

LICENCIATURA EN INFORMÁTICA

INGENIERÍA DE SOFTWARE

FERNANDO LÓPEZ SALVADOR
NANCY OBED MARTÍNEZ MIGUEL
LUIS FERNANDO VÁZQUEZ FABIÁN

M . T . C . A . R OLANDO PEDRO GABRIEL

Índice de contenido

1 Personal Involucrado	7
2 Introducción	9
3 Especificación de Requisitos	9
3.1 Próposito del proyecto	9
3.2 Viabilidad	9
3.3 Factibilidad	9
3.4 Ámbito del proyecto	10
3.4.1 Validación y Registro de usuario	10
3.4.2 Punto de venta	10
3.4.3 Ticket	10
3.4.4 Agregar registro de los productos	10
3.4.5 Modificación del producto	11
3.4.6 Eliminar productos	11
3.5 Requerimientos Funcionales	11
3.6 Requerimientos No Funcionales	13
3.7 Requerimientos de interfaz	13
3.8 Mapa de Navegación	19
4 Análisis	20
4.1 Diagrama de Objetos	20
4.2 Diagrama de Componentes	21
4.3 Diagrama de Despliegue	22
4.4 Diagramas de Secuencia	23
4.4.1 Diagrama de secuencia de Login	23
4.4.2 Diagrama de Secuencia inventario	24
4.4.3 Diagrama de Secuencia del usuario	26
4.4.4 Diagrama de Secuencia del Proveedores	28
4.4.5 Diagrama de Secuencia del Clientes	29
4.4.6 Diagrama de Secuencia de la información de la tienda	30
4.4.7 Diagrama de Secuencia de la información de la consulta de ventas	31
4.5 Documentación	31
4.6 Diagrama de Estados	42
4.6.1 Diagrama de Estados Cliente	42
4.6.2 Diagrama de Estados Proveedor	44
4.6.3 Diagrama de Estados Productos	46
4.6.4 Diagrama de Estados Actualizar Proveedores	48
4.6.5 Diagrama de Estados para realizar una Venta	50
4.6.6 Diagrama de Estados para el Logeo	52
4.7 Caso de Uso	54
4.7.1 Caso de Uso de un login	54
4.7.2 Caso de Uso del Inventario	56
4.7.3 Caso de Uso del Usuario	58
4.7.4 Caso de Uso de Proveedores	60
4.8 Diagramas de colaboración	62
4.8.1 Diagramas de colaboración del Login	62
4.8.2 Diagramas de colaboración de la venta	62

4.8.3	Diagramas de colaboración de la Factura	63
4.8.4	Diagramas de colaboración del producto	64
4.8.5	Diagramas de colaboración del proveedor	65
5	Diseño	71
5.1	Introducción	71
5.2	Descripción del sistema	71
5.3	Arquitectura del sistema	72
5.3.1	Arquitectura Física	72
5.3.2	Arquitectura Lógica	76
5.4	Diseño de Datos	83
5.4.1	Modelo Conceptual	83
5.4.2	Diagrama entidad de la base de datos	84
5.4.3	Script de creación de la base de datos	85
5.4.4	Diccionario de datos	89
6	Código del proyecto	91
6.1	clase conexion	91
6.2	clase Clientes	91
6.3	clase Productos	94
6.4	clase Proveedores	98
6.5	Login	100
6.6	Principal	101
6.7	función para listar los clientes	102
6.8	función para listar los proveedores	102
6.9	función para listar los usuarios	102
6.10	función para listar los productos	103
6.11	función para listar limpiar las tablas del programa	103
6.12	función para registrar una venta	103
6.13	función para registrar una venta	104
6.14	función para calcular el precio	104
6.15	Agregar un Nuevo Cliente	104
6.16	Eliminar un Cliente	105
7	Conclusiones	106
7.1	Conclusión	106
7.2	Definiciones y Abreviaturas	106
7.2.1	Definiciones	106
7.2.2	Abreviaturas	106
7.3	Referencias	106

Índice de Tablas

1	Requerimientos Funcionales 1	11
2	Requerimientos Funcionales 2	12
3	Requerimientos Funcionales 4	12
4	Requerimientos Funcionales 5	12
5	Requerimientos no Funcionales 1	13
6	Requerimientos no Funcionales 2	13
7	Requerimientos no Funcionales 3	13
8	Requerimientos no Funcionales 3	13
9	Equipos Utilizados HP Pavilion	72
10	Equipos Utilizados HP 15-bs	72
11	Equipos Utilizados	73
12	Samsung note 5	74
13	Motorola	74
14	Xiaomi	75

Indice de Figuras

1	Fernando	7
2	Luis	7
3	Nancy	7
4	Prof.Rolando	8
5	Prof.Everardo	8
6	Prof.Lirio	8
7	Mapa de Navegación	19
8	Diagrama de Objetos	20
9	Diagrama de Objetos	21
10	Diagrama de componentes	21
11	Diagrama de componentes	22
12	Diagrama de despliegue	22
13	login	23
14	Diagrama de Usuarios	27
15	Diagrama de Estados cliente	42
16	Diagrama de Estados cliente	43
17	Diagrama de Estados proveedor	44
18	Diagrama de Estados proveedor	45
19	Diagrama de Estados Productos	46
20	Diagrama de Estados Productos	47
21	Diagrama de Estados Actualizar Proveedores	48
22	Diagrama de Estados Actualizar Proveedores	49
23	Diagrama de Estados Realizar Venta	50
24	Diagrama de Estados Realizar Venta	51
25	Diagrama de Estados del Logeo.	52
26	Diagrama de Estados del Logeo.	53
27	Caso de uso del Logeo.	54
28	Diagrama de caso de uso del Logeo.	55
29	Caso de uso del inventario.	56
30	Diagrama de caso de uso del inventario.	57
31	Caso de uso de Usuarios.	58
32	Diagrama de caso de uso de Usuarios.	59
33	Caso de uso de Proveedores.	60
34	Diagrama de caso de uso de Proveedores.	61
35	Diagrama de colaboración Login.	62
36	Diagrama de colaboración de Venta.	62
37	Diagrama de colaboración de Factura.	63
38	Diagrama de colaboración del producto.	64
39	Diagrama de colaboración del proveedor.	65
40	Pavilon	72
41	Custom	73
42	Galaxy note 5	74
43	Motorola	74
44	xiaomi	75
45	Apache netbeans	76
46	Balsamiq	77
47	PgAdmin	78
48	Modelo conceptual del sistema	83

49	Base de datos	84
50	Diccionario de la tabla clientes	89
51	Diccionario de la tabla Vendedor	89
52	Diccionario de la tabla ventas	90
53	Diccionario de la tabla detalle	90

1 Personal Involucrado

Fernando López Salvador(Desarrollador) estudiante de la Universidad de la Sierra Sur, domina los lenguajes de programación como lo son C Y java, también domina los gestores de base de datos MySQL Y Postgres.



Figure 1: Fernando

Luis Fernando Vásquez Fabián(Auxiliar) estudiante de la Universidad de la Sierra Sur, quien tiene una gran habilidad para diseñar y mantener unido al equipo.



Figure 2: Luis

Nancy Obed Martínez Miguel(Master) estudiante de la Universidad de la Sierra Sur, quien tiene una creatividad muy interesante .



Figure 3: Nancy

M.T.C.A. Rolando Pedro Gabriel(Usuario Final) es el encargado de revisar y dar su punto de vista del documento aquí realizado. Está altamente calificado para realizarlo ya que es profesor investigador perteneciente al área de informática.



Figure 4: Prof.Rolando

M.T.E. Everardo de Jesús Pacheco Antonio(Usuario Final) como es el usuario final dará su punto de vista de acuerdo a la parte de codificación ya que es el maestro encargado de la materia paradigmas de programación y el dará los resultado de este después de probarlo.



Figure 5: Prof.Everardo

M.C.C Lirio Ruiz Guerra (Usuario final) La profesora como usuario final dara la aprobación final sobre el tema de seguridad sobre la base de datos del sistema de ventas desarrollado durante el semestre.



Figure 6: Prof.Lirio

2 Introducción

En la actualidad las herramientas tecnologicas se han convertido en uno de los factores más importantes en cuanto a los usos institucionales de las empresas, estas son utilizados diariamente para el manejo de Información y su proyección social. Así que dentro del amplio ámbito de la informática se busca que día a día se puedan lograr los avances necesarios para poder facilitar la vida de las personas, ya que conforme pasa tiempo surgen las necesidades de cambiar algunas cosas. Por lo que en este proyecto se enfoca en implementar un software de ámbito comercial, este software será diseñado para el control de un sistema de ventas de una tienda de abarrotes, buscando una optimización de los datos que esta maneja.

Por lo que estas aplicaciones mejoran el control administrativo llevando un seguimiento preciso de todas las transacciones que se realizan dentro de la tienda en un tiempo real, para generar reportes detallados de las cuentas que van a permitir a los administradores la cantidad correcta de productos, en el momento adecuado lo cual va a permitir al negocio o empresa mejorar servicio al cliente reduciendo cualquier anomalía. Los sistemas de ventas tienen la ventaja de ser personalizados para cumplir con las necesidades específicas de un negocio, por ejemplo, en la tienda las organizaciones de venta al menudeo pueden ayudar a localizando precios de venta de los productos y costos actuales de los productos para poder trabajar más rápidamente. En años recientes el mercado ha incrementado y se han ido generando diversos negocios o empresas en todas partes y eso ha llevado a las mismas ofrecer una mejor atención por lo que es muy importante contar con un sistema de ventas rápido y sencillo.

3 Especificación de Requisitos

3.1 Próposito del proyecto

El sistema de ventas está orientada a la gestión de empresas comerciales o pequeños negocios, es un software práctico que va a permitir un mejor control del negocio. Se trata de un software para la gestión de clientes y productos. Donde se puede ingresar como administrador y realizar los que se necesite así sea agregar, eliminar o modificar un producto o de ser un vendedor buscar los productos que el cliente desee y consultar precios, cantidades en existencia y realizar su venta. Así mismo generara un informe de la venta y su respectivo ticket. El sistema se hace cargo de registrar las ventas del producto o servicio de la organización, evalúa sus características o necesidades del negocio, llevando un detalle del registro de todas las transacciones.

3.2 Viabilidad

Este proyecto es viable debido a que se cuenta con las posibilidades para poder trasladarse a dichos establecimientos al que se realizará el proyecto, todo esto con la finalidad de platicar con el cliente y así se pueda proporcionar cada una de las necesidades o requerimientos que este quiera que se implemente al software.

3.3 Factibilidad

El proyecto es factible debido a que en la UNIVERSIDAD DE LA SIERRA SU sur contamos con un centro de tecnologías de la informática y comunicación donde contamos con las herramientas necesarias ya que se cuenta con computadoras internet, y la asesoría de los

docentes quienes están altamente calificados para ayudarnos a desarrollar dicho software. A si mismo también durante nuestra carrera hasta el día de hoy hemos aprendido conocimientos en distintos lenguajes de programación como C, Java, Css, Html y lenguaje ensamblador los cuales podemos aplicar en el proyecto a resolver.

3.4 Ámbito del proyecto

En la actualidad debido a la falta de conocimiento de muchas personas que administran establecimientos o pequeñas empresas dedicadas a las ventas no tienen en orden la administración de las mismas, como podrían ser las ventas, registros o actualizaciones de productos. Por lo que muchas veces suelen tener perdidas de ganancias o desperdicio de la misma mercancía.

Pero también se sabe que en un establecimiento deben de tener de manera ordenada los roles que se desempeñan dentro del lugar, es decir, ver que cosas puede hacer el administrador del negocio y que puede realizar el vendedor. Así que se desea implementar un sistema de control de ventas en una tienda de abarrotes y así dar solución a la problemática ya antes mencionada. Evitando así perdidas monetarias y descontrol de su inventario, este sistema de venta estará diseñado de manera que el usuario pueda interactuar fácilmente y obtenga buenos resultados al utilizarlo.

Por lo que se describirá la estructura ya pensada de como se va ir desarrollando el sistema, el cual se va a distribuir de la siguiente manera:

3.4.1 Validación y Registro de usuario

En esta parte se implementará la validación de usuario con la finalidad de mantener seguro los datos del usuario, evitando así que personas no autorizadas puedan acceder al sistema haciendo un mal uso del sistema de ventas. También dependerá del rol que ingrese al sistema, así se le irán mostrando sus respectivas ventanas por ejemplo si entra del lado del vendedor:

3.4.2 Punto de venta

Donde el vendedor va a generar una venta es decir el cliente comienza a tomar los productos que serán mostrados desde una tabla y así puede ir comprobando si se encuentra en existencia dicho producto, el precio y entre otras cosas.

3.4.3 Ticket

En esta parte se piensa que después de que el vendedor genere su venta le muestre como resultado un mensaje donde le detalle el nombre del producto, el precio y de igual manera le muestre la suma total de su compra. En caso contrario y se iniciará sesión como administradores pues lo primero que le mostraría es una ventana de ventas en la que a un costado izquierdo encontrara todos los botones activados para el dicho desplazamiento entre ventanas.

3.4.4 Agregar registro de los productos

El usuario podrá realizar sus registros diarios de las ventas y adquisición de Productos nuevos, además podrá registrar las características de cada producto como nombre, id, existencias, precio entre otras.

3.4.5 Modificación del producto

En cualquier momento el usuario podrá acceder a su inventario para poder hacer las modificaciones necesarias siempre y cuando se requiera. En este caso se le activará un pequeño formulario muy similar al de agregar y así podrá modificar lo que necesite para posteriormente dar clic en el botón actualizar.

3.4.6 Eliminar productos

El usuario podrá en cualquier momento eliminar los productos pasados que ya no necesite, esto con la finalidad de que no requiera de más espacio de memoria evitando de esta manera la necesidad de hacer más gastos innecesarios. Este apartado estará diseñado de manera simple para que el usuario pueda navegar sin problemas, además de que contará con aviso de alerta cuando el usuario elimine un registro, de esta forma el usuario estará seguro si desea eliminar tal registro, evitando así el riesgo de eliminar registros importantes.

Este sistema tendrá el objetivo de ser una gran herramienta que pueda ayudar al usuario llevar una mejor organización de sus operaciones en su negocio y también está claro que esta también debe cumplir las expectativas de cliente en cuanto al diseño del software, ya que el diseño se realizará de forma que pueda ser más agradable y entendible para el usuario, este sistema contará con diferentes herramientas, como el registro de ganancias en donde podrá ver si su negocio está resultando así como también contara con página en donde puedan consultar los modelos productos que ha vendido y así el usuario podrá ver que compras son más recomendables les hará a sus proveedores.

3.5 Requerimientos Funcionales

Código de requerimiento	RF01
Nombre	Envío y recepción de archivos
Propósito	Permitir el acceso de información a la base de datos así como la facilidad para enviar información
Descripción	El usuario realiza la encuesta para la información, a continuación acepta los términos para así poder enviar su información a la base de datos
Entrada	Formulario de perfil del cliente
Salida	Mensaje de acción satisfactoria
Prioridad	Alta

Table 1: Requerimientos Funcionales 1

Código de requerimiento	RF03
Nombre	Operaciones Crud
propósito	Crear, Actualizar, borrar usuarios
descripción	Una vez ubicados en la página inicial del software el usuario debe capturar los datos de cliente
Entrada	Formulario de entrada de llenado de información del cliente
Salida	Mensaje que indique que el registro fue exitoso
Prioridad	Alta

Table 2: Requerimientos Funcionales 2

Código de requerimiento	RF04
Nombre	Facturación de pedidos de venta
propósito	Crear la factura de la venta
descripción	La facturación de pedidos de venta se realizará por medio de una pantalla de pedidos pendientes de facturación.
Entrada	Formulario de entrada de llenado de información de la compra
Salida	Mensaje que indique que el proceso fue exitoso
Prioridad	Media

Table 3: Requerimientos Funcionales 4

Código de requerimiento	RF05
Nombre	Maestro de clientes
propósito	Mantener un mejor control sobre la información de los clientes
descripción	El sistema manejará un registro maestro de clientes. Sólo los usuarios autorizados podrán ingresar nuevos clientes, modificar los datos o eliminarlos.
Entrada	Formulario de entrada de llenado de información del cliente
Salida	Mensaje que indique que el proceso fue exitoso
Prioridad	Alta

Table 4: Requerimientos Funcionales 5

3.6 Requerimientos No Funcionales

Código de requerimiento	RNF01
Nombre	Actualización del software
propósito	Realizar mantenimiento y actualizaciones del software
descripción	Ir a la empresa a darle mantenimiento al software y actualizar bibliotecas

Table 5: Requerimientos no Funcionales 1

Código de requerimiento	NRF02
Nombre	Mantenimiento
propósito	Un software con un buen funcionamiento
descripción	Las acciones que se harán para mejorar el software (cambio de contraseña, alguna mejora, limpieza de registro de sistema).

Table 6: Requerimientos no Funcionales 2

Código de requerimiento	NRF03
Nombre	Aplicación para celular
propósito	El software no contara con aplicación para celulares
Prioridad	baja

Table 7: Requerimientos no Funcionales 3

Código de requerimiento	NRF04
Nombre	Ordenes de entrega
propósito	El sistema no contara con opción para envio a domicilio
Prioridad	baja

Table 8: Requerimientos no Funcionales 3

3.7 Requerimientos de interfaz

Validación de usuario

En la siguiente ventana lo que nos muestra es el **Login** o el inicio de sesión en el cual es necesario ingresar los datos correctos para así poder ingresar al sistema, ya sea como Vendedor o como administrador para realizar las acciones necesarias, en caso de ingresar los datos de manera errónea se muestra un mensaje diciendo que el usuario o la contraseña son incorrectos.

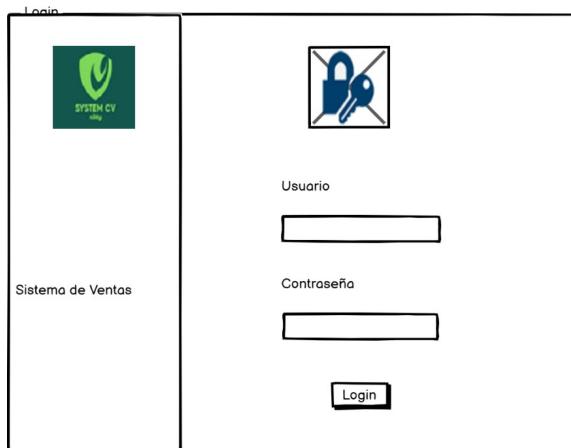


Figura 1: Login.

Punto de venta

Una vez validado el inicio de sesión se muestra la siguiente ventana donde se realizaran las ventas las opciones que encontramos en esta ventana son **Nueva Venta, Productos, Clientes, Ventas, Proveedores, Información, Usuarios, Salir**. Para ejecutar una venta se necesita ingresar el código en la opción de código a continuación se muestra la descripción del producto, posteriormente se ingresa la cantidad de producto a comprar y se muestra el precio en caso de no contar con el producto se muestra un mensaje de Stock no disponible, la X es para eliminar un producto que no se desea comprar, en la parte inferior encontramos con el cliente en el cual ingresamos el código del cliente el cual desea hacer la venta una vez ingresado el código se muestra el código de cliente y su respectivo nombre y para finalizar la venta se presiona en finalizar o en el botón de la impresora lo que significa realizar la venta, finalizada la venta se muestra el ticket y para salir se presiona la opción salir que se encuentra al final en la izquierda de la ventana para si regresar al **Login** en caso de querer ingresar un nuevo usuario.

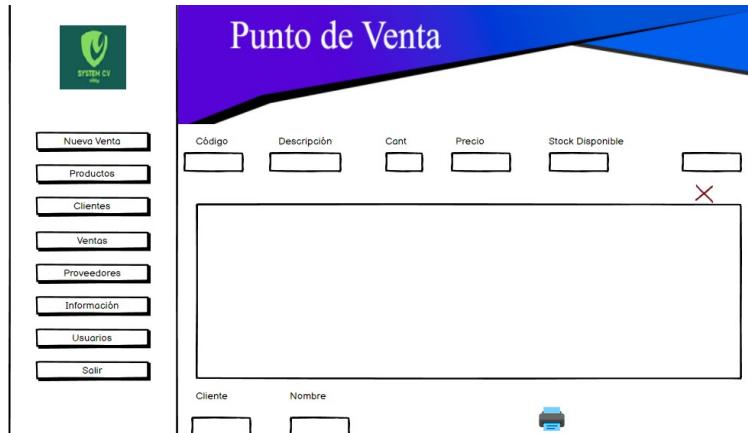


Figura 2: Punto de Venta.

Ventana de Clientes

En caso de que el usuario ingrese como **Administrador** la ventana que se muestra es la de clientes con la opción de **agregar modificar o eliminar clientes**, en caso de ingresar como **Vendedor** solo estará visible la tabla desactivando así la opción de modificar agregar o eliminar, que contiene a los clientes para así poder realizar consultas.

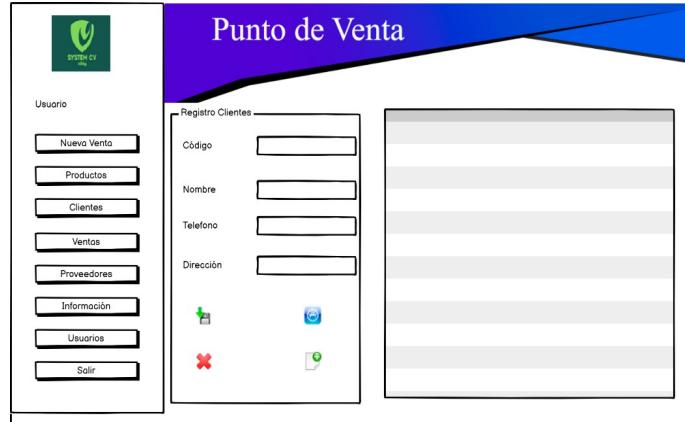


Figura 3: Clientes.

Ventana de Inventario

En la opción de inventario nuevamente si ingresamos como **Administrador** estarán habilitadas las opciones para agregar, modificar o eliminar productos, en caso de ingresar como **Vendedor** solo está habilitado la tabla que contiene la información de los productos para facilitar las consultas.

Figura 4: Inventario.

