

UNIVERSIDAD TECNOLOGICA DEL PERU



TEMA: PROYECTO AVANCE FINAL

IMPLEMENTACIÓN DEL SISTEMA INFORMÁTICO PARA EL PROCESO DE ADMINISTRACION DE VENTAS DE "PAUL PIZZA"

CURSO: PROGRAMACIÓN ORIENTADA A OBJETOS

PROFESOR:

WILLABALDO MARCELINO ESTRADA ARO

Integrantes:

- **Khalil Alexander Bautista Paiva** **U20244885**

LIMA - PERÚ

2022

ÍNDICE

1.....	3
2. CAPÍTULO 1: ASPECTOS GENERALES	3
1. DEFINICIÓN DEL PROBLEMA:	3
1 1 La forma de realizar un comprobante de venta	3
1.2 Mal sistema de empleados	4
1.3 Lento sistema para sacar un reporte de ventas	4
DEFINICIÓN DE LOS OBJETIVOS:	4
2.1 OBJETIVO GENERAL:	4
2.2 OBJETIVOS ESPECÍFICOS:.....	4
2.3 ALCANCES Y LIMITACIONES.....	5
CAPÍTULO 2: MARCO TEÓRICO	6
2.4 Antecedentes Generales	6
2.5 <u>Fundamentos teóricos:</u>	8
CAPÍTULO 3	10
3.1. Requerimientos funcionales y no funcionales.....	10
3.1.1. Requerimientos funcionales	10
3.1.2. Requerimientos no funcionales	11
3.1.3. Prototipo del software (1 pantalla por cada RF).....	11
3.1.4. Diagrama de Clases	23
4. Conclusiones	60
Bibliografía:	67

1.

2. CAPÍTULO 1: ASPECTOS GENERALES

1. DEFINICIÓN DEL PROBLEMA:

En este estudio, estamos considerando ejecutar realmente el proceso de gestión de ventas de pizzas. Las empresas del rubro Pizzería que se especializan en la venta de alimentos situada en Lurigancho chosica tiene problemas con la administración del negocio, sistema antiguo que hace que todo sea más difícil, la venta de los productos, el manejo de los empleados y por último el control de los productos.

Por lo tanto, este proyecto busca encontrar posibles soluciones a los problemas que existen en muchas estructuras. Configurando el sistema y sacarle el máximo provecho. Sin embargo, para llegar a este punto de la posible solución, primero dividimos y analizamos los procesos más relevantes que ya se están realizando de forma manual y lenta, de la siguiente manera.

1

DESCRIPCIÓN DEL PROBLEMA:

Actualmente, el restaurante cuenta con un Sistema en malas condiciones para administrar el negocio y esto puede desencadenar diferentes pérdidas monetarias, pero sobre todo de clientes, ya que debido a un error en el se genera algunos problemas para el restaurante y esto produciría reclamos al restaurante, pérdida de tiempo y deteriorar la imagen de la pizzería.

En el Área de venta:

1.1 *La forma de realizar un comprobante de venta*

Cuando el cliente se dirige a la caja para cancelar lo que pidió, la cajera llena una boleta o factura manualmente en una hoja y eso provoca tener un sistema desorganizado, ya que esa boleta al no estar registrada se podría perder, creando un problema en el sistema financiero de la pizzería.

1.2 *Mal sistema de empleados*

La empresa no tiene un organizador para los empleados, tiene dificultades para poder controlar a sus empleados, ya que no hay sistema que monitoree a sus empleados, como por ejemplo cuando días trabajo, si falto por motivos personales, o si falto un día y no especificó, cuántas pizzas vendió, o quien estaba en la caja o preparando la pizza.

1.3 *Lento sistema para sacar un reporte de ventas*

Para el informe de ventas diarias, semanales y mensuales, la empresa tiene un método lento, antiguo, que hace que todo sea más trabajoso.

2 DEFINICIÓN DE LOS OBJETIVOS:

Teniendo en cuenta los problemas anteriores, se han propuesto varios objetivos que se describen en detalle a continuación para comprender los objetivos de este proyecto.

2.1 *OBJETIVO GENERAL:*

Diseñar e implementar un sistema informático para el proceso de ventas de la pizzería “PAUL PIZZA” en el distrito de Lurigancho, con la finalidad de mejorar el área de administrativa y atención al cliente.

2.2 *OBJETIVOS ESPECÍFICOS:*

- *Implementar el sistema en la pizzería que permita reducir el tiempo de atención al cliente.*

- *El sistema podrá administrar mejor el área de ventas para ser guardados en el sistema facilitando así el proceso que actualmente lo están realizando manualmente.*

- *Poder administrar los reportes de ventas del día, mensuales y anuales y así tener las finanzas en orden.*

2.3 ALCANCES Y LIMITACIONES

ALCANCES:

Atreves de la implementación del software. El gerente podrá administrar mejor el sistema de ventas, reportes y empleados, por lo cual. tendrá un sistema más eficiente en su rubro administrativo.

Son:

- El sistema podrá buscar los productos, facturas y reportes.*
- Ingresar el producto con todas sus características necesarias para su venta del día.*
- El usuario podrá eliminar los productos que ya no estén en stock o un producto que ya no esté en venta, de igual manera, podrá eliminar el registro de algún cliente que ya no crea conveniente que esté en el sistema registrado.*
- El sistema tendrá un filtro de usuarios como un vendedor y administrador la cual tienen diferentes opciones al poder manejar la aplicación.*
- Tendrá la opción de mantenimiento de usuarios por lo cual el administrador podrá modificar o eliminar algún vendedor para que no pueda entrar al sistema.*
- El usuario podrá anular alguna venta que por equivocación se realizó.*

LIMITACIONES:

El aplicativo no presentará una función para crearle una cuenta al cliente. Por lo cual, eso demandaría que el sistema se colapse o que presente problemas en su sistema de pedidos,cobros.

3 CAPÍTULO 2: MARCO TEÓRICO

12.1 Antecedentes Generales

Marco internacional:

- *Burgos (2018) realizó la siguiente investigación “Desarrollo de un sistema web para la gestión de pedidos en un restaurante.Aplicación a un caso de estudio” en la escuela Politécnica Nacional Facultad de Ingeniería de Sistemas Quito Ecuador. Denominó a su sistema con el nombre de SYSPER (Sistema de Pedidos para Restaurantes), mismo que permitió gestionar los pedidos de una manera rápida, segura y amigable con el cliente.*
- *Mora (2019) desarrolló la siguiente investigación “Software de gestión de productos en el restaurante Alejho de Melgar,Tolima” en la Corporación Universitaria Minuto de Dios Facultad de Ingeniería de Software Cundinamarca Colombia. Estableció un sistema de control interno que permite al administrador tener pleno conocimiento de los movimientos financieros del restaurante, generando más beneficios y fiabilidad de los resultados obtenidos generando un impacto influyente a la hora de minimizar tiempo y costos en la realización de las actividades diarias del restaurante.*
- *Hott & Toro (2011) elaboró la siguiente investigación “Sistema para la implementación masiva de delivery online de comida” en la universidad Técnico Federico Santa María Departamento de electrónica Valparaíso Chile. Permitió a cualquier restaurante implementar su propio servicio de reparto a domicilio online facilitará una interfaz web para sus clientes y así, estos puedan realizar pedidos en internet. En la cual la conclusión es lograr construir, tanto para el lado del cliente como para el administrador, que poseen interfaces intuitivas y que minimizan los posibles errores cometidos por los usuarios.*

Marco Nacional

- *En la actualidad, debido a la reciente recuperación económica y al panorama positivo y optimista que atraviesa la economía peruana, empresarios extranjeros y peruanos han invertido en la expansión y creación de nuevos negocios en el sector de las comidas rápidas. Ya que ven en el país un mercado prometedor, a causa de su rápida expansión y evolución en este sector. Siendo este, bajo la modalidad de franquicia, el segundo sector con mayor crecimiento en el país.*
- *El sector de las pizzas en la ciudad de Lima había tenido a través de los años un lento dinamismo, puesto que ésta no contaba con la infraestructura adecuada y la ciudad era muy pequeña. Sin embargo, importantes compañías como 500 Grados y La Linterna Pizza han tenido presencia en la ciudad. Teniendo estas dos últimas una larga y exitosa trayectoria, y esto se debe a su excelente sistema de ventas que facilita el proceso. Por eso es importante la implementación de un sistema de ventas para la empresa “Paul Pizza”, para facilitar el trabajo y tener toda la información accesible y disponible.*
- *En PERÚ como en muchos países emergentes, el sector de las comidas rápidas se está convirtiendo en un polo de desarrollo comercial y turístico que ha permitido que el mercado de las comidas rápidas se encuentre en un proceso de maduración y crecimiento, con un futuro prometedor y enorme potencial.*

Fundamentos teóricos:

¿Qué es Java?

Java es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas de punta.

Una de las principales características por las que Java se ha hecho muy famoso, es que es un lenguaje independiente de la plataforma. Eso quiere decir que si se hace un programa en Java podrá funcionar en cualquier ordenador del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc. Esto lo consigue porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente.

¿Qué es Netbeans?

Es un programa que sirve como IDE (un entorno de desarrollo integrado) que nos permite programar en diversos lenguajes. El desarrollo de software se ha diversificado mucho basándonos en la cantidad de lenguajes que existen para la programación. Sin embargo, hay lenguajes que van imponiéndose como estándares, entre ellos tenemos a Java, PHP, HTML, C++, C#, Ruby.

El problema que se presenta a la mayoría de los programadores es contar con un entorno de desarrollo que sea completo, eficaz, fácil de usar y sea en lo posible gratuito. Todos esos requerimientos los podemos encontrar en NetBeans. NetBeans es ideal para trabajar con el lenguaje de desarrollo JAVA (y todos sus derivados), así como también nos ofrece un excelente entorno para programar en PHP. También se puede descargar una vez instalado NetBeans, los complementos para programar en C++. La IDE de NetBeans es perfecta. Tiene un excelente balance entre una interfaz con múltiples opciones y el editor puede autocompletar nuestro código.

¿Qué es Microsoft SQL Server?

Microsoft SQL Server es uno de los principales sistemas de gestión de bases de datos relacional del mercado que presta servicio a un amplio abanico de aplicaciones de software destinadas a la inteligencia empresarial y análisis sobre entornos corporativos.

Microsoft SQL Server es ideal para almacenar toda la información deseada en bases de datos relacionales, como también para administrar dichos datos sin complicaciones, gracias a su interfaz

visual y a las opciones y herramientas que tiene. Es algo vital, especialmente en webs que tienen la opción de registrar usuarios para que inicien sesión.

Para las compañías, emplear esta herramienta es esencial por las facilidades que plantea y las utilidades con las que cuenta. Si se tiene un listado de clientes, un catálogo de productos o incluso una gran selección de contenidos multimedia disponible, Microsoft SQL Server ayuda a gestionarlo absolutamente todo. Es básico para el buen funcionamiento de una web o de cualquier aplicación.

Su componente principal está compuesto por un motor relacional encargado del procesamiento de comandos, consultas, así como del almacenamiento de archivos, bb.dd., tablas y búferes de datos. Sus niveles secundarios están destinados a la gestión de la memoria, programación y administración de las interacciones de solicitud y respuesta con los servidores que alojan las bases de datos.

¿Qué es SGBD?

Es un manejador de bases de datos, cuya función es servir de interfaz entre la base de datos, el usuario y las distintas aplicaciones utilizadas. Como su propio nombre indica, el objetivo de los sistemas manejadores de base de datos es precisamente el de manejar un conjunto de datos para convertirlos en información relevante para la organización, ya sea a nivel operativo o estratégico.

Lo hace mediante una serie de rutinas de software para permitir su uso de una manera segura, sencilla y ordenada. Se trata, en suma, de un conjunto de programas que realizan tareas de forma interrelacionada para facilitar la construcción y manipulación de bases de datos, adoptando la forma de interfaz entre éstas, las aplicaciones y los mismos usuarios. Su uso permite realizar un mejor control a los administradores de sistemas y, por otro lado, también obtener mejores resultados a la hora de realizar consultas que ayuden a la gestión empresarial mediante la generación de la tan perseguida ventaja competitiva.

CAPÍTULO 3

DESARROLLO DE LA SOLUCIÓN

3.1. Requerimientos funcionales y no funcionales.

1. 3.1.1. Requerimientos funcionales

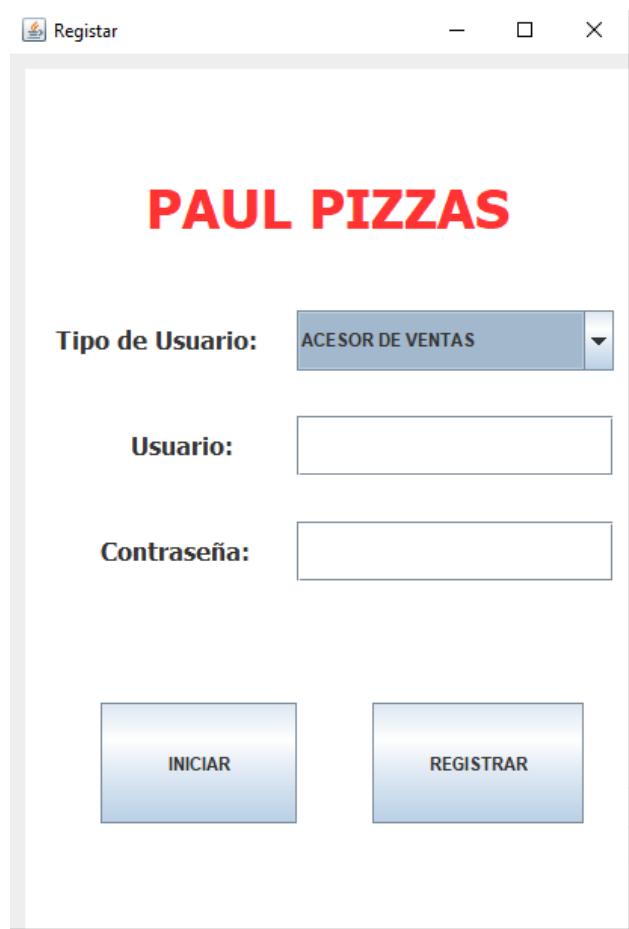
RF1	<i>El sistema debe permitir Ingresar al administrar y al empleado</i>
RF2	<i>El sistema permitirá registrar a los nuevos y antiguos empleados para que tengan acceso al sistema</i>
RF3	<i>El Sistema abrirá un menú para el administrador, donde podrá controlar todo el sistema y también a los empleados</i>
RF4	<i>El sistema abrirá un menú para el empleado, donde podrá controlar, las facturas, ventas, productos y registrar su reporte de cada día</i>
RF5	<i>El sistema permitirá al administrador y al empleado visualizar los productos disponibles en el negocio.</i>
RF6	<i>El sistema Permitirá al administrador y al empleado consultar las facturas registradas por los empleados.</i>
RF7	<i>El sistema permitirá al administrador y al empleado registrar la factura y su detalle</i>
RF8	<i>El sistema permitirá al administrador y al empleado consultar los detalles de la factura</i>
RF9	<i>El sistema permitirá realizar un reporte diario al administrador y al empleado, donde el empleado tendrá que registrar cada día los reportes y dinero del restaurante</i>
RF10	<i>El sistema permitirá realizar buscar reporte de una semana o cierta fecha solo al administrador</i>
RF11	<i>El sistema permitirá realizar búsquedas de reportes de un mes al administrador</i>
RF12	<i>El sistema permitirá al administrador, administrar los productos, verificar el stock de cada producto, modificar su precio, agregar o eliminar productos</i>
RF13	<i>El sistema permitirá al administrador gestionar y administrar a los empleados y verificar el rendimiento del empleado</i>

3.1.2. Requerimientos no funcionales

RNF1	<i>El sistema es construido con el lenguaje Java mediante Netbeans</i>
RNF2	<i>El sistema debe contar con una interfaz amigable para el usuario.</i>
RNF3	<i>El sistema debe responder ante un clic mínimo 0.3 milisegundos.</i>
RNF4	<i>La base de datos del sistema debe soportar un almacenamiento máximo de 10 T por año.</i>
RNF5	<i>El sistema es compatible con las versiones de Windows XP hacia adelante.</i>
RNF6	<i>El sistema debe tener una BD en MySQL.</i>

3.2. Prototipo del software (1 pantalla por cada RF).

RF1: Login para el administrador y empleado



RF2 Registro Usuario

REGISTRO USUARIO

Nombre de Usuario:

Nombre:

Apellido:

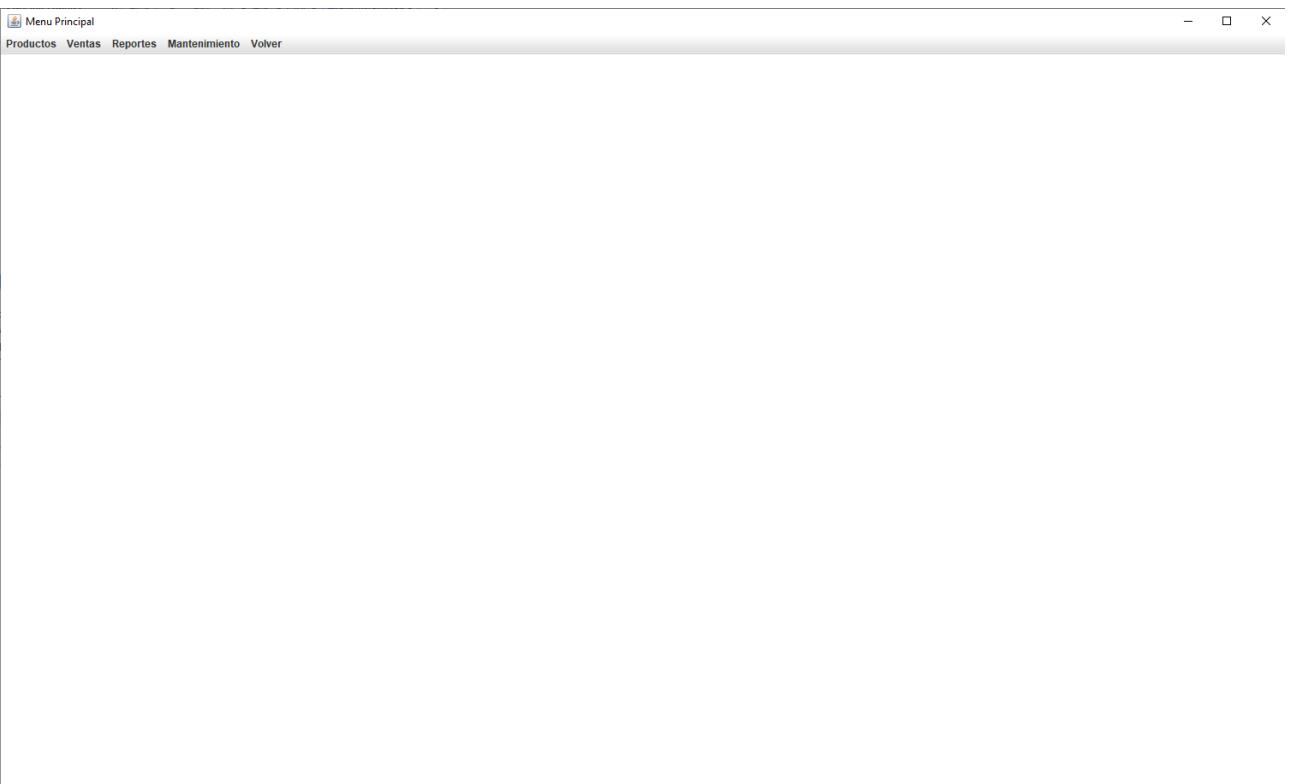
Email:

