

UNIVERSIDAD DE LA SIERRA SUR

Licenciatura en Informática

INGENIERÍA DE SOFTWARE

Integrantes del equipo:

Silva López Gamaliel

Vásquez Hernández Vicente Leonel

Vásquez Santiago German

506-A

M.T.C.A. Rolando Pedro Gabriel

Miahuatlán de Porfirio Díaz Oaxaca

Colaboradores

El siguiente trabajo fue elaborado por tres desarrolladores los cuales tienen una formación en distintos lenguajes. A continuación se describe cada uno de ellos y su formación.

Gamaliel Silva López: Tiene conocimiento con la programación de software en distintos lenguajes como: SQL, C, java, HTML, python, PHP, CSS, javascript, lenguaje ensamblador y c++.

Vicente Leonel Vásquez Hernández: Tiene un amplio conocimiento con la programación de software enfocado en los distintos lenguajes como: SQL, C, java, HTML, CSS, javascript, lenguaje ensamblador y python.

German Vasquez Santiago: Está familiarizado con la programación de software y tiene un amplio conocimiento en distintos lenguajes como: C++, C, java, HTML, CSS, javascript, lenguaje ensamblador, SQL, python y entre otras.

Índice de contenido

1	Introducción	12
2	Especificación de requerimientos	13
2.1	Propósito	13
2.2	Alcance (ámbito del proyecto)	13
2.3	Estudio de viabilidad	16
2.4	Estudio de factibilidad	16
2.5	Requisitos específicos	17
2.5.1	Requerimientos funcionales	17
2.5.2	Requerimientos no funcionales	19
2.5.3	Requerimientos de interfaz	20
2.6	Mapa de navegación	31
3	Análisis	32
3.1	Diagramas de estructura UML	32
3.1.1	Diagrama de flujo de datos	32
3.1.2	Diagrama de clases y paquetes	36
3.1.3	Diagrama de objetos	44
3.1.4	Diagrama de componentes	46
3.1.5	Diagrama de despliegue	48

3.1.6	Diagrama de actividades	50
3.2	Diagramas de comportamiento UML	63
3.2.1	Diagramas de caso de uso	63
3.2.2	Diagramas de estado	81
3.2.3	Diagramas de secuencia	101
3.2.4	Diagramas de colaboración	115
4	Diseño	116
4.1	Introducción	116
4.2	Descripción del sistema	116
4.3	Arquitectura del sistema	116
4.3.1	Física	116
4.3.2	Lógica	119
4.4	Estándar de codificación	125
4.4.1	Nombre de variable	125
4.4.2	Sangría	125
4.4.3	Comentario	125
4.4.4	Indentación	126
4.4.5	Métodos	126
4.4.6	Espacios	126
4.4.7	Encabezado	127

4.4.8	Número de caracteres por línea	128
4.4.9	Nombre de interfaces	128
4.4.10	Nombre de clases	128
4.4.11	Nombre de Objetos	129
4.4.12	Constantes	129
4.4.13	Listas	129
4.5	Diseño de datos	130
4.5.1	Diseño conceptual de la base de datos . .	130
4.5.2	Diseño lógico entidad relación	131
4.5.3	Esquema relacional	132
4.5.4	Diseño Físico	133
4.5.5	Diccionario de datos	135
5	Código fuente	136
6	Conclusiones	157
7	Definición, acrónimos y abreviaturas	158
8	Anexo	161
8.1	Evidencias fotográficas	161
9	Referencias	165

Índice de figuras

1	Requerimientos Funcional Registros de herramientas.	17
2	Requerimientos Funcional Registros de ventas.	17
3	Requerimientos Funcional Consulta de productos.	18
4	Requerimientos Funcional Consulta de ventas.	18
5	Requerimientos Funcional Registro de usuario.	18
6	Requerimientos Funcional Consulta de usuarios	19
7	Requerimientos no funcional Seguridad.	19
8	Requerimientos no funcional Mensajes de error.	19
9	Requerimientos no funcional Look and feel.	19
10	Requerimientos no funcional Restricciones de contenido.	20
11	Requerimientos no funcional Confidencialidad.	20
12	Requerimientos no funcional Robustez.	20
13	Ventana login.	21
14	Ventana principal.	22
15	Ventana ventas.	23
16	Ventana seleccionar producto.	24
17	Ventana consulta ventas.	25
18	Ventana alta de productos.	26
19	Ventana consulta de productos.	27