Ventana de Proveedores

También contamos con la ventana de Proveedores en la cual almacenamos a todos los proveedores a los cuales compramos mercancía en la cual guardamos el **nombre**, **dirección**, **teléfono** y en la parte derecha contamos con una tabla la cual nos muestra el total de proveedores con los cuales contamos actualmente teniendo así un mejor control de nuestros proveedores en esta solamente tendrá el acceso el **Administrador**.

Figura 5: Proveedores.

Ventana de Ventas

En esta opción nos permitirá mostrar o consultar las ventas totales realizadas, en dicha tabla nos muestra el **Id** de la venta el nombre del cliente, el nombre del vendedor y el total a pagar, en esta ventana no están habilitadas las modificaciones simplemente es para poder hacer las consultas de las ventas realizadas previamente donde se mostraran las ventas así como la información relevante de estas.

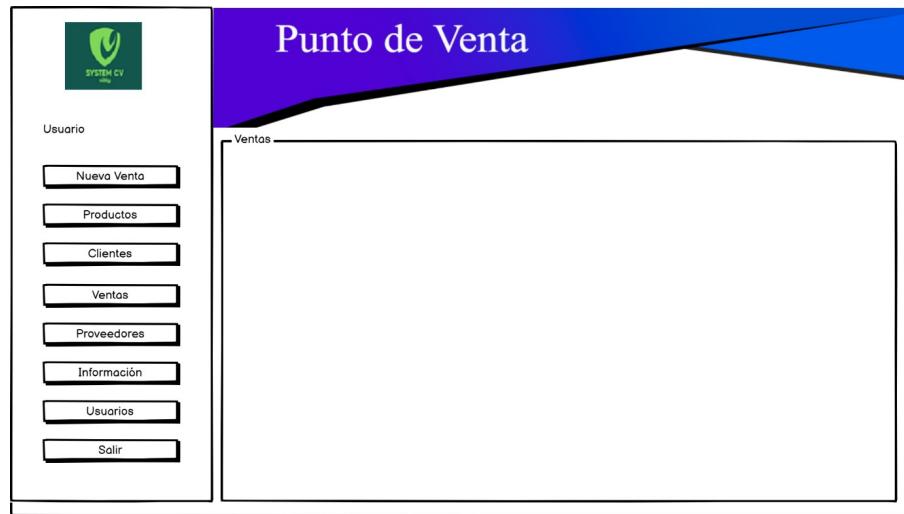


Figura 6: Ventas.

Ventana de Usuarios

En la opción de Usuarios nos permite ingresar nuevos usuarios al sistema dándole cierto privilegio en caso de ser Administrador y en caso de ser vendedor simplemente los privilegios básicos, a dicha ventana solo tendrá acceso el Administrador.

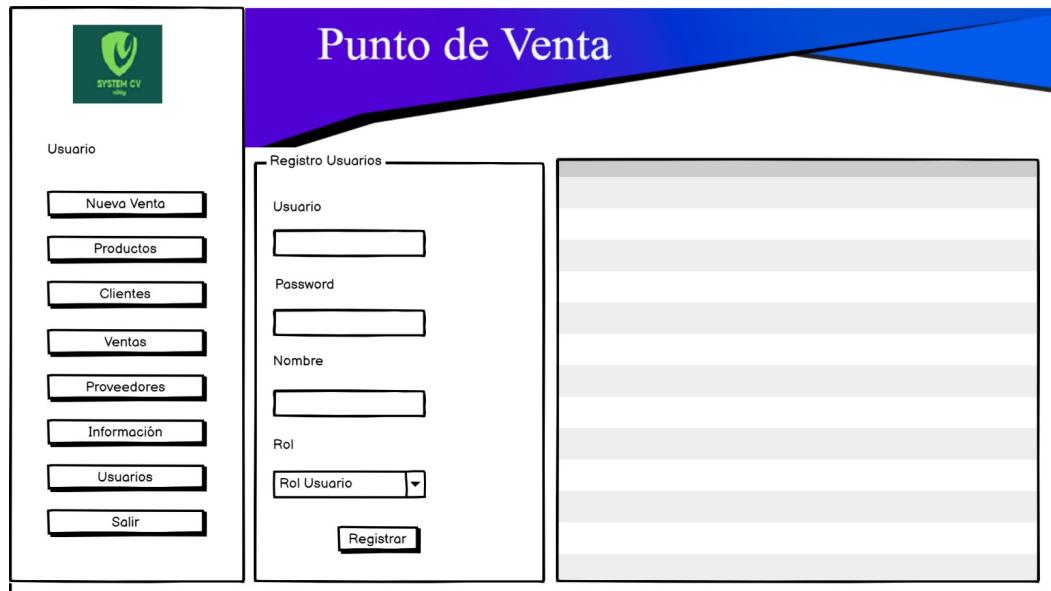


Figura 7: Usuarios.

Ventana de Información

En la opción de Información nos permite acceder a los datos de la tienda contando con la información de esta misma como el **nombre** la **dirección** el **teléfono** o también un **mensaje** que deseé mostrar la tienda al momento de realizar una compra, en caso de ingresar como administrador este nos permitirá modificar dicha información, si ingresa como vendedor no tendrá permitido modificar los datos en esta ventada, pero se mostraran de igual manera al terminar una venta en el ticket respectivo de cada compra.

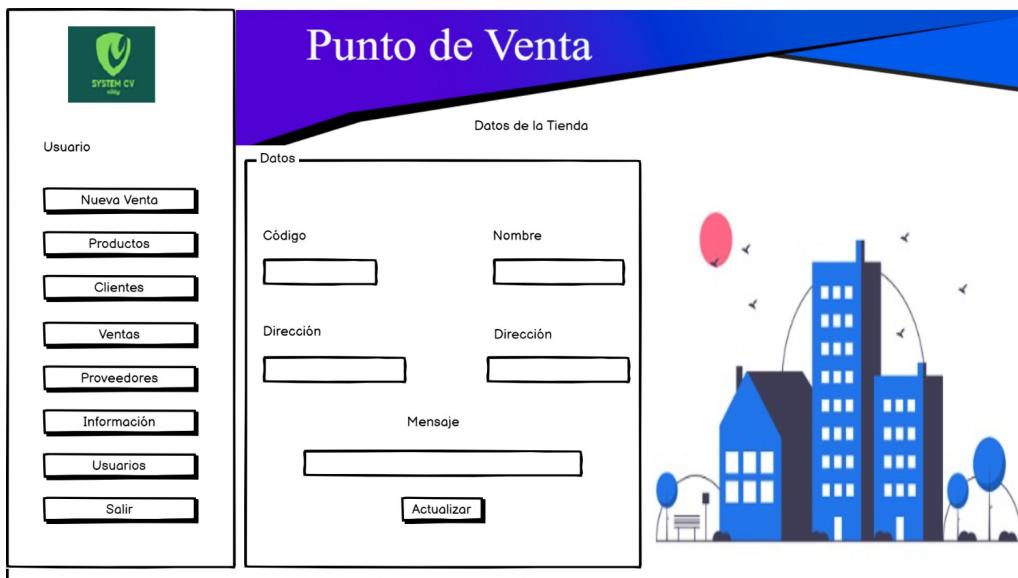


Figura 8: Información de la tienda.

3.8 Mapa de Navegación

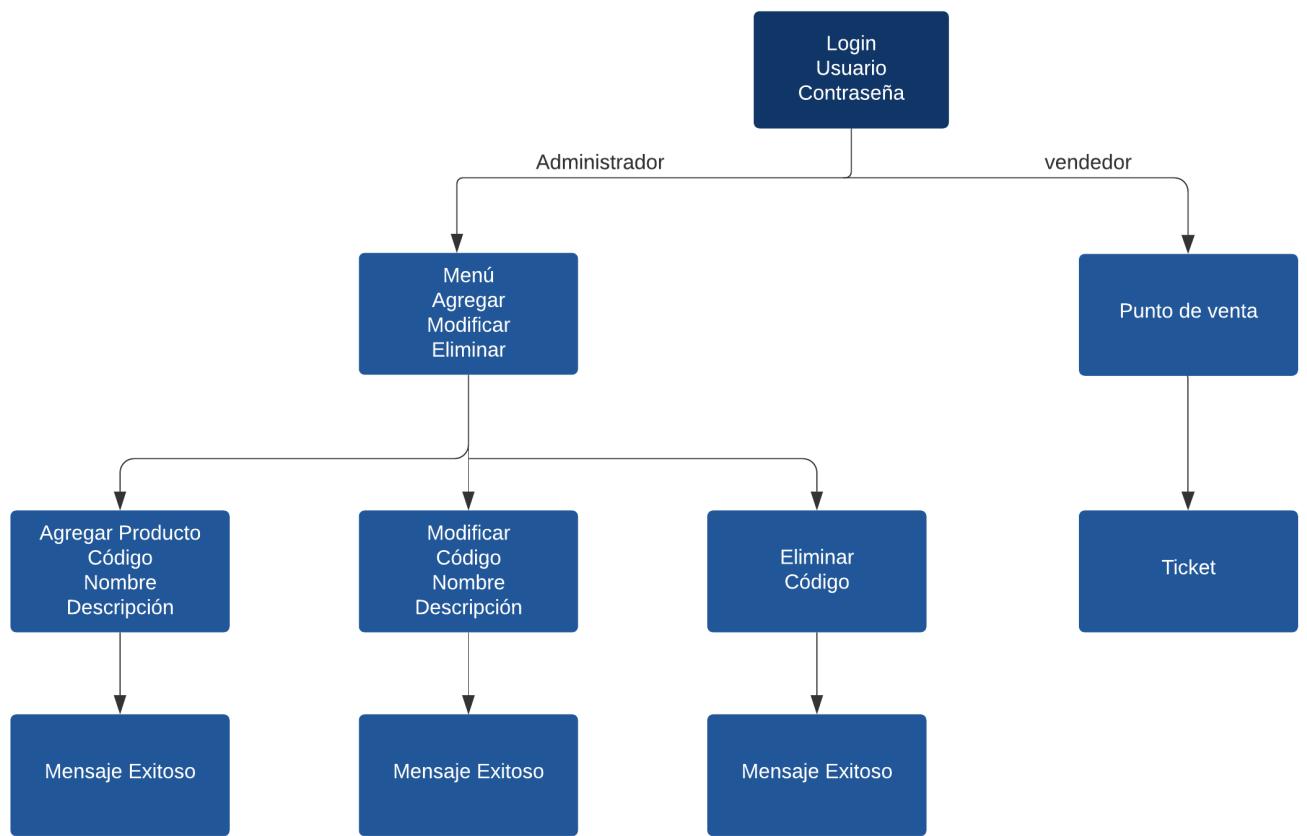


Figure 7: Mapa de Navegación

4 Análisis

4.1 Diagrama de Objetos

En la siguiente imagen se muestra el diagrama de Objetos

```
@startuml
object Usuario
Usuario : NombreUsuario:
Usuario : Contraseña:
object Vendedor
Vendedor : Nombre: Luis
Vendedor : Contraseña:000
Vendedor : Rol: Vendendor
object Administrador
Administrador : NombreUsuario
Administrador : Contraseña:111
Administrador : Rol:SuperUsuario

object Venta
Venta : ID:00001
Venta : Descripcion:S/N
Venta : Cantidad:2
Venta : PrecioU.: $20
Venta : Total:$40

object Producto
Producto : NombreProducto:Sabritas
Producto : Cantidad:2

object Proveedores
Proveedores : NID:000000123
Proveedores : Nombre:Fernando
Proveedores : Telefono:95122834
Proveedores : Direcciones:calle palma

Venta--|>Producto
Usuario--|> Vendedor
Usuario --|> Administrador
@enduml
```

Figure 8: Diagrama de Objetos

En la siguiente imagen se muestra el diagrama de Objetos

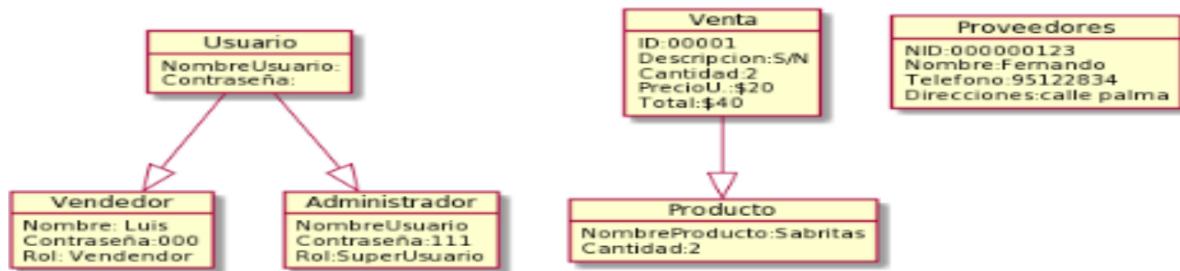


Figure 9: Diagrama de Objetos

4.2 Diagrama de Componentes

En la siguiente imagen se muestra el diagrama de Componentes

```
@startuml
package Diseño<<Interface>> {
    component Principal
    component Ventas
    component Inventario
    component Clientes
    component Información
    component Principal
}
package Operaciones <<component>> {
    component altaProductos
    component bajaProductos
    component altaUsuarios
    component bajaUsuarios
    component altaClientes
    component bajaclientes
    component altaProveedores
    component bacaproveedores
}
package Lógico{
    component GestiónProductos
    component GestiónLogin
    entity ingreso
    component GestiónVentas
}

database postgres #pink{
    component SistemaNikky
}
GestiónLogin-->Principal
GestiónLogin-->SistemaNikky
Operaciones-->SistemaNikky
@enduml
```

Figure 10: Diagrama de componentes

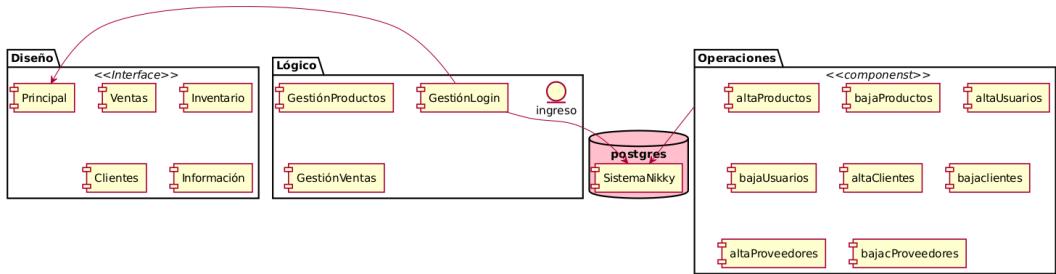


Figure 11: Diagrama de componentes

4.3 Diagrama de Despliegue

En la siguiente imagen se muestra el diagrama de Despliegue del programa.

```
@startuml
node Iniciodesesión as i
node Principal as p
node Ventas as v
node Clientes as c
node Proveedores as pv
node Usuarios as u
node Producto as pd
node JDBC as j
node BaseDeDatos as bd
i->p
p-->v
p-->c
p-->pd
p-->pv
p-->u
v-->j
c-->j
pv-->j
u-->j
pd-->j
j-->bd
@enduml
```

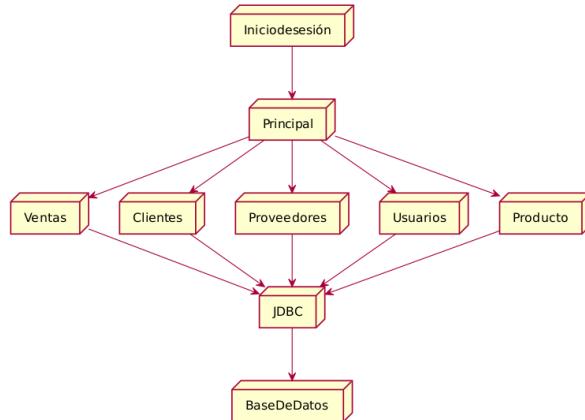


Figure 12: Diagrama de despliegue

4.4 Diagramas de Secuencia

4.4.1 Diagrama de secuencia de Login

En el siguiente diagrama se muestra la secuencia del inicio o el Logeo del programa.

Diagramas de Secuencia

1.-Login

@startuml

note right: nancy

actor Administrador

Activate login

Administrador -> login: 1.-Ingresa sus datos

database baseDeDatos #pink

login -> login: 2.-Ingresa su Usuario

login -> login: 3.-Ingresa su Contraseña

login -> baseDeDatos:4.- Válidar los datos

baseDeDatos --> Administrador: 5.- El administrador entra al sistema

@enduml

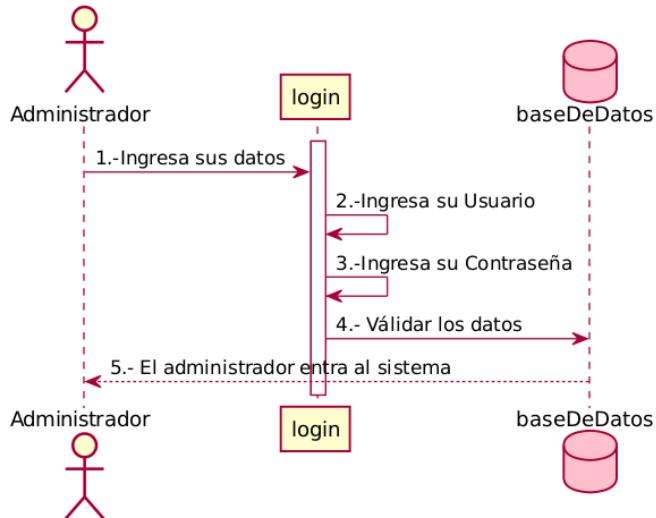


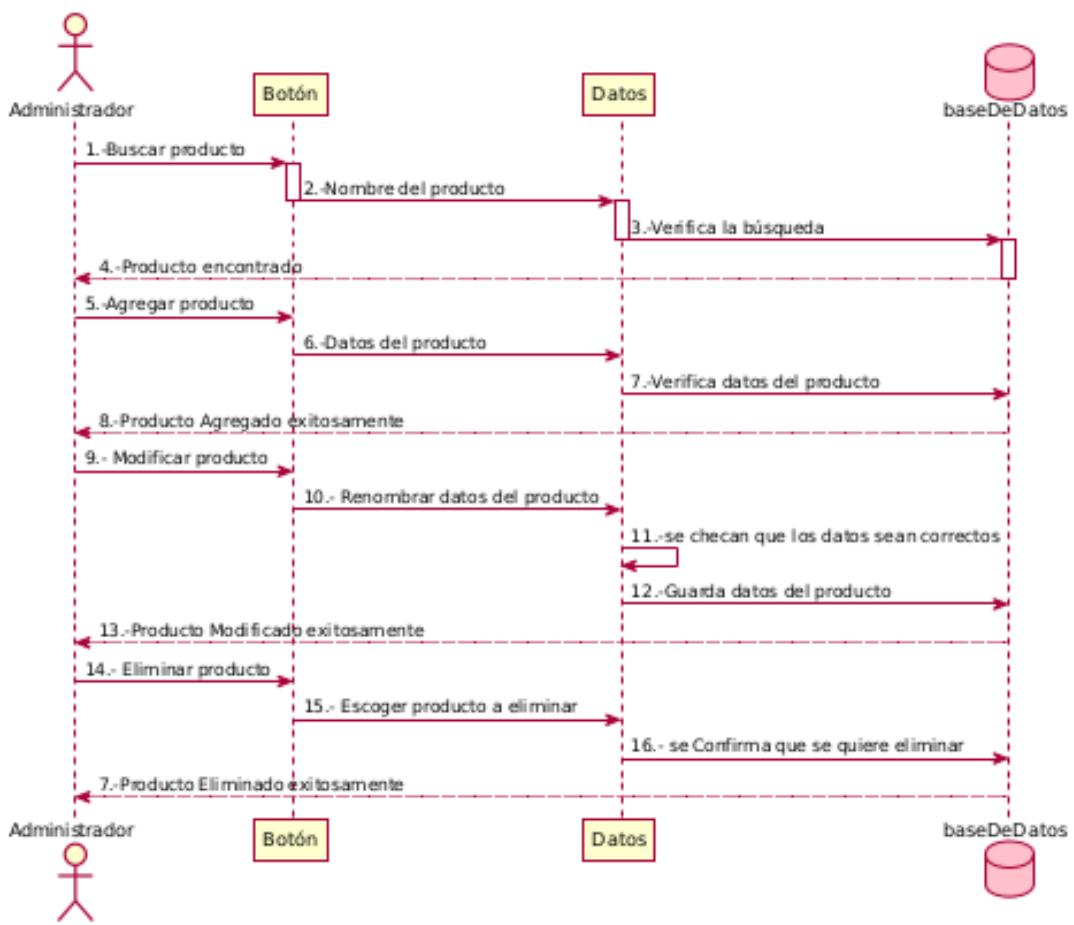
Figura 1: Diagrama de secuencia de login

Figure 13: login

4.4.2 Diagrama de Secuencia inventario

En la siguiente imagen se muestra el diagrama de secuencia del inventario.

```
@startuml
actor Administrador
Administrador->Botón:1.-Buscar producto
Activate Botón
Botón->Datos:2.-Nombre del producto
deactivate Botón
database baseDeDatos #pink
Activate Datos
Datos->baseDeDatos :3.-Verifica la búsqueda
deactivate Datos
Activate baseDeDatos
Administrador--baseDeDatos:4.-Producto encontrado
deactivate baseDeDatos
Administrador->Botón:5.-Aregar producto
Botón->Datos:6.-Datos del producto
Datos->baseDeDatos :7.-Verifica datos del producto
Administrador--baseDeDatos:8.-Producto Agregado exitosamente
Administrador->Botón:9.- Modificar producto
Botón->Datos:10.- Renombrar datos del producto
Datos->Datos:11.-se checan que los datos sean correctos
Datos->baseDeDatos :12.-Guarda datos del producto
Administrador--baseDeDatos:13.-Producto Modificado exitosamente
Administrador->Botón:14.- Eliminar producto
Botón->Datos:15.- Escoger producto a eliminar
Datos->baseDeDatos:16.- se Confirma que se quiere eliminar
Administrador--baseDeDatos:7.-Producto Eliminado exitosamente
@enduml
```



4.4.3 Diagrama de Secuencia del usuario

En la siguiente imagen se muestra el diagrama de secuencias del usuario.