Contraseña:

Confirmar Contraseña:

Registrar **Cancelar**

RF3



RF4



RF5

Visualizar

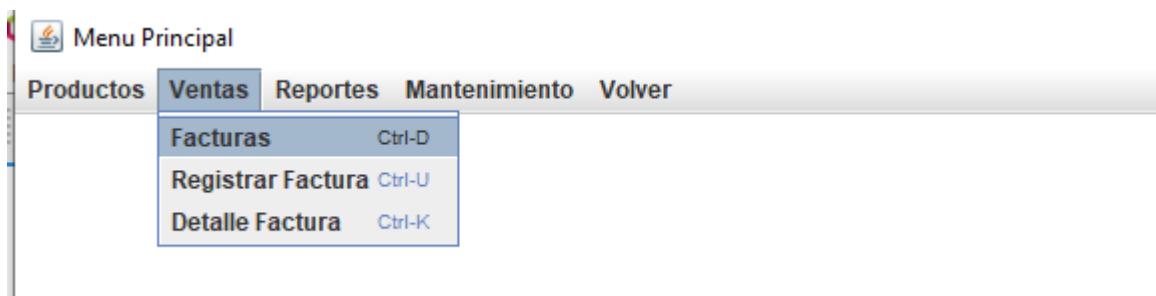
Productos

Cantidad de Productos : 78

Filtro Por Producto:

		Categoría	Stock	Tamaño
		ct1	s1	t1
1	pr1	Pizza Americana	\$12.00	s1
2	pr10	Pizza Delicia	\$16.00	s1
3	pr11	Pizza Choripizza	\$17.00	s1
4	pr12	Pizza Lomo Ahumado	\$120.00	s1
5	pr13	Pizza Paul Especial	\$25.00	s1
6	pr14	Pizza Americana	\$23.00	s1
7	pr15	Pizza Mozarella	\$24.00	s1
8	pr16	Pizza Jamon	\$25.00	s1
9	pr17	Pizza Salame	\$25.00	s1
10	pr18	Pizza Oriental	\$26.00	s1
11	pr19	Pizza Hawaina	\$26.00	s1
12	pr2	Pizza Mozarella	\$12.00	s1
13	pr20	Pizza Vegetariana	\$26.00	s1
14	pr21	Pizza tocino	\$26.00	s1
15	pr22	Pizza Francesa	\$40.00	s1
16	pr23	Pizza Americana	\$41.00	s1
17	pr24	Pizza Mozarella	\$42.00	s1
18	pr25	Pizza Jamon	\$43.00	s1
19	pr26	Pizza Salame	\$44.00	s1
20	pr27	Pizza Oriental	\$45.00	s1
21	pr28	Pizza Hawaina	\$44.00	s1

RF6



Facturas

Codigo Factura:

DNI Cliente:

Empleado: ▾

Fecha:

Total:

Registro de Facturas

Seleccione Fecha

Ingrese la factura

num	codigo Factura	DNI Cliente	Codigo empleado	Fecha	Total
1	23423456	13140430	2022-07-19	76.0	
2	23456435	13140428	2022-07-19	229.0	
3	23454323	13140436	2022-07-19	72.0	
4	02734532	13140428	2022-07-23	46.0	

Cantidad de Facturas : 4

RF7



Menu Principal

Productos Ventas Reportes Mantenimiento Volver

Facturas Ctrl-D

Registrar Factura Ctrl-U

Detalle Factura Ctrl-K

Registro Factura

REGISTRAR FACTURA

Codigo Producto: Nombre Producto: Cantidad: Precio: Agregar Producto

codigo Producto	Descripcion	Cantidad	Precio Unitario	Precio Total

Dni Cliente: Código Empleado: Bautista Total:
Fecha:

RF8



Productos Ventas Reportes Mantenimiento Volver

Facturas Ctrl-D
Registrar Factura Ctrl-U
Detalle Factura Ctrl-K

Detalle Factura

Registro del Detalle de la Facturas

Ingrese la factura

num	codigo detalle	Codigo producto	Descripcion	Cantidad	Precio Unitario	Precio Total	Codigo Factura
1	4	pr10	Pizza Delicia	3	16.0	48.0	1
2	5	pr50	Coca Cola	4	7.0	28.0	1
3	6	pr10	Pizza Delicia	3	16.0	48.0	2
4	7	pr50	Coca Cola	4	7.0	28.0	2
5	8	pr30	Pizza tocino	3	47.0	141.0	2
6	9	pr52	Coca Cola	1	12.0	12.0	2
7	10	pr10	Pizza Delicia	2	16.0	32.0	3
8	11	pr1	Pizza Americana	2	13.0	26.0	3
9	12	pr50	Coca Cola	2	7.0	14.0	3
10	13	pr10	Pizza Delicia	2	16.0	32.0	4
11	14	pr50	Coca Cola	2	7.0	14.0	4

Cantidad de Detalle de Facturas:

RF9

Menu Principal

- Productos
- Ventas
- [Reportes](#)
- Mantenimiento
- Volver

Empleado

- [Reporte Diario](#) Ctrl-C
- [Reporte Semanal](#) Ctrl-S
- [Reporte Mensual](#) Ctrl-M

Reporte Diario

num	NºReporte	Codigo emp.	Pizza Vend...	Pastas Ven...	Bebidas Ve...	Piqueos Ve...	Platos Crioll...	Fecha	Dinero Inicial	Dinero Final	Dinero Sobre...
1	1	123123	40	10	50	14	5	2022-07-06	\$/ 200,00	\$/ 500,00	\$/ 300,00
2	2	13140429	45	11	55	14	5	2022-05-30	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
3	3	13140429	43	12	55	14	5	2022-07-03	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
4	4	13140431	42	13	52	14	5	2022-07-04	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
5	5	123123	36	14	53	14	5	2022-07-02	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
6	6	13140429	47	15	35	14	5	2022-07-01	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
7	7	123123	50	16	55	14	5	2022-05-06	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
8	8	13140431	43	14	60	14	5	2022-06-06	\$/ 4200,00	\$/ 500,00	-\$/ 3700,00
9	9	13140429	42	13	53	14	5	2022-06-26	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
10	10	13140431	45	13	53	14	5	2022-05-06	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
11	11	123123	48	15	56	14	5	2022-07-10	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
12	12	13140431	50	17	57	14	5	2021-12-26	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
13	16	13140432	23	42	21	32	42	2022-07-09	\$/ 1000,00	\$/ 400,00	-\$/ 600,00

Reporte del dia

Cantidad de Reporte Diario : 13

RF10

Menu Principal

- Productos
- Ventas
- Reportes**
- Mantenimiento
- Volver

Empleado

- Reporte Diario Ctrl-C
- Reporte Semanal Ctrl-S
- Reporte Mensual Ctrl-M

Reporte Semanal

Filtro por Nº reporte:

Filtro por fecha:



Buscar **Restaurar**

num	NºReporte	Codigo em.	Pizza Vend.	Pastas Ven.	Bebidas Ve.	Piqueos Ve...	Platos Criol...	Fecha	Dinero Inicial	Dinero Final	Dinero Sob...
1	1	123123	40	10	50	14	5	2022-07-06	\$/200.00	\$/500.00	\$/300.00
2	2	13140429	45	11	55	14	5	2022-05-30	\$/1200.00	\$/500.00	\$/700.00
3	3	13140429	43	12	55	14	5	2022-07-03	\$/2200.00	\$/500.00	\$/1700.00
4	4	13140431	42	13	52	14	5	2022-07-04	\$/3200.00	\$/500.00	\$/2700.00
5	5	123123	36	14	53	14	5	2022-07-02	\$/1200.00	\$/500.00	\$/700.00
6	6	13140429	47	15	35	14	5	2022-07-01	\$/2200.00	\$/500.00	\$/1700.00
7	7	123123	50	16	55	14	5	2022-05-08	\$/3200.00	\$/500.00	\$/2700.00
8	8	13140431	43	14	60	14	5	2022-05-08	\$/4200.00	\$/500.00	\$/3700.00
9	9	13140429	42	13	53	14	5	2022-06-28	\$/2200.00	\$/500.00	\$/1700.00
10	10	13140431	45	13	53	14	5	2022-05-06	\$/1200.00	\$/500.00	\$/700.00
11	11	123123	48	15	56	14	5	2022-07-10	\$/3200.00	\$/500.00	\$/2700.00
12	12	13140431	50	17	57	14	5	2021-12-26	\$/2200.00	\$/500.00	\$/1700.00
13	16	13140432	23	42	21	32	42	2022-07-09	\$/1000.00	\$/400.00	\$/600.00

Cantidad de Reporte Diario

RF11

Menu Principal

Productos Ventas **Reportes** Mantenimiento Volver

Empleado

Reporte Diario Ctrl-C
Reporte Semanal Ctrl-S
Reporte Mensual Ctrl-M

Reporte Mensual

Busqueda Por Mes:

Busqueda Por Año:



Restaurar

num	NºReporte	Codigo em.	Pizza Vend.	Pastas Ven.	Bebidas Ve.	Piqueos Ve...	Platos Criol...	Fecha	Dinero Inicial	Dinero Final	Dinero Sob...
1	1	123123	40	10	50	14	5	2022-07-06	\$/200.00	\$/500.00	\$/300.00
2	2	13140429	45	11	55	14	5	2022-05-30	\$/1200.00	\$/500.00	\$/700.00
3	3	13140429	43	12	55	14	5	2022-07-03	\$/2200.00	\$/500.00	\$/1700.00
4	4	13140431	42	13	52	14	5	2022-07-04	\$/3200.00	\$/500.00	\$/2700.00
5	5	123123	36	14	53	14	5	2022-07-02	\$/1200.00	\$/500.00	\$/700.00
6	6	13140429	47	15	35	14	5	2022-07-01	\$/2200.00	\$/500.00	\$/1700.00
7	7	123123	50	16	55	14	5	2022-05-06	\$/3200.00	\$/500.00	\$/2700.00
8	8	13140431	43	14	60	14	5	2022-06-06	\$/4200.00	\$/500.00	\$/3700.00
9	9	13140429	42	13	53	14	5	2022-06-28	\$/2200.00	\$/500.00	\$/1700.00
10	10	13140431	45	13	53	14	5	2022-05-06	\$/1200.00	\$/500.00	\$/700.00
11	11	123123	48	15	56	14	5	2022-07-10	\$/3200.00	\$/500.00	\$/2700.00
12	12	13140431	50	17	57	14	5	2021-12-26	\$/2200.00	\$/500.00	\$/1700.00
13	16	13140432	23	42	21	32	42	2022-07-09	\$/1000.00	\$/400.00	\$/600.00

Cantidad de Reporte Diario

RF12

Menu Principal

Productos Ventas Reportes Mantenimiento Volver

Empleado Productos Ctrl-P Empleados Ctrl-E

PRODUCTOS

Codigo Producto:

Nombre Producto:

Precio:

Stock: Disponible

Categoría: Bebidas

Tamaño: 1 LT

Lista de Productos

Num	Codigo Producto	Nombre Producto	Precio	Stock	Categoría	Tamaño
1	pr1	Pizza Americana	S/ 12.00	s1	ct1	t1
2	pr2	Pizza Mozzarella	S/ 12.00	s1	ct1	t1
3	pr3	Pizza Jamon	S/ 14.00	s1	ct1	t1
4	pr4	Pizza Salame	S/ 13.00	s1	ct1	t1
5	pr5	Pizza Oriental	S/ 15.00	s1	ct1	t1

Agregar **Eliminar**

Consultar **Modificar**

Cantidad de Productos : 5

RF13

Menu Principal

Productos Ventas Reportes Mantenimiento Volver

Empleado Productos Ctrl-P Empleados Ctrl-E

Empleados

num	codigo empleado	Nombre	Apellido	Direccion	Telefono	Sueldo	Edad	Dias Trabajados
1	123123	khallil alexander	Huaman Acevedo	Av brasil 123	234567892	S/ 1000,00	23	23
2	12345678	Leomel Andres	Messi Villanueva	Av brasil 123	123456789	S/ 1200,00	23	13
3	23412367	Claudia	Aguirre	Av angeles 123	234542657	S/ 1200,00	25	8
4	73140427	roberto Javier	Montalvar messi	Av Brasil 1234	938434343	S/ 1250,00	26	6
5	73149247	Brayan	Medrano	santo chocano	983748347	S/ 950,00	25	25

Lista de Empleados

DNI Empleado:

Nombre:

Apellido:

Direccion:

Telefono:

Sueldo:

Edad:

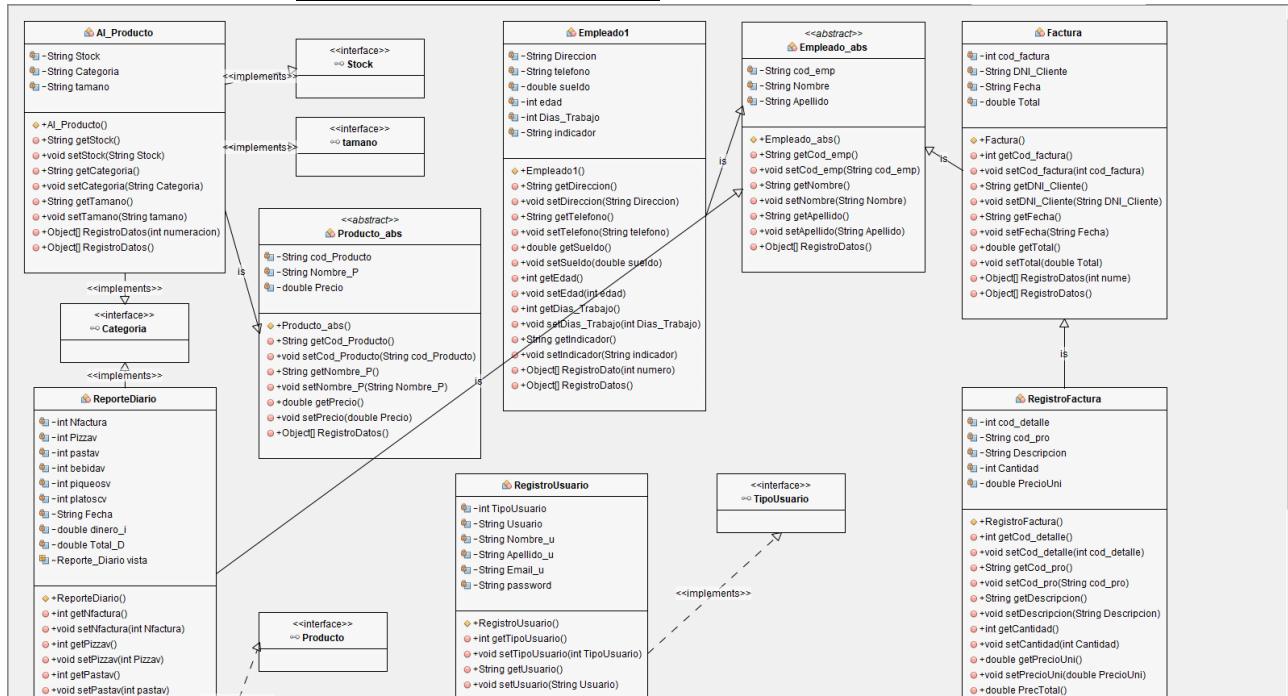
Dias Trabajado

Agregar **Modificar**

Consultar **Eliminar**

Cantidad de Empleados : 5

3.3. Diagrama de Clases



3.4 Códigos de la clases

3.4.1 código de la clase registro

```

1 package Modelo;
2
3
4 public class RegistroUsuario implements TipoUsuario {
5
6     //atributos
7     private int TipoUsuario;
8     private String Usuario;
9     private String Nombre_u;
10    private String Apellido_u;
11    private String Email_u;
12    private String password;
13
14    //constructor
15    public RegistroUsuario(){}
16    //getter y setter
17    public int getTipoUsuario() {return TipoUsuario;}
18    public void setTipoUsuario(int TipoUsuario) {this.TipoUsuario = TipoUsuario;}
19    public String getUsuario() {return Usuario;}
20    public void setUsuario(String Usuario) {this.Usuario = Usuario;}
21    public String getNombre_u() {return Nombre_u;}
22    public void setNombre_u(String Nombre_u) {this.Nombre_u = Nombre_u;}
23    public String getApellido_u() {return Apellido_u;}
24    public void setApellido_u(String Apellido_u) {this.Apellido_u = Apellido_u;}
25    public String getEmail_u() {return Email_u;}
26    public void setEmail_u(String Email_u) {this.Email_u = Email_u;}
27    public String getPassword() {return password;}
28    public void setPassword(String password) {this.password = password;}
29 }
30

```

3.4.2 clase producto

```

package Modelo;
import java.text.DecimalFormat;

public class Al_Producto extends Producto_abs implements Stock,tamano,Categoría {
    private String Stock;
    private String Categoría;
    private String tamano;
//Metodo
    public Al_Producto(){}
    //getter y setter
    public String getStock() {return Stock;}
    public void setStock(String Stock) {this.Stock = Stock;}
    public String getCategoría() {return Categoría;}
    public void setCategoría(String Categoría) {this.Categoría = Categoría;}
    public String getTamano() {return tamano;}
    public void setTamano(String tamano) {this.tamano = tamano;}
    //metodo
    public Object[] RegistroDatos(int numeracion) {
        DecimalFormat df = new DecimalFormat("S/ #0.00");
        Object[] fila = (numeracion,super.getCod_Producto(),super.getNombre_P(),df.format(super.getPrecio()))
                      ,this.getStock(),this.getCategoría(),this.getTamano());
        return fila;
    }
    @Override
    public Object[] RegistroDatos() {
        throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
    }
}

```

3.4.3 clase empleado

```

package Modelo;
import java.text.DecimalFormat;
public class Empleado1 extends Empleado_abs{
    //atributos
    private String Direccion;
    private String telefono;
    private double sueldo;
    private int edad;
    private int Dias_Trabajo;
    private String indicador;
    //Constructor
    public Empleado1(){}
    //getter y setter
    public String getDireccion() {return Direccion;}
    public void setDireccion(String Direccion) {this.Direccion = Direccion;}
    public String gettelefono() {return telefono;}
    public void settelefono(String telefono) {this.telefono = telefono;}
    public double getsueldo() {return sueldo;}
    public void setsueldo(double sueldo) {this.sueldo = sueldo;}
    public int getEdad() {return edad;}
    public void setEdad(int edad) {this.edad = edad;}
    public int getDias_Trabajo() {return Dias_Trabajo;}
    public void setDias_Trabajo(int Dias_Trabajo) {this.Dias_Trabajo = Dias_Trabajo;}
    public String getIndicador() {return indicador;}
    public void setIndicador(String indicador) {this.indicador = indicador;}
    //metodo
    public Object[] RegistroData(int numero) {
        DecimalFormat df = new DecimalFormat("S/ #0.00");
        Object[] fila = (numero,super.getCod_emp(),super.getNombre(),super.getApellido(),
                        this.getDireccion(),this.getTelefono(),df.format(this.getSueldo()),this.getEdad(),this.getDias_Trabajo());
        return fila;
    }
    @Override
    public Object[] RegistroDatos() {
        throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
    }
}