20	Ventana actualizar	28
21	Ventana agregar usuarios.	29
22	Ventana consulta de usuarios.	30
23	Mapa de navegación	31
24	Diagrama de flujo de datos para venta	32
25	Diagrama de flujo de datos para productos	33
26	Diagrama de flujo de datos para usuarios	34
27	Diagrama de flujo de datos para el logueo	35
28	Diagrama de clases primera parte	42
29	Diagrama de clases segunda parte parte	43
30	Diagrama de objetos	45
31	Diagrama de componentes	47
32	Diagrama de despliegue	49
33	Diagrama de actividades actualizar producto	51
34	Diagrama de actividades actualizar usuario.	52
35	Diagrama de actividades agregar productos.	53
36	Diagrama de actividades consultar productos.	55
37	Diagrama de actividades consultar usuario.	57
38	Diagrama de actividades inicio de sesion.	58
39	Diagrama de actividades alta de productos	59

40	Diagrama de actividades alta de usuarios.	60
41	Diagrama de actividades realizar venta.	62
42	Caso de uso 1 inicio de sesión.	63
43	Documentación del caso de uso 1 de inicio de sesión.	64
44	Caso de uso 2 principal.	65
45	Documentación del caso de uso 2 principal.	66
46	Caso de uso 3 realiza venta.	67
47	Documentación de caso de uso 3 realiza venta.	68
48	Caso de uso 3.1 seleccionar producto.	69
49	Documentación de caso de uso 3.1 seleccionar producto.	70
50	Caso de uso 4 consultar ventas.	71
51	Documentación de Caso de uso 4 Consultar ventas.	72
52	Caso de uso 5 alta de productos	72
53	Documentación de caso de uso 5 dar alta de productos.	73
54	Caso de uso 6 consultar productos.	73
55	Documentación de caso de uso 6 consultar productos.	74
56	Caso de uso 6.1 actualizar.	75
57	Documentación de caso de uso 6.1 actualizar.	76
58	Caso de uso 7 de agregar usuario.	77
59	Documentación de caso de uso 7 de agregar usuario.	78

60	Caso de uso 8 de consultar usuario.	79
61	Documentación de caso de uso 8 de consultar ususrio.	80
62	Diagrama de estados del login	82
63	Diagrama de estados de la interfaz principal	84
64	Diagrama de estados de la interfaz realiza venta	86
65	Diagrama de estados de la interfaz agragar producto de venta	88
66	Diagrama de estados de la interfaz consulta ventas	90
67	Diagrama de estados de la interfaz alta de productos	92
68	Diagrama de estados de la interfaz consulta productos	94
69	Diagrama de estados de la interfaz actualiza productos	96
70	Diagrama de estados de la interfaz alta usuarios	98
71	Diagrama de estados de la interfaz consulta usuarios	100
72	Diagrama de secuencia consultar venta	102
73	Diagrama de secuencia inicio de sesion	104
74	Diagrama de secuencia inventario	107
75	Diagrama de secuencia registro de usuarios	109
76	Diagrama de secuencia usuario	112
77	Diagrama de secuencia venta	114
78	Diagrama de colaboración del administrador	115
79	Diagrama de colaboración del empleado	115

80	Componentes de los equipos utilizados	117
81	Componentes de los equipos utilizados	118
82	Entorno de desarrollo	119
83	Componentes de los equipos utilizados	120
84	Software pgadmin4	120
85	Logotipo de photoshop	121
86	Logotipo de dia	122
87	Logotipo de Process Dashboard	123
88	Logotipo de Github	124
89	Formato de encabezado.	127
90	Partes de la declaración de una clase o interfaces.	128
91	Mapa entidad relación	131
92	Esquema relacional	132
93	Diccionario de datos	135
94	Diccionario de datos	135
95	Diccionario de datos	135
96	Diccionario de datos	135
97	Definición, acrónimos y abreviaturas	158
98	Definición, acrónimos y abreviaturas	159
99	Definición, acrónimos y abreviaturas	160