3.-Registrar Usuario

```
@startuml  
actor Usuarios  
Activate Datos  
database baseDeDatos  
Usuarios->Datos: 1.-Nuevo Usuario  
Datos->Datos:2.-Nombre Usuario  
Datos->Datos:3.-Nombre real  
Datos->Datos:4.-Contrasena  
Datos->Datos:5.- Rol  
Datos->baseDeDatos:6.-verificar datos  
activate baseDeDatos  
baseDeDatos-->Usuarios:7.- Se registro su usuario correctamente  
deactivate baseDeDatos  
@enduml
```

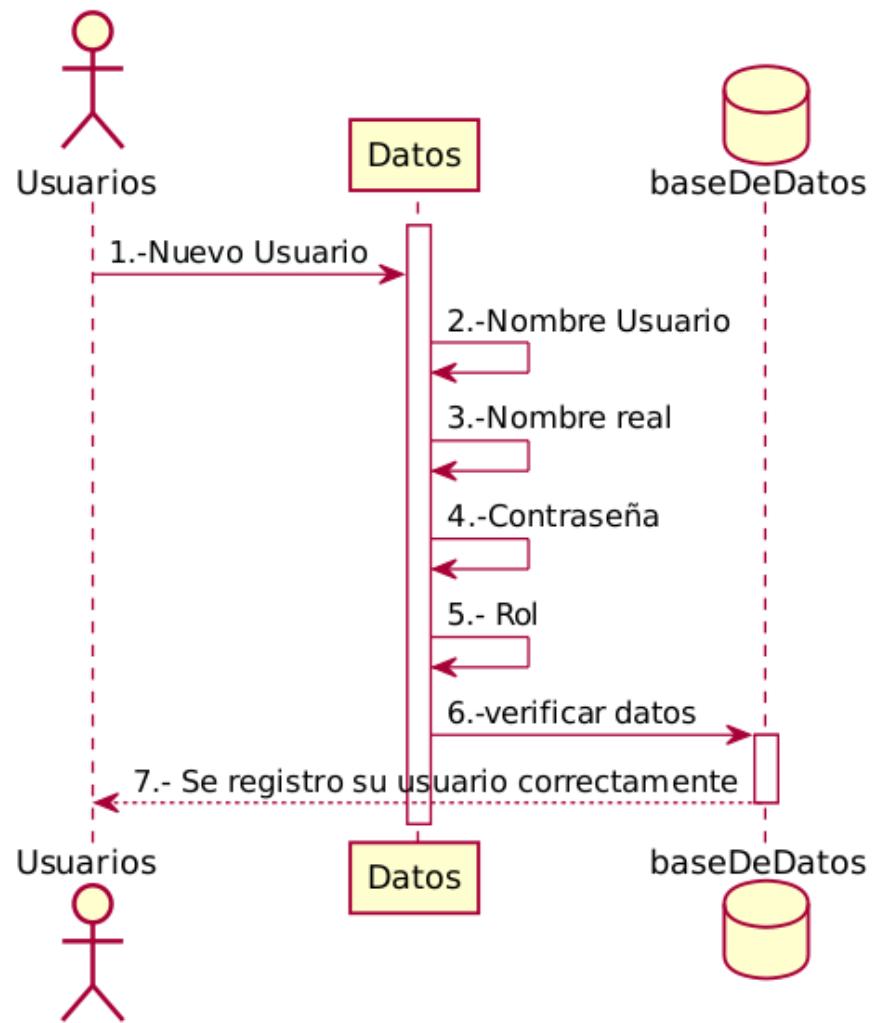


Figure 14: Diagrama de Usuarios

4.4.4 Diagrama de Secuencia del Proveedores

En la siguiente imagen se muestra el diagrama de secuencias de Proveedores.

4.-Registrar Proveedores

```
@startuml
actor Administrador
Activate Datos
database baseDeDatos
Administrador->Datos: 1.-Nuevo proveedor
Datos->Datos:2.-Nombre proveedor
Datos->Datos:3.-Código de proveedor
Datos->Datos:4.-Teléfono
Datos->Datos:5.- Dirección
Datos->baseDeDatos:6.-verificar datos
activate baseDeDatos
baseDeDatos-->Administrador:7.- Se registro un nuevo proveedor correctamente
deactivate baseDeDatos
@enduml
```

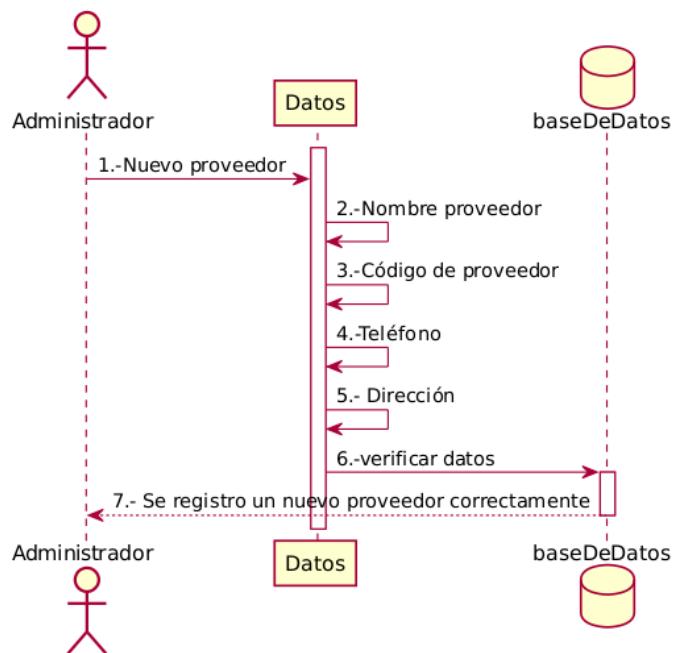


Figura 4: Secuencia-Proveedor

4.4.5 Diagrama de Secuencia del Clientes

En la siguiente imagen se muestra el diagrama de secuencias de los clientes.

```
5.-Registro de Clientes
@startuml
actor Administrador
Activate Datos
database baseDeDatos #pink
Administrador->Datos: 1.-Nuevo Cliente
Datos->Datos:2.-Nombre Cliente
Datos->Datos:3.-Código de Cliente
Datos->Datos:4.-Teléfono
Datos->Datos:5.- Dirección
Datos->baseDeDatos:6.-verificar datos
activate baseDeDatos
baseDeDatos-->Administrador:7.- Se registro un nuevo Cliente correctamente
deactivate baseDeDatos
@enduml
```

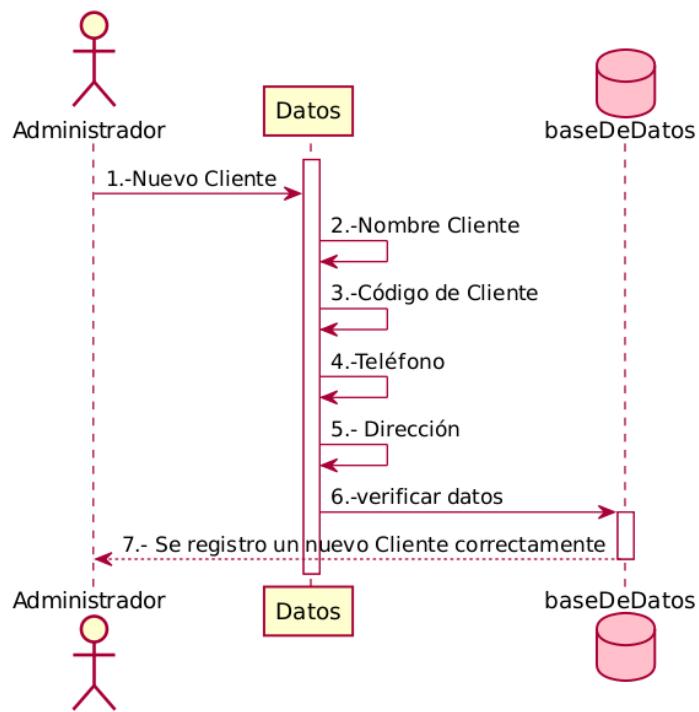


Figura 5: Secuencia-Clientes

4.4.6 Diagrama de Secuencia de la información de la tienda

En la siguiente imagen se muestra el diagrama de secuencias de la información de la tienda.

6.-Información de la Tienda

```
@startuml
actor Administrador
Activate Datos
database baseDeDatos #pink
Administrador-> Datos: 1.-Edita la informacion
Datos->Datos:2.-Nombre empresa
Datos->Datos:3.-Código de empresa
Datos->Datos:4.-Teléfono
Datos->Datos:5.- Dirección
Datos->baseDeDatos:6.-verificar datos
activate baseDeDatos
baseDeDatos-->público:7.- Se muestra la información de la empresa
deactivate baseDeDatos
@enduml
```

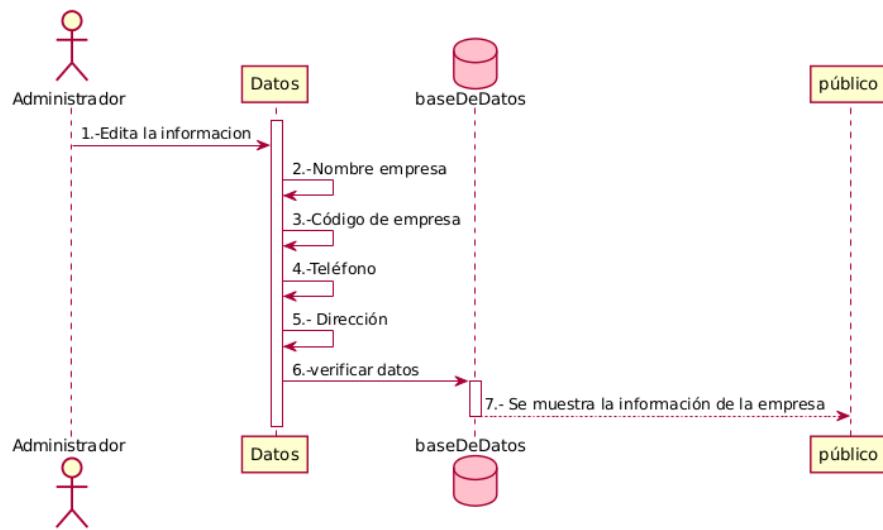


Figura 6: Secuencia-empresa

4.4.7 Diagrama de Secuencia de la información de la consulta de ventas

En la siguiente imagen se muestra el diagrama de la consulta de ventas.

7.-Consulta Ventas

```
@startuml  
actor Vendedor  
database baseDeDatos #pink  
Vendedor->baseDeDatos :1.-Búска las ventas realizadas  
Activate baseDeDatos  
baseDeDatos -->Vendedor:2.-Verifica por fechas  
deactivate baseDeDatos  
baseDeDatos-->3.-Vendedor:Devuelve el reporte de las ventas  
@enduml
```

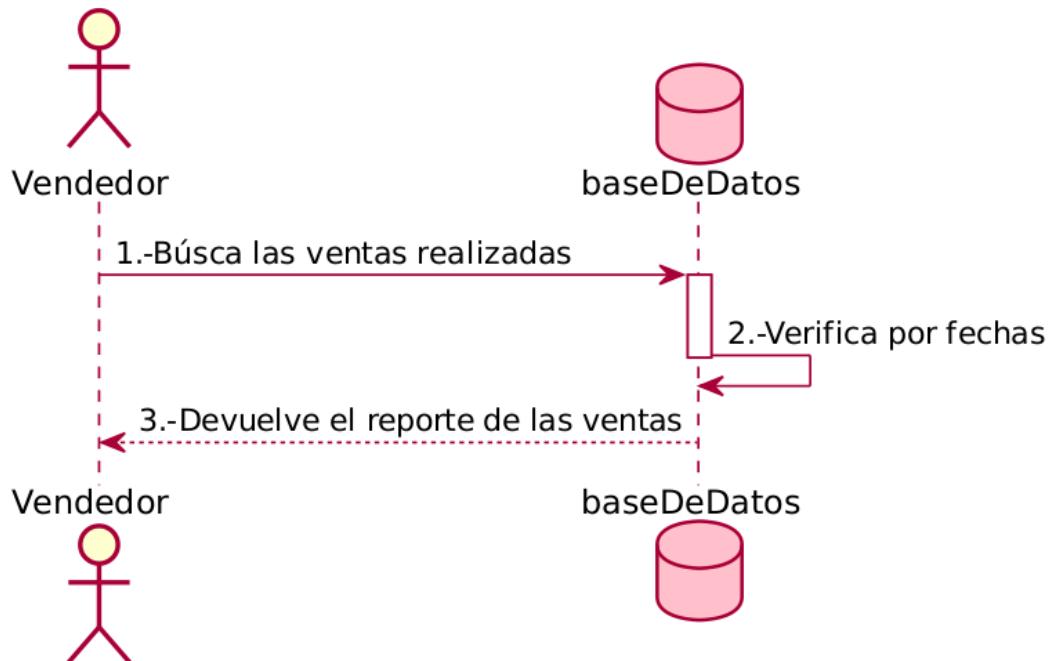


Figura 7: Secuencia-ConsultaVentas

4.5 Documentación

A continuación se muestra la documentación de los casos de uso con sus respectivas tablas dependiendo cada caso de uso.

Documentación

CASO DE USO

Id:	CU – 0		
Nombre:	Login		
Creado por:	L.S.Fernando	Actualizado por	L.S. Fernando
Fecha de creación:	3/12/21	Fecha de última revisión:	06/12/21
Actores:	Base de datos, Administrador		
Descripción:	Muestra pantalla para iniciar sesión		
Disparador:			
Predi-condiciones:			
Pos-condiciones:			
Flujo normal:	<p>1.- Ingresa el nombre del usuario con el que se va ingresar al sistema.</p> <p>2.- Ingresa la contraseña que corresponde al usuario con el que va ingresar</p> <p>3.- Oprimir el botón “Ingresar”.</p> <p>4.-En caso de no encontrar al usuario o la contraseña limpia los campos y muestra un mensaje de datos incorrectos.</p> <p>5.- Ingresa al sistema.</p>		
Flujos alternos:			
Includes:	N/A		
Frecuencia de uso:	Frecuente.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notes:	N/A		

CASO DE USO			
Id:	CU – 1		
Nombre:	Inventario		
Creado por:	L.S.F	Actualizado por	V.F.L.F
Fecha de creación:	01/12/21	Fecha de ultima revisión:	01/12/21
Actores:	Base de datos y Administrador.		
Descripción:	Se agregan los productos al inventario		
Disparador:	En el menú de la pantalla 2, presionar el botón Agregar producto		
Predi-condiciones:			
Pos-condiciones:	Se guardan los datos y se limpian los campos de la ventana 3		
Flujo normal:	1.- Ingresar el código del producto 2.- Ingresar el nombre del producto 2.1 Si ingresa un numero le mostrará una ventana emergente con un mensaje “El dato debe ser ingresado con letras” 3.- Agregar una breve descripción del producto 3.1-Si ingresa un numero le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con letras” 4.-Agregar el precio de compra 4.1-Si ingresa una letra le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con números” 5.- Agregar el precio de venta 5.1-Si ingresa una letra le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con números” 6.- Ingresar la cantidad de productos que se agregarán 7.- Oprime al botón “Aregar” 7.1- Mostrará una ventana emergente con el mensaje “Datos ingresados correctamente”. 7.2- Oprimir el botón “ok” para cerrar la pantalla emergente 8.- Si oprimen el botón Actualizar se activan los campos para hacer las actualizaciones . 9.- si Oprimes el botón Mostrar Producto se despliega una tabla con todos los productos		
Flujos alternos:	Para cerrar ésta pantalla le dan clic la pestaña Cerrar sesión		
Includes:	N/A		
Frecuencia de uso:	Frecuente.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notes:	N/A		

CASO DE USO

Id: CU – 3

Nombre: Clientes

Creado por: L.S.Fernando **Actualizado por:** V.F.L.Fernando

Fecha de creación: **Fecha de ultima revisión:**

	2/12/21	3/12/21
Actores:	Base de datos, Administrador.	
Descripción:	Agregar, elimina y modifica nuevos clientes	
Disparador:	En el menú de la pantalla 3, oprimir el botón Agregar clientes	
Predi-condiciones:		
Pos-condiciones:	Se guardan los datos y se limpian los campos	
Flujo normal:	<p>1.- Ingresar el nombre del cliente 2.- Ingresar su apellido</p> <p>2.1 Si ingresa un número le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con letras”</p> <p>3.- Ingresar el nombre de usuario</p> <p>3.1-Si ingresa un número le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con letras”</p> <p>4.- ingresar la contraseña</p> <p>4.1-Si ingresa un número le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con letras”</p> <p>5.- Oprimir el botón “Agregar cliente”</p> <p>5.1- Si alguno de los campos no esta llenado se enviara un mensaje ”Rellenar todos los campos”</p> <p>5.1.1-Oprimir el botón OK para salir de la ventana emergente y regresa al paso 5.</p> <p>5.2- Si los datos son correctos aparecerá el mensaje “se agregó correctamente”</p> <p>5.2.1.- presionar OK para salir de la ventana se limpian los campos y regresa.</p> <p>7.-si oprime el botón “Modificar Cliente”se muestra los datos de cliente para que se pueda modificar.</p> <p>8.- se Oprime el botón” Eliminar cliente” y se elimina el cliente seleccionado</p>	
Flujos alternos:	Se cierra la ventana dando clic en pestaña “ Cerrar sesión ”	
Includes:	N/A	
Frecuencia de uso:	Frecuente.	
Requerimientos especiales:	N/A	
Supuestos:	N/A	
Issues y Notes:	N/A	

CASO DE USO		
Id:	CU – 4	
Nombre:	Proveedores	
Creado por:	V.F.L.Fernando	Actualizado por M.M.Nancy Obed
Fecha de creación:	06/12/21	Fecha de ultima revisión:
Actores:	Base de datos y administrador	
Descripción:	Se hacen los registros,modificaciones y borrados de proveedores	
Disparador:	Inicio de sesión	
Predi-condiciones:		
Pos-condiciones:		
Flujo normal:	1.- En el menú principal elegir la opción “Archivo” , en el submenu seleccionar “Nueva venta” 2.- Aparecerá una tabla. 2.-Para agregar un producto a la venta escribir el código en la caja de texto	

4.- En la tabla aparecerán los datos del producto al que corresponde ese código
5.-En la parte superior aparecerá el subtotal y posteriormente el total en caso de aplicar algún descuento
6.-Oprimir el botón aceptar
 6.1.-Guarda los datos de la venta
7.- El botón cancelar cancelará la venta y limpiara la tabla

Flujos alternos: Regresa a la pantalla 2

Includes: N/A

Frecuencia de uso: Frecuente.

Requerimientos especiales: N/A

Supuestos: N/A

Issues y Notes: N/A

CASO DE USO

Id:	CU – 5
Nombre:	Ventas
Creado por:	Actualizado por
L.S.Fernando	L.S.Fernando
Fecha de creación:	Fecha de ultima revisión: 07/12/21
06/12/21	
Actores:	Base de datos y vendedor
Descripción:	Ejecuta venta de productos
Disparador:	Inicio de sesión
Predi-condiciones:	
Pos-condiciones:	
Flujo normal:	<p>1. Oprimes el Botón “Aregar” para comenzar a agregar un producto a la venta.</p> <p>2.-Si ya terminas sus compras le das un clic al botón Finalizar venta</p> <p>3.- También si durante la venta el cliente ya no termina o ya no quiere las cosas le oprimes el botón Cancelar venta”</p> <p>4 .- para moverte en esta pantalla tiene tres pestañas proveedor.</p> <p>4.1 si quieres irte a proveedores le das clic en la pestañas proveedor.</p>
Flujos alternos:	Para cerrar esta pantalla le dan clic a la pestaña cerrar sesión
Includes:	N/A
Frecuencia de uso:	Frecuente.
Requerimientos especiales:	N/A
Supuestos:	N/A
Issues y Notes:	N/A

CASO DE USO		
Id:	CU – 6	
Nombre:	JVentas	
Creado por:	L.S.Fernando	Actualizado por M.M.Nancy Obed
Fecha de creación:	05/12/21	Fecha de ultima revisión:
Actores:	Base de datos y administrador	
Descripción:	Se hacen las consultas de los registros de ventas	
Disparador:	Ventas	
Predi-condiciones:		
Pos-condiciones:		
Flujo normal:	1.- En el menú principal elegir la opción “General” , en el submenu seleccionar “Ventas” 2.- Aparecerá una tabla. 3.-Para regresar pulsar el botón “Regresar”	
Flujos alternos:	Regresa a la pantalla Ventas	
Includes:	N/A	
Frecuencia de uso:	Frecuente.	
Requerimientos especiales:	N/A	
Supuestos:	N/A	
Issues y Notes:	N/A	

CASO DE USO

Id:	CU – 7
Nombre:	JClientes
Creado por:	Actualizado por
	M.M.Nancy Obed
Fecha de creación:	Fecha de ultima revisión:
05/12/21	L.S.Fernando
Actores:	Base de datos y administrador
Descripción:	Se hacen las consultas de los registros de Clientes
Disparador:	Ventas
Predi-condiciones:	
Pos-condiciones:	
Flujo normal:	<p>1.- En el menú principal elegir la opción “General” , en el sub-menu seleccionar “Clientes”</p> <p>2.- Aparecerá una tabla.</p> <p>3.-Si desea realizar una búsqueda en específico ingresar dicho dato en el espacio en blanco de la pantalla.</p> <p>3.-Para regresar pulsar el botón “Regresar”.</p>
Flujos alternos:	Regresa a la pantalla Ventas
Includes:	N/A
Frecuencia de uso:	Frecuente.
Requerimientos especiales:	N/A
Supuestos:	N/A
Issues y Notes:	N/A

CASO DE USO		
Id:	CU – 7	
Nombre:	JProducto	
Creado por:	V.F.L.Fernando	Actualizado por
		L.S.Fernando
Fecha de creación:	05/12/21	Fecha de ultima revisión:
Actores:	Base de datos y administrador	
Descripción:	Se hacen las consultas de los registros de Productos	
Disparador:	Ventas	
Predi-condiciones:		
Pos-condiciones:		
Flujo normal:	1.- En el menú principal elegir la opción “General” , en el sub-menu seleccionar “Producto” 2.- Aparecerá una tabla. 3.-Si desea realizar una búsqueda en específico ingresar dicho dato en el espacio en blanco de la pantalla. 3.-Para regresar pulsar el botón “Regresar”.	
Flujos alternos:	Regresa a la pantalla Ventas	
Includes:	N/A	
Frecuencia de uso:	Frecuente.	
Requerimientos especiales:	N/A	
Supuestos:	N/A	
Issues y Notes:	N/A	