```

3.4.4 clase abstracta empleado

```

package Modelo;

public abstract class Empleado_abs {
    //atributos
    private String cod_emp;
    private String Nombre;
    private String Apellido;
    //metodo
    public Empleado_abs() {}

    //getter
    public String getCod_emp() {return cod_emp;}
    public void setCod_emp(String cod_emp) {this.cod_emp = cod_emp;}
    public String getNombre() {return Nombre;}
    public void setNombre(String Nombre) {this.Nombre = Nombre;}
    public String getApellido() {return Apellido;}
    public void setApellido(String Apellido) {this.Apellido = Apellido;}
    //metodo
    public abstract Object[] RegistroDatos();

}

```

3.4.5 clase factura

```

package Modelo;

import java.text.DecimalFormat;

public class Factura extends Empleado_abs{
    //atributos
    private String cod_factura;
    private String DNI_Cliente;
    private String descripcion;
    private String Fecha;
    private double Total;
    private static int contador=0;
    //constructor
    public Factura(){}
    //getter y setter
    public String getCod_factura() {return cod_factura;}
    public void setCod_factura(String cod_factura) {this.cod_factura = cod_factura;}
    public String getDNI_Cliente() {return DNI_Cliente;}
    public void setDNI_Cliente(String DNI_Cliente) {this.DNI_Cliente = DNI_Cliente;}
    public String getDescripcion() {return descripcion;}
    public void setDescripcion(String descripcion) {this.descripcion = descripcion;}
    public String getFecha() {return Fecha;}
    public void setFecha(String Fecha) {this.Fecha = Fecha;}
    public double getTotal() {return Total;}
    public void setTotal(double Total) {this.Total = Total;}
    //metodo
    public Object[] RegistroDatos(int nume) {
        DecimalFormat df = new DecimalFormat("S/ #0.00");
        Object[] fila = {nume,this.getCod_factura(),this.getDNI_Cliente(),super.getCod_emp(),this.getDescripcion(),this.getFecha(),df.format(this.getTotal())};
        return fila;
    }

    @Override
    public Object[] RegistroDatos() {
        throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
    }
}

```

3.4.6 clase abstracta producto

```

package Modelo;

public abstract class Producto_abs {
    private String cod_Producto;
    private String Nombre_P;
    private double Precio;
    public Producto_abs(){}
    // getter y setter
    public String getCod_Producto() {return cod_Producto;}
    public void setCod_Producto(String cod_Producto) {this.cod_Producto = cod_Producto;}
    public String getNombre_P() {return Nombre_P;}
    public void setNombre_P(String Nombre_P) {this.Nombre_P = Nombre_P;}
    public double getPrecio() {return Precio;}
    public void setPrecio(double Precio) {this.Precio = Precio;}
    //metodo
    public abstract Object[] RegistroDatos();
}

```

3.4.7 clase reporte diario

```

package Modelo;
//libreria
import vista.Reporte_Diario;
import vista.*;
import java.text.DecimalFormat;
public class ReporteDiario extends Empleado_abs implements Producto,Categoría{
    //atributos
    private int Nfactura;
    private int Pizzav;
    private int pastav;
    private int bebidav;
    private int piqueosv;
    private int platoscv;
    private String Fecha;
    private double dinero_i;
    private double Total_D;
    Reporte_Diario vista;
    //constructor
    public ReporteDiario(){}
    //getter y setter
    public int getNfactura() {return Nfactura;}
    public void setNfactura(int Nfactura) {this.Nfactura = Nfactura;}
    public int getPizzav() {return Pizzav;}
    public void setPizzav(int Pizzav) {this.Pizzav = Pizzav;}
    public int getPastav() {return pastav;}
    public void setPastav(int pastav) {this.pastav = pastav;}
    public int getBebidav() {return bebidav;}
    public void setBebidav(int bebidav) {this.bebidav = bebidav;}
    public int getPiqueosv() {return piqueosv;}
    public void setPiqueosv(int piqueosv) {this.piqueosv = piqueosv;}
    public int getPlatoscv() {return platoscv;}
    public void setPlatoscv(int platoscv) {this.platoscv = platoscv;}
    public String getFecha() {return Fecha;}
    public void setFecha(String Fecha) {this.Fecha = Fecha;}
    public double getDinero_i() {return dinero_i;}
    public void setDinero_i(double dinero_i) {this.dinero_i = dinero_i;}
    public double getTotal_D() {return Total_D;}
    public void setTotal_D(double Total_D) {this.Total_D = Total_D;}
    //metodo
    public double Sobrante(){ return Total_D - dinero_i;}
    public Object[] RegistroDatos(int numero) {
        DecimalFormat df = new DecimalFormat("S/ #0.00");
        double Sobrante=0;
        Object[] fila = {numero,this.getNfactura(),super.getCod_emp(),this.getPizzav(),this.getPastav(),this.getBebidav(),this.getPiqueosv(),this.getPlatoscv(),
        ,this.getFecha(),df.format(this.getDinero_i()),df.format(this.getTotal_D()),df.format(Sobrante())};
    }
}

```

3.4.8 clase RegistroFactura

```

package Modelo;

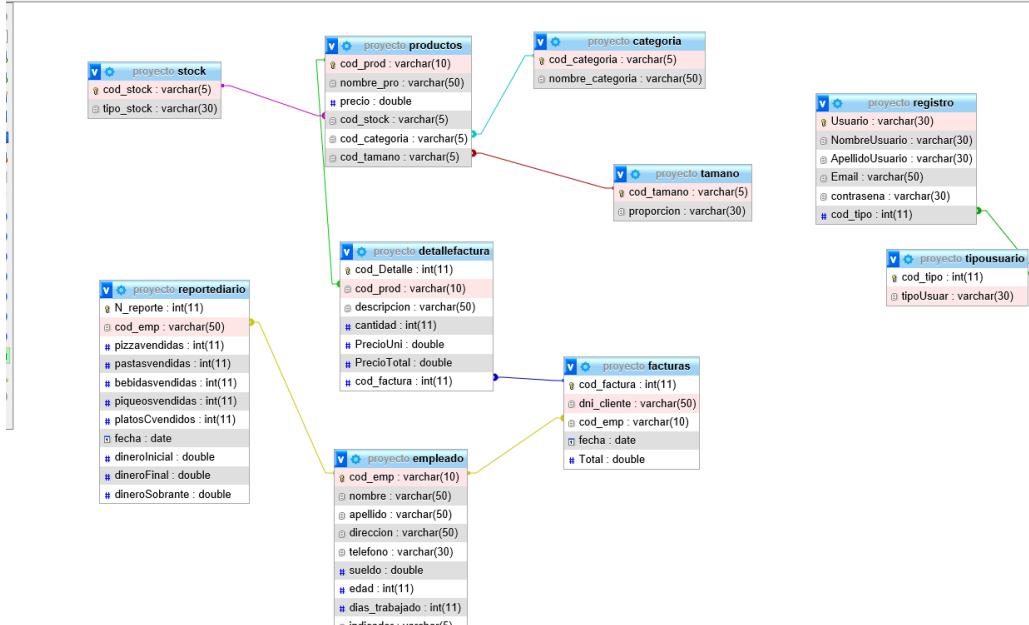
public class RegistroFactura extends Factura {
    //atributos
    private int cod_detalle;
    private String cod_pro;
    private String Descripcion;
    private int Cantidad;
    private double PrecioUni;
    public RegistroFactura(){
    }
    //getter y setter
    public int getCod_detalle() {return cod_detalle;}
    public void setCod_detalle(int cod_detalle) {this.cod_detalle = cod_detalle; }
    public String getCod_pro() {return cod_pro;}
    public void setCod_pro(String cod_pro) {this.cod_pro = cod_pro;}
    public String getDescripcion() { return Descripcion;}
    public void setDescripcion(String Descripcion) {this.Descripcion = Descripcion;}
    public int getCantidad() { return Cantidad;}
    public void setCantidad(int Cantidad) {this.Cantidad = Cantidad;}
    public double getPrecioUni() {return PrecioUni;}
    public void setPrecioUni(double PrecioUni) {this.PrecioUni = PrecioUni; }

    //metodo
    public double PrecTotal(){ return PrecioUni*Cantidad; }

    public Object[] RegistroDatosDetalle(int nume) {
        Object[] fila = {nume,this.getCod_detalle(),this.getCod_pro(),this.getDescripcion(),this.getCantidad(),this.getPrecioUni(),this.PrecTotal(),super.getCod_factura()};
        return fila;
    }
}

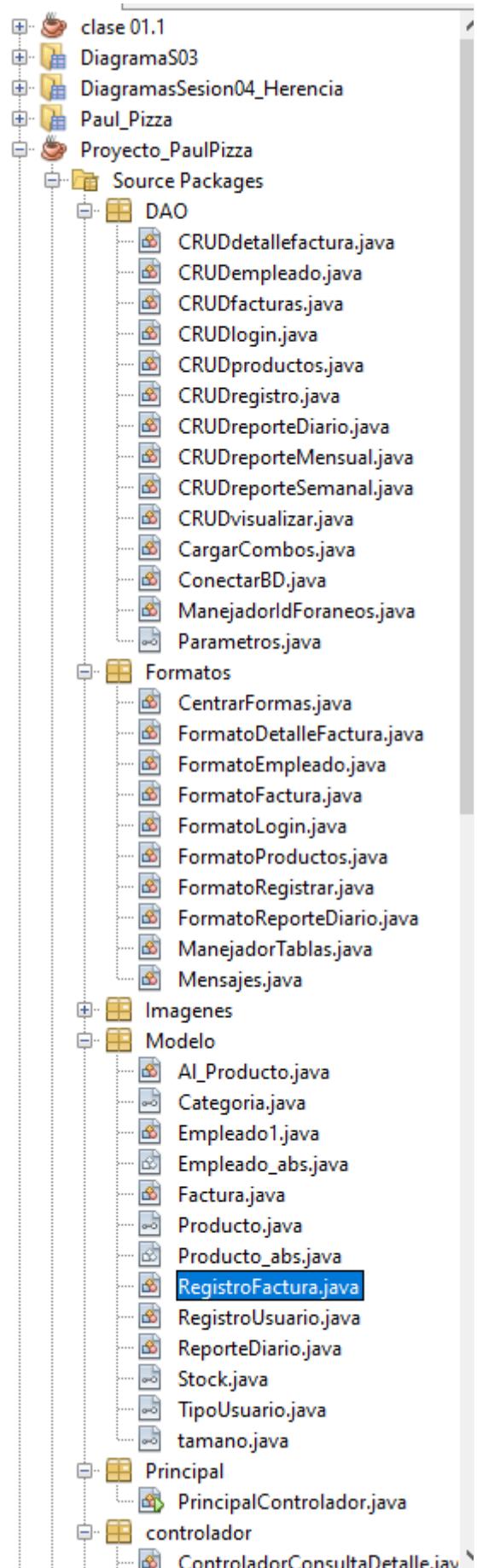
```

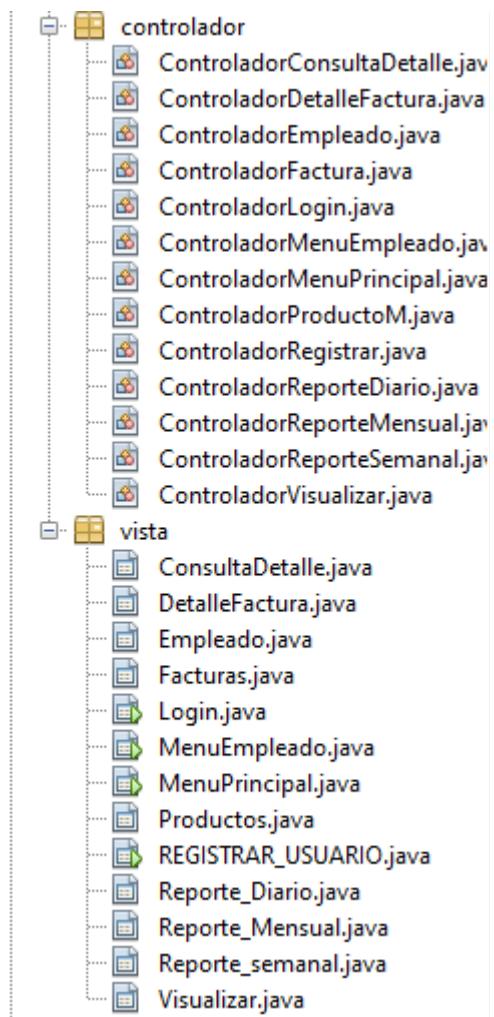
3.5. Diagrama de la base de Datos.



3.6. Evidencia del software del MCV y DAO.

3.6.1 MCV





3.6.1.1 controlador empleado

```
package controlador;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.Empleado;
import vista.MenuPrincipal;
import Modelo.*;
import DAO.*;
import Formatos.*;
public class ControladorEmpleado implements ActionListener{
    //atributos
    Empleado vista;
    MenuPrincipal menu;
    Empleado emp;
    CRUDempleado crud;
    FormatoEmpleado fm;
    //constructor
    public ControladorEmpleado(Empleado em) {
        this.vista = em;
        vista.jbtnAgregarE.addActionListener(this);
        vista.jbtnConsultarE.addActionListener(this);
        vista.jbtnModificarE.addActionListener(this);
        vista.jbtnEliminarEl.addActionListener(this);
        vista.setVisible(true);
        crud = new CRUDempleado();
        crud.MostrarEmpleadoEnTabla(vista.jtblListaEmpleado,vista.jlbListaEmpleado);
        vista.setTitle("Empleado");
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == vista.jbtnAgregarE){
            emp = FormatoEmpleado.LeerProducto(vista);
            crud = new CRUDempleado();
            crud.InsertarRegistroEmpleado(emp);
            crud.MostrarEmpleadoEnTabla(vista.jtblListaEmpleado,vista.jlbListaEmpleado);
            FormatoEmpleado.LimpiarEntradas(vista);

        }
        try{
            if(e.getSource() == vista.jbtnModificarE){
                emp = FormatoEmpleado.LeerProducto(vista);
                crud = new CRUDempleado();
                crud.ActualizarEmpleado(emp);
                crud.MostrarEmpleadoEnTabla(vista.jtblListaEmpleado,vista.jlbListaEmpleado);
            }
        }
    }
}
```

```

        FormatoEmpleado.LimpiarEntradas(vista);

    }

    }catch (Exception ex){
        Mensajes.M1("ERROR no se puede modificar .."+ex);
    }

    try{
        if(e.getSource() == vista.btnExitEliminarE1){
            int respuesta = Mensajes.M3("Confirmar!!!","¿Deseas eliminar al empleado?");
            if(respuesta==0){
                emp = FormatoEmpleado.LeerProducto(vista);
                crud = new CRUDempleado();
                crud.EliminarEmpleado(emp);
                crud.MostrarEmpleadoEnTabla(vista.jtblListaEmpleado,vista.jlbListaEmpleado);
                FormatoEmpleado.LimpiarEntradas(vista);
            }
        }catch (Exception ex){
            Mensajes.M1("ERROR no se puede eliminar .."+ex);
        }

        if(e.getSource() == vista.btnExitConsultarE){
            String cod_emp = Mensajes.M2("Ingrese el codigo del empleado a buscar ");
            crud = new CRUDempleado();
            emp = crud.ObtenerRegistroEmp(cod_emp);
            if(emp==null){
                Mensajes.M1("El ID "+emp+" no existe en la tabla Empleado..");
            }else{
                vista.jtxtCodigoE.setText(emp.getCod_emp());
                vista.jtxtNombreE.setText(emp.getNombre());
                vista.jtxtApellidoE.setText(emp.getApellido());
                vista.jtxtDireccionE.setText(emp.getDireccion());
                vista.jtxtTelefonoE.setText(emp.getTelefono());
                vista.jtxtEdadE.setText(Integer.toString(emp.getEdad()));
                vista.jtxtSueldoE.setText(Double.toString(emp.getSueldo()));
                vista.jspnDiasTrabajoE.setValue(Integer.toString((emp.getDias_Trabajo())));
            }
        }
    }
}
}

```

3.6.1.2 controlador factura

```
package controlador;
//libreria|
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import Modelo.*;
import DAO.*;
import Formatos.*;
import DAO.CRUDfacturas;
import java.text.ParseException;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.text.SimpleDateFormat;
import javax.swing.table.DefaultTableModel;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
public class ControladorFactura implements ActionListener{
    //atributos
    Facturas vista;
    Factura fa;
    CRUDfacturas crud;
    FormatoFactura fac;
    CargarCombos cc;
    //constructor
    public ControladorFactura(Facturas fac){
        this.vista = fac;
        vista.jConsultaFactura.addActionListener(this);
        vista.btnExitBuscarF.addActionListener(this);
        vista.btnExitBuscarFechaF.addActionListener(this);
        vista.btnExitConsultarF.addActionListener(this);
        vista.btnExitModificarF.addActionListener(this);
        vista.btnExitEliminarF.addActionListener(this);
        vista.btnExitRegistrarF1.addActionListener(this);
        Calendar cal = new GregorianCalendar();
        vista.jfechaFConsulta.setCalendar(cal);
        vista.jFechaF1.setCalendar(cal);
        cc = new CargarCombos();
        cc.CargarNombreEmpleado(vista.jtxtCodigo_Epleado);
        crud = new CRUDfacturas();
        crud.MostrarFacturasEnTabla(vista.jtblFacturasF,vista.jtxtCFacturas);
        vista.setTitle("Factura");
        vista.setVisible(true);
    }
}
```

```

@Override
public void actionPerformed(ActionEvent e) {
    try{
        if(e.getSource() == vista.jbtnModificarF){
            fa = FormatoFactura.LeerProducto(vista);
            crud = new CRUDfacturas();
            crud.ActualizarFactura(fa);
            crud.MostrarFacturasEnTabla(vista.jtblFacturasF,vista.jtxtCFacturas);
            FormatoFactura.LimpiarEntradas(vista);

        }
    }catch (Exception ex){
        Mensajes.M1("ERROR no se puede modificar .."+ex);
    }
    try{
        if(e.getSource() == vista.jbtnEliminarF){
            int respuesta = Mensajes.M3("Confirmar!!!","¿Deseas eliminar al empleado?");
            if(respuesta==0){
                fa = FormatoFactura.LeerProducto(vista);
                crud = new CRUDfacturas();
                crud.EliminarFactura(fa);
                crud.EliminarDetalleFactura(fa);
                crud.MostrarFacturasEnTabla(vista.jtblFacturasF,vista.jtxtCFacturas);
                FormatoFactura.LimpiarEntradas(vista);
            }
        }}catch (Exception ex){
        Mensajes.M1("ERROR no se puede eliminar .."+ex);
    }
    if(e.getSource() == vista.jbtnConsultarF){
        String cod_fa = Mensajes.M2("Ingrese el codigo del empleado a buscar ");
        crud = new CRUDfacturas();
        fa = crud.ObtenerRegistroFactura(cod_fa);
        if(fa==null){
            Mensajes.M1("El ID "+fa+" no existe en la tabla Factura..");
        }else{
            try {
                int fecha = vista.jtblFacturasF.getSelectedRow();
                DefaultTableModel model = (DefaultTableModel)vista.jtblFacturasF.getModel();
                vista.jtxtCod_Factura.setText(Integer.toString(fa.getCod_factura()));
                vista.jtxtDNI.setTxt

