integer.toString(ganancia));
    }
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(formInRangoVenta.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(formInRangoVenta.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(formInRangoVenta.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(formInRangoVenta.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    }
    //</editor-fold>
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new formInRangoVenta().setVisible(true);
        }
    });
}
// Variables declaration - do not modify
private javax.swing.JButton btnAceptar;
private javax.swing.JButton btnAtras;
private javax.swing.JButton btnCancelar;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.ButtonGroup buttonGroup2;
private com.toedter.calendar.JDateChooser dcDesde;
private com.toedter.calendar.JDateChooser dcHasta;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JLabel jLabel1;

```

```
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JSeparator jSeparator3;
private javax.swing.JLabel lblDesde;
private javax.swing.JLabel lblHasta;
private javax.swing.JRadioButton rbtnAmbos;
private javax.swing.JRadioButton rbtnDiurno;
private javax.swing.JRadioButton rbtnMes;
private javax.swing.JRadioButton rbtnRango;
private javax.swing.JRadioButton rbtnVespertino;
private javax.swing.JTable tblEg;
private javax.swing.JTable tblIng;
private javax.swing.JTextField txtFecha;
private javax.swing.JTextField txtGanancia;
private javax.swing.JTextField txtTotalEg;
private javax.swing.JTextField txtTotalIng;
// End of variables declaration
}
```