4.6 Diagrama de Estados

4.6.1 Diagrama de Estados Cliente

En la siguiente imagen se muestra el diagrama de Estados

```
@startuml
[*] --> Nuevo
Nuevo : Nuevo cliente
Nuevo--> DatosCliente

DatosCliente--> Validacion
Validacion: Validacion de datos
Validacion-->Error
Error: Error de datos
Error-->Validacion
Validacion-->Guardar
Guardar --> [*]
@enduml
```

Figure 15: Diagrama de Estados cliente

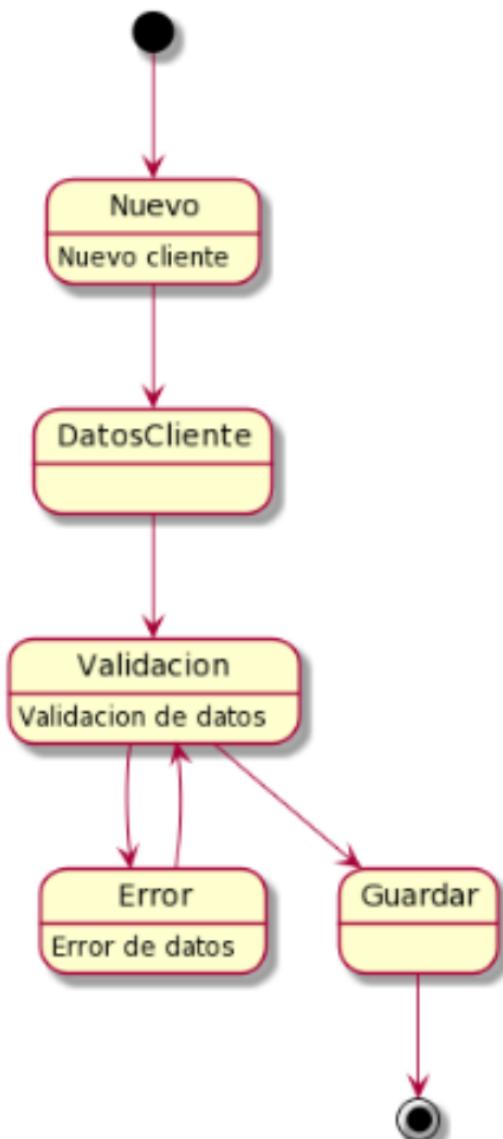


Figure 16: Diagrama de Estados cliente

4.6.2 Diagrama de Estados Proveedor

En la siguiente imagen se muestra el diagrama de Estados

```
@startuml
[*] --> Nuevo
Nuevo : Nuevo Proveedor
Nuevo--> DatosProveedor
DatosProveedor--> Validacion
Validacion: Validacion de datos
Validacion-->Error
Error: Error de datos
Error-->Validacion
Validacion-->Guardar
Guardar --> [*]
@enduml
```

Figure 17: Diagrama de Estados proveedor

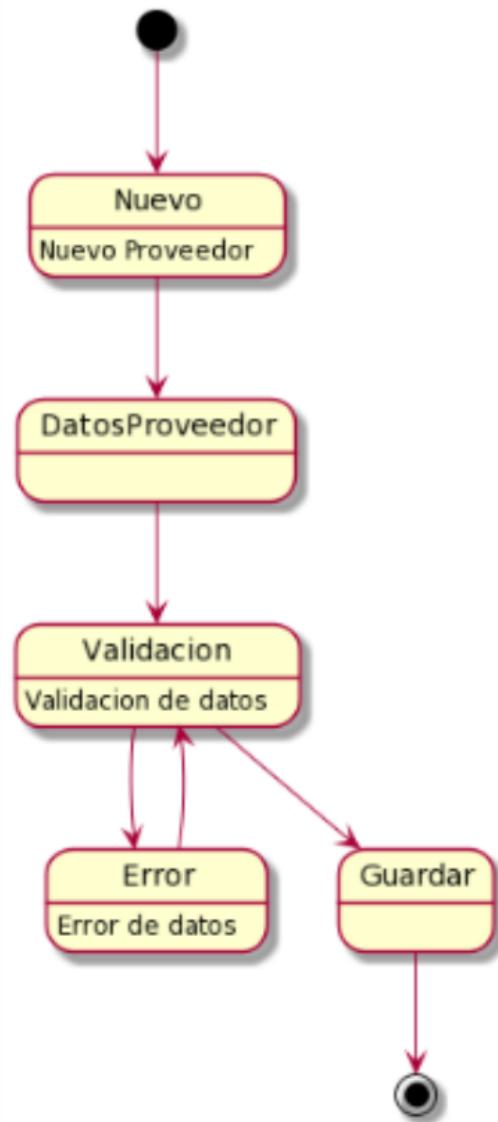


Figure 18: Diagrama de Estados proveedor

4.6.3 Diagrama de Estados Productos

En la siguiente imagen se muestra el diagrama de los Productos

```
@startuml
[*] --> ActualizarProductos
ActualizarProductos--> IngresaDatos
IngresaDatos-->ValidarDatos
ValidarDatos-->Actualizar:Datos Correctos
ValidarDatos-->Error: Datos incorrectos
Error: Error de datos
Error-->IngresaDatos
Actualizar --> [*]
@enduml|
```

Figure 19: Diagrama de Estados Productos

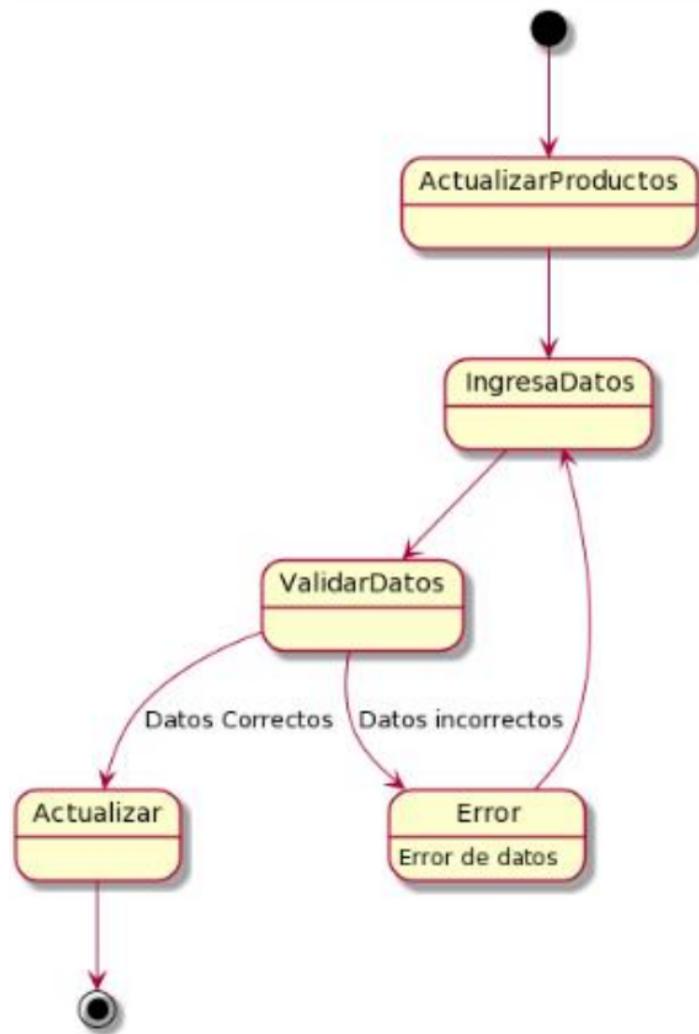


Figure 20: Diagrama de Estados Productos

4.6.4 Diagrama de Estados Actualizar Proveedores

En la siguiente imagen se muestra el diagrama donde se actualizan los Proveedores

```
@startuml
[*] --> ActualizarProveedores
ActualizarProveedores--> IngresarDatos
IngresarDatos--> ValidarDatos
ValidarDatos--> Actualizar:Datos Correctos
ValidarDatos--> Error: Datos incorrectos
Error: Error de datos
Error--> IngresarDatos
Actualizar --> [*]
@enduml
```

Figure 21: Diagrama de Estados Actualizar Proveedores



Figure 22: Diagrama de Estados Actualizar Proveedores

4.6.5 Diagrama de Estados para realizar una Venta

En la siguiente imagen se muestra el diagrama donde se realiza la venta

```
@startuml
[*] --> SeleccionarProducto
SeleccionarProducto: Descripcion
SeleccionarProducto--> AgregarProductos
SeleccionarProducto--> EliminarProductos
AgregarProductos--> Total
EliminarProductos-->Total
Total -->pagar
Total-->Cancelar
pagar --> [*]
Cancelar --> [*]
@enduml
```

Figure 23: Diagrama de Estados Realizar Venta

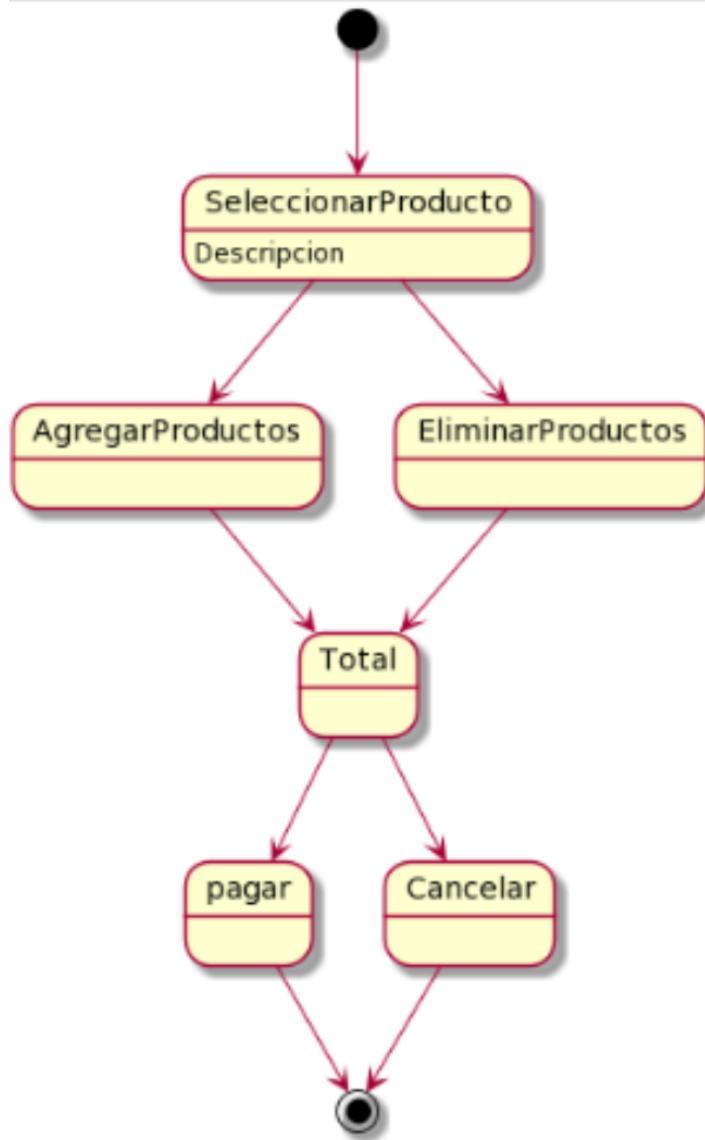


Figure 24: Diagrama de Estados Realizar Venta

4.6.6 Diagrama de Estados para el Logeo

En la siguiente imagen se muestra el diagrama donde se realiza el Logeo.

```
@startuml
[*] --> login
login : Ventana de login
login--> CapturaDatos
CapturaDatos-->Ingresar
Ingresar : ingresa usuario, contraseña
Ingresar--> Validacion
Validacion: Validacion de datos
Validacion-->Error
Error: Error de datos
Error-->Validacion
Validacion-->VentanaPrincipal
VentanaPrincipal --> [*]
@enduml
```

Figure 25: Diagrama de Estados del Logeo.

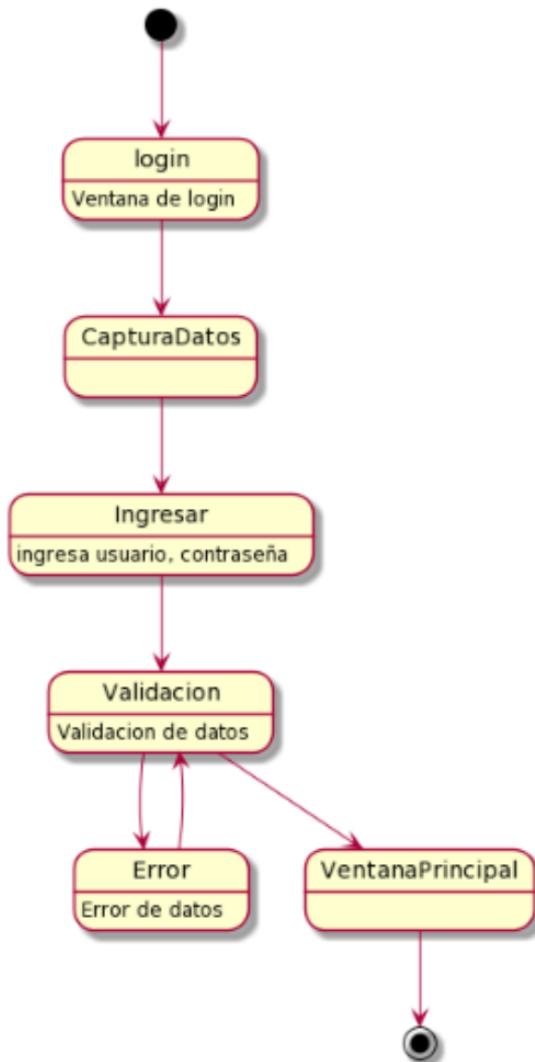


Figure 26: Diagrama de Estados del Logeo.

4.7 Caso de Uso

4.7.1 Caso de Uso de un login

En la siguiente imagen se muestra el código dónde se realiza el Logeo.

```
@startuml
left to right direction
actor "Vendedor" as fc
actor "Administrador" as a
actor "base de datos" as bd
rectangle LOGIN {
    usecase "Logeo" as UC1
    usecase "Validar datos" as UC2
    usecase "Usuario y contraseña" as UC3
}
fc --> UC1
a --> UC1
UC1-->UC2
UC2-->bd
UC1 <..UC3 :<<extends>>
@enduml
```

Figure 27: Caso de uso del Logeo.

En la siguiente imagen se muestra el diagrama de Caso de uso del Login.

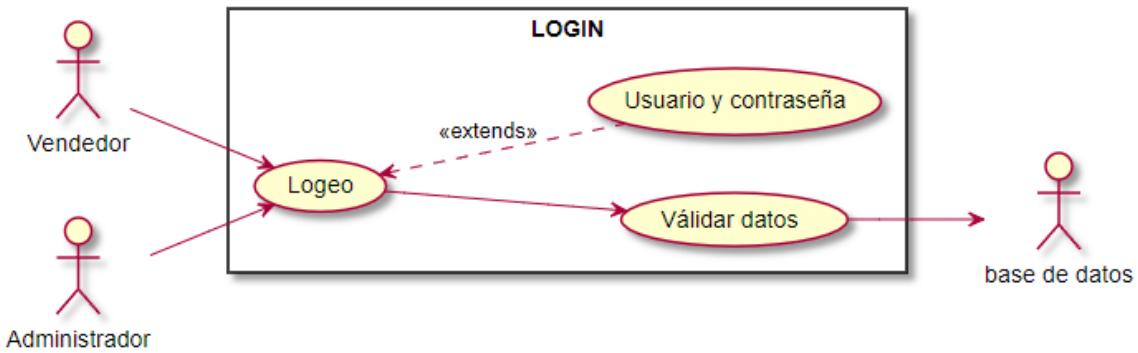


Figure 28: Diagrama de caso de uso del Logeo.

4.7.2 Caso de Uso del Inventario

En la siguiente imagen se muestra el código donde se realiza el inventario.

```
@startuml
actor "Administrador" as fc
actor "Base de Datos" as bd
rectangle Inventario {
    usecase "Gestionar producto" as UC1
    usecase "Alta producto" as UC2
    usecase "Actualizar producto" as UC3
    usecase "Baja producto" as UC4
    usecase "Mostrar producto" as UC5
}
fc-->UC1
UC1-->bd
UC1 <..UC2 :<<extends>>
UC1 <..UC3 : <<extends>>
UC1 <..UC4 :<<extends>>
UC1 <..UC5 :<<extends>>
@enduml
```

Figure 29: Caso de uso del inventario.

En la siguiente imagen se muestra el diagrama de Caso de uso del inventario.

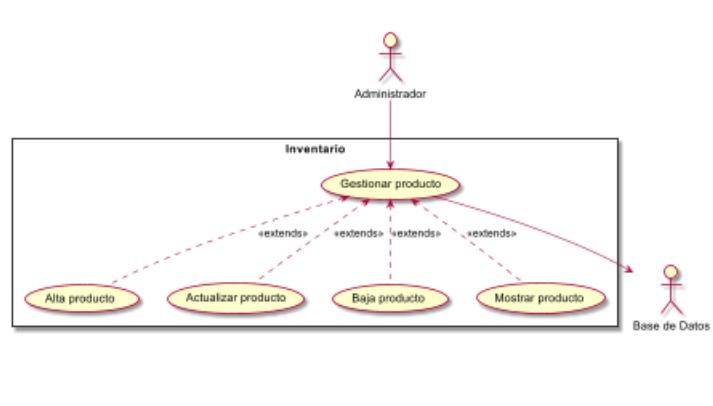


Figure 30: Diagrama de caso de uso del inventario.

4.7.3 Caso de Uso del Usuario

En la siguiente imagen se muestra el código dónde se realiza alta de usuarios.

```
@startuml
actor "Administrador" as fc
actor "Base de Datos" as bd
rectangle Usuarios {
    usecase "Gestionar Usuario" as UC1
    usecase "Alta Usuario" as UC2
    usecase "Actualizar Usuario" as UC3
    usecase "Baja Usuario" as UC4
    usecase "Mostrar Usuario" as UC5
}
fc-->UC1
UC1-->bd
UC1 <..UC2 :<<extends>>
UC1 <..UC3 : <<extends>>
UC1 <..UC4 :<<extends>>
UC1 <..UC5 :<<extends>>
@enduml
```

Figure 31: Caso de uso de Usuarios.

En la siguiente imagen se muestra el diagrama de Caso de uso de Usuarios.

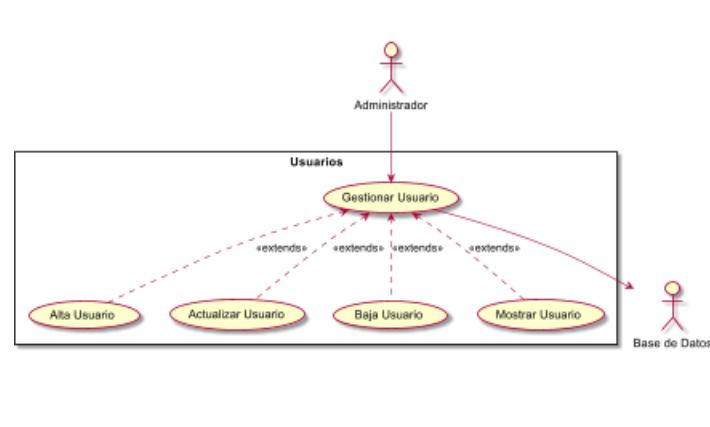


Figure 32: Diagrama de caso de uso de Usuarios.

4.7.4 Caso de Uso de Proveedores

En la siguiente imagen se muestra el código donde se realiza lo de proveedores.

```
@startuml
actor "Administrador" as fc
actor "Base de Datos" as bd
rectangle Proveedores {
    usecase "Gestionar Proveedor" as UC1
    usecase "Alta Proveedor" as UC2
    usecase "Actualizar Proveedor" as UC3
    usecase "Baja proveedor" as UC4
    usecase "Mostrar proveedor" as UC5
}
fc-->UC1
UC1-->bd
UC1 <..UC2 :<<extends>>
UC1 <..UC3 : <<extends>>
UC1 <..UC4 :<<extends>>
UC1 <..UC5 :<<extends>>
@enduml
```

Figure 33: Caso de uso de Proveedores.

En la siguiente imagen se muestra el diagrama de Caso de uso de proveedores.

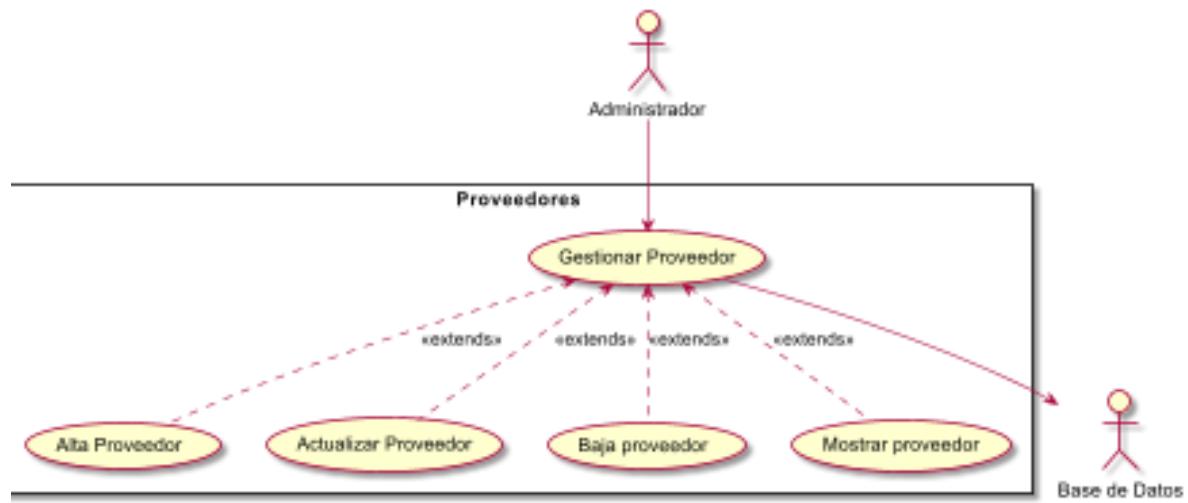


Figure 34: Diagrama de caso de uso de Proveedores.

4.8 Diagramas de colaboración

4.8.1 Diagramas de colaboración del Login

En la siguiente imagen se muestra el diagrama de colaboración del login.