```

3.6.1.3 Controlador Detalle factura

```

package controlador;
//libreria
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import Modelo.*;
import DAO.*;
import Formatos.*;
import DAO.CRUDfacturas;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import javax.swing.table.DefaultTableModel;
public class ControladorDetalleFactura implements ActionListener{
    //atributos
    DetalleFactura vista;
    Factura fa;
    RegistroFactura fact;
    CRUDdetallefactura crud;
    CRUDfacturas crud;
    FormatoFactura fac;
    CargarCombos cc;
    ManejadorIdForaneos mif;
    DefaultTableModel modelotabla;
    double totalPagar;
    String[] TitulosTabla={"codigo Producto","Descripcion","Cantidad","Precio Unitario","Precio Total"};
    //constructor
    public ControladorDetalleFactura(DetalleFactura fac){
        this.vista = fac;
        vista.btnRegisFac.addActionListener(this);
        vista.botAgreProFac.addActionListener(this);
        cc = new CargarCombos();
        cc.CargarNombreEmpleado(vista.cbcEmpleadoFac);
        modelotabla = new DefaultTableModel(null,TitulosTabla);
        this.vista.TablaDetalle.setModel(modelotabla);
        vista.setTitle("Registro Factura");
        vista.setVisible(true);
    }
    void LimpiarEntradas(){
        vista.txtCodProFac.setText("");
        vista.txtNomProFac.setText("");
        vista.txtCantidadFac.setText("");
        vista.txtPrecioFac.setText("");
        vista.txtCodProFac.requestFocus();
    }
}

```

```

        vista.txtCodProFac.requestFocus();
    }

    void TotalPagar(){
        totalPagar = 0;
        int numFila = vista.TablaDetalle.getRowCount();
        for ( int i = 0;i<numFila;i++){
            double cal = Double.parseDouble(String.valueOf(vista.TablaDetalle.getModel().getValueAt(i, 4)));
            totalPagar = totalPagar + cal;
        }
        vista.TotalFac.setText(Double.toString(totalPagar));
    }

    void LimpiarTablaDetalle(){
        for(int i = 0 ;i < modelotabla.getRowCount() ; i++){
            modelotabla.removeRow(i);
            i = i-1;
        }
    }

    void InsertarCompra(){
        mif = new ManejadorIdForaneos();
        SimpleDateFormat sf = new SimpleDateFormat("yyyy-MM-dd");
        crud = new CRUDdetallefactura();
        String dni = vista.txtDNIIfac.getText();
        String cod = mif.RecuperarCodEmp(vista.cbcEmpleadoFac.getSelectedItem().toString());
        String fecha = sf.format(vista.jdateFactura.getDate());
        Double Total = Double.parseDouble(vista.TotalFac.getText());
        if(cruda.InsertarRegistroFactura(dni, cod, fecha, Total)){
            for(int i = 0 ;i < vista.TablaDetalle.getRowCount() ; i++){
                int id = mif.RecuperarIdFactura();
                String cod_pro = vista.TablaDetalle.getValueAt(i, 0).toString();
                String des = vista.TablaDetalle.getValueAt(i, 1).toString();
                int cant = Integer.parseInt(vista.TablaDetalle.getValueAt(i, 2).toString());
                double preu = Double.parseDouble(vista.TablaDetalle.getValueAt(i, 3).toString());
                double total = cant*preu;
                crud.InsertardetalleFactura(cod_pro, des, cant, preu,total, id);
            }
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == vista.botAgreProFac){
            RegistroFactura m = new RegistroFactura();
            int cant = Integer.parseInt(vista.txtCantidadFac.getText());
            }
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            if(e.getSource() == vista.botAgreProFac){
                RegistroFactura m = new RegistroFactura();
                int cant = Integer.parseInt(vista.txtCantidadFac.getText());
                String cod = vista.txtCodProFac.getText();
                String des = vista.txtNomProFac.getText();
                double pre = Double.parseDouble(vista.txtPrecioFac.getText());
                ArrayList lista = new ArrayList();
                int item = 1;
                lista.add(item);
                lista.add(cod);
                lista.add(des);
                lista.add(cant);
                lista.add(pre);
                lista.add(pre*cant);
                Object[] obj = new Object[5];
                obj[0] = lista.get(1);
                obj[1] = lista.get(2);
                obj[2] = lista.get(3);
                obj[3] = lista.get(4);
                obj[4] = lista.get(5);
                modelotabla.addRow(obj);
                vista.TablaDetalle.setModel(modelotabla);
                LimpiarEntradas();
                TotalPagar();
            }
            try{
                if(e.getSource() == vista.btnRegisFac){
                    InsertarCompra();
                    FormatoDetalleFactura.LimpiarEntradas(vista);
                    LimpiarTablaDetalle();
                }
            }catch (Exception ex){
                Mensajes.M1("ERROR no se puede modificar .."+ex);
            }
        }
    }
}

```

3.6.1.4 Controlador Consultar Factura

```

    package controlador;
    //libreria
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import vista.*;
    import Modelo.*;
    import DAO.*;
    import Formatos.*;
    import DAO.CRUDfacturas;
    import java.text.SimpleDateFormat;
    import java.util.ArrayList;
    import javax.swing.table.DefaultTableModel;
    public class ControladorConsultaDetalle implements ActionListener{
        //atributos
        ConsultaDetalle vista;
        RegistroFactura fact;
        CRUDdetallefactura cruda;
        DefaultTableModel modelotabla;
        double totalPagar;
        String[] TitulosTabla={"codigo Producto","Descripcion","Cantidad","Precio Unitario","Precio Total"};
        //constructor
        public ControladorConsultaDetalle(ConsultaDetalle fac){
            this.vista = fac;
            vista.jConsultaFactura.addActionListener(this);
            vista.setTitle("Detalle Factura");
            vista.setVisible(true);
        }
        @Override
        public void actionPerformed(ActionEvent e) {
            throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
        }
    }
}

```

3.6.1.5 Controlador Login

```

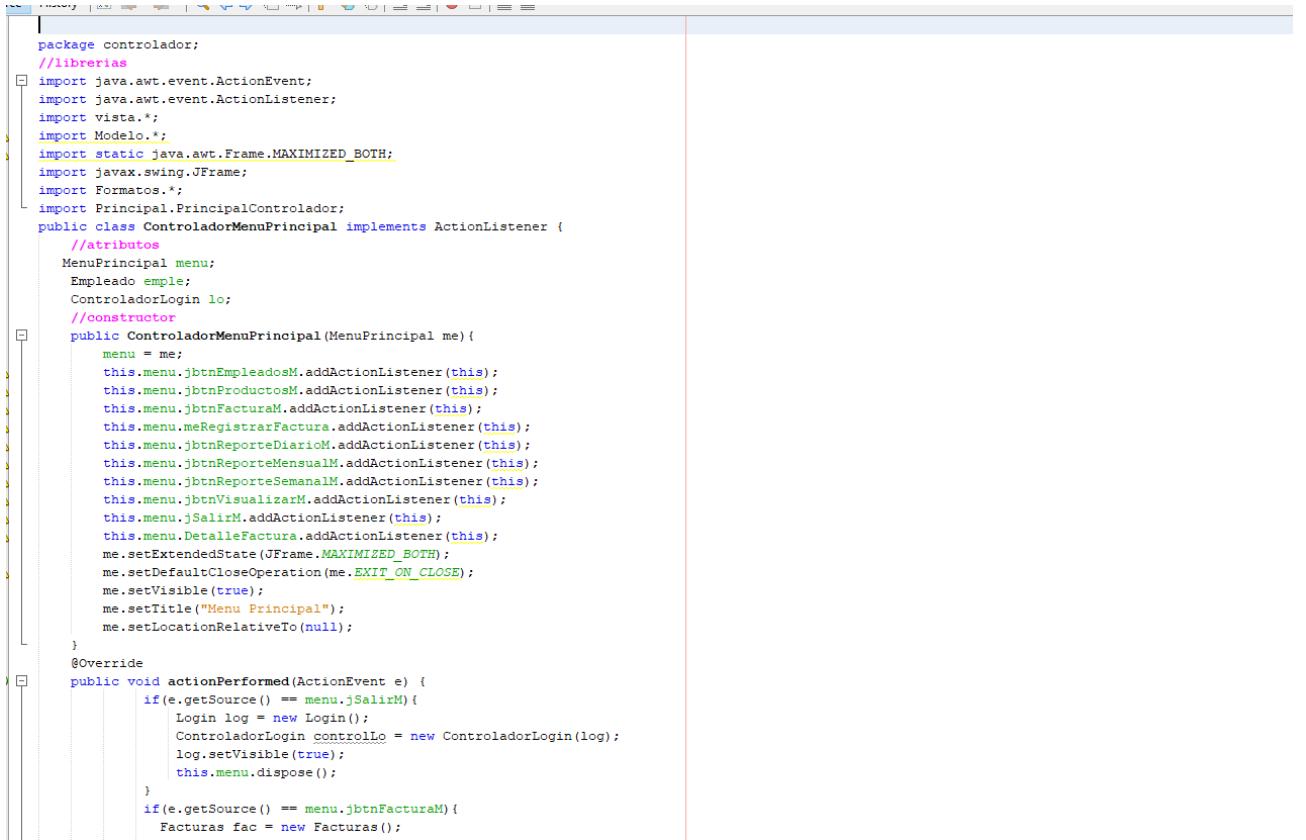
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import vista.*;
    import vista.Login;
    import Modelo.RegistroUsuario;
    import DAO.*;
    import Formatos.*;
    public class ControladorLogin implements ActionListener{
        //atributos
        Login vista;
        REGISTRAR_USUARIO re;
        CRUDlogin crud;
        RegistroUsuario regis;
        FormatoLogin forlo;
        MenuPrincipal menu = new MenuPrincipal();
        CargarCombos cc;
        //constructor
        public ControladorLogin(Login lo){
            vista=lo;
            vista.jbtnIniciar.addActionListener(this);
            vista.jbtnRegistrar.addActionListener(this);
            vista.setVisible(true);
            vista.setLocationRelativeTo(null);
            vista.setTitle("Ingrese Usuario");
            cc = new CargarCombos();
            cc.CargarUsuarioCombo(vista.jcbcTipoUsuario);
        }
        @Override
        public void actionPerformed(ActionEvent e) {
            if(e.getSource() == vista.jbtnIniciar){
                RegistroUsuario log = new RegistroUsuario();
                CRUDlogin crud = new CRUDlogin();
                String usu = vista.jtxtUsuario.getText();
                String contra = vista.jtxtContrasena.getText();
                ManejadorIdForaneos id = new ManejadorIdForaneos();
                int codtipo = id.RecuperarTipoU(vista.jcbcTipoUsuario.getSelectedItem().toString());
                crud.IngresarSistema(usu, contra,codtipo);
                FormatoLogin.LimpiarEntradas(vista);
                vista.setVisible(false);
            }
            if(e.getSource() == vista.jbtnRegistrar){
                REGISTRAR_USUARIO ventana = new REGISTRAR_USUARIO();
                ControladorRegistrar controlre = new ControladorRegistrar(ventana);
                ventana.setVisible(true);
                this.vista.dispose();
            }
        }
    }
}

```

3.6.1.6 Controlador MenuEmpleado

```
package controlador;
//librerias
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import static java.awt.Frame.MAXIMIZED_BOTH;
import java.awt.event.ComponentListener;
import Formatos.*;
import javax.swing.JFrame;
public class ControladorMenuEmpleado implements ActionListener {
    //atributos
    MenuEmpleado menu;
    Empleado emp;
    Productos pro;
    ControladorLogin lo;
    //constructor
    public ControladorMenuEmpleado(MenuEmpleado emp){
        menu = emp;
        this.menu.buttonFacturasE.addActionListener(this);
        this.menu.buttonReporteDiarioE.addActionListener(this);
        this.menu.VisualizarE.addActionListener(this);
        this.menu.jSalirE.addActionListener(this);
        this.menu.meRegistrarFactura.addActionListener(this);
        this.menu.DetalleFactura.addActionListener(this);
        emp.setExtendedState(JFrame.MAXIMIZED_BOTH);
        emp.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        emp.setVisible(true);
        emp.setTitle("Menu Empleado");
        emp.setLocationRelativeTo(null);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == menu.jSalirE){
            try{
                Login log = new Login();
                ControladorLogin controlLo = new ControladorLogin(log);
                log.setVisible(true);
                this.menu.dispose();
            }catch (Exception ex){
                Mensajes.Mi("ERROR no se puede registrar .."+ex);
            }
        }
        if(e.getSource() == menu.buttonFacturasE){
            Facturas fac = new Facturas();
            menu.jescritorio.add(fac);
            menu.jescritorio.revalidate();
            ControladorFactura fa = new ControladorFactura(fac);
            CentrarFormas.CPanel(menu.jescritorio,fa);
        }
        if(e.getSource() == menu.meRegistrarFactura){
            DetalleFactura defac = new DetalleFactura();
            menu.jescritorio.add(defac);
            ControladorDetalleFactura cfac = new ControladorDetalleFactura(defac);
            CentrarFormas.CPanel(menu.jescritorio,defac);
        }
        if(e.getSource() == menu.DetalleFactura){
            ConsultaDetalle defac = new ConsultaDetalle();
            menu.jescritorio.add(defac);
            ControladorConsultaDetalle cdf = new ControladorConsultaDetalle(defac);
            CentrarFormas.CPanel(menu.jescritorio,defac);
        }
        if(e.getSource() == menu.VisualizarE){
            Visualizar vis = new Visualizar();
            ControladorVisualizar visu = new ControladorVisualizar(vis);
            menu.jescritorio.add(vis);
            CentrarFormas.CPanel(menu.jescritorio,vis);
        }
        if(e.getSource() == menu.buttonReporteDiarioE){
            Reporte_Diario re = new Reporte_Diario();
            ControladorReporteDiario crd = new ControladorReporteDiario(re);
            menu.jescritorio.add(re);
            CentrarFormas.CPanel(menu.jescritorio,re);
        }
    }
}
```

3.6.1.7 Controlador Menu administrador



The screenshot shows a Java code editor with the following code:

```
package controlador;
//librerias
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import Modelo.*;
import static javax.swing.JFrame.MAXIMIZED_BOTH;
import javax.swing.JFrame;
import Formatos.*;
import Principal.PrincipalControlador;
public class ControladorMenuPrincipal implements ActionListener {
    //atributos
    MenuPrincipal menu;
    Empleado emple;
    ControladorLogin lo;
    //constructor
    public ControladorMenuPrincipal(MenuPrincipal me) {
        menu = me;
        this.menu.jbtnEmpleadosM.addActionListener(this);
        this.menu.jbtnProductosM.addActionListener(this);
        this.menu.jbtnFacturaM.addActionListener(this);
        this.menu.meRegistrarFactura.addActionListener(this);
        this.menu.jbtnReporteDiarioM.addActionListener(this);
        this.menu.jbtnReporteMensualM.addActionListener(this);
        this.menu.jbtnReporteSemanalM.addActionListener(this);
        this.menu.jbtnVisualizarM.addActionListener(this);
        this.menu.jSalirM.addActionListener(this);
        this.menu.DetalleFactura.addActionListener(this);
        me.setExtendedState(JFrame.MAXIMIZED_BOTH);
        me.setDefaultCloseOperation(me.EXIT_ON_CLOSE);
        me.setVisible(true);
        me.setTitle("Menu Principal");
        me.setLocationRelativeTo(null);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == menu.jSalirM){
            Login log = new Login();
            ControladorLogin controlLo = new ControladorLogin(log);
            log.setVisible(true);
            log.dispose();
        }
        if(e.getSource() == menu.jbtnFacturaM){
            Facturas fac = new Facturas();
        }
    }
}
```

```

        menu.jescritorio.add(fac);
        ControladorFactura fa = new ControladorFactura(fac);
        CentrarFormas.CPanel(menu.jescritorio,fa);
    }
    if(e.getSource() == menu.meRegistrarFactura){
        DetalleFactura defac = new DetalleFactura();
        menu.jescritorio.add(defac);
        ControladorDetalleFactura cfac = new ControladorDetalleFactura(defac);
        CentrarFormas.CPanel(menu.jescritorio,defac);
    }
    if(e.getSource() == menu.DetalleFactura){
        ConsultaDetalle defac = new ConsultaDetalle();
        menu.jescritorio.add(defac);
        ControladorConsultaDetalle cdf = new ControladorConsultaDetalle(defac);
        CentrarFormas.CPanel(menu.jescritorio,defac);
    }
    if(e.getSource() == menu.jbtnReporteMensualM){
        Reporte_Mensual rm = new Reporte_Mensual();
        ControladorReporteMensual crm = new ControladorReporteMensual(rm);
        menu.jescritorio.add(rm);
        CentrarFormas.CPanel(menu.jescritorio,rm);
    }
    if(e.getSource() == menu.jbtnReporteSemanalM){
        Reporte_semanal rs = new Reporte_semanal();
        ControladorReporteSemanal crs = new ControladorReporteSemanal(rs);
        menu.jescritorio.add(rs);
        CentrarFormas.CPanel(menu.jescritorio,rs);
    }
    if(e.getSource() == menu.jbtnVisualizarM){
        Visualizar vis = new Visualizar();
        ControladorVisualizar visu = new ControladorVisualizar(vis);
        menu.jescritorio.add(vis);
        CentrarFormas.CPanel(menu.jescritorio,vis);
    }
    if(e.getSource() == menu.jbtnEmpleadosM){
        Empleado em = new Empleado();
        menu.jescritorio.add(em);
        ControladorEmpleado emp = new ControladorEmpleado(em);
        CentrarFormas.CPanel(menu.jescritorio,em);
    }
    if(e.getSource() == menu.jbtnReporteDiarioM){
        Reporte_Diario re = new Reporte_Diario();
        ControladorReporteDiario crd = new ControladorReporteDiario(re);
        menu.jescritorio.add(re);
        CentrarFormas.CPanel(menu.jescritorio,re);
    }

```

3.6.1.8 Controlador Producto

```
package controlador;
//libreria
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import vista.Productos;
import Modelo.Al_Producto;
import DAO.*;
import Formatos.Formatoproductos;
import Formatos.*;
import java.text.SimpleDateFormat;
import javax.swing.table.DefaultTableModel;
import java.util.Date;
public class ControladorProductoM implements ActionListener{
    //atributos
    Productos vista;
    Al_Producto prod;
    CRUDproductos crud;
    CargarCombos cc;
    Formatoproductos fpro;
    //constructor
    public ControladorProductoM(Productos p){
        this.vista = p;
        vista.jbtnConsultarP.addActionListener(this);
        vista.jbtnEliminarP.addActionListener(this);
        vista.jbtnModificarP.addActionListener(this);
        vista.jbtnAgregarP1.addActionListener(this);
        cc = new CargarCombos();
        cc.CargarCategoriasEnCombo(vista.jCategoriaP);
        cc.CargarstockEnCombo(vista.jtxtStockP);
        cc.CargartamanoEnCombo(vista.jcbcTamañoP);
        crud = new CRUDproductos();
        crud.MostrarProductoEnTabla(vista.jtblListaProductos,vista.lblProductos);
        vista.setTitle("Productos");
        vista.setVisible(true);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == vista.jbtnAgregarP1){
            prod = Formatoproductos.LeerProducto(vista);
            crud = new CRUDproductos();
            crud.InsertarRegistroProd(prod);
            crud.MostrarProductoEnTabla(vista.jtblListaProductos,vista.lblProductos);
            Formatoproductos.LimpiarEntradas(vista);
        }
    }
}
```

```

    }
}

try{
    if(e.getSource() == vista.jbtnModificarP){
        prod = FormatoProductos.LeerProducto(vista);
        crud = new CRUDproductos();
        crud.ActualizarProducto(prod);
        crud.MostrarProductoEnTabla(vista.jtblListaProductos,vista.lblProductos);
        FormatoProductos.LimpiarEntradas(vista);
    }
} catch (Exception ex){
    Mensajes.M1("ERROR no se puede modificar .."+ex);
}

try{
    if(e.getSource() == vista.jbtnEliminarP){
        int respuesta = Mensajes.M3("Confirmar!!!","¿Deseas eliminar el producto?");
        if(respuesta==0){
            prod = FormatoProductos.LeerProducto(vista);
            crud = new CRUDproductos();
            crud.EliminarFactura(prod);
            crud.MostrarProductoEnTabla(vista.jtblListaProductos,vista.lblProductos);
            FormatoProductos.LimpiarEntradas(vista);
        }
    } catch (Exception ex){
        Mensajes.M1("ERROR no se puede eliminar .."+ex);
    }
    if(e.getSource() == vista.jbtnConsultarP){
        String cod_pro = Mensajes.M2("Ingrese el codigo del producto a buscar ");
        crud = new CRUDproductos();
        prod = crud.ObtenerRegistroProducto(cod_pro);
        if(prod==null){
            Mensajes.M1("El ID "+prod+" no existe en la tabla Factura..");
        }else{
            vista.jtxtCodigoProducto.setText(prod.getCod_Producto());
            vista.jtxtNombrePro.setText(prod.getNombre_P());
            vista.jtxtPrecioP.setText(Double.toString(prod.getPrecio()));
        }
    }
}
}
}