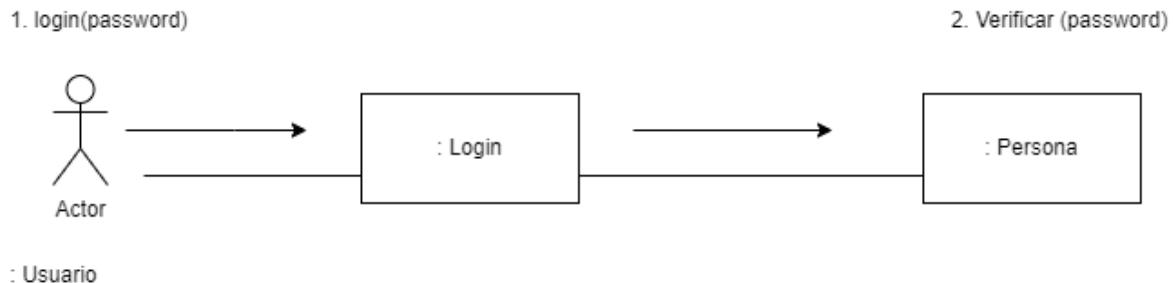


Figure 35: Diagrama de colaboración Login.

4.8.2 Diagramas de colaboración de la venta

En la siguiente imagen se muestra el diagrama de colaboración de la venta.

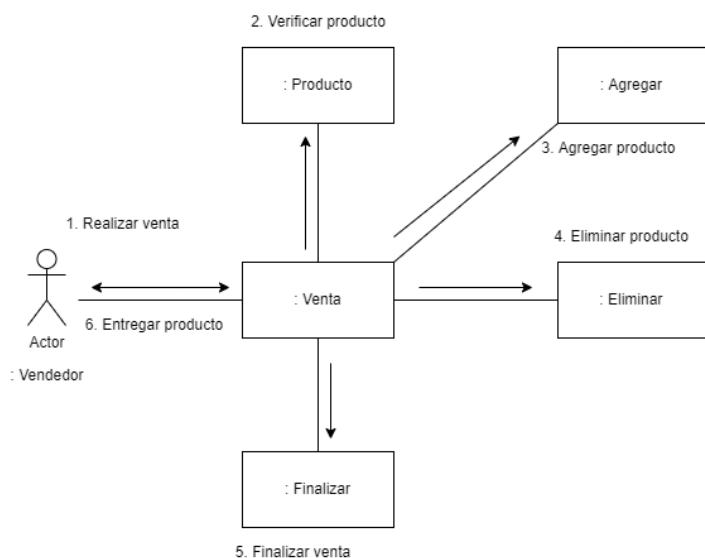


Figure 36: Diagrama de colaboración de Venta.

4.8.3 Diagramas de colaboración de la Factura

En la siguiente imagen se muestra el diagrama de colaboración de la Factura.

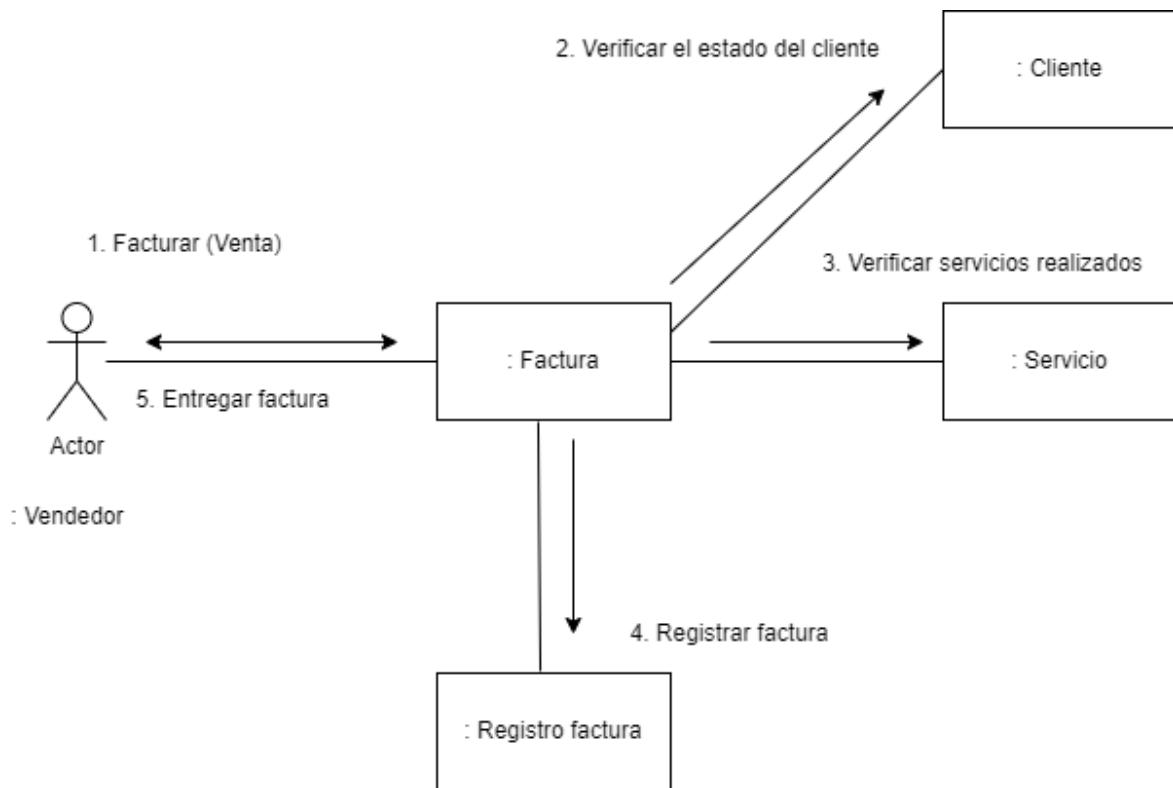


Figure 37: Diagrama de colaboración de Factura.

4.8.4 Diagramas de colaboración del producto

En la siguiente imagen se muestra el diagrama de colaboración del producto.

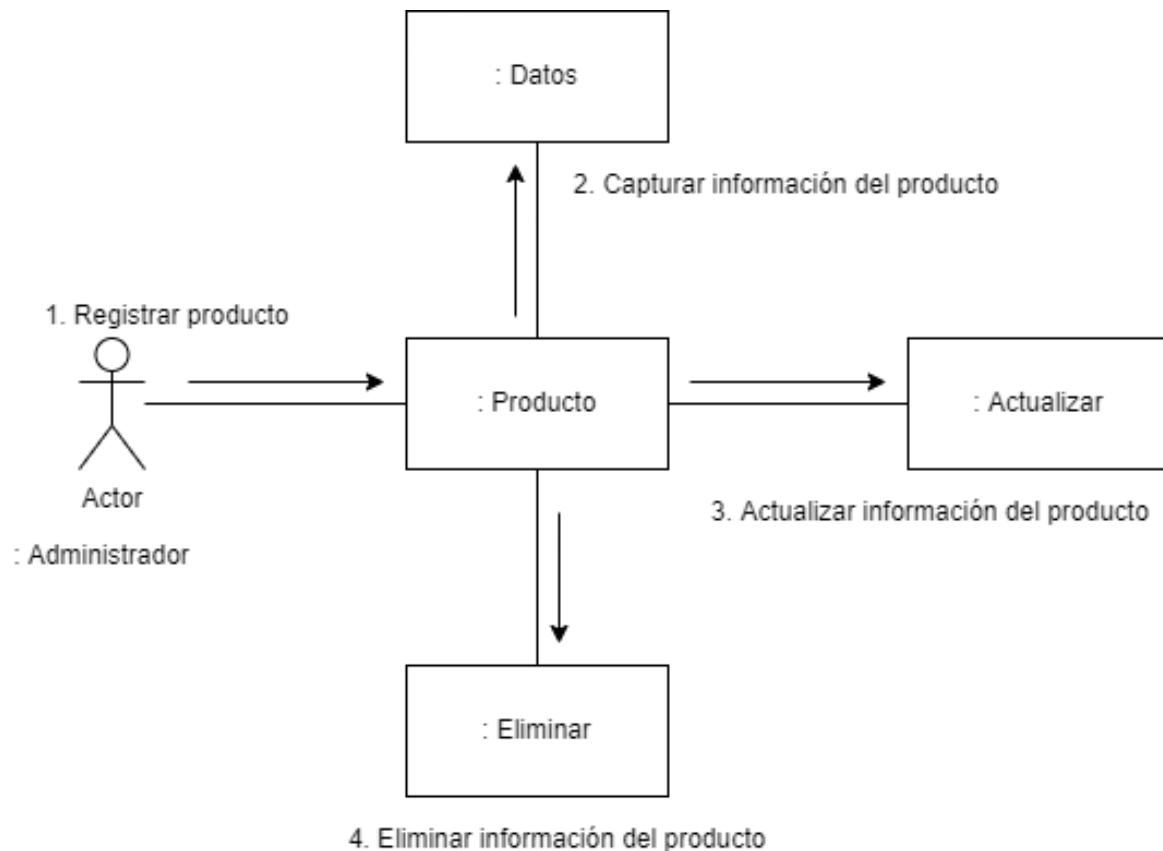


Figure 38: Diagrama de colaboración del producto.

4.8.5 Diagramas de colaboración del proveedor

En la siguiente imagen se muestra el diagrama de colaboración del proveedor.

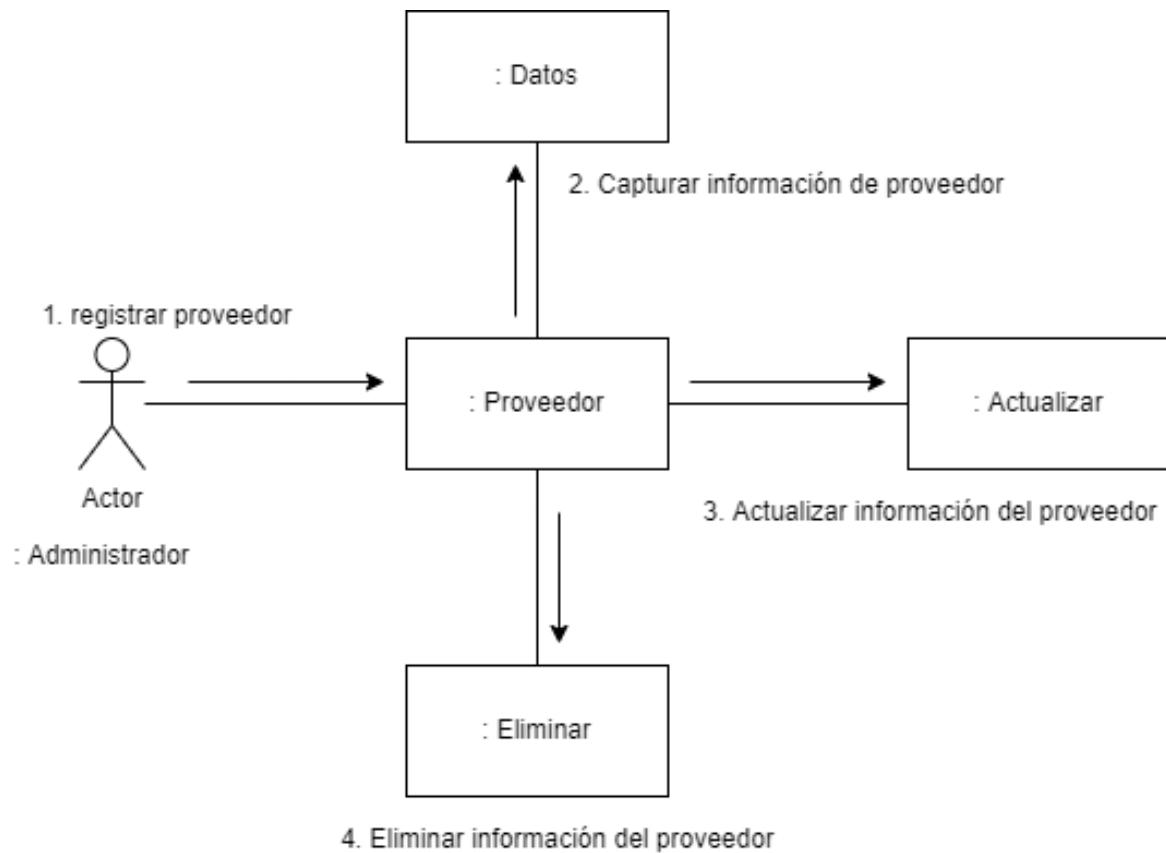


Figure 39: Diagrama de colaboración del proveedor.

Descripción Base datos

El sistema de ventas está orientada a la gestión de empresas comerciales o pequeños negocios, es un software práctico que va permitir un mejor control del negocio. Se trata de un software para la gestión de clientes y productos. Donde puedes ingresar como administrador y realizar los que se necesite así sea ingresar, eliminar o modificar un producto o de ser un vendedor buscar los productos que el cliente desee y consultar precios, cantidades en existencia y realizar su venta. Así mismo generará un informe de la venta y su respectivo ticket. El sistema se hace cargo de registrar las ventas del producto o servicio de la organización, evalúa sus características o necesidades del negocio, llevando un detalle del registro de todas las transacciones.

La base de datos contiene la tabla inventario en la cual se almacena el id del producto el cual es un tipo de valor serial para identificar el producto, contiene la descripción en el cual se hace una pequeña descripción del producto a almacenar es de tipo varchar, también contiene el precio del producto el cual es de tipo flotante el cual se almacenara el valor del producto, por último tenemos la cantidad de producto con la que contamos en esta se almacena el total del producto en inventario y es de tipo entero.

Tabla inventario

	Columna	Tipo de dato	Descripción
PK	Id_prod	Serial	Número de identificación para producto
	Descripción	VARCHAR (20)	Descripción de producto
	precio	Float	Número de precio de producto
	Stock	Integer	Número de stock de producto

En la tabla clientes tenemos el id_clientes el cual es la primary key de tipo serial y sirve para identificar al cliente, también tenemos la variable nombre de tipo varchar que contiene el nombre del cliente, la variable dirección contiene la dirección del cliente y es de tipo varchar, contamos también con la variable numero la cual es de tipo varchar y contiene el número de teléfono del cliente, la variable correo de tipo varchar con tiene la dirección de correo electrónico del cliente.

Tabla Clientes

	Columna	Tipo de dato	Descripción
PK	Id_clientes	Serial	Número de identificación para cliente
	nombre	VARCHAR (20)	Nombre completo del cliente
	dirección	VARCHAR (20)	Dirección completa del cliente
	teléfono	VARCHAR (12)	Teléfono de contacto
	correo	VARCHAR (20)	Correo electrónico de contacto
	contraseña	VARCHAR (20)	Contraseña del cliente

La tabla proveedores contiene el id_prov el cual almacena el id del cliente siendo esta la primary key de la tabla, contamos con variable nombre de tipo varchar la cual almacena el nombre de los proveedores, la variable dirección de tipo varchar contiene la dirección de los proveedores, la variable teléfono contiene el número de teléfono de los proveedores y por último la variable correo de tipo varchar en el cual almacenamos el correo de los proveedores.

Tabla proveedores

Columna		Tipo de dato	Descripción
PK	Id_prov	Serial	Número de identificación para proveedor
	Nombre	Varchar (20)	Nombre completo del proveedor
	dirección	Varchar (20)	Dirección completa del proveedor
	teléfono	Varchar (12)	Teléfono de contacto
	Correo	Varchar (20)	Correo electrónico de contacto

En la tabla ventas contamos con la variable id_venta la cual es serial y es la primary key de la tabla sirve para identificar la venta, contamos con el id_prod el cual es una llave foránea de la tabla inventario sirve para identificar el producto, tenemos el id_clientes el cual es una llave foránea de la tabla clientes la cual sirve para identificar al cliente, también el id_prov el cual es una llave foránea y sirve para identificar al proveedor, tenemos la cantidad el cual es un tipo de variable integrer y sirve para saber la cantidad de producto a vender, la variable precio sirve para saber la cantidad total a pagar y por ultimo la variable pago la cual almacena la cantidad ingresada a pagar .

	Columna	Tipo de dato	Descripción
PK	Id_venta	Serial	Código de identificación para ventas
FK	Id_prod	Integer	Número de identificación para producto
FK	Id_clientes	Integer	Número de identificación para cliente
FK	Id_prov	Integer	Número de identificación para proveedor

Diccionario de datos

Tabla usuarios

	Columna	Tipo de dato	Descripción
	usuario	Varchar (20)	Nombre de usuario
	contraseña	Varchar (20)	Contraseña de usuario
	tipo	Varchar (20)	Descripción de tipo de usuario

Tabla inventario

	Columna	Tipo de dato	Descripción
PK	Id_prod	Serial	Número de identificación para producto
	Descripción	Varchar (20)	Descripción de producto
	precio	Float	Número de precio de producto
	Stock	Integer	Número de stock de producto

Tabla Clientes

	Columna	Tipo de dato	Descripción
PK	Id_clientes	Serial	Número de identificación para cliente
	nombre	Varchar (20)	Nombre completo del cliente
	dirección	Varchar (20)	Dirección completa del cliente
	teléfono	Varchar (12)	Teléfono de contacto
	correo	Varchar (20)	Correo electrónico de contacto
	contraseña	Varchar (20)	Contraseña del cliente

Tabla proveedores

	Columna	Tipo de dato	Descripción
PK	Id_prov	Serial	Número de identificación para proveedor
	Nombre	Varchar (20)	Nombre completo del proveedor
	dirección	Varchar (20)	Dirección completa del proveedor
	teléfono	Varchar (12)	Teléfono de contacto
	Correo	Varchar (20)	Correo electrónico de contacto

Tabla ventas

	Columna	Tipo de dato	Descripción
PK	Id_venta	Serial	Código de identificación para ventas
FK	Id_prod	Integer	Número de identificación para producto
FK	Id_clientes	Integer	Número de identificación para cliente
FK	Id_prov	Integer	Número de identificación para proveedor

Diagrama relacional BD

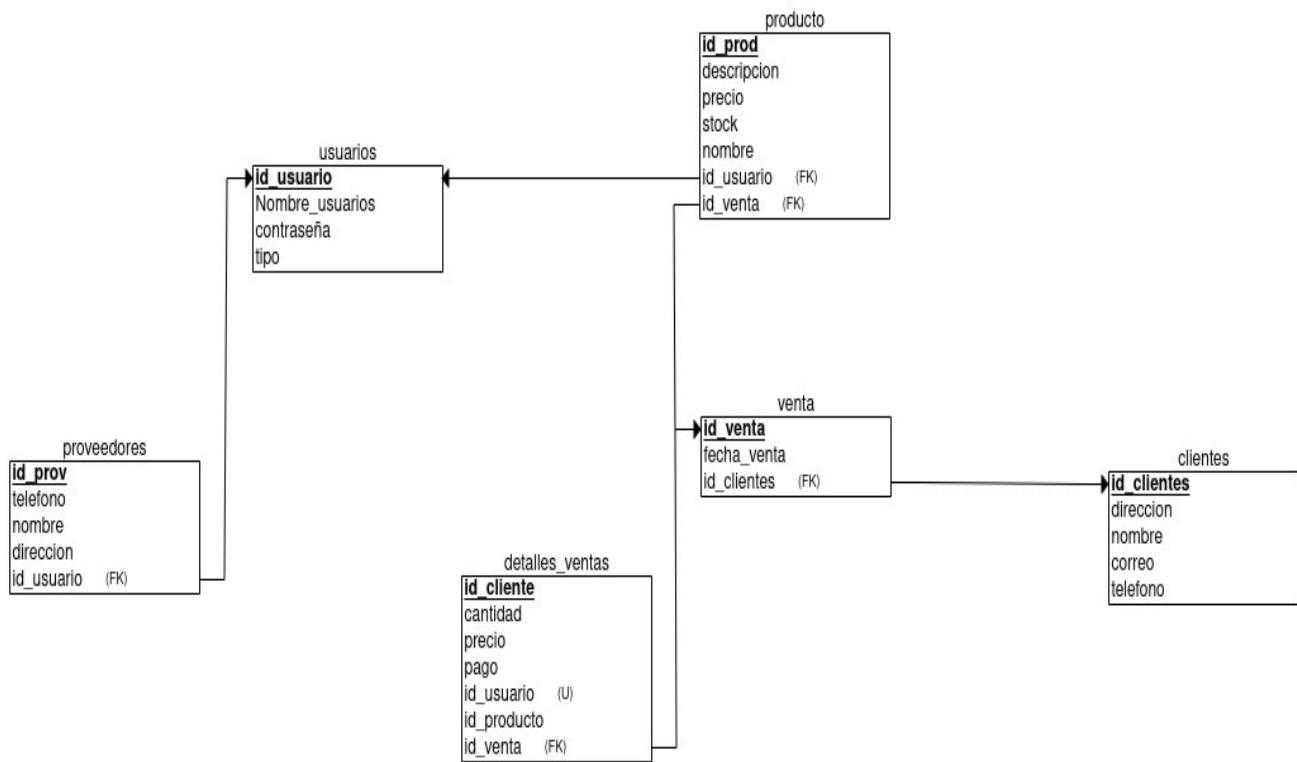
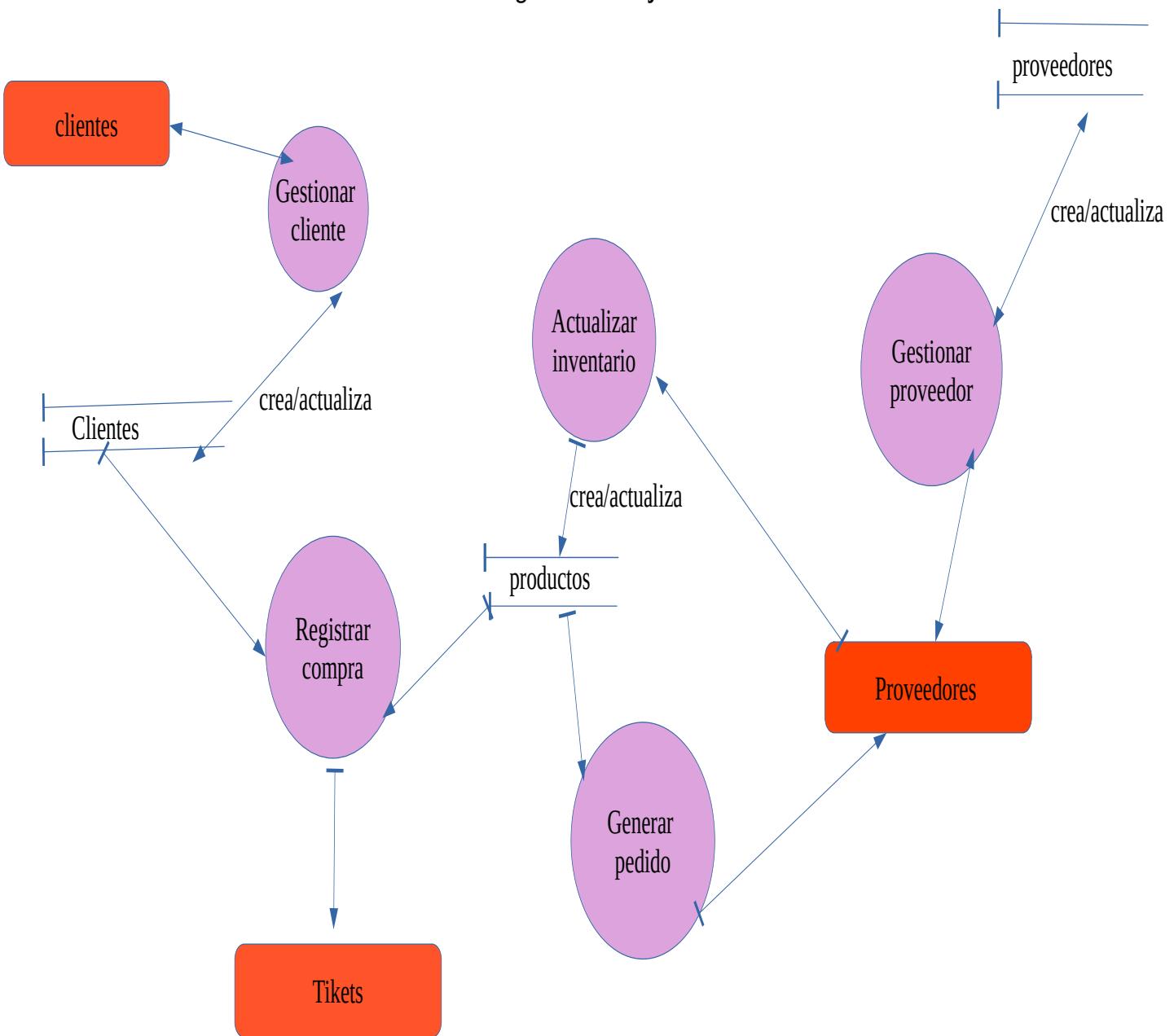


Diagrama de Flujo de Datos



5 Diseño

5.1 Introducción

A continuación les mostraremos las herramientas que utilizamos para la elaboración y desarrollo de nuestro programa, así como los programas para el diseño de interfaces, el diseño del código y la creación de la base de datos. Por lo que estas aplicaciones mejoran el control administrativo llevando un seguimiento preciso de todas las transacciones que se realizan dentro de la tienda en un tiempo real, para generar reportes detallados de las cuentas que van a permitir a los administradores la cantidad correcta de productos, en el momento adecuado lo cual va a permitir al negocio o empresa mejorar servicio al cliente reduciendo cualquier anomalía. Los sistemas de ventas tienen la ventaja de ser personalizados para cumplir con las necesidades específicas de un negocio, por ejemplo, en la tienda las organizaciones de venta al menudeo pueden ayudar a localizando precios de venta de los productos y costos actuales de los productos para poder trabajar más rápidamente.

5.2 Descripción del sistema

El sistema automatizado de ventas es donde un cliente paga por un artículo. Entonces, básicamente, es el sistema que permite a las partes proceder con la transacción entre un cliente y una tienda, este sistema ayudara a las pequeñas tiendas a tener un funcionamiento adecuado para el manejo de artículos y con respecto a el almacenamiento de información, así también en caso de ser necesario el registro y modificación de los clientes que frecuentemente visitan la tienda. Por lo que en este proyecto se enfoca en implementar un software de ámbito comercial, este software será diseñado para el control de un sistema de ventas de una tienda de abarrotes, buscando una optimización de los datos que esta maneja.

5.3 Arquitectura del sistema

5.3.1 Arquitectura Física

A continuación les mostraremos algunos de los equipos que utilizamos en la elaboración de nuestro proyecto incluyendo equipos personales así como equipos de la respectiva universidad. Dicha universidad nos proporciono 3 equipos independientes a cada uno de los integrantes del equipo en las cuales realizamos algunas de las actividades, pero en mayor medida los equipos que más utilizábamos eran los propios debido a la comodidad que estos nos proporcionan.

Equipos o computadoras

Equipos utilizados			
Fabricante	Modelo	Procesador	Memoria
HP	Pavilon	Ryzen 3	8 Ram

Table 9: Equipos Utilizados HP Pavilon



Figure 40: Pavilon

Equipos utilizados			
Fabricante	Modelo	Procesador	Memoria
HP	15-bs08	Intel I5	12 Ram

Table 10: Equipos Utilizados HP 15-bs



Figure 41: Custom

Equipos utilizados			
Fabricante	Modelo	Procesador	Memoria
Custom	Custom	Ryzen 5	16 Ram

Table 11: Equipos Utilizados

Equipos portatiles o Celulares

Equipos moviles utilizados			
Fabricante	Modelo	Procesador	Memoria
Samsung	Note5	Snapdragon	8 Ram

Table 12: Samsung note 5



Figure 42: Galaxy note 5

Equipos moviles utilizados			
Fabricante	Modelo	Procesador	Memoria
Motorola	Power	Snapdragon	8 Ram

Table 13: Motorola



Figure 43: Motorola

Equipos moviles utilizados			
Fabricante	Modelo	Procesador	Memoria
Xiaomi	Redmi	Snapdragon	12 Ram

Table 14: Xiaomi



Figure 44: xiaomi

5.3.2 Arquitectura Lógica

A continuación les mostraremos algunos de los programas o software que utilizamos para el desarrollo de nuestro sistema, muchos de ellos son encontrados de forma gratuita, es decir, que no se necesita licencia para poder utilizarlos o algunos de ellos tienen solo la versión de prueba la cual limita los campos a los cuales podemos acceder.

Programas para la codificación

Estos son los programas enfocados al desarrollo del programa en el código fuente el principal que utilizamos es Java con la opción de "Apache Netbeans" con su versión 1.12.3, la cual nos facilito el desarrollo de la aplicación debido a que teníamos ya experiencia en dicho programa

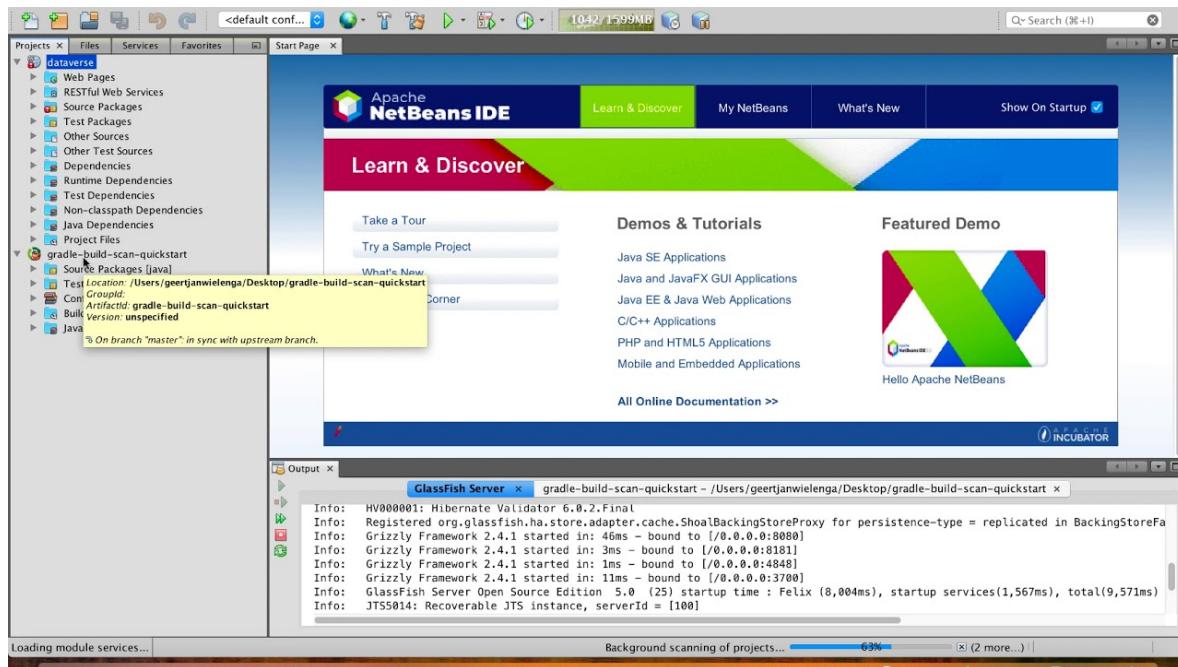


Figure 45: Apache netbeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Programas para el diseño de prototipos

Estos son los programas enfocados al diseño de prototipos en este caso nosotros utilizamos el programa gratuito en línea el cual se llama "balsamiq" a herramienta para hacer prototipos de proyectos. Es un programa de escritorio, es decir, solo tienes que registrarte para poder empezar a utilizarlo sin ningún tipo de descarga. Interfaz fácil de usar: como se ha creado con AIR es instalable tanto en Windows como Linux y Mac OS X. Te permite escoger entre un montón de objetos prediseñados como: barras de estado, menús, barras de progreso, etc. Además, te permite exportar el diseño que realices en PNG, PDF e incluso al portapapeles.

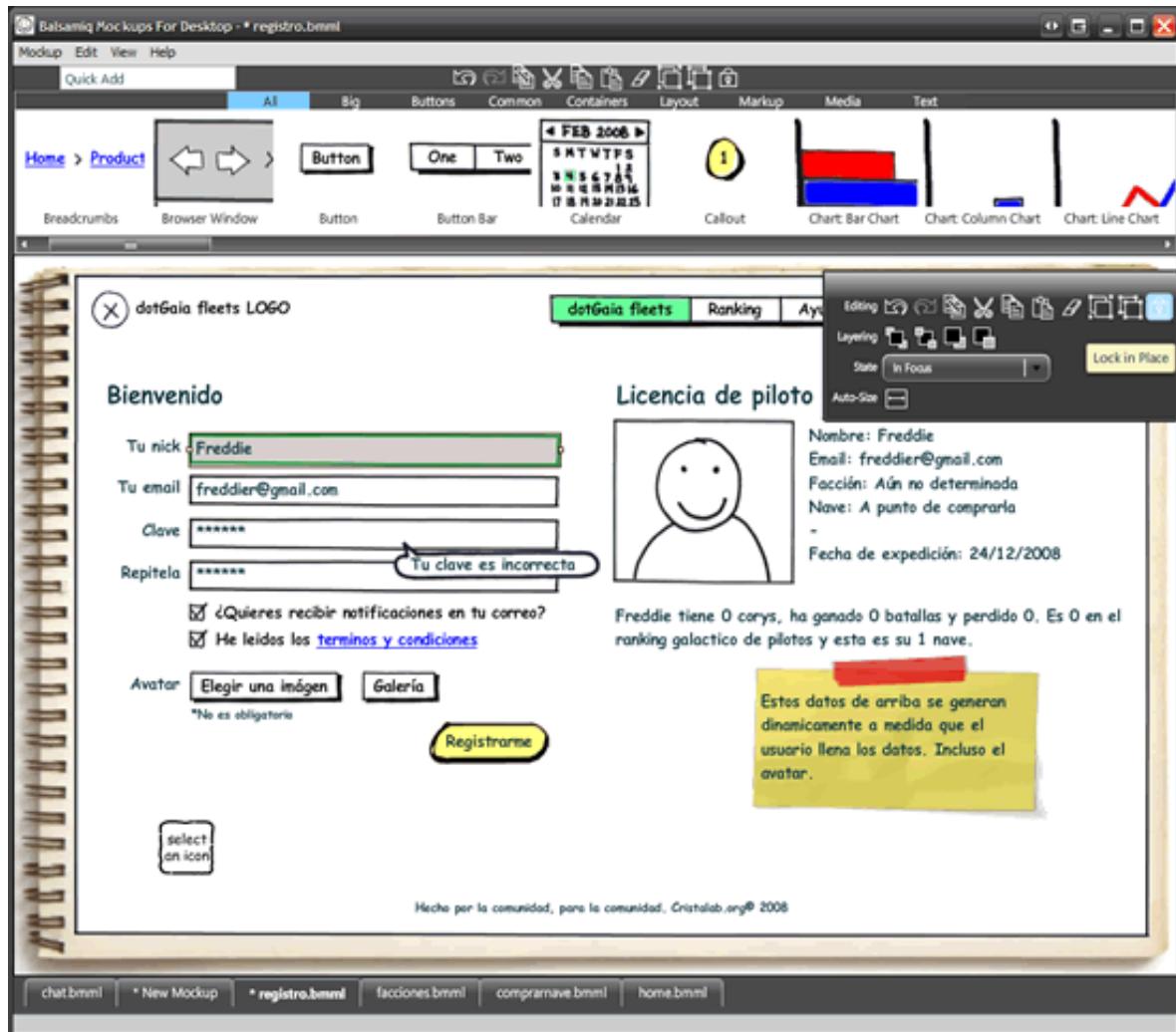


Figure 46: Balsamiq

Programas para la Base de datos

Estos son los programas enfocados al desarrollo o creación de la base de datos utilizamos el PgAdmin el cual es una aplicación para el manejo y gestaría de la base de datos, la cual no habíamos utilizado anteriormente por lo cual fue un reto implementarla en nuestro proyecto, así como el lenguaje que utilizamos es el "posgresSQL" apoyado de PGAdmin el PGAdmin en PgAdmin en su versión 4 para poder utilizar su interfaz grafica de manera más simple y fácil.

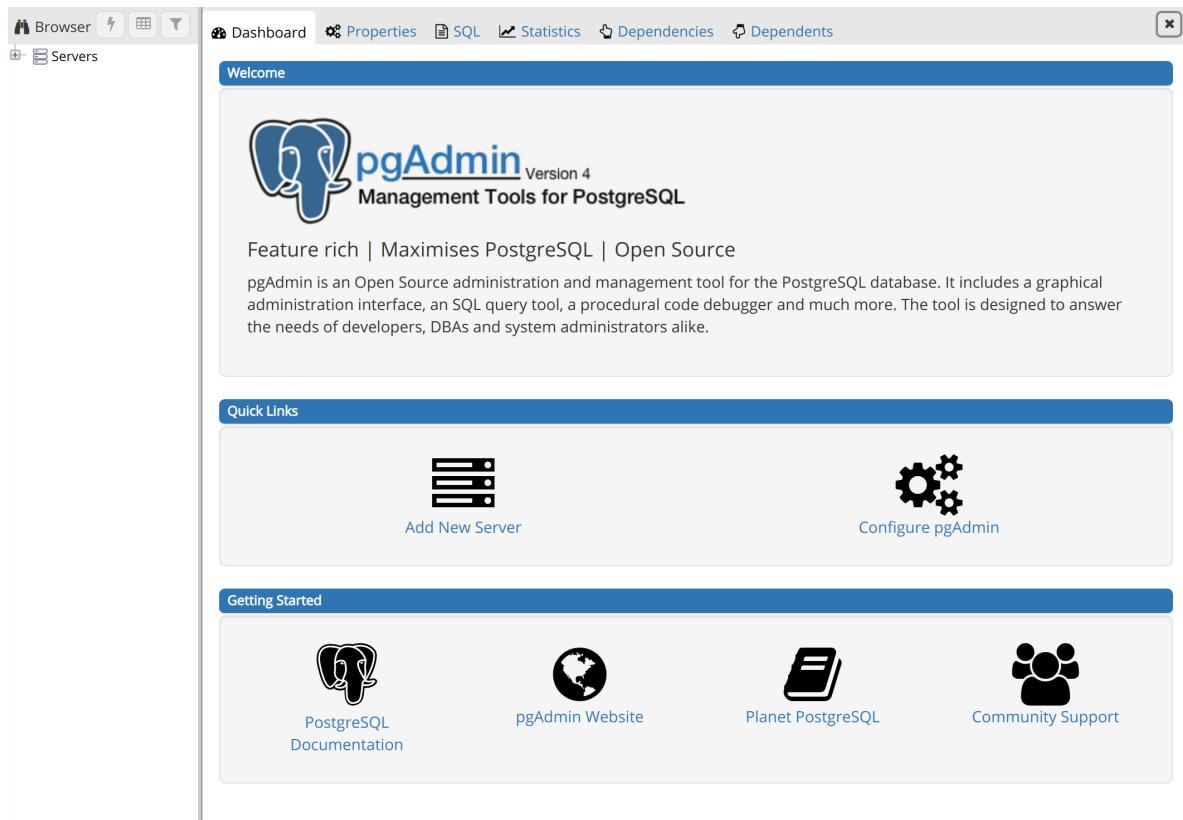


Figure 47: PgAdmin

PgAdmin es una herramienta indispensable para gestionar y administrar PostgreSQL, la base de datos de código abierto más avanzada del mundo. Por lo tanto pgAdmin es la herramienta para gestionar nuestras bases de datos espaciales PostGIS.

Estándar de codificación Java

Nomenclatura

El idioma por defecto a la hora de dar sentido funcional al nombre de clases, variables, constantes, etc. será una mezcla entre la nomenclatura tradicional en inglés y la nomenclatura funcional adoptada.

Resumiendo, aquella codificación que por estandarización y/o aceptación se pueda escribir en inglés se mantendrá así por convenio, casos como insert, update, delete, create, retrieve, list, set, get, newInstance, Delegate.

Para la parte funcional se utilizará castellano, por lo tanto la nomenclatura de los métodos será: getListEmpresa en sustitución de getListCompany o insertBanco en lugar de insertarBanco.

Paquetes

Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales. El paquete base queda definido como es.gobcantabria, en este paquete no se definirá ninguna clase.

Se tendrá, así mismo, otro nivel extra dentro del paquete definido como el nombre del proyecto o del módulo



Nombres de Interfaces

Los nombres de interfaces utilizarán el sufijo **Interface** y estarán compuestos por palabras con la primera letra en mayúscula (CamelCase). Se debe evitar el uso de abreviaciones que dificulten la comprensión del código.

Nombres de clases

Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas (CamelCase).

Debemos intentar mantener los nombres de clases simples y descriptivos.

Debemos usar palabras completas y evitar acrónimos y abreviaturas (se permiten DAO, DTO, URL, HTML, etc.).

Si la clase cumpliese algún patrón determinado o tuviese una funcionalidad específica es recomendable definirlo en el nombre.

Paquete	Funcionalidad	Nombre
bussines.dao	Data Access Object (Interface)	<nombre>DAO
bussines.dao.impl	Data Access Object (Implementation)	<nombre>DAOImpl
bussines.exception	Excepciones	<nombre>Exception
bussines.service	Service	<nombre>Service
bussines.helper	Helper	<nombre>Helper
bussines.dto	Data Transfer Objects	<nombre>DTO
util	Clases de constantes.	<scope>Keys <nombre>Keys
web.controller	Controller	<nombre>Controller
web.filter	Filter	<nombre>Filter
web.model	Model	<nombre>Model
web.listener	Listener	<nombre>Listener

Nombres específicos de gestiones

Cuando se trata de gestionar una entidad determinada (Ej. Usuario) se definen los nombres de clases, demás ficheros implicados con la siguientes reglas:

Clase: <<FuncionalidadGenerica>><<Entidad>><<Especificación de Clase>>

Ejemplo: Usuario

UsuarioAction, FindUsuarioAction

Métodos

Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (lowerCamelCase).

No se permiten caracteres especiales.

Variables

Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos, con la salvedad de que aquí sí importa más la relación entre la regla mnemónica y la longitud del nombre.

El primer bucle siempre será el que tenga la variable i como iterador. (Esta variable se definirá para el bucle en cuestión).

Constantes

Los nombres de constantes de clases deberían escribirse todo en mayúsculas con las palabras separadas por subrayados ("_"). Todas serán declaradas como public static final

```
public static final String PROPERTY_URL_SERVICIO = "urlServicio";
```

Estilo de codificación

Comentarios

Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se tiene, por tanto, que evitar referencias al diseño funcional de la misma.

El uso abusivo de los comentarios es desaconsejable, principalmente por el trabajo extra necesario para su correcto mantenimiento. Es preferible rediseñar el código para una mejor compresión del mismo.

Estándar de Codificación JEE 13ESTÁNDAR DE CODIFICACIÓN JEE

Se tienen que evitar el uso de caracteres especiales dentro de los comentarios así como el uso de cajas u otro tipo de gráfico creado mediante códigos ASCII.

La estructura de los diferentes tipos de comentarios y su uso general se presenta en la clase base de codificación.

Son obligatorios los siguientes comentarios de documentación:

Comentario de la clase / interface:

- Prescripción genérica de la clase y su responsabilidad.
- Autor.

Todas las variables tipo private o protected han de ser obligatoriamente comentadas.

Reglas generales a la hora de escribir comentarios de documentación.

- Siempre se escribe en tercera persona.
- Los caracteres especiales tales como tildes y eñes se han de codificar con su código HTML correspondiente.
- Las descripciones siempre deberían empezar por un verbo.

Los demás tags permitidos (@since, @serial, etc..) el uso es menos común y por lo tanto no se define una manera de utilizarlos.

Usar el atributo <code> para las palabras reservadas de java, nombres de clases, métodos, interfaces, propiedades, argumentos y ejemplos de código.

El uso del atributo @link tendrá que ser mínimo, para evitar llenar el documento de enlaces. Como norma general solo se incluirán links cuando la referencia sea necesaria y sólo en la primera aparición de la misma.

Declaraciones

Para la declaración de las variables se utiliza una declaración de cada vez y no se permiten dejar variables locales sin inicializar salvo en el caso de que sean propiedades de un objeto bean.

La codificación correcta sería:

```
public static Integer entero = new Integer(0);
```

La declaración de las variables locales a una clase, método o bloque de código se realizan al principio del mismo y no justo antes de necesitarse la utilización de la variable.

La única excepción a esta regla son las variables que gestionan los bucles for.

Las variables de avance de bucles for no podrán ser modificadas de ninguna manera fuera de la propia sentencia del bucle.

La duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase se tiene que evitar.

Sentencias

Normas básicas son :

- Una sentencia por línea de código.
- Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un if.

La declaración de los bucles for serán usualmente de la forma :

```
for (int i = 0; i < condicion ; i++)
```

Son obligatorias las tres condiciones del bucle for: inicialización, condición de finalización y actualización del valor de la variable de avance.

La variable de avance del bucle nunca podrá ser modificada dentro del propio bucle.

Buenas prácticas

Constantes

Como norma general todas las constantes numéricas no deberían codificarse directamente, salvo la excepción de -1, 0 y 1.

Propiedades

El acceso/modificación de las propiedades de una clase (no constantes) siempre mediante métodos de acceso get/set.

La asignación de variables / propiedades no podrá ser consecutiva.

Variable1 = variable2 = "hola mundo"	No válido
--------------------------------------	-----------

No utilizar el operador asignación en sitios donde se pueda confundir con el operador igualdad. Ni dentro de expresiones complejas.

Métodos

Como norma general no se debe acceder a un método estático desde una instancia de una clase, debemos utilizar la clase en sí misma.

5.4 Diseño de Datos

5.4.1 Modelo Conceptual

En la siguiente imagen se muestra el modelo conceptual del sistema de ventas

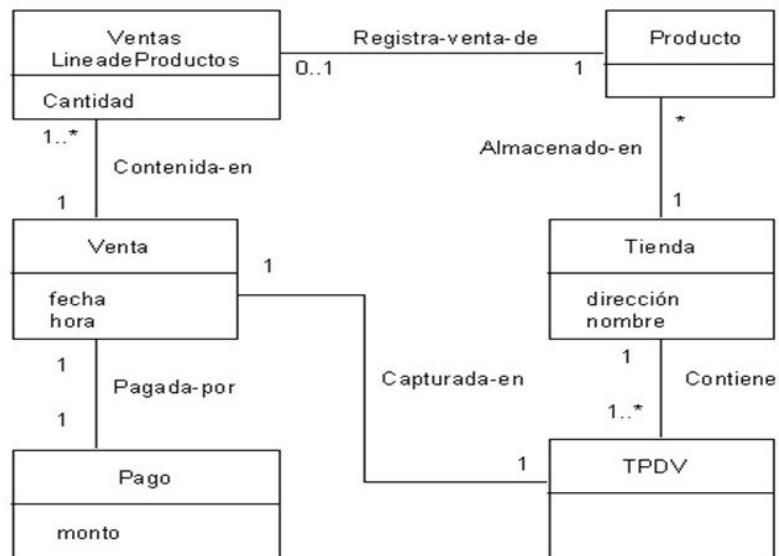


Figure 48: Modelo conceptual del sistema

5.4.2 Diagrama entidad de la base de datos

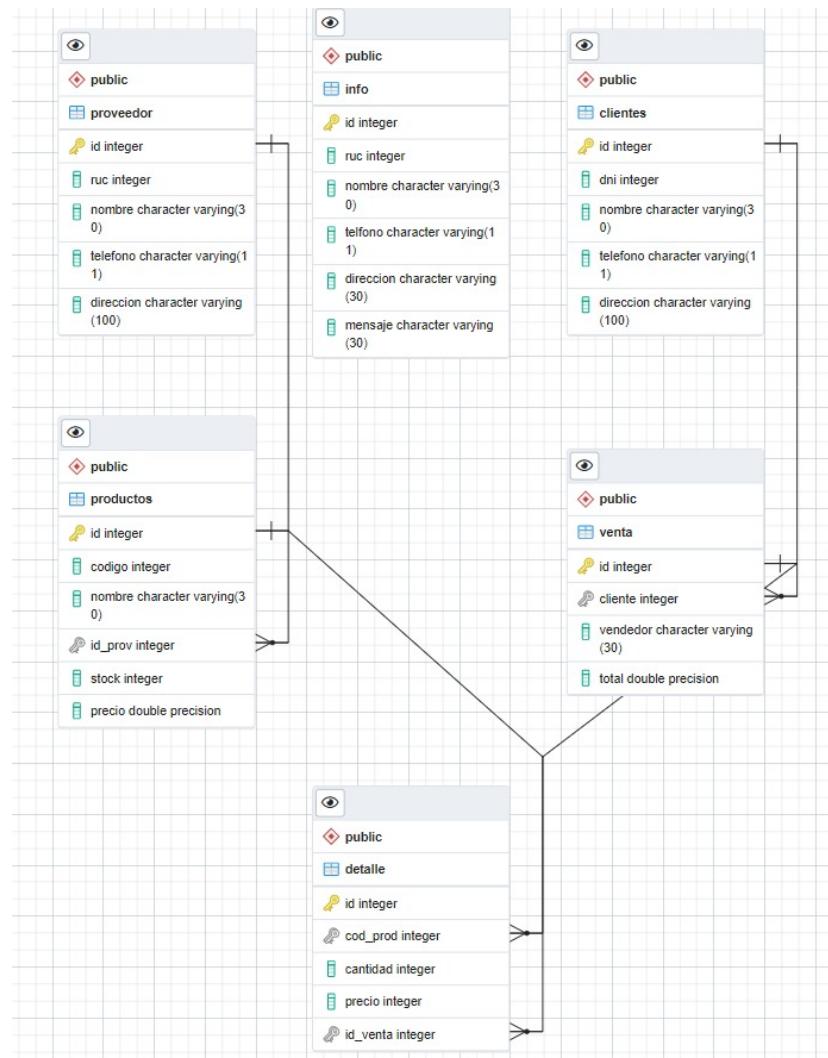


Figure 49: Base de datos

5.4.3 Script de creación de la base de datos

Creacion de la base de datos y de las tablas

```
create database Ventas;

create table clientes (
    id serial primary key,
    dni integer,
    nombre varchar(30) ,
    telefono varchar(11),
    direccion varchar(100)
);

create table proveedor(
    id serial primary key,
    ruc integer,
    nombre varchar(30),
    telefono varchar(11),
    direccion varchar(100)
);

create table productos(
    id serial primary key,
    codigo integer,
    nombre varchar(30),
    id_prov integer,
    stock integer,
    precio double precision ,
    foreign key (id_prov) references proveedor(id) on delete no action on update cascade
);

create table venta(
    id serial primary key,
    cliente integer,
    vendedor varchar(30),
    total double precision,
    foreign key(cliente) references clientes(id) on delete no action on update cascade
);

create table detalle(
    id serial primary key,
    cod_prod integer,
    cantidad integer,
    precio integer,
    id_venta integer,
    foreign key (cod_prod) references productos(id) on delete no action on update cascade,
    foreign key (id_venta) references venta(id) on delete no action on update cascade
);

create table info(
```

```
id serial primary key,  
ruc integer,  
nombre varchar(30),  
telefono varchar(11),  
direccion varchar(30),  
mensaje varchar(30)  
);
```

Creacion de roles y permisos

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC
```

```
Revocando privilegios del rol 'public'
```

```
REVOKE ALL ON DATABASE ventas FROM PUBLIC
```

"Rol solo lectura, en este rol solo esta permitido las consultas que se realizaran en el Programa permitiendo solo el acceso a la información de las tablas mas no la modificaion de estas mismas.

Este rol esta diseñado para los vendedores de una tienda los cuales solo necesitan realizar consultas para conocer los datos de los clientes o de algunos productos, por lo cual los permisos de estos mismos son minimos para le manejo de la base de datos".

```
CREATE ROLE Vendedor
```

```
GRANT CONNECT ON DATABASE venta TO Vendedor
```

```
GRANT USAGE ON SCHEMA ventas TO Vendedor
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA ventas TO Vendedor
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA ventas GRANT SELECT ON TABLES TO Vendedor
```

"Rol lectura-escritura, en este rol se permite la modificación de las tablas y de su contenido asi como la manipulación de la información que contiene

cada tabla, permitiendo eliminar, modificar, insertar el contenido en las tablas o también eliminar las tablas".

"Este rol esta diseñado para los o el administrador de la tienda el cual por el contrario del vendedor necesita realizar varias acciones en la base de datos como modificar la cantidad de productos en existencia el modificar datos de los clientes si es necesario o eliminar algunas cosas que sean necesarias".

```
CREATE ROLE Administrador
GRANT CONNECT ON DATABASE venta TO Administrador
GRANT USAGE, CREATE ON SCHEMA ventas TO Administrador
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA ventas TO Administrador
ALTER DEFAULT PRIVILEGES IN SCHEMA ventas GRANT SELECT, INSERT, UPDATE,
DELETE ON TABLES TO Administrador
GRANT USAGE ON ALL SEQUENCES IN SCHEMA Ventas TO Administrador
ALTER DEFAULT PRIVILEGES IN SCHEMA ventas GRANT USAGE ON SEQUENCES TO Administrador
```

"Se crean los usuarios con sus respectivas contraseñas"

```
CREATE USER Vendedores WITH PASSWORD 'Ceti123'
CREATE USER Administradores WITH PASSWORD 'UNYSIS123'
```

"Se Concede el permiso correspondiente a cada uno de los usuarios
Para el usuario Vendedor se asignan los privilegios de Vendedor"

```
GRANT Vendedor TO vendedores
```

"Para el usuario Administrador se le asignan los privilegios de Administradores"

GRANT Administrador TO Administradores.

5.4.4 Diccionario de datos

En la siguiente imagen se muestra el diccionario de datos de la tabla clientes

CLIENTE			
Campo	Tipo	Tamaño	Descripción
ID_C	Int	11	Primary key
Nombre	String	50	Nombre del cliente
Apellido	String	50	Apellido del cliente
Documento	Int	11	Documento de identidad del cliente
Teléfono	Int	11	Teléfono del cliente
Relaciones: ID_C con ID_VE		Campos claves: ID_C	

Figure 50: Diccionario de la tabla clientes

En la siguiente imagen se muestra el diccionario de datos de la tabla Vendedor

VENDEDOR			
Campo	Tipo	Tamaño	Descripción
ID_V	Int	11	Primary key
Nombre	String	50	Nombre del vendedor
Apellido	String	50	Apellido del vendedor
Documento	Int	11	Documento de identidad del vendedor
Teléfono	Int	11	Teléfono del vendedor
Relaciones: ID_V con ID_VE		Campos claves: ID_V	

Figure 51: Diccionario de la tabla Vendedor

En la siguiente imagen se muestra el diccionario de datos de la tabla ventas del programa

VENTAS			
Campo	Tipo	Tamaño	Descripción
ID_VE	Int	11	Primary key
FK_ID_V	Int	11	Foreing Key
FK_ID_DV	Int	11	Foreing Key
FK_ID_C	Int	11	Foreing Key

Relaciones: ID_VE con ID_C/**ID_VE con ID_V/ID_VE con ID_DV** **Campos claves:** ID_VE

Figure 52: Diccionario de la tabla ventas

En la siguiente imagen se muestra el diccionario de datos de la tabla detalle de las ventas del respectivo programa

DETALLE DE VENTA			
Campo	Tipo	Tamaño	Descripción
ID_DV	Int	11	Primary key
FK_ID_P	Int	11	Foreing Key
Fecha	Date	10	Fecha de la compra
Cantidad	Int	5	Cantidad del producto comprado
SubTotal	Double	11	SubTotal de la compra
IVA	Double	3	IVA del producto
Total	Double	11	Total de la compra
Descripción	String	300	Descripción del producto

Relaciones: ID_DV con ID_VE **Campos claves:** ID_DV

Figure 53: Diccionario de la tabla detalle

6 Código del proyecto

6.1 clase conexion

se muestra la clase conexion con la cual accedemos a la base de datos para poder realizar las acciones correspondientes del programa.

```
public class Conexion {  
    String pass="Shiro970916";  
    String user="postgres";  
    String host="localhost";  
    Connection con;  
  
    public Connection getConnection() {  
        try {  
            String access = "jdbc:ucanaccess:/D:/ventas.accdb";  
            String myBD = "jdbc:mysql://localhost:3306/venta";  
            con = DriverManager.getConnection(myBD, "root", "");  
            Class.forName("org.postgresql.Driver");  
            con= DriverManager.getConnection("jdbc:postgresql:"+host+  
                "ventas", user ,pass);  
            System.out.println("conexion establecida");  
            return con;  
        } catch (Exception e) {  
            System.out.println(e.toString());  
        }  
        return null;  
    }  
}
```

6.2 clase Clientes

se muestra la clase clientes en la cual declaramos sus respectivas variables para y las operaciones que se realizara con los clientes dentro del programa.

```
public class ClienteDao {  
  
    Conexion cn = new Conexion();  
    Connection con;  
    PreparedStatement ps;  
    ResultSet rs;  
  
    public boolean RegistrarCliente(Cliente c1){  
        String sql = "INSERT INTO clientes (dni , nombre ,  
        telefono , direccion)  
        VALUES(?, ?, ?, ?)";  
        try {  
            con = cn.getConnection();  
            ps = con.prepareStatement(sql);  
            ps.setString(1, c1.getDni());  
            ps.setString(2, c1.getNombre());  
        }  
    }  
}
```

```

        ps.setString(3, cl.getTelefono());
        ps.setString(4, cl.getDireccion());
        ps.execute();
        return true;
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e.toString());
        return false;
    } finally{
        try {
            con.close();
        } catch (SQLException e) {
            System.out.println(e.toString());
        }
    }
}

public List ListarCliente(){
    List<Cliente> ListaCl = new ArrayList();
    String sql = "SELECT * FROM clientes";
    try {
        con = cn.getConnection();
        ps = con.prepareStatement(sql);
        rs = ps.executeQuery();
        while (rs.next()) {
            Cliente cl = new Cliente();
            cl.setId(rs.getInt("id"));
            cl.setDni(rs.getString("dni"));
            cl.setNombre(rs.getString("nombre"));
            cl.setTelefono(rs.getString("telefono"));
            cl.setDireccion(rs.getString("direccion"));
            ListaCl.add(cl);
        }
    } catch (SQLException e) {
        System.out.println(e.toString());
    }
    return ListaCl;
}

public boolean EliminarCliente(int id){
    String sql = "DELETE FROM clientes WHERE id = ?" ;
    try {
        ps = con.prepareStatement(sql);
        ps.setInt(1, id);
        ps.execute();
        return true;
    } catch (SQLException e) {
        System.out.println(e.toString());
        return false;
    } finally{

```

```

        try {
            con . close ();
        } catch (SQLException ex) {
            System . out . println (ex . toString ());
        }
    }

    public boolean ModificarCliente (Cliente cl){
        String sql = "UPDATE clientes SET dni=? , nombre=? , telefono=? ,
-----direccion=? WHERE id=?";
        try {
            ps = con . prepareStatement (sql);
            ps . setString (1, cl . getDni ());
            ps . setString (2, cl . getNombre ());
            ps . setString (3, cl . getTelefono ());
            ps . setString (4, cl . getDireccion ());
            ps . setInt (5, cl . getId ());
            ps . execute ();
            return true;
        } catch (SQLException e) {
            System . out . println (e . toString ());
            return false;
        } finally {
            try {
                con . close ();
            } catch (SQLException e) {
                System . out . println (e . toString ());
            }
        }
    }

    public Cliente Buscarcliente (int dni){
        Cliente cl = new Cliente ();
        String sql = "SELECT * FROM clientes WHERE dni = ? ";
        try {
            con = cn . getConnection ();
            ps = con . prepareStatement (sql);
            ps . setInt (1, dni);
            rs = ps . executeQuery ();
            if (rs . next ()) {
                cl . setId (rs . getInt ("id"));
                cl . setNombre (rs . getString ("nombre"));
                cl . setTelefono (rs . getString ("telefono"));
                cl . setDireccion (rs . getString ("direccion"));
            }
        } catch (SQLException e) {
            System . out . println (e . toString ());
        }
    }
}

```

```

    return cl;
}
}

```

6.3 clase Productos

se muestra la clase productos en la cual declaramos sus respectivas variables para y las operaciones que se realizara con los clientes dentro del programa.

```

public class ProductosDao {
    Connection con;
    Conexion cn = new Conexion();
    PreparedStatement ps;
    ResultSet rs;

    public boolean RegistrarProductos(Productos pro){
        String sql = "INSERT INTO productos_(codigo ,_nombre ,_proveedor ,
        _stock ,_precio )_VALUES_(? ,? ,? ,? ,?)";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(sql);
            ps.setString(1, pro.getCodigo());
            ps.setString(2, pro.getNombre());
            ps.setInt(3, pro.getProveedor());
            ps.setInt(4, pro.getStock());
            ps.setDouble(5, pro.getPrecio());
            ps.execute();
            return true;
        } catch (SQLException e) {
            System.out.println(e.toString());
            return false;
        }
    }

    public List ListarProductos(){
        List<Productos> Listapro = new ArrayList();
        String sql = "SELECT.pr .id _AS_id_proveedor ,_pr .nombre _AS
        nombre_proveedor ,_p.* _FROM_proveedor _pr _INNER_JOIN_productos
        _p _ON_pr .id _=_p .proveedor _ORDER_BY_p . id _DESC";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(sql );
            rs = ps.executeQuery();
            while ( rs.next() ) {
                Productos pro = new Productos();
                pro.setId(rs.getInt("id"));
                pro.setCodigo(rs.getString("codigo"));
                pro.setNombre(rs.getString("nombre"));
                pro.setProveedor(rs.getInt("id_proveedor"));

```

```

        pro.setProveedorPro(rs.getString("nombre_proveedor"));
        pro.setStock(rs.getInt("stock"));
        pro.setPrecio(rs.getDouble("precio"));
        Listapro.add(pro);
    }
} catch (SQLException e) {
    System.out.println(e.toString());
}
return Listapro;
}

public boolean EliminarProductos(int id){
    String sql = "DELETE FROM productos WHERE id = ?" ;
    try {
        ps = con.prepareStatement(sql);
        ps.setInt(1, id);
        ps.execute();
        return true;
    } catch (SQLException e) {
        System.out.println(e.toString());
        return false;
    } finally{
        try {
            con.close();
        } catch (SQLException ex) {
            System.out.println(ex.toString());
        }
    }
}

public boolean ModificarProductos(Productos pro){
    String sql = "UPDATE productos SET codigo=?, nombre=?, proveedor=?,
stock=?, precio=? WHERE id=?";
    try {
        ps = con.prepareStatement(sql);
        ps.setString(1, pro.getCodigo());
        ps.setString(2, pro.getNombre());
        ps.setInt(3, pro.getProveedor());
        ps.setInt(4, pro.getStock());
        ps.setDouble(5, pro.getPrecio());
        ps.setInt(6, pro.getId());
        ps.execute();
        return true;
    } catch (SQLException e) {
        System.out.println(e.toString());
        return false;
    } finally{
        try {
            con.close();
        }
    }
}

```

```

        } catch (SQLException e) {
            System.out.println(e.toString());
        }
    }

public Productos BuscarPro(String cod){
    Productos producto = new Productos();
    String sql = "SELECT * FROM productos WHERE codigo = ?";
    try {
        con = cn.getConnection();
        ps = con.prepareStatement(sql);
        ps.setString(1, cod);
        rs = ps.executeQuery();
        if (rs.next()) {
            producto.setId(rs.getInt("id"));
            producto.setNombre(rs.getString("nombre"));
            producto.setPrecio(rs.getDouble("precio"));
            producto.setStock(rs.getInt("stock"));
        }
    } catch (SQLException e) {
        System.out.println(e.toString());
    }
    return producto;
}

public Productos BuscarId(int id){
    Productos pro = new Productos();
    String sql = "SELECT pr.id AS id_proveedor , pr.nombre AS nombre_proveedor , p.* FROM proveedor pr INNER JOIN productos p ON p.proveedor = pr.id WHERE p.id = ?";
    try {
        con = cn.getConnection();
        ps = con.prepareStatement(sql);
        ps.setInt(1, id);
        rs = ps.executeQuery();
        if (rs.next()) {
            pro.setId(rs.getInt("id"));
            pro.setCodigo(rs.getString("codigo"));
            pro.setNombre(rs.getString("nombre"));
            pro.setProveedor(rs.getInt("proveedor"));
            pro.setProveedorPro(rs.getString("nombre_proveedor"));
            pro.setStock(rs.getInt("stock"));
            pro.setPrecio(rs.getDouble("precio"));
        }
    } catch (SQLException e) {
        System.out.println(e.toString());
    }
    return pro;
}

```

```

public Proveedor BuscarProveedor(String nombre){
    Proveedor pr = new Proveedor();
    String sql = "SELECT * FROM proveedor WHERE nombre = ?";
    try {
        con = cn.getConnection();
        ps = con.prepareStatement(sql);
        ps.setString(1, nombre);
        rs = ps.executeQuery();
        if (rs.next()) {
            pr.setId(rs.getInt("id"));
        }
    } catch (SQLException e) {
        System.out.println(e.toString());
    }
    return pr;
}
public Config BuscarDatos(){
    Config conf = new Config();
    String sql = "SELECT * FROM config";
    try {
        con = cn.getConnection();
        ps = con.prepareStatement(sql);
        rs = ps.executeQuery();
        if (rs.next()) {
            conf.setId(rs.getInt("id"));
            conf.setRuc(rs.getString("ruc"));
            conf.setNombre(rs.getString("nombre"));
            conf.setTelefono(rs.getString("telefono"));
            conf.setDireccion(rs.getString("direccion"));
            conf.setMensaje(rs.getString("mensaje"));
        }
    } catch (SQLException e) {
        System.out.println(e.toString());
    }
    return conf;
}

public boolean ModificarDatos(Config conf){
    String sql = "UPDATE config SET ruc=?, nombre=?, telefono=?, "
    "direccion=?, mensaje=? WHERE id=?";
    try {
        ps = con.prepareStatement(sql);
        ps.setString(1, conf.getRuc());
        ps.setString(2, conf.getNombre());
        ps.setString(3, conf.getTelefono());
        ps.setString(4, conf.getDireccion());
        ps.setString(5, conf.getMensaje());
        ps.setInt(6, conf.getId());
        ps.execute();
    }

```

```

        return true;
    } catch (SQLException e) {
        System.out.println(e.toString());
        return false;
    } finally{
        try {
            con.close();
        } catch (SQLException e) {
            System.out.println(e.toString());
        }
    }
}

```

6.4 clase Proveedores

se muestra la clase proveedores en la cual declaramos sus respectivas variables para y las operaciones que se realizara con los clientes dentro del programa.