```

3.6.1.9 Controlador Registrar

```
package controlador;
//libreria
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import vista.REGISTRAR_USUARIO;
import vista.Login;
import Modelo.*;
import DAO.*;
import Formatos.FormatoRegistrar;
public class ControladorRegistrar implements ActionListener {
    //atributos
    REGISTRAR_USUARIO vista;
    CRUDregistro crud;
    RegistroUsuario regis;
    Login lo;
    //constructor
    public ControladorRegistrar(REGISTRAR_USUARIO re) {
        vista=re;
        vista.jbtnRegistrarR.addActionListener(this);
        vista.jbtnCancelar.addActionListener(this);
        vista.setVisible(true);
        vista.setLocationRelativeTo(null);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()== vista.jbtnRegistrarR){
            regis = FormatoRegistrar.LeerProducto(vista);
            crud = new CRUDregistro();
            crud.InsertarRegistroProd(regis);
            FormatoRegistrar.LimpiarEntradas(vista);
        }
        if(e.getSource()== vista.jbtnCancelar){
            Login log = new Login();
            ControladorLogin controlLo = new ControladorLogin(log);
            log.setVisible(true);
            this.vista.dispose();
        }
    }
}
```

3.6.1.10 Controlador ReporteDiario

```
package controlador;
//libreria
] import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import vista.Reporte_Diario;
import Modelo.*;
import DAO.*;
import Formatos.*;
import java.text.ParseException;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.ArrayList;
import java.text.SimpleDateFormat;
import javax.swing.table.DefaultTableModel;
import java.util.Date;
import java.util.logging.Level;
- import java.util.logging.Logger;

public class ControladorReporteDiario implements ActionListener{
    //atributos
    Reporte_Diario vista;
    DefaultTableModel modelotabla;
    CRUDreporteDiario crud;
    FormatoReporteDiario red;
    CargarCombos cc;
    ReporteDiario re;
    //constructor
] public ControladorReporteDiario(Reporte_Diario rd){
    this.vista = rd;
    vista.btnExitConsultarRD.addActionListener(this);
    vista.btnExitEliminarRD.addActionListener(this);
    vista.btnExitModificarRD.addActionListener(this);
    vista.btnExitRegistrarRD1.addActionListener(this);
    Calendar cal = new GregorianCalendar();
    vista.jtxtFechaR.setCalendar(cal);
    cc = new CargarCombos();
    cc.CargarNombreEmpleado(vista.jtxtEmpleadoR);
    crud = new CRUDreporteDiario();
    crud.MostrarReporteEnTabla(vista.jtblReporteDiaRB,vista.jblCRD);
    vista.jtxNReporte.setEnabled(false);
    vista.setTitle("Reporte Diario");
    vista.setVisible(true);
}
] public String categoria(){
```

```

        }
    }

    void LeerCategoria(){
        re = new ReporteDiario();
        ManejadorIdForaneos mif = new ManejadorIdForaneos();
        re.setCod_emp(mif.RecuperarCodEmp(vista.jtxtEmpleadoR.getSelectedItem().toString()));
        re.setPizzav(Integer.parseInt(vista.jtxtPizzaVendida.getText().toString()));
        re.setPastav(Integer.parseInt(vista.jtxtPastasVendida.getText().toString()));
        re.setBebidav(Integer.parseInt(vista.jtxtBebidasVendidas.getText().toString()));
        re.setPiqueosv(Integer.parseInt(vista.jtxtPiqueoVendido.getText().toString()));
        re.setPlatoscv(Integer.parseInt(vista.jtxtPlatosVendidos.getText().toString()));
        SimpleDateFormat sf = new SimpleDateFormat("yyyy-MM-dd");
        re.setFecha(sf.format(vista.jtxtFechaR.getDate()));
        re.setDinero_i(Double.parseDouble(vista.jtxtDineroInicialRD.getText()));
        re.setTotal_D(Double.parseDouble(vista.jtxtTotalRD.getText()));
        re.Sobrante();
    }

}

@Override
public void actionPerformed(ActionEvent e) {
    try{
        if(e.getSource() == vista.btnExitReporteRD1){
            LeerCategoria();
            CRUDreporteDiario crud = new CRUDreporteDiario();
            crud.InsertarRegistroReporteDiario(re);
            crud.MostrarReporteEnTabla(vista.jtblReporteDiaRB,vista.jblCRD);
            FormatoReporteDiario.LimpiarEntradas(vista);
        }
    }catch (Exception ex){
        Mensajes.M1("ERROR no se puede registrar .."+ex);
    }
    try{
        if(e.getSource() == vista.btnExitModificarRD){
            LeerCategoria();
            re.setNfactura(Integer.parseInt(vista.jtxNReporte.getText()));
            CRUDreporteDiario crud = new CRUDreporteDiario();
            crud.ActualizarReporte(re);
            crud.MostrarReporteEnTabla(vista.jtblReporteDiaRB,vista.jblCRD);
            FormatoReporteDiario.LimpiarEntradas(vista);
        }
    }catch (Exception ex){
        Mensajes.M1("ERROR no se puede modificar .."+ex);
    }
    try{
        try{
            if(e.getSource() == vista.btnExitEliminarRD){
                int respuesta = Mensajes.M3("Confirmar!!!","¿Deseas eliminar el reporte?");
                if(respuesta==0){
                    int idcat = Integer.parseInt(vista.jtxNReporte.getText());
                    CRUDreporteDiario crud = new CRUDreporteDiario();
                    crud.EliminarReporte(idcat);
                    crud.MostrarReporteEnTabla(vista.jtblReporteDiaRB,vista.jblCRD);
                    FormatoReporteDiario.LimpiarEntradas(vista);
                }
            }
        }catch (Exception ex){
            Mensajes.M1("ERROR no se puede eliminar .."+ex);
        }
        if(e.getSource() == vista.btnExitConsultarRD){
            String repo = Mensajes.M2("Ingrese el codigo del reporte a buscar ");
            crud = new CRUDreporteDiario();
            re = crud.ObtenerRegistroReporte(repo);
            if(re==null){
                Mensajes.M1("El ID "+repo+" no existe en la tabla Empleado..");
            }else{
                try {
                    int fecha = vista.jtblReporteDiaRB.getSelectedRow();
                    DefaultTableModel model = (DefaultTableModel)vista.jtblReporteDiaRB.getModel();
                    vista.jtxNReporte.setText(Integer.toString(re.getNfactura()));
                    vista.jtxtPizzaVendida.setText(Integer.toString(re.getPizzav()));
                    vista.jtxtPastasVendida.setText(Integer.toString(re.getPastav()));
                    vista.jtxtBebidasVendidas.setText(Integer.toString(re.getBebidav()));
                    vista.jtxtPiqueoVendido.setText(Integer.toString(re.getPiqueosv()));
                    vista.jtxtPlatosVendidos.setText(Integer.toString(re.getPlatoscv()));
                    Date date = new SimpleDateFormat("yyyy-MM-dd").parse((String)model.toString());
                    vista.jtxtFechaR.setDate(date);
                    vista.jtxtDineroInicialRD.setText(Double.toString(re.getDinero_i()));
                    vista.jtxtTotalRD.setText(Double.toString(re.getTotal_D()));
                } catch (ParseException ex) {
                    Logger.getLogger(ControladorReporteDiario.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }
    }
}
}

```

3.6.1.11 Controlador ReporteSemanal

```

package controlador;
//librerias
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import vista.Reporte_Diario;
import Modelo.*;
import DAO.*;
import Formatos.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.table.DefaultTableModel;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.ArrayList;

public class ControladorReporteSemanal implements ActionListener{
    //atributos
    Reporte_semanal vista;
    DefaultTableModel modelotabla;
    CRUDreporteSemanal crud;
    CargarCombo cc;
    ReporteDiario re;
    Reporte_Diario rvista;
    //constructor
    public ControladorReporteSemanal(Reporte_semanal rs){
        this.vista = rs;
        vista.jbuscarRS.addActionListener(this);
        vista.RestauraraRS.addActionListener(this);
        crud = new CRUDreporteSemanal();
        crud.MostrarReporteEnTabla(vista.jtblRsemanal,vista.jbCDP);
        vista.setTitle("Reporte Semanal");
        vista.setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()== vista.jbuscarRS){
            crud = new CRUDreporteSemanal();
        }
        if(e.getSource()== vista.RestauraraRS){
            crud = new CRUDreporteSemanal();
            crud.MostrarReporteEnTabla(vista.jtblRsemanal,vista.jbCDP);
        }
    }
}

```

3.6.1.12 Controlador Reporte Mensual

```

package controlador;
//libreria
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import vista.Reporte_Diario;
import Modelo.*;
import DAO.*;
import javax.swing.table.DefaultTableModel;
public class ControladorReporteMensual implements ActionListener{
    //atributos
    Reporte_Mensual vista;
    DefaultTableModel modelotabla;
    CRUDreporteMensual crud;
    CargarCombos cc;
    ReporteDiario re;
    Reporte_Diario rvista;
    //constructor
    public ControladorReporteMensual(Reporte_Mensual rm){
        this.vista = rm;

        vista.Restaurar.addActionListener(this);
        crud = new CRUDreporteMensual();
        crud.MostrarReporteEnTabla(vista.jtblRmensual,vista.jCDR);
        vista.setTitle("Reporte Mensual");
        vista.setVisible(true);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == vista.Restaurar){
            crud = new CRUDreporteMensual();
            crud.MostrarReporteEnTabla(vista.jtblRmensual,vista.jCDR);
        }
    }
}

```

3.6.1.13 Controlador visualizar

```

package controlador;
//libreria
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import vista.*;
import Modelo.Al_Producto;
import DAO.*;
import Formatos.Mensajes;
public class ControladorVisualizar implements ActionListener{
    //atributos
    Visualizar vista;
    Al_Producto prod;
    CRUDproductos crud;
    CargarCombos cc;
    CRUDvisualizar crudv;
    //constructor
    public ControladorVisualizar(Visualizar pro){
        this.vista = pro;
        vista.jbtnFCategoria.addActionListener(this);
        vista.jbtnProductos.addActionListener(this);
        vista.jbtnStock.addActionListener(this);
        vista.jbtnFiltrar.addActionListener(this);
        vista.jbtnRestaurar.addActionListener(this);
        cc = new CargarCombos();
        cc.CargarCombocategorias(vista.jcboCategoriaV);
        cc.CargarCombostock(vista.jcboStockV);
        cc.CargarCombotamano(vista.jFiltroTamano);
        crud = new CRUDproductos();
        crud.MostrarProductoEnTabla(vista.jtblVisualizar,vista.JCDP);
        vista.setTitle("Visualizar");
        vista.setVisible(true);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == vista.jbtnFstock){
            try{
                crudv = new CRUDvisualizar();
                crudv.MostrarStockEnTabla(vista.jtblVisualizar,vista.jcboStockV.getSelectedItem().toString());
            }catch (Exception ex){
                Mensajes.Mi("ERROR no se puede Filtrar .."+ex);
            }
        }
        if(e.getSource() == vista.jbtnFCategoria){
            try{
                crudv = new CRUDvisualizar();

```

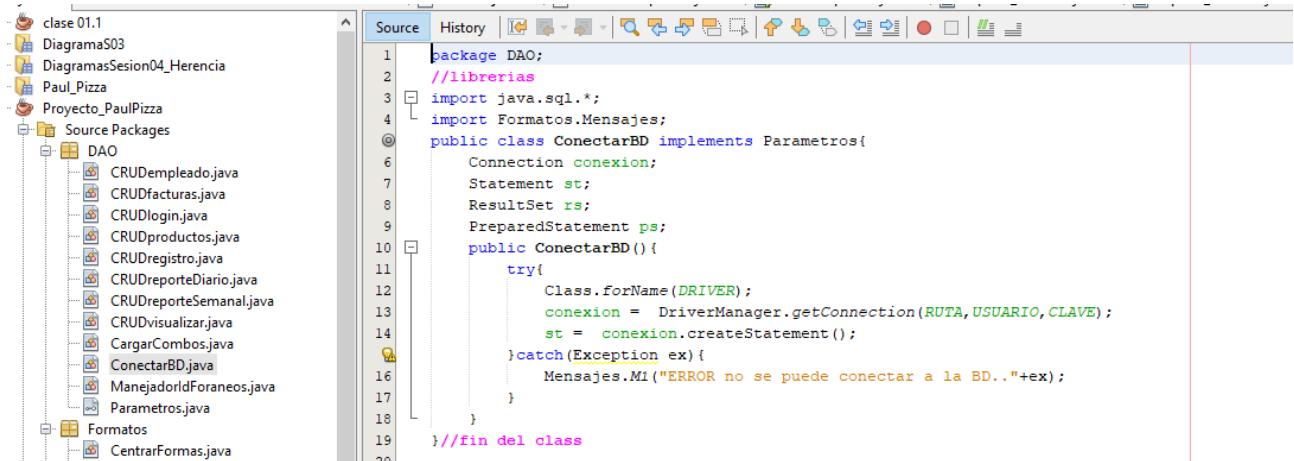
```

        try{
            crudv = new CRUDvisualizar();
            crudv.MostrarCategoriaEnTabla(vista.jtblVisualizar,vista.jcbcCategoriaV.getSelectedItem().toString());
        }catch (Exception ex){
            Mensajes.M1("ERROR no se puede filtrar .."+ex);
        }
    }
    if(e.getSource() == vista.jbtnFProducto){
        try{
            crudv = new CRUDvisualizar();
            crudv.MostrarTamanoEnTabla(vista.jtblVisualizar,vista.jFiltroTamano.getSelectedItem().toString());
        }catch (Exception ex){
            Mensajes.M1("ERROR no se puede filtrar .."+ex);
        }
    }
    if(e.getSource() == vista.jbtnRestaurar){
        try{
            crud = new CRUDproductos();
            crud.MostrarProductoEnTabla(vista.jtblVisualizar,vista.JCDP);
        }catch (Exception ex){
            Mensajes.M1("ERROR no se puede mostrar datos .."+ex);
        }
    }
    if(e.getSource() == vista.jbtnFiltrar){
        try{
            crudv = new CRUDvisualizar();
            crudv.MostrarProductoEnTabla(vista.jtblVisualizar,vista.jcbcProductoV.getText());
        }catch (Exception ex){
            Mensajes.M1("ERROR no se puede filtrar .."+ex);
        }
    }
}
}

```

3.6.2 DAO

3.6.2.1 Conectar base de datos



The screenshot shows the NetBeans IDE interface. On the left is the Project Explorer pane, which lists several Java files under the DAO package. On the right is the Source editor pane, which displays the code for the `ConectarBD` class.

```

package DAO;
//librerias
import java.sql.*;
import Formatos.Mensajes;
public class ConectarBD implements Parametros{
    Connection conexion;
    Statement st;
    ResultSet rs;
    PreparedStatement ps;
    public ConectarBD(){
        try{
            Class.forName(DRIVER);
            conexion = DriverManager.getConnection(RUTA,USUARIO,CLAVE);
            st = conexion.createStatement();
        }catch(Exception ex){
            Mensajes.M1("ERROR no se puede conectar a la BD.."+ex);
        }
    }
}

```

3.6.2.2 Cargar combos

```

package DAO;
//librerias
import javax.swing.JComboBox;
import Formatos.*;
import java.sql.*;
public class CargarCombos extends ConectarBD{
    public CargarCombos(){}
    //metodo que carga los nombres de las categorias en un combo
    public void CargarCategoriasEnCombo(JComboBox combo){
        try{
            rs = st.executeQuery("SELECT nombre_categoria FROM categoria ORDER BY 1;");
            while(rs.next()){
                combo.addItem(rs.getString(1));
            }
            rs.close();
        }catch(Exception ex){
            Mensajes.Mi("ERROR no puede cargar categorias en el combo..."+ex);
        }
    }
    //metodo que carga los nombres de la compañia de los proveedores en un combo
    public void CargarstockEnCombo(JComboBox combo){
        try{
            rs = st.executeQuery("SELECT tipo_stock FROM stock ORDER BY 1;");
            while(rs.next()){
                combo.addItem(rs.getString(1));
            }
            rs.close();
        }catch(Exception ex){
            Mensajes.Mi("ERROR no puede cargar proveedores en el combo..."+ex);
        }
    }
    //metodo que carga los nombres de la compañia de los proveedores en un combo
    public void CargartamanoEnCombo(JComboBox combo){
        try{
            rs = st.executeQuery("SELECT proporcion FROM tamano ORDER BY 1;");
            while(rs.next()){
                combo.addItem(rs.getString(1));
            }
            rs.close();
        }catch(Exception ex){
            Mensajes.Mi("ERROR no puede cargar proveedores en el combo..."+ex);
        }
    }
    public void CargarUsuarioCombo (JComboBox combo){
        try{
            rs = st.executeQuery("SELECT tipoUiciar FROM tipousuario ORDER BY 1;");
            while(rs.next()){
                combo.addItem(rs.getString(1));
            }
            rs.close();
        }catch(Exception ex){
            Mensajes.Mi("ERROR no puede cargar los tipos de usuario..."+ex);
        }
    }
    public void CargarNombreEmpleado (JComboBox combo){
        try{
            rs = st.executeQuery("SELECT apellido FROM empleado ORDER BY 1;");
            while(rs.next()){
                combo.addItem(rs.getString(1));
            }
            rs.close();
        }catch(Exception ex){
            Mensajes.Mi("ERROR no puede cargar los tipos de usuario..."+ex);
        }
    }
}

```

3.6.2.3 CargarIDForaneo

```

package DAO;
import Formatos.*;
public class ManejadorIdForaneos extends ConectarBD{
public ManejadorIdForaneos(){}
//metodo que recuperar el Id del nombre de la categoria
public String RecuperarIdCategoria(String nomcat){
    String idcat = null;
    try{
        rs = st.executeQuery("SELECT cod_categoria FROM categoria WHERE nombre_categoria='"+nomcat+"'");
        if(rs.next()){
            idcat = rs.getString(1);
        }
    }catch(Exception ex){
        Mensajes.M1("ERROR no se puede recuperar el IDcategoria.."+ex);
    }
    return idcat;
}
//metodo que recupera el codigo del stock por medio de su nombre
public String RecuperarIdstock(String nomcom){
    String idstock = null;
    try{
        rs = st.executeQuery("SELECT cod_stock FROM stock WHERE tipo_stock='"+nomcom+"'");
        if(rs.next()){
            idstock = rs.getString(1);
        }
    }catch(Exception ex){
        Mensajes.M1("ERROR no se puede recuperar el codigo del stock.."+ex);
    }
    return idstock;
}
//metodo que recupera el codigo del tamano por medio de su nombre
public String RecuperarIdtamano(String nomtamo){
    String idtamano = null;
    try{
        rs = st.executeQuery("SELECT cod_tamano FROM tamano WHERE proporcion='"+nomtamo+"'");
        if(rs.next()){
            idtamano = rs.getString(1);
        }
    }catch(Exception ex){
        Mensajes.M1("ERROR no se puede recuperar el codigo del tamano.."+ex);
    }
    return idtamano;
}
//metodo que recupera el codigo del cod del tipo por medio de su nombre
public int RecuperarTipoU(String nomtipo){
    int tipou=0;
    int tipou=0;
    try{
        rs = st.executeQuery("SELECT cod_tipo FROM tipousuario WHERE tipoUsuar='"+nomtipo+"'");
        if(rs.next()){
            tipou = rs.getInt(1);
        }
    }catch(Exception ex){
        Mensajes.M1("ERROR no se puede recuperar el codigo del tamano.."+ex);
    }
    return tipou;
}
//metodo que recupera el codigo del codigo del empleado por medio de su apellido
public String RecuperarCodEmp(String codempl){
    String coemp = null;
    try{
        rs = st.executeQuery("SELECT cod_emp FROM empleado WHERE apellido='"+codempl+"';");
        if(rs.next()){
            coemp= rs.getString(1);
        }
    }catch(Exception ex){
        Mensajes.M1("ERROR no se puede recuperar el codigo del empleado.."+ex);
    }
    return coemp;
}

```

3.6.2.3 CRUD empleado

```

package DAO;
//libreria
import Formatos.Mensajes;
import Modelo.Empleadol;
import javax.swing.JTable;
import javax.swing.JLabel;
import javax.swing.table.DefaultTableModel;
import java.sql.SQLException;
public class CRUDEmpleado extends ConectarBD {
    //constructor
    public CRUDEmpleado() { }
    //metodo
    public void MostrarEmpleadoEnTabla(JTable tabla,JLabel etiqueta){
        String titulos[]={"num","codigo empleado","Nombre","Apellido","Direccion","Telefono","Sueldo","Edad","Dias Trabajados"};
        DefaultTableModel modelo = new DefaultTableModel(null,titulos);
        tabla.setModel(modelo);
        try{
            rs = st.executeQuery("SELECT cod_emp, nombre, apellido, direccion, telefono, sueldo, edad, dias_trabajado,indicador FROM empleado WHERE indicador='S'");
            int contreg=0;
            while(rs.next()){
                contreg++;
                Empleado em = new Empleado();
                em.setCod_emp(rs.getString(1));
                em.setNombre(rs.getString(2));
                em.setApellido(rs.getString(3));
                em.setDireccion(rs.getString(4));
                em.setTelefono(rs.getString(5));
                em.setSueldo(rs.getDouble(6));
                em.setEdad(rs.getInt(7));
                em.setDias_Trabajo(rs.getInt(8));
                em.setIndicador(rs.getString(9));
                modelo.addRow(em.RegistroDatos(contreg));
            }
            etiqueta.setText("Cantidad de Empleados : "+contreg);
            rs.close();
        }catch(SQLException ex){
            Mensajes.Mi("ERROR no se puede mostrar los registros de empleado.." +ex);
        }
    }//fin del metodo
    // metodo para insertar datos
    public void InsertarRegistroEmpleado(Empleado emp){
        try{
            ps = conexion.prepareStatement("INSERT INTO empleado(cod_emp, nombre, apellido, direccion, telefono, sueldo, edad, dias_trabajado, indicador) VALUES (?,?,?,?,?,?,?,?,?'S')");
            //actualizando los parametros
            ps.setString(1,emp.getCod_emp());
            ps.setString(2,emp.getNombre());
            ps.setString(3,emp.getApellido());
            ps.setString(4,emp.getDireccion());
            ps.setString(5,emp.getTelefono());
            ps.setDouble(6,emp.getSueldo());
            ps.setInt(7,emp.getEdad());
            ps.setInt(8,emp.getDias_Trabajo());
            ps.executeUpdate();
            ps.close();
            Mensajes.Mi("Registro Insertado");
        }catch(SQLException ex){
            Mensajes.Mi("ERROR no se puede insertar la categoria .." +ex);
        }
    }
    //metodo para obtener el registro
    public Empleado ObtenerRegistroEmp(String codemp){
        Empleado codem=null;
        try{
            rs = st.executeQuery("SELECT cod_emp, nombre, apellido, direccion, telefono, sueldo, edad, dias_trabajado,indicador FROM empleado where cod_emp ='"+codemp+"'");
            if(rs.next()){
                codem = new Empleado();
                codem.setCod_emp(rs.getString(1));
                codem.setNombre(rs.getString(2));
                codem.setApellido(rs.getString(3));
                codem.setDireccion(rs.getString(4));
                codem.setTelefono(rs.getString(5));
                codem.setSueldo(rs.getDouble(6));
                codem.setEdad(rs.getInt(7));
                codem.setDias_Trabajo(rs.getInt(8));
                codem.setIndicador(rs.getString(9));
            }
        }catch(Exception ex){
            Mensajes.Mi("ERROR no se puede recuperar el registro de empleados.." +ex);
        }
        return codem;
    }
    //fin metodo
    //metodo que recibir un empleado y actualiza al empleado
    public void ActualizarEmpleado(Empleado emp){
        try{
            ps = conexion.prepareStatement("UPDATE empleado SET nombre=? ,apellido=? ,direccion=? ,telefono=? ,sueldo=? ,edad=? ,dias_trabajado=? WHERE cod_emp=?");
            ps.setString(1,emp.getNombre());
            ps.setString(2,emp.getApellido());
            ps.setString(3,emp.getDireccion());
            ps.setString(4,emp.getTelefono());
            ps.setDouble(5,emp.getSueldo());
        }
    }
}

```

```

        ps.setInt(6,emp.getEdad());
        ps.setInt(7,emp.getDias_Trabajo());
        ps.setString(8,emp.getCod_emp());
        ps.executeUpdate();
        Mensajes.M1("Registro Actualizado");
        ps.close();
    }catch(Exception ex){
        Mensajes.M1("ERROR no se puede actualizar el registro del empleado.." +ex);
    }
}

//metodo eliminar datos
public void EliminarEmpleado(Empleadol emp) {
    try{
        ps = conexion.prepareStatement("DELETE from empleado WHERE cod_emp=?");
        ps.setString(1,emp.getCod_emp());
        ps.executeUpdate();
        Mensajes.M1("Registro eliminado de la tabla Empleado");
        ps.close();
    }catch(Exception ex){
        Mensajes.M1("ERROR no se puede eliminar el Empleado.." +ex);
    }
}
}

```

3.6.2.4 CRUD FACTURAS

```

package DAO;
//libreria
import Formatos.*;
import Modelo.Factura;
import javax.swing.JTable;
import javax.swing.JLabel;
import javax.swing.table.DefaultTableModel;
import java.sql.SQLException;
public class CRUDfacturas extends ConectarBD {
    //constructor
    public CRUDfacturas(){}
    //metodo para oteber datos en la tabla
    public void MostrarFacturasEnTabla(JTable tabla,JLabel etiqueta){
        String titulos[]{"num","codigo Factura","DNI Cliente","Codigo empleado","Descripcion","Fecha","Total"};
        DefaultTableModel modelo = new DefaultTableModel(null,titulos);
        tabla.setModel(modelo);
        try{
            rs = st.executeQuery("SELECT cod_factura, dni_cliente, cod_emp, descripcion,fecha,total FROM facturas");
            int contreg=0;
            while(rs.next()){
                contreg++;
                Facture fa = new Factura();
                fa.setCod_factura(rs.getString(1));
                fa.setDNI_Cliente(rs.getString(2));
                fa.setCod_emp(rs.getString(3));
                fa.setDescripcion(rs.getString(4));
                fa.setFecha(rs.getString(5));
                fa.setTotal(rs.getDouble(6));
                modelo.addRow(fa.RegistroDatos(contreg));
            }
        }catch(SQLException ex){
            Mensajes.M1("ERROR no se puede mostrar los registros de empleado.." +ex);
        }
    }
    //fin del metodo
    //metodo para insertar datos
    public void InsertarRegistroFactura(Factura fac){
        try{
            ps = conexion.prepareStatement("INSERT INTO facturas(cod_factura,dni_cliente, cod_emp, descripcion, fecha, Total) VALUES (?,?,?,?,?,?)");
            //actualizando los parametros
            ps.setString(1,fac.getCod_factura());
            ps.setString(2,fac.getDNI_Cliente());
            ps.setString(3,fac.getCod_emp());
            ps.setString(4,fac.getDescripcion());
        }
    }
}

```

```

        ps.setString(5,fac.getFecha());
        ps.setDouble(6,fac.getTotal());
        ps.executeUpdate();
        ps.close();
        Mensajes.MI("Registro Insertado");
    }catch(SQLException ex){
        Mensajes.MI("ERROR no se puede insertar la Factura .."+ex);
    }
}

//metodo para obtener el registro mediante el codigo de factura
public Factura ObtenerRegistroFactura(String codfac){
    Factura codfa=null;
    try{
        rs = st.executeQuery("SELECT cod_factura, dni_cliente, cod_emp, descripcion,fecha,total FROM facturas where cod_factura ='"+codfac+"'");
        if(rs.next()){
            codfa = new Factura();
            codfa.setCod_factura(rs.getString(1));
            codfa.setDNI_Cliente(rs.getString(2));
            codfa.setCod_emp(rs.getString(3));
            codfa.setDescripcion(rs.getString(4));
            codfa.setFecha(rs.getString(5));
            codfa.setTotal(rs.getDouble(6));
        }
    }catch(SQLException ex){
        Mensajes.MI("ERROR no se puede recuperar el registro de la factura.."+ex);
    }
    return codfa;
}//fin metodo
//metodo para actualizar datos
public void ActualizarFactura(Factura fac){
    try{
        ps = conexion.prepareStatement("UPDATE facturas SET dni_cliente=? ,cod_emp=?,descripcion=?,fecha=?,total=? WHERE cod_factura=?");
        ps.setString(1,fac.getDNI_Cliente());
        ps.setString(2,fac.getCod_emp());
        ps.setString(3,fac.getDescripcion());
        ps.setString(4,fac.getFecha());
        ps.setDouble(5,fac.getTotal());
        ps.setString(6,fac.getCod_factura());
        ps.executeUpdate();
        Mensajes.MI("Registro Actualizado");
        ps.close();
    }catch(SQLException ex){
        Mensajes.MI("ERROR no se puede actualizar el registro del empleado.."+ex);
    }
}

//metodo para eliminar datos
public void EliminarFactura(Factura fac){
    try{
        ps = conexion.prepareStatement("DELETE from facturas WHERE cod_factura=?");
        ps.setString(1,fac.getCod_factura());
        ps.executeUpdate();
        Mensajes.MI("Registro eliminado de la tabla factura");
        ps.close();
    }catch(SQLException ex){
        Mensajes.MI("ERROR no se puede eliminar el Empleado.."+ex);
    }
}
}

```

3.6.2.5 CRUD LOGIN

```

package DAO;
//libreria
import Modelo.RegistroUsuario;
import Formatos.Mensajes;
import vista.MenuPrincipal;
import vista.MenuEmpleado;
import controlador.ControladorMenuPrincipal;
import controlador.ControladorMenuEmpleado;
import java.sql.SQLException;
import java.util.ArrayList;
import vista.*;
public class CRUDlogin extends ConectarBD{
    //constructor
    public CRUDlogin() {
        MenuPrincipal menu = new MenuPrincipal();
        MenuEmpleado menu2 = new MenuEmpleado();
        RegistroUsuario lo = new RegistroUsuario();
        Login login = new Login();
    }
    //metodo que ingresar al sistema
    public void IngresarSistema(String usuario, String contrasena, int codtipo) {
        String sql = "SELECT Usuario, contrasena FROM registro WHERE Usuario='"+usuario+"' AND contrasena='"+contrasena+"' and cod_tipo =" + codtipo;
        try {
            ps = conexion.prepareStatement(sql);
            rs = ps.executeQuery();
            if (rs.next()) {
                usuario = rs.getString("usuario");
                contrasena = rs.getString("contrasena");
                if (codtipo == 2) {
                    ControladorMenuPrincipal cmp = new ControladorMenuPrincipal(menu);
                    menu.setVisible(true);
                    Mensajes.Mi("Bienvenido al Sistema");
                    login.setVisible(false);
                }
                if (codtipo == 1) {
                    ControladorMenuEmpleado cme = new ControladorMenuEmpleado(menu2);
                    menu2.setVisible(true);
                    Mensajes.Mi("Bienvenido al Sistema");
                    login.setVisible(false);
                }
            } else {
                Mensajes.Mi("Usuario o Contraseña incorrecto");
            }
        } catch (SQLException ex) {
            Mensajes.Mi("ERROR no se puede Ingresar al Sistema.." + ex);
        }
    }
}

```

3.6.2.6 CRUD PRODUCTOS

```

package DAO;
//libreria
import Modelo.Al_Producto;
import Formatos.Mensajes;
import javax.swing.JTable;
import javax.swing.JLabel;
import javax.swing.table.DefaultTableModel;
import Formatos.HaneadorFila;
import javax.swing.table.DefaultTableModel;
import java.sql.SQLException;
public class CRUDproductos extends ConectarBD{
    public CRUDproductos() {
    }
    //metodo para mostrar datos en la tabla
    public void MostrarProductoEnTabla(JTable tabla, JLabel etiqueta) {
        String[] TitulosTabla={"Num","Codigo Producto","Nombre Producto","Precio","Stock","Categoria","Tamaño"};
        DefaultTableModel modelo = new DefaultTableModel(null,TitulosTabla);
        tabla.setModel(modelo);
        try {
            rs = st.executeQuery("SELECT cod_prod, nombre_pro, precio, cod_stock, cod_categoria,cod_tamano FROM productos;");
            int contreg=0;
            while(rs.next()){
                contreg++;
                Al_Producto prod = new Al_Producto();
                prod.setCod_Producto(rs.getString(1));
                prod.setNombre_F(rs.getString(2));
                prod.setPrecio(rs.getDouble(3));
                prod.setStock(rs.getString(4));
                prod.setCategoria(rs.getString(5));
                prod.setTamano(rs.getString(6));
                modelo.addRow(prod.RegistroDatos(contreg));
            }
            etiqueta.setText("Cantidad de Productos : "+contreg);
            conexion.close();
        }catch(Exception ex){
            Mensajes.Mi("ERROR no se puede mostrar los registros de productos.." +ex);
        }
    }
    //fin del metodo
    //metodo que inserta en registro a la tabla productos
    public void InsertarRegistroProd(Al_Producto pro) {
        try {
            ps = conexion.prepareStatement("INSERT INTO productos(cod_prod, nombre_pro, precio, cod_stock, cod_categoria,cod_tamano) VALUES (?, ?, ?, ?, ?, ?);");
            ps.setString(1,pro.getCod_Producto());
            ps.setString(2,pro.getNombre_P());
            ps.setDouble(3,pro.getPrecio());
            ps.setString(4,pro.getStock());
        }
    }
}

```

```

        ps.setString(5,pro.getCategoría());
        ps.setString(6,pro.getTamaño());
        ps.executeUpdate();
        ps.close();
        Mensajes.Mi(" registrado correctamente!!!");
    }catch(Exception ex){
        Mensajes.Mi("ERROR no se puede registrar.." +ex);
    }
}

//metodo para obtener datos mediante el codigo del producto
public Al_Producto ObtenerRegistroProducto(String codpro){
    Al_Producto codpo=null;
    try{
        rs = st.executeQuery("SELECT cod_prod, nombre_pro, precio, cod_stock, cod_categoria,cod_tamano FROM productos where cod_factura ='"+codpro+"'");
        if(rs.next()){
            codpo = new Al_Producto();
            codpo.setCod_Producto(rs.getString(1));
            codpo.setNombre_P(rs.getString(2));
            codpo.setPrecio(rs.getDouble(3));
            codpo.setStock(rs.getString(4));
            codpo.setCategoria(rs.getString(5));
            codpo.setTamaño(rs.getString(6));
        }
    }catch(SQLException ex){
        Mensajes.Mi("ERROR no se puede recuperar el registro de la factura.." +ex);
    }
    return codpo;
}//fin metodo
//metodo para actualizar productos
public void ActualizarProducto(Al_Producto pro){
    try{
        ps = conexion.prepareStatement("UPDATE productos SET nombre_pro=?,precio=?,cod_stock=?,cod_categoria=?,cod_tamano=? WHERE cod_prod=?");
        ps.setString(1,pro.getNombre_P());
        ps.setDouble(2,pro.getPrecio());
        ps.setString(3,pro.getStock());
        ps.setString(4,pro.getCategoría());
        ps.setString(5,pro.getTamaño());
        ps.setString(6,pro.getCod_Producto());
        ps.executeUpdate();
        Mensajes.Mi("Registro Actualizado");
        ps.close();
    }catch(SQLException ex){
        Mensajes.Mi("ERROR no se puede actualizar el registro del producto.." +ex);
    }
}

//metodo eliminar datos de la tabla productos
public void EliminarFactura(Al_Producto pro){
    try{
        ps = conexion.prepareStatement("DELETE from productos WHERE cod_prod=?");
        ps.setString(1,pro.getCod_Producto());
        ps.executeUpdate();
        Mensajes.Mi("Registro eliminado de la tabla Producto");
        ps.close();
    }catch(SQLException ex){
        Mensajes.Mi("ERROR no se puede eliminar el Producto.." +ex);
    }
}

//metodo para filtrar la categoria en tabla
public void MostrarCategoriaEnTabla(JTable tabla,String codpro){
    String[] TitulosTabla={"Num","Codigo Producto","Nombre Producto","Precio","Stock","Categoria","Tamaño"};
    DefaultTableModel modelo = new DefaultTableModel(null,TitulosTabla);
    tabla.setModel(modelo);
    try{
        rs = st.executeQuery("SELECT cod_prod, nombre_pro, precio, cod_stock, cod_categoria,cod_tamano FROM productos WHERE cod_categoria ='"+codpro+"'");
        int contreg=0;
        while(rs.next()){
            contreg++;
            Al_Producto prod = new Al_Producto();
            prod.setCod_Producto(rs.getString(1));
            prod.setNombre_P(rs.getString(2));
            prod.setPrecio(rs.getDouble(3));
            prod.setStock(rs.getString(4));
            prod.setCategoria(rs.getString(5));
            prod.setTamaño(rs.getString(6));
            modelo.addRow(prod.RegistroDatos(contreg));
        }
        //fin while
        conexion.close();
    }catch(Exception ex){
        Mensajes.Mi("ERROR no se puede mostrar los registros de productos.." +ex);
    }
}
}