```

public class ProveedorDao {
    Connection con;
    Conexion cn = new Conexion();
    PreparedStatement ps;
    ResultSet rs;
    public boolean RegistrarProveedor(Proveedor pr){
        String sql = "INSERT INTO proveedor(ruc , nombre , telefono ,
-----direccion ) VALUES(?, ?, ?, ?)";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(sql);
            ps.setString(1, pr.getRuc());
            ps.setString(2, pr.getNombre());
            ps.setString(3, pr.getTelefono());
            ps.setString(4, pr.getDireccion());
            ps.execute();
            return true;
        } catch (SQLException e) {
            System.out.println(e.toString());
            return false;
        } finally{
            try {
                con.close();
            } catch (SQLException e) {
                System.out.println(e.toString());
            }
        }
    }

    public List ListarProveedor(){
        List<Proveedor> Listapr = new ArrayList();
        String sql = "SELECT * FROM proveedor";

```

```

try {
    con = cn.getConnection();
    ps = con.prepareStatement(sql);
    rs = ps.executeQuery();
    while (rs.next()) {
        Proveedor pr = new Proveedor();
        pr.setId(rs.getInt("id"));
        pr.setRuc(rs.getString("ruc"));
        pr.setNombre(rs.getString("nombre"));
        pr.setTelefono(rs.getString("telefono"));
        pr.setDireccion(rs.getString("direccion"));
        Listapr.add(pr);
    }
} catch (SQLException e) {
    System.out.println(e.toString());
}
return Listapr;
}

public boolean EliminarProveedor(int id){
    String sql = "DELETE FROM proveedor WHERE id = ? ";
    try {
        con = cn.getConnection();
        ps = con.prepareStatement(sql);
        ps.setInt(1, id);
        ps.execute();
        return true;
    } catch (SQLException e) {
        System.out.println(e.toString());
        return false;
    } finally{
        try {
            con.close();
        } catch (SQLException e) {
            System.out.println(e.toString());
        }
    }
}

public boolean ModificarProveedor(Proveedor pr){
    String sql = "UPDATE proveedor SET ruc=?, nombre=?,
    telefono=?, direccion=? WHERE id=?";
    try {
        con = cn.getConnection();
        ps = con.prepareStatement(sql);
        ps.setString(1, pr.getRuc());
        ps.setString(2, pr.getNombre());
        ps.setString(3, pr.getTelefono());
    }
}

```

```

        ps.setString(4, pr.getDireccion());
        ps.setInt(5, pr.getId());
        ps.execute();
        return true;
    } catch (SQLException e) {
        System.out.println(e.toString());
        return false;
    } finally{
        try {
            con.close();
        } catch (SQLException e) {
            System.out.println(e.toString());
        }
    }
}
}

```

6.5 Login

se muestra la el Login donde se valida el ingreso del usuario para asi poder accedes a las respectivas ventanas.

```

public class Login extends javax.swing.JFrame {
    login lg = new login();
    LoginDAO login = new LoginDAO();
    public Login() {
        initComponents();
        this.setLocationRelativeTo(null);
        txtCorreo.setText("losf.97@hotmail.com");
        txtPass.setText("fernando");
    }
    public void validar(){
        String correo = txtCorreo.getText();
        String pass = String.valueOf(txtPass.getPassword());
        if (!"".equals(correo) || !"".equals(pass)) {

            lg = login.log(correo, pass);
            if (lg.getCorreo()!= null && lg.getPass() != null) {
                Sistema sis = new Sistema(lg);
                sis.setVisible(true);
                dispose();
            } else{
                JOptionPane.showMessageDialog(null,
                    "Correo_o_la_Contrase_a_incorrecta");
            }
        }
    }
}

```

6.6 Principal

se muestra el código de la ventana principal donde se muestran todas las instancias que se realizan en el programa.

```
public final class Sistema extends javax.swing.JFrame {
    Date fechaVenta = new Date();
    String fechaActual = new SimpleDateFormat("dd/MM/yyyy").
format(fechaVenta);
    Cliente cl = new Cliente();
    ClienteDao client = new ClienteDao();
    Proveedor pr = new Proveedor();
    ProveedorDao PrDao = new ProveedorDao();
    Productos pro = new Productos();
    ProductosDao proDao = new ProductosDao();
    Venta v = new Venta();
    VentaDao Vdao = new VentaDao();
    Detalle Dv = new Detalle();
    Config conf = new Config();
    Eventos event = new Eventos();
    login lg = new login();
    LoginDAO login = new LoginDAO();
    DefaultTableModel modelo = new DefaultTableModel();
    DefaultTableModel tmp = new DefaultTableModel();
    int item;
    double Totalpagar = 0.00;

    public Sistema() {
        initComponents();
    }
    public Sistema (login priv){
        initComponents();
        this.setLocationRelativeTo(null);
        Midate.setDate(fechaVenta);
        txtIdCliente.setVisible(false);
        txtIdVenta.setVisible(false);
        txtIdPro.setVisible(false);
        txtIdproducto.setVisible(false);
        txtIdProveedor.setVisible(false);
        txtIdConfig.setVisible(false);
        txtIdCV.setVisible(false);
        ListarConfig ();
        if (priv.getRol().equals("Asistente")) {
            btnProductos.setEnabled(false);
            btnProveedor.setEnabled(false);
            LabelVendedor.setText(priv.getNombre());
        }else{
            LabelVendedor.setText(priv.getNombre());
        }
    }
}
```

6.7 función para listar los clientes

se muestra el código de la función para listar a los clientes de la base de datos en su respectiva tabla.

```
public void ListarCliente() {
    List<Cliente> ListarCl = client.ListarCliente();
    modelo = (DefaultTableModel) TableCliente.getModel();
    Object[] ob = new Object[6];
    for (int i = 0; i < ListarCl.size(); i++) {
        ob[0] = ListarCl.get(i).getId();
        ob[1] = ListarCl.get(i).getDni();
        ob[2] = ListarCl.get(i).getNombre();
        ob[3] = ListarCl.get(i).getTelefono();
        ob[4] = ListarCl.get(i).getDireccion();
        modelo.addRow(ob);
    }
    TableCliente.setModel(modelo);
}
```

6.8 función para listar los proveedores

se muestra el código de la función para listar a los proveedores de la base de datos en su respectiva tabla.

```
public void ListarProveedor() {
    List<Proveedor> ListarPr = PrDao.ListarProveedor();
    modelo = (DefaultTableModel) TableProveedor.getModel();
    Object[] ob = new Object[5];
    for (int i = 0; i < ListarPr.size(); i++) {
        ob[0] = ListarPr.get(i).getId();
        ob[1] = ListarPr.get(i).getRuc();
        ob[2] = ListarPr.get(i).getNombre();
        ob[3] = ListarPr.get(i).getTelefono();
        ob[4] = ListarPr.get(i).getDireccion();
        modelo.addRow(ob);
    }
    TableProveedor.setModel(modelo);
}
```

6.9 función para listar los usuarios

se muestra el código de la función para listar a los usuarios de la base de datos en su respectiva tabla.

```
public void ListarUsuarios() {
    List<login> Listar = login.ListarUsuarios();
    modelo = (DefaultTableModel) TableUsuarios.getModel();
    Object[] ob = new Object[4];
```

```

        for (int i = 0; i < Listar.size(); i++) {
            ob[0] = Listar.get(i).getId();
            ob[1] = Listar.get(i).getNombre();
            ob[2] = Listar.get(i).getCorreo();
            ob[3] = Listar.get(i).getRol();
            modelo.addRow(ob);
        }
        TableUsuarios.setModel(modelo);
    }
}

```

6.10 función para listar los productos

se muestra el código de la función para listar a los productos de la base de datos en su respectiva tabla.

```

public void ListarProductos() {
    List<Productos> ListarPro = proDao.ListarProductos();
    modelo = (DefaultTableModel) TableProducto.getModel();
    Object[] ob = new Object[6];
    for (int i = 0; i < ListarPro.size(); i++) {
        ob[0] = ListarPro.get(i).getId();
        ob[1] = ListarPro.get(i).getCodigo();
        ob[2] = ListarPro.get(i).getNombre();
        ob[3] = ListarPro.get(i).getProveedorPro();
        ob[4] = ListarPro.get(i).getStock();
        ob[5] = ListarPro.get(i).getPrecio();
        modelo.addRow(ob);
    }
    TableProducto.setModel(modelo);
}

```

6.11 función para listar limpiar las tablas del programa

se muestra el código de la función para limpiar la tabla del programa.

```

public void LimpiarTable() {
    for (int i = 0; i < modelo.getRowCount(); i++) {
        modelo.removeRow(i);
        i = i - 1;
    }
}

```

6.12 función para registrar una venta

se muestra el código de la función realizar una venta en la se guardara los campos necesarios para posteriormente identificar la venta con facilidad.

```

private void RegistrarVenta() {
    int cliente = Integer.parseInt(txtIdCV.getText());
}

```

```

String vendedor = LabelVendedor.getText();
double monto = Totalpagar;
v.setCliente(cliente);
v.setVendedor(vendedor);
v.setTotal(monto);
v.setFecha(fechaActual);
Vdao.RegistrarVenta(v);
}

```

6.13 función para registrar una venta

se muestra el código de la función realizar una venta en la se guardara los campos necesarios para posteriormente identificar la venta con facilidad.

```

private void RegistrarVenta() {
    int cliente = Integer.parseInt(txtIdCV.getText());
    String vendedor = LabelVendedor.getText();
    double monto = Totalpagar;
    v.setCliente(cliente);
    v.setVendedor(vendedor);
    v.setTotal(monto);
    v.setFecha(fechaActual);
    Vdao.RegistrarVenta(v);
}

```

6.14 función para calcular el precio

se muestra el código de la función que realiza el la suma para saber la cantidad a pagar por los productos comprados.

```

private void TotalPagar() {
    Totalpagar = 0.00;
    int numFila = TableVenta.getRowCount();
    for (int i = 0; i < numFila; i++) {
        double cal = Double.parseDouble(String.
            valueOf(TableVenta.getModel().getValueAt(i, 4)));
        Totalpagar = Totalpagar + cal;
    }
    LabelTotal.setText(String.format("%.2f", Totalpagar));
}

```

6.15 Agregar un Nuevo Cliente

se muestra el código de la función el botón para agregar un nuevo cliente a la base de datos.

```

if (!"".equals(txtDniCliente.getText()) || !"".
    equals(txtNombreCliente.getText()) || !"".
    equals(txtTelefonoCliente.getText()) || !"".
    equals(txtDireccionCliente.getText())) {
    cl.setDni(txtDniCliente.getText());
}

```

```

        cl.setNombre(txtNombreCliente.getText());
        cl.setTelefono(txtTelefonoCliente.getText());
        cl.setDireccion(txtDireccionCliente.getText());
        client.RegistrarCliente(cl);
        JOptionPane.showMessageDialog(null, "Cliente Registrado");
        LimpiarTable();
        LimpiarCliente();
        ListarCliente();
        btnEditarCliente.setEnabled(false);
        btnEliminarCliente.setEnabled(false);
        btnGuardarCliente.setEnabled(true);
    } else {
        JOptionPane.showMessageDialog(null, "Los campos estan vacios");
    }
}

```

6.16 Eliminar un Cliente

se muestra el codigo de la función el boton para eliminar un cliente a la base de datos.

```

if (!"".equals(txtIdCliente.getText())) {
    int pregunta = JOptionPane.showConfirmDialog(null,
        "Esta seguro de eliminar");
    if (pregunta == 0) {
        int id = Integer.parseInt(txtIdCliente.getText());
        client.EliminarCliente(id);
        LimpiarTable();
        LimpiarCliente();
        ListarCliente();
    }
}

```

7 Conclusiones

7.1 Conclusión

Al finalizar el proyecto queda más claro la forma de documentación de una aplicación de un sistema software, mediante el análisis y estudios de factibilidad realizados durante el tránsito del semestre, podemos destacar que se ha expuesto teóricamente los pasos a seguir para la realización de un sistema de software y documentación del mismo. Resultó ser una buena experiencia ya que no teníamos el conocimiento de cómo es el proceso de documentación y desarrollo de un sistema de software, apoyándonos de metodologías y procesos de elaboración de sistemas informáticos.

7.2 Definiciones y Abreviaturas

7.2.1 Definiciones

Software — Conjunto de programas y rutinas que permite a la computadora realizar determinadas tareas.

Ticket — Comprobante de pago, que se emite en operaciones de compra realizadas con consumidores o usuarios finales, emitido en moneda nacional.

Interfaz—En informática, se conoce como interfaz de usuario al medio que permite a una persona interactuar con un sistema de cómputo.

Script—Fragmentos de código que tienen como objetivo realizar o añadir funciones dentro de un sistema informático.

Nomenclatura—Conjunto de términos o palabras propias utilizadas en una ciencia, técnica, o especialidad, o por un autor.

ASCII—Sistema de codificación de caracteres alfanuméricos que asigna un número del 0 al 127 a cada letra.

Variable—Es un tipo de dato utilizado en el área de informática, es donde se guardan temporalmente que se utilizan en el programa. Abreviaturas

7.2.2 Abreviaturas

RF01—Requerimientos funcionales 01

RF02—Requerimientos funcionales 02

RF03—Requerimientos funcionales 03

RF04—Requerimientos funcionales 04

RF05—Requerimientos funcionales 05

RNF01—Requerimientos no funcionales 01

RNF02—Requerimientos no funcionales 02

RNF03—Requerimientos no funcionales 03

RNF04—Requerimientos no funcionales 04

7.3 Referencias

A, Elmasri Ramez. Fundamentos de sistemas de base de datos . En Fundamentos de sistemas de base de datos, 5ta edición España : Addison Wesley, 2002.

Abraham, Silberschatz. Fundamentos de base de datos. En Fundamentos de base de datos, de Silberschatz Abraham, 5ta edición España: McGraw Hill, 2006.

Ceballos Sierra, Francisco Javier. JAVA 2: Manual de usuario y tutorial. En JAVA 2: Manual de usuario y tutorial, de Francisco Javier Ceballos Sierra. México: Alfa Omega, 2006.

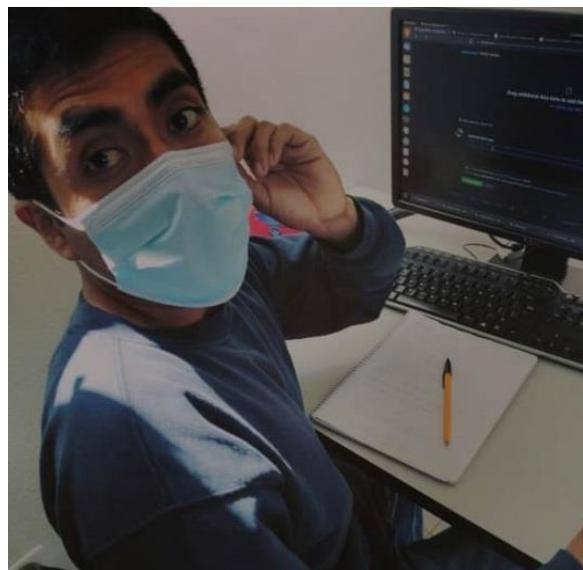
- Deitel, Harvey M. Como programar en Java. En Como programar en Java, de Harvey M. Deitel. México: Pearson educación, 2012.
- Durango, Alicia. Diseño de software. En Diseño de software, de Alicia Durango. Atenea Campus: USA, 2015. J., Date C. Introducción a los sistemas de bases de datos . En Introducción a los sistemas de bases de datos, 3ra edición México: Pearson Education, 2001.
- Joyanes Aguilar, Luis. Programación en C, C++,JAVA y UML. En Programación en C, C++, JAVA y UML, de Luis Joyanes Aguilar. España: McGraw Hill, 2010.
- Kendall, Kenneth E. Análisis y Diseño de sistemas. En Análisis y Diseño de sistemas , de Kenneth E Kendall. Pearson : México, 2005.
- Pressman, Roger S. Ingeniería de software. Un enfoque práctico. En Ingeniería de software. Un enfoque Práctico, de Roger S. Pressman. México: McGraw Hill, 2010.
- Somerville, lan. Ingeniería de software. En Ingeniería de software, de lan Somerville. México: Pearson, 2011.

Anexo

Galería de imágenes



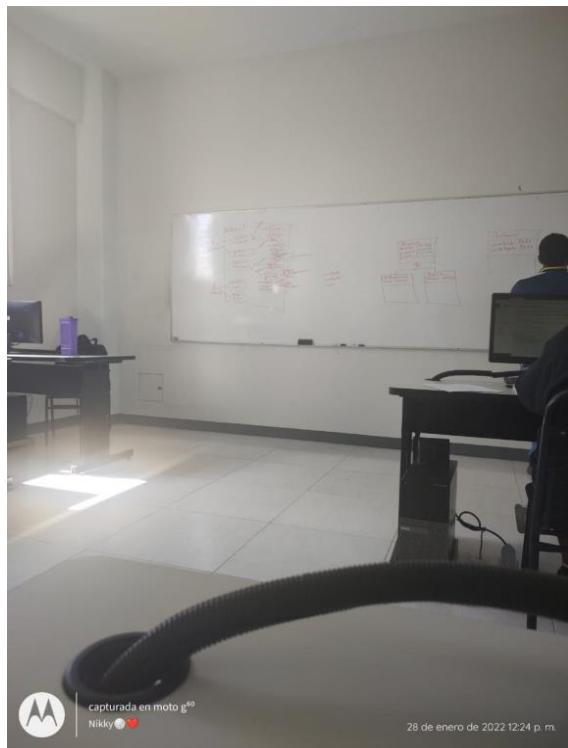
En clase de Base Datos aprendiendo bases teóricas para la seguridad de nuestro sistema de software



El compañero Luis Fernando actualizado los repositorios en GitHub



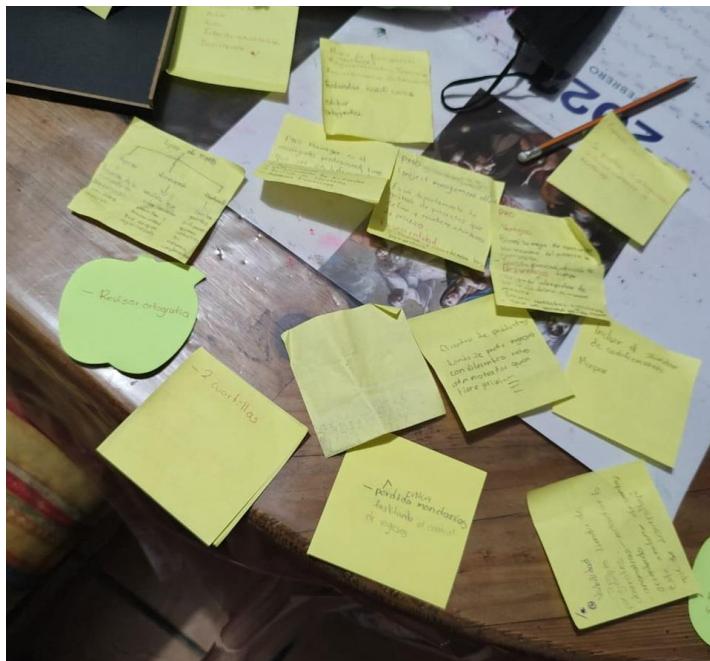
Instituto de informática después de una larga jornada de trabajo en equipo



La compañera Nancy tomado apuntes de los diagramas de clases



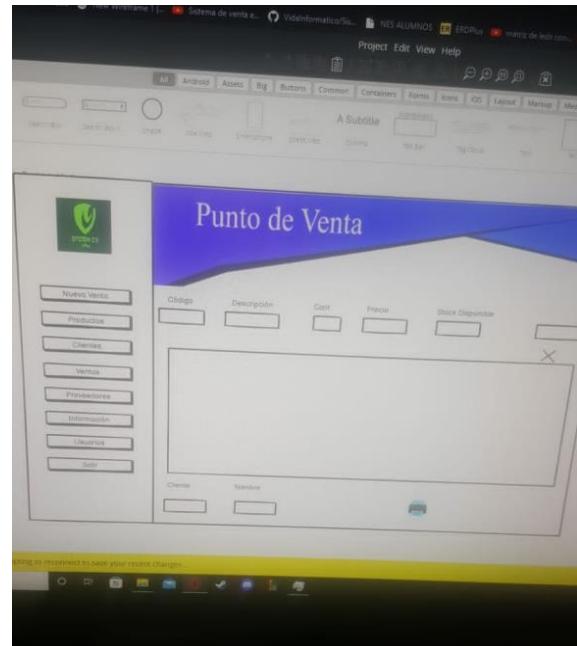
Los compañeros Nancy y Luis Fernando trabajando en el desarrollo del proyecto



Notas escritas en post stick sobre requerimientos clave para el desarrollo del sistema



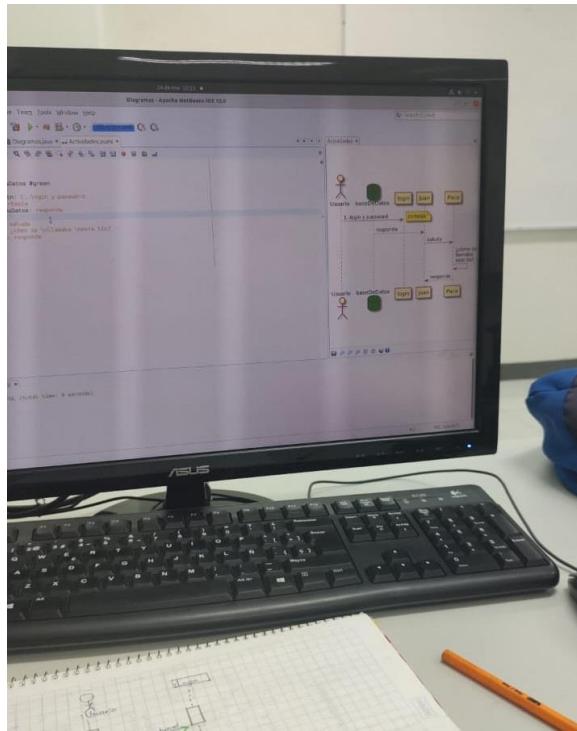
Los compañeros Luis Fernando y Nancy trabajando sin parar



Captura de pantalla sobre el desarrollo de ventanas del sistema de ventas



La compañera Nancy en una asesoría impartida por los compañeros de clase sobre el desarrollo del sistema



La compañera Nancy diseñando los diagramas de secuencia