```

3.6.2.7 CRUD REGISTRO

```

package DAO;
//libreria
import Modelos.RegistroUsuario;
import Formatos.Mensajes;
public class CRUDregistro extends ConectarBD{
    //metodo que inserta en registro de datos
    public void InsertarRegistroProd(RegistroUsuario re){
        try{
            ps = conexion.prepareStatement(
                "INSERT INTO registro(Usuario, NombreUsuario, ApellidoUsuario, Email, contrasena,cod_tipo)" +
                " values(?,?,?,?,?,?)");
            ps.setString(1,re.getUsuario());
            ps.setString(2,re.getNombre_u());
            ps.setString(3,re.getApellido_u());
            ps.setString(4,re.getEmail_u());
            ps.setString(5,re.getPassword());
            ps.executeUpdate();
            ps.close();
            Mensajes.Mi(" registrado correctamente!!!");
        }catch(Exception ex){
            Mensajes.Mi("ERROR no se puede registrar.."+ex);
        }
    }
}

```

3.6.2.8 CRUD REPORTEDIARIO

```

package DAO;
//libreria
import Formatos.*;
import Modelos.ReporteDiario;
import javax.swing.JTable;
import javax.swing.JLabel;
import javax.swing.table.DefaultTableModel;
import java.sql.SQLException;
public class CRUDDiario extends ConectarBD {
    public CRUDDiario(){}
    //metodo para insertar datos en la tabla
    public void MostrarReporteEnTabla(JTable tabla,JLabel etiqueta){
        String titulos[]={"num","Codigo empleado","Producto vendido","Cantidad de Productos","Fecha","Dinero Inicial","Dinero Final","Dinero Sobrante"};
        DefaultTableModel modelo = new DefaultTableModel(null,titulos);
        tabla.setModel(modelo);
        try{
            rs = st.executeQuery("SELECT N_reporte,cod_emp, productoVendido, cantidadProductos, fecha, dineroInicial, dineroFinal, dineroSobrante FROM ReporteDiario");
            int contreg=0;
            while(rs.next()){
                contreg++;
                ReporteDiario red = new ReporteDiario();
                red.setNfactura(rs.getString(1));
                red.setCod_emp(rs.getString(2));
                red.setProducto(rs.getString(3));
                red.setCantidad(rs.getInt(4));
                red.setFecha(rs.getString(5));
                red.setDinero_i(rs.getDouble(6));
                red.setTotal_D(rs.getDouble(7));
                modelo.addRow(red.RegistroDatos(contreg));
            }
            etiqueta.setText("Cantidad de Reporte Diario : "+contreg);
            conexion.close();
        }catch(SQLException ex){
            Mensajes.Mi("ERROR no se puede mostrar los Reporte Diario.."+ex);
        }
    }
    //fin del metodo
    //metodo para insertar datos en la base de datos
    public void InsertarRegistroReporteDiario(ReporteDiario rep){
        try{
            ps = conexion.prepareStatement("INSERT INTO ReporteDiario(cod_emp, productoVendido, cantidadProductos, fecha, dineroInicial, dineroFinal, dineroSobrante) VALUES (?,?,?,?,?,?,?,?)");
            //actualizar los parametros
            ps.setString(1,rep.getCod_emp());
            ps.setString(2,rep.getProducto());
            ps.setInt(3,rep.getCantidad());
            ps.setDouble(4,rep.getDinero_i());
            ps.setDouble(5,rep.getDinero_i());
            ps.setDouble(6,rep.getDinero_i());
            ps.setDouble(7,rep.getSobrante() );
            ps.executeUpdate();
            Mensajes.Mi("Registro Insertado");
        }catch(SQLException ex){
            Mensajes.Mi("ERROR no se puede insertar el Reporte Diario .."+ex);
        }
    }
    //metodo para obtener el registro mediante el Numero de reporte
    public ReporteDiario ObtenerRegistroReporte(String codrept){
        ReporteDiario codrd=null;
        try{
            rs = st.executeQuery("SELECT N_reporte,cod_emp, productoVendido, cantidadProductos, fecha, dineroInicial, dineroFinal, dineroSobrante FROM ReporteDiario where N_reporte = '"+codrept+"'");
            if(rs.next()){
                codrd=new ReporteDiario();
                codrd.setNfactura(rs.getString(1));
                codrd.setCod_emp(rs.getString(2));
                codrd.setProducto(rs.getString(3));
                codrd.setCantidad(rs.getInt(4));
                codrd.setFecha(rs.getString(5));
                codrd.setDinero_i(rs.getDouble(6));
                codrd.setTotal_D(rs.getDouble(7));
            }
        }catch(SQLException ex){
            Mensajes.Mi("ERROR no se puede recuperar el registro de reporte.."+ex);
        }
        return codrd;
    }
    //metodo que recibe una reporte y lo actualiza
    public void ActualizarReporte(ReporteDiario nrd){
        try{
            ps = conexion.prepareStatement("UPDATE ReporteDiario SET cod_emp=? ,productoVendido=? ,cantidadProductos=? ,fecha=? ,dineroInicial=? ,dineroFinal=? ,dineroSobrante=? WHERE N_reporte=?");
            ps.setString(1,nrd.getCod_emp());
            ps.setString(2,nrd.getProducto());
            ps.setInt(3,nrd.getCantidad());
            ps.setString(4,nrd.getFecha());
            ps.setDouble(5,nrd.getDinero_i());
            ps.setDouble(6,nrd.getTotal_D());
            ps.setDouble(7, nrd.Sobrante());
            ps.setString(8,nrd.getNfactura());
            ps.executeUpdate();
            Mensajes.Mi("Registro Actualizado");
        }

```

```

        Mensajes.M1("Registro Actualizado");
        ps.close();
    }catch(SQLException ex){
        Mensajes.M1("ERROR no se puede actualizar el registro del reporte.." +ex);
    }
}

//metodo elimina un reporte
public void EliminarReporte(ReporteDiario codrd) {
    try{
        ps = conexion.prepareStatement("DELETE from ReporteDiario WHERE N_reporte=?");
        ps.setString(1,codrd.getCodFactura());
        ps.executeUpdate();
        Mensajes.M1("Registro eliminado del Reporte");
        ps.close();
    }catch(SQLException ex){
        Mensajes.M1("ERROR no se puede eliminar el n de reporte.." +ex);
    }
}
}

```

3.6.2.9 CRUD REPORTESEMANAL

The screenshot shows the Eclipse IDE interface with the project structure on the left and the code editor on the right.

Project Structure:

- Paul Pizza
- Proyecto_PaulPizza
- Source Packages
 - DAO
 - CRUDempleado.java
 - CRUDfacturas.java
 - CRUDfacturas.java
 - CRUDproductos.java
 - CRUDregistro.java
 - CRUDreporteDiario.java
 - CRUDreporteSemanal.java
 - CRUDvisualizar.java
 - CargarCombos.java
 - ConectarBD.java
 - ManejadorDeReportes.java
 - Formatos
 - FormatoFormas.java
 - FormatoEmpleado.java
 - FormatoFactura.java
 - FormatoLogin.java
 - FormatoProductos.java
 - FormatoRegistrar.java
 - FormatoReporteDiario.java
 - ManejadorTablas.java
 - Mensajes.java
 - Imagenes
 - paul pizza.png
 - reporte.png
 - Modelo
 - Al_Producto.java
 - Categoria.java
 - Empleado.java
 - Empleado_abs.java
 - Factura.java
 - Producto.java
 - Producto_abs.java
 - ReporteDiario.java
 - ReporteDiario.java
 - Stock.java
 - Usuario.java
 - Principal
 - Principal.java
 - PrincipalControlador.java
 - controlador
 - ControladorEmpleado.java
 - Controladorfactura.java

Code Editor (CRUDeReporteSemanal.java):

```

package DAO;
//librerias
import Formatos.*;
import Modelo.ReporteDiario;
import javax.swing.JTable;
import javax.swing.JLabel;
import javax.swing.table.DefaultTableModel;
import Formatos.ManejadorTablas;
import Modelo.Empleado;
import java.sql.SQLException;
public class CRUDeReporteSemanal extends ConectarBD {
    public CRUDeReporteSemanal() {
    }
    //metodo para mostrar la tabla
    public void MostrarReporteEnTabla(JTable tabla,Label etiqueta){
        String titulos[]={"num","Reporte","Codigo empleado","Producto vendido","Cantidad de Productos","Fecha","Dinero Inicial","Dinero Final","Dinero Sobrante"};
        DefaultTableModel modelo = new DefaultTableModel(null,titulos);
        tabla.setModel(modelo);
        try{
            rs = st.executeQuery("SELECT N_reporte,cod_emp, productoVendido, cantidadProductos, fecha, dineroInicial, dineroFinal, dineroSobrante FROM ReporteDiario");
            int contreg=0;
            while(rs.next()){
                ContadorReporteDiario red = new ReporteDiario();
                red.setCodFactura(rs.getString(1));
                red.setCod_emp(rs.getString(2));
                red.setProducto(rs.getString(3));
                red.setCantidad(rs.getInt(4));
                red.setFecha(rs.getString(5));
                red.setDinero_i(rs.getDouble(6));
                red.setTotal_D(rs.getDouble(7));
                modelo.addRow(red.RegistroDatos(contreg));
                contreg++;
            }
        }catch(SQLException ex){
            etiqueta.setText("Cantidad de Reporte Diario : "+contreg);
            etiqueta.setVisible(true);
            etiqueta.setForeground(Color.red);
        }
    }
    //metodo para filtrar por fecha
    public void FiltraFecha(ReporteDiario rep){
        try{
            ps = conexion.prepareStatement("SELECT * FROM ReporteDiario WHERE fecha > ? and fecha< ?");
            //actualizando los parametros
            ps.setString(1,rep.getFecha());
            ps.setString(2,rep.getFecha());
            ps.setString(2,rep.getFecha());
            ps.executeUpdate();
            ps.close();
            Mensajes.M1("Filtrado Correctamente");
        }catch(SQLException ex){
            Mensajes.M1("ERROR no se puede filtrar.." +ex);
        }
    }
    //fin del metodo
}

```

3.6.2.10 CRUD REPORTEMENSUAL

```

package DAO;
//libreria
import Formatos.*;
import Modelo.ReporteDiario;
import javax.swing.JTable;
import javax.swing.JLabel;
import javax.swing.table.DefaultTableModel;
import Formatos.ManejadorTablas;
import Modelo.Empleado;
import java.sql.SQLException;
import java.util.Date;
public class CRUDReporteMensual extends ConectarBD {
    public CRUDReporteMensual() {
        //metodo para filtrar por mes y año
        public void MostrarFiltroFecha(JTable tabla, String fechaf) {
            String titulos[] = {"Num", "NºReporte", "Codigo empleado", "Pizza Vendidas", "Pastas Vendidas", "Bebidas Vendidas", "Piqueos Vendidas", "Platos Criollos Vendidos", "Fecha", "Dinero Inicial", "Dinero Final", "Dinero Sobra"};
            DefaultTableModel modelo = new DefaultTableModel(null, titulos);
            tabla.setModel(modelo);
            try {
                rs = st.executeQuery("SELECT N_reporte,cod_emp, pizzavendidas, pastasvendidas, bebidasvendidas, piqueosvendidas, platosCvendidos, fecha, dineroInicial, dineroFinal, dineroSobranteFROM");
                tabla.setModel(modelo);
                while(rs.next()) {
                    contreg++;
                    ReporteDiario red = new ReporteDiario();
                    red.setNfactura(rs.getString(1));
                    red.setCod_emp(rs.getString(2));
                    red.setPizzav(rs.getInt(3));
                    red.setPastav(rs.getInt(4));
                    red.setBebidav(rs.getInt(5));
                    red.setPiqueosv(rs.getInt(6));
                    red.setPlatoscv(rs.getInt(7));
                    red.setFecha(rs.getString(8));
                    red.setDinero_i(rs.getDouble(9));
                    red.setTotal_D(rs.getDouble(10));
                    modelo.addRow(red.RegistroDatos(contreg));
                }
            } catch (SQLException ex) {
                Mensajes.MI("ERROR no se puede mostrar el filtro.."+ex);
            }
        }
    }
}

```

3.6.2.11 CRUD VISUALIZAR

```

package DAO;
//libreria
import Modelo.Al_Producto;
import Formatos.Mensajes;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import java.sql.SQLException;
public class CRUDvisualizar extends ConectarBD {
    public CRUDvisualizar() {
        //metodo para mostrar datos en la tabla
        public void MostrarStockEnTabla(JTable tabla, String jcbcStockV) {
            String[] TitulosTabla = {"Num", "Codigo Producto", "Nombre Producto", "Precio", "Stock", "Categoria", "Tamaño"};
            DefaultTableModel modelo = new DefaultTableModel(null, TitulosTabla);
            tabla.setModel(modelo);
            try {
                rs = st.executeQuery("SELECT cod_prod, nombre_pro, precio, cod_stock, cod_categoria, cod_tamano FROM productos WHERE cod_stock ='" + jcbcStockV + "'");
                int contreg=0;
                while(rs.next()) {
                    contreg++;
                    Al_Producto prod = new Al_Producto();
                    prod.setCod_Producto(rs.getString(1));
                    prod.setNombre_P(rs.getString(2));
                    prod.setPrecio(rs.getDouble(3));
                    prod.setStock(rs.getString(4));
                    prod.setCategoria(rs.getString(5));
                    prod.setTamano(rs.getString(6));
                    modelo.addRow(prod.RegistroDatos(contreg));
                }
            } catch (SQLException ex) {
                Mensajes.MI("ERROR no se puede mostrar los registros de productos.."+ex);
            }
        }
        //fin del metodo
        public void MostrarCategoriaEnTabla(JTable table, String jcbcCategoriaV) {
            String[] TitulosTabla = {"Num", "Codigo Producto", "Nombre Producto", "Precio", "Stock", "Categoria", "Tamaño"};
            DefaultTableModel modelo = new DefaultTableModel(null, TitulosTabla);
            tabla.setModel(modelo);
            try {
                rs = st.executeQuery("SELECT cod_prod, nombre_pro, precio, cod_stock, cod_categoria, cod_tamano FROM productos WHERE cod_categoria ='" + jcbcCategoriaV + "'");
                int contreg=0;
                while(rs.next()) {
                    contreg++;
                    Al_Producto prod = new Al_Producto();
                    prod.setCod_Producto(rs.getString(1));
                    prod.setNombre_P(rs.getString(2));

```

3.6.2.11 CRUD VISUALIZAR

```

        prod.setPrecio(rs.getDouble(3));
        prod.setStock(rs.getString(4));
        prod.setCategoria(rs.getString(5));
        prod.setTamaño(rs.getString(6));
        modelo.addRow(prod.RegistroDatos(contreg));
    }//fin while
    conexion.close();
}catch(SQLException ex){
    Mensajes.Mi("ERROR no se puede mostrar los registros de productos.."+ex);
}
}

public void MostrarTamanoEnTabla(JTable tabla, String jFiltroTamano){
String[] TitulosTabla={"Num","Codigo Producto","Nombre Producto","Precio","Stock","Categoria","Tamaño"};
DefaultTableModel modelo = new DefaultTableModel(null,TitulosTabla);
tabla.setModel(modelo);
try{
    rs = st.executeQuery("SELECT cod_prod, nombre_pro, precio, cod_stock, cod_categoria,cod_tamano FROM productos WHERE cod_tamano ='"+jFiltroTamano+"'");
    int contreg=0;
    while(rs.next()){
        contreg++;
        Al_Producto prod = new Al_Producto();
        prod.setCod_Producto(rs.getString(1));
        prod.setNombre_P(rs.getString(2));
        prod.setPrecio(rs.getDouble(3));
        prod.setStock(rs.getString(4));
        prod.setCategoria(rs.getString(5));
        prod.setTamaño(rs.getString(6));
        modelo.addRow(prod.RegistroDatos(contreg));
    }
    conexion.close();
}catch(SQLException ex){
    Mensajes.Mi("ERROR no se puede mostrar los registros de productos.."+ex);
}
}

public void MostrarProductoEnTabla(JTable tabla, String jFiltroproducto){
String[] TitulosTabla={"Num","Codigo Producto","Nombre Producto","Precio","Stock","Categoria","Tamaño"};
DefaultTableModel modelo = new DefaultTableModel(null,TitulosTabla);
tabla.setModel(modelo);
try{
    rs = st.executeQuery("SELECT cod_prod, nombre_pro, precio, cod_stock, cod_categoria,cod_tamano FROM productos WHERE nombre_pro like '%"+jFiltroproducto+"%'");
    int contreg=0;
    while(rs.next()){
        contreg++;
        Al_Producto prod = new Al_Producto();
        prod.setCod_Producto(rs.getString(1));
        prod.setNombre_P(rs.getString(2));
        prod.setPrecio(rs.getDouble(3));
        prod.setStock(rs.getString(4));
        prod.setCategoria(rs.getString(5));
        prod.setTamaño(rs.getString(6));
        modelo.addRow(prod.RegistroDatos(contreg));
    }
    conexion.close();
}catch(SQLException ex){
    Mensajes.Mi("ERROR no se puede mostrar los registros de productos.."+ex);
}
}

public void MostrarEnTabla(JTable tabla, String jFiltroprecio){
String[] TitulosTabla={"Num","Codigo Producto","Nombre Producto","Precio","Stock","Categoria","Tamaño"};
DefaultTableModel modelo = new DefaultTableModel(null,TitulosTabla);
tabla.setModel(modelo);
try{
    rs = st.executeQuery("SELECT cod_prod, nombre_pro, precio, cod_stock, cod_categoria,cod_tamano FROM productos WHERE nombre_pro like '%"+jFiltroprecio+"%'");
    int contreg=0;
    while(rs.next()){
        contreg++;
        Al_Producto prod = new Al_Producto();
        prod.setCod_Producto(rs.getString(1));
        prod.setNombre_P(rs.getString(2));
        prod.setPrecio(rs.getDouble(3));
        prod.setStock(rs.getString(4));
        prod.setCategoria(rs.getString(5));
        prod.setTamaño(rs.getString(6));
        modelo.addRow(prod.RegistroDatos(contreg));
    }
    rs.close();
}catch(SQLException ex){
    Mensajes.Mi("ERROR no se puede mostrar los registros de productos.."+ex);
}
}
}

```

3.6.2.12 CRUD DETALLE FACTURA

```

package DAO;
import Formatos.*;
import Modelo.Factura;
import Modelo.RegistroFactura;
import javax.swing.JTable;
import javax.swing.JLabel;
import javax.swing.table.DefaultTableModel;
import java.sql.SQLException;
import java.util.Date;
public class CRUDDetallefactura extends ConectarBD {
    int r;
    //constructor
    public CRUDDetallefactura() {
        //metodo para insertar datos
    }
    public boolean InsertarRegistroFactura(String dni, String cod_em, String fecha, double total) {
        try {
            ps = conexion.prepareStatement("INSERT INTO facturas(dni_cliente, cod_emp, fecha, Total) VALUES (?, ?, ?, ?)");
            //actualizando los parametros
            ps.setString(1, dni);
            ps.setString(2, cod_em);
            ps.setString(3, fecha);
            ps.setDouble(4, total);
            ps.executeUpdate();
            ps.close();
            Mensajes.MI("Registro Insertado");
            return true;
        } catch (SQLException ex) {
            Mensajes.MI("ERROR no se puede insertar la Factura .."+ex);
            return false;
        }
    }
    public boolean Insertardetallefactura(String pro, String des, int cant, double preu, double total, int cof_fa) {
        try {
            ps = conexion.prepareStatement("INSERT INTO detallefactura(cod_prod, descripcion, cantidad, PrecioUni, PrecioTotal, cod_factura) VALUES (?, ?, ?, ?, ?, ?)");
            //actualizando los parametros
            ps.setString(1, pro);
            ps.setString(2, des);
            ps.setInt(3, cant);
            ps.setDouble(4, preu);
            ps.setDouble(5, total);
            ps.setInt(6, cof_fa);
            ps.executeUpdate();
            ps.close();
            return true;
        } catch (SQLException ex) {
            Mensajes.MI("ERROR no se puede insertar la Factura .."+ex);
            return false;
        }
    }
    public void MostrarCodDetalleFactura(JTable tabla, String cod_factura) {
        String titulos[] = {"num", "Codigo detalle", "Codigo producto", "Descripcion", "Cantidad", "Precio Unitario", "Precio Total", "Codigo Factura"};
        DefaultTableModel modelo = new DefaultTableModel(null, titulos);
        tabla.setModel(modelo);
        try {
            rs = st.executeQuery("SELECT * FROM detallefactura WHERE cod_factura like '%"+cod_factura+"%'");
            int contreg=0;
            while(rs.next()){
                contreg++;
                RegistroFactura fa = new RegistroFactura();
                fa.setCod_detalle(rs.getInt(1));
                fa.setCod_pro(rs.getString(2));
                fa.setDescripcion(rs.getString(3));
                fa.setCantidad(rs.getInt(4));
                fa.setPrecioUni(rs.getDouble(5));
                fa.PrecTotal();
                fa.setCod_factura(rs.getInt(7));

                modelo.addRow(fa.RegistroDatosDetalle(contreg));
            }
            rs.close();
        } catch (SQLException ex) {
            Mensajes.MI("ERROR no se puede mostrar los registros de Factura.."+ex);
        }
    }
}

```

3.7 Descripción de los formulario

3.7.1 Login: el formulario permite ingresar al administrar y empleados pero dependiendo de quien Sea se le mostrara diferente tipo de menú



Registrar

-

□

X

PAUL PIZZAS

Tipo de Usuario: ACESOR DE VENTAS ▾

Usuario:

Contraseña:

INICIAR

REGISTRAR

3.7.2 Registro : El formulario permite registrar solo usuarios de empleados para ingresar el sistema

The screenshot shows a registration form titled "REGISTRO USUARIO". The form includes fields for "Nombre de Usuario", "Nombres", "Apellidos", "Email", and "Contraseña". At the bottom are two buttons: "Registrar" and "Regresar".

REGISTRO USUARIO

Nombre de Usuario:

Nombres:

Apellidos:

Email:

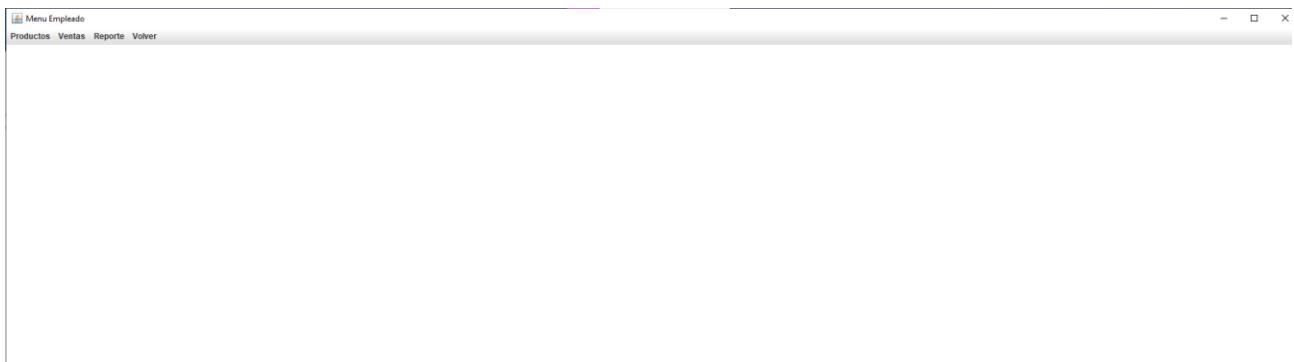
Contraseña:

Registrar **Regresar**

3.7.3 Menu administrador : el formulario permite al administrador tener un menú con mas funciones donde pueda administrar los empleados , productos y reportes semanales y mensuales



3.7.4 Menu empleado: el formulario permite al empleado visualizar los productos, insertar facturas y reportes diarios



3.7.5 Visualizar : el formulario permite al empleado observar y filtrar todos los productos que hay en la base de datos que el administrador inserta, actualizar o elimina.

Visualizar

Productos

Cantidad de Productos : 78

Filtro Por Producto:

	Num	Codigo Producto	Nombre Producto	Precio	Stock	Categoría	Tamaño
1	pr1		Pizza Americana	S/ 12.00	s1	ct1	t1
2	pr10		Pizza Delicia	S/ 16.00	s1	ct1	t1
3	pr11		Pizza Choripizza	S/ 17.00	s1	ct1	t1
4	pr12		Pizza Lomo Ahumado	S/ 20.00	s1	ct1	t1
5	pr13		Pizza Paul Espacial	S/ 25.00	s1	ct1	t1
6	pr14		Pizza Americana	S/ 23.00	s1	ct1	t2
7	pr15		Pizza Mozzarella	S/ 24.00	s1	ct1	t2
8	pr16		Pizza Jamon	S/ 25.00	s1	ct1	t2
9	pr17		Pizza Salame	S/ 25.00	s1	ct1	t2
10	pr18		Pizza Oriental	S/ 26.00	s1	ct1	t2
11	pr19		Pizza Hawaina	S/ 26.00	s1	ct1	t2
12	pr2		Pizza Mozzarella	S/ 12.00	s1	ct1	t1
13	pr20		Pizza Vegetariana	S/ 26.00	s1	ct1	t2
14	pr21		Pizza tocino	S/ 26.00	s1	ct1	t2
15	pr22		Pizza Francesa	S/ 40.00	s1	ct1	t2
16	pr23		Pizza Americana	S/ 41.00	s1	ct1	t3
17	pr24		Pizza Mozzarella	S/ 42.00	s1	ct1	t3
18	pr25		Pizza Jamon	S/ 43.00	s1	ct1	t3
19	pr26		Pizza Salame	S/ 44.00	s1	ct1	t3
20	pr27		Pizza Oriental	S/ 45.00	s1	ct1	t3
21	pr28		Pizza Hawaina	S/ 44.00	s1	ct1	t3

Restaurar Filtrar Categoría Filtrar Tamaño Filtrar Stock Filtrar

3.7.6 Facturas: el formulario permite al empleado y administrador consultar ,filtrar, modifica y eliminar facturas y observarlo en la tabla

Factura

Facturas

Código Factura:

DNI Cliente:

Empleado: Bautista

Fecha: 2022-07-29

Total:

Consultar Modificar Eliminar

Seleccione Fecha

Ingrese la factura

num	codigo Factura	DNI Cliente	Código empleado	Fecha	Total
1	1	23423456	13140430	2022-07-19	76.0
2	2	23456435	13140428	2022-07-19	229.0
3	3	23454323	13140436	2022-07-19	72.0
4	4	82734532	13140428	2022-07-23	46.0

Cantidad de Facturas : 4

3.7.7 Facturas: el formulario permite al empleado y administrar registrar una factura mediante la inserción del detalle de la factura, con el DNI del cliente y el DNI del empleado

Registro Factura

REGISTRAR FACTURA

Codigo Producto:	<input type="text"/>	Nombre Producto:	<input type="text"/>	
Cantidad:	<input type="text"/>	Precio:	<input type="text"/>	
Agregar Producto				
codigo Producto	Descripcion	Cantidad	Precio Unitario	Precio Total

Dni Cliente: Código Empleado:
Total:

Fecha:

[Registrar Factura](#)

3.7.8 Facturas: el formulario permite al empleado y administrar consultar ,filtrar los detalles de la factura, mediante el código de la factura.

Detalle Factura

Registro del Detalle de la Facturas

Ingrese la factura

num	codigo detalle	Codigo producto	Descripción	Cantidad	Precio Unitario	Precio Total	Codigo Factura
1	4	pr10	Pizza Delicia	3	16.0	48.0	1
2	5	pr50	Coca Cola	4	7.0	28.0	1
3	6	pr10	Pizza Delicia	3	16.0	48.0	2
4	7	pr50	Coca Cola	4	7.0	28.0	2
5	8	pr30	Pizza locino	3	47.0	141.0	2
6	9	pr52	Coca Cola	1	12.0	12.0	2
7	10	pr10	Pizza Delicia	2	16.0	32.0	3
8	11	pr1	Pizza Americana	2	13.0	26.0	3
9	12	pr50	Coca Cola	2	7.0	14.0	3
10	13	pr10	Pizza Delicia	2	16.0	32.0	4
11	14	pr50	Coca Cola	2	7.0	14.0	4

Cantidad de Detalle de Facturas:

3.7.9 Reporte Diario en este parte del aplicativo se podrá registrar las ventas diarias por parte De cada empleado, almacenando la cantidad vendida y cantidad de dinero total que hubo en el Negocio, para poder así ayudar al administrador a ver que empleado fue el responsable del dia.

Reporte Diario

num	NºReporte	Codigo emp..	Pizza Vend..	Pastas Ven..	Bebidas Ve..	Piqueos Ve..	Platos Crioll..	Fecha	Dinero Inicial	Dinero Final	Dinero Sobre...
1	1	123123	40	10	50	14	5	2022-07-06	S/ 200.00	S/ 500.00	S/ 300.00
2	2	13140429	45	11	55	14	5	2022-05-30	S/ 1200.00	S/ 500.00	S/ 700.00
3	3	13140429	43	12	55	14	5	2022-07-03	S/ 2200.00	S/ 500.00	S/ 1700.00
4	4	13140431	42	13	52	14	5	2022-07-04	S/ 3200.00	S/ 500.00	S/ 2700.00
5	5	123123	36	14	53	14	5	2022-07-02	S/ 1200.00	S/ 500.00	S/ 700.00
6	6	13140429	47	15	35	14	5	2022-07-01	S/ 2200.00	S/ 500.00	S/ 1700.00
7	7	123123	50	16	55	14	5	2022-05-06	S/ 3200.00	S/ 500.00	S/ 2700.00
8	8	13140431	43	14	60	14	5	2022-06-06	S/ 4200.00	S/ 500.00	S/ 3700.00
9	9	13140429	42	13	53	14	5	2022-06-26	S/ 2200.00	S/ 500.00	S/ 1700.00
10	10	13140431	45	13	53	14	5	2022-05-04	S/ 1200.00	S/ 500.00	S/ 700.00
11	11	123123	48	15	56	14	5	2022-07-10	S/ 3200.00	S/ 500.00	S/ 2700.00
12	12	13140431	50	17	57	14	5	2021-12-26	S/ 2200.00	S/ 500.00	S/ 1700.00
13	16	13140432	23	42	21	32	42	2022-07-09	S/ 1000.00	S/ 400.00	S/ 600.00

Cantidad de Reporte Diario : 13

Reporte Diario

Nº de Reporte:

Codigo Empleado: **Bautista**

Pizzas vendidas:

Pastas vendidas:

bebidas vendidas:

Piqueos vendidos:

Platos C. vendidos:

Fecha: 2022-07-09

Dinero Inicial:

Total de Dinero:

Registrar **Eliminar**

Consultar **Modificar**

3.7.10 Reporte Semanal este parte del aplicativo es solo para administrados, aquí el Administrador puede consultar los reportes diarios registrados, poniendo un inicio y un final

Reporte Semanal

Filtro por N° reporte:

Filtro por fecha:



Buscar **Restaurar**

Reporte Semanal

num	NºReporte	Codigo em...	Pizza Vend...	Pastas Ven...	Bebidas Ve...	Piqueos Ve...	Platos Criol...	Fecha	Dinero Inicial	Dinero Final	Dinero Sob...
1	1	123123	40	10	50	14	5	2022-07-06	\$/ 200,00	\$/ 500,00	-\$/ 300,00
2	2	13140429	45	11	55	14	5	2022-06-30	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
3	3	13140429	43	12	55	14	5	2022-07-03	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
4	4	13140431	42	13	52	14	5	2022-07-04	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
5	5	123123	36	14	53	14	5	2022-07-02	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
6	6	13140429	47	15	35	14	5	2022-07-01	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
7	7	123123	50	16	55	14	5	2022-05-06	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
8	8	13140431	43	14	60	14	5	2022-06-06	\$/ 4200,00	\$/ 500,00	-\$/ 3700,00
9	9	13140429	42	13	53	14	5	2022-06-26	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
10	10	13140431	45	13	53	14	5	2022-05-06	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
11	11	123123	48	15	56	14	5	2022-07-10	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
12	12	13140431	50	17	57	14	5	2021-12-26	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
13	16	13140432	23	42	21	32	42	2022-07-09	\$/ 1000,00	\$/ 400,00	-\$/ 600,00

Cantidad de Reporte Diario

3.7.11 Reporte Mensual, En esta parte solo para el administrador podrá filtrar todos los Los reportes seleccionando un mes y años, así será más fácil el control de la gestión del negocio.

Reporte Mensual

Busqueda Por Mes:

Busqueda Por Año:



Restaurar

Reporte Mensual

num	NºReporte	Codigo em...	Pizza Vend...	Pastas Ven...	Bebidas Ve...	Piqueos Ve...	Platos Criol...	Fecha	Dinero Inicial	Dinero Final	Dinero Sob...
1	1	123123	40	10	50	14	5	2022-07-06	\$/ 200,00	\$/ 500,00	-\$/ 300,00
2	2	13140429	45	11	55	14	5	2022-05-30	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
3	3	13140429	43	12	55	14	5	2022-07-03	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
4	4	13140431	42	13	52	14	5	2022-07-04	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
5	5	123123	36	14	53	14	5	2022-07-02	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
6	6	13140429	47	15	35	14	5	2022-07-01	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
7	7	123123	50	16	55	14	5	2022-05-06	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
8	8	13140431	43	14	60	14	5	2022-06-06	\$/ 4200,00	\$/ 500,00	-\$/ 3700,00
9	9	13140429	42	13	53	14	5	2022-06-26	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
10	10	13140431	45	13	53	14	5	2022-05-06	\$/ 1200,00	\$/ 500,00	-\$/ 700,00
11	11	123123	48	15	56	14	5	2022-07-10	\$/ 3200,00	\$/ 500,00	-\$/ 2700,00
12	12	13140431	50	17	57	14	5	2021-12-26	\$/ 2200,00	\$/ 500,00	-\$/ 1700,00
13	16	13140432	23	42	21	32	42	2022-07-09	\$/ 1000,00	\$/ 400,00	-\$/ 600,00

Cantidad de Reporte Diario

3.7.12 Productos, en esta parte solo para el administrador, aquí el administrador podrá agregar Modificar, eliminar productos, cambiarle el precio, el stock disponible para que los empleados Al momento de consultar puedan verificar los productos que hay en el restaurante.

Productos

PRODUCTOS

Codigo Producto:	<input type="text"/>
Nombre Producto:	<input type="text"/>
Precio:	<input type="text"/>
Stock:	<input type="text"/> Disponible
Categoría:	<input type="text"/> Bebidas
Tamaño:	<input type="text"/> 1 LT

Cantidad de Productos : 5

Num	Codigo Producto	Nombre Producto	Precio	Stock	Categoría	Tamaño
1	pr1	Pizza Americana	S/ 12,00	s1	ct1	t1
2	pr2	Pizza Mozzarella	S/ 12,00	s1	ct1	t1
3	pr3	Pizza Jamon	S/ 14,00	s1	ct1	t1
4	pr4	Pizza Salame	S/ 13,00	s1	ct1	t1
5	pr5	Pizza Oriental	S/ 15,00	s1	ct1	t1

3.7.13 Empleados, En este apartado solo para administrador, el administrador podrá agregar Nuevos empleados, monitorizar sus asistencias, modificar sus sueldos y así poder también Controlar mejor a sus empleados

Empleados

Empleados

DNI Empleado:	<input type="text"/>
Nombre:	<input type="text"/>
Apellido:	<input type="text"/>
Direccion:	<input type="text"/>
Telefono:	<input type="text"/>
Sueldo:	<input type="text"/>
Edad:	<input type="text"/>
Días Trabajado	<input type="text"/> 0

Cantidad de Empleados : 5

num	codigo empleado	Nombre	Apellido	Direccion	Telefono	Sueldo	Edad	Días Trabajados
1	123123	khalil alexander	Huaman Acevedo	Av brasil 123	234567892	S/ 1000,00	23	23
2	12345678	Leonel Andres	Messi Villanueva	Av brasil 123	123456789	S/ 1200,00	23	13
3	23412367	Claudia	Aguirre	Av angeles 123	234542657	S/ 1200,00	25	8
4	73140427	roberto Javier	Montalvar messi	Av Brasil 1234	983434343	S/ 1250,00	26	6
5	73149247	Brayan	Medrano	santo chocano	983748347	S/ 950,00	25	25

4. Conclusiones

- Finalmente, se puede concluir que es fundamental obtener un buen informe de investigación sobre el rubro y las características de la empresa. Puesto que, esto ayudaría a la creación de un sistema de venta adecuado a los requerimientos y necesidades del mercado.
- La empresa Paul Pizza cuenta con un sistema de administración de venta más eficiente que ha logrado optimizar el tiempo de sus sistemas administrativos.
- La empresa Paul Pizza cuenta con un sistema de ventas orientado a la administración del negocio manteniendo un control entre ellas. Asimismo, mantiene un orden en la búsqueda y visualización de sus facturas.
- La empresa Paul Pizzas cuenta con una base de datos que maneja y controla la información de nuestra compañía. Tales como, datos de los clientes, trabajadores, productos, entre otros.
- El sistema mantiene una organización y almacenamiento de los datos pertenecientes a las facturas. De esta manera, se mejora servicio del trabajador hacia el cliente

Bibliografía:

- Jaime Andres (2012) *Plan de Mercadeo para la empresa Mr. Pizza*. pp. 20-23 Recuperado de:
<https://red.uao.edu.co/bitstream/handle/10614/2991/TAD00929.pdf;jsessionid=6D43D9C11D52B67B9E713C734286F8A8?sequence=1>
- Sandra Monteluisa (2016) *Plan de Negocio Pizzalegro*. pp. 15-20 Recuperado de:
<https://www.ujcm.edu.pe/sites/default/files/field/archivos/EP/Comercial/12.1.pdf>
- Lopez G. y Rodriguez J. (2017) *PLAN DE NEGOCIO DE UNA PIZZERÍA UTILIZANDO INGREDIENTES NO TRADICIONALES EN EL SECTOR CÉNTRICO DE GUAYAQUIL*. pp. 25-26 Recuperado de:
<http://repositorio.ug.edu.ec/bitstream/redug/47469/1/TESIS%20PIZZERIA%20PIZZA%20FINAL.pdf>
- Campos E. y Mora D. (2019) *SOFTWARE DE GESTIÓN DE PRODUCTOS EN EL RESTAURANTE ALEJHO DE MELGAR, TOLIMA*. pp. 22-24 Recuperado de:
https://repository.uniminuto.edu/bitstream/10656/7574/1/T.TI_MoraGomezDanyJhoani_2019.pdf

- Carlos Burgos (2015) *DESARROLLO DE UN SISTEMA WEB PARA LA GESTIÓN DE PEDIDOS EN UN RESTAURANTE. APLICACIÓN A UN CASO DE ESTUDIO.* p.20

Recuperado de : <https://bibdigital.epn.edu.ec/bitstream/15000/10337/3/CD-6157.pdf>