100	presentación proyecto	161
101	presentación proyecto	162
102	presentación proyecto	162
103	Trabajando en el proyecto de Ferrimax	163
104	Trabajando en el proyecto de Ferrimax	164

1 Introducción

La necesidad de usar un software cada vez es más recurrente en las medianas y pequeñas empresas. Surge la necesidad de automatizar información, cada negocio poco a poco opta por comprar o hacerse de un sistema que le ayude a mejorar el manejo de su información. Por eso el desarrollo de software cada vez es más importante, cabe mencionar que requiere de otras áreas para complementarse y así poder realizar un buen sistema, una de ellas es la ingeniería de software, que es la encargada de ofrecer métodos y técnicas para desarrollar y mantener software de calidad. Es aquí donde se requiere de la especificación de requisitos de software, que se refiere a la descripción completa del comportamiento del sistema que se va a desarrollar.

A continuación, se presenta el avance de la especificación de requisitos de software aplicándolo en un sistema punto de ventas de una ferretería. Esta especificación de requerimientos es creada para ser un conjunto de información que ayude a los desarrolladores del sistema a entender y analizar todos los requerimientos y requisitos que el cliente desea. Además, es un informe útil para que el cliente final describa cuales son las necesidades que requiere dentro de su negocio, para así tener una mejor visión de lo que se va a realizar.

También se describirá todas las interfaces del sistema. Se incluirán los estudios de viabilidad y factibilidad para encontrar la solución óptima del proyecto y así mismo el análisis de posibles riesgos que pueda llevar.

2 Especificación de requerimientos

2.1 Propósito

Definir y presentar de forma ordenada los requisitos y especificaciones que deberá cumplir el software a construir, el cual permitirá hacer una gestión de las actividades y emitir diferentes funciones de la ferretería. Esto consistirá en una resolución de problemas desde lo más primordial hasta las cuestiones básicas como las operaciones que se realizan al momento de generar una compra. Además, el objetivo de esta especificación es definir de manera clara y precisa las funcionalidades y restricciones que tendrá el sistema que se desea construir, y va dirigida al equipo de desarrollo de software y a las personas que harán uso del sistema.

Este documento será un medio de comunicación entre cada uno de los roles implicados en el desarrollo de software y por lo mismo está sujeto a revisiones, tanto de los desarrolladores como de los usuarios, hasta obtener su aprobación. En cuanto esto ocurra el documento funcionará como base al equipo de desarrollo para la construcción del nuevo sistema.

2.2 Alcance (ámbito del proyecto)

Con los avances de la tecnología cada vez es más necesario automatizar datos en diferentes áreas, con esto surge la necesidad de crear un sistema para tener un mejor manejo de la información. En este caso el sistema lleva por nombre FERRI-MAX, en ella se busca desarrollar e implementar una aplicación de escritorio que servirá para gestionar los datos de un punto de venta, esta se basará en una ferretería. Además, que tenga la finalidad de fortalecer su área de las tecnologías de gestión de datos.

El FERRI-MAX es un sistema de ventas que tiene la finalidad de atender las problemáticas que se presenta en las tiendas de esta área, así como el registro de pro-

ducto, sistemas de control de ventas, consultar productos, registros de usuarios. Los puntos mencionados anteriormente se presentarán en el sistema de ventas de una manera muy bien estructurada, organizada e intuitiva, con la finalidad de hacerlo entendible y fácil de manejar para los usuarios.

Gracias al desarrollo de FERRI-MAX se espera:

- Un mejoramiento en reporte de ventas.
- Un control de ingresos.
- Mejorar la seguridad del sistema.
- Un mejor servicio del sistema.
- Un mejor servicio al cliente.
- Agilizar el proceso de ventas.
- Facilitar el trabajo del administrador y del dueño del negocio.
- Tener guardada de forma segura toda la información del sistema.
- Que el sistema sea fácil de usar.
- Que los usuarios puedan acceder a sus funciones establecidas.

Por otra parte, se buscará que el sistema no cuente con errores, o con fallas a la hora de ejecutarse, todo esto a través de un buen diseño y una buena programación, así como el buen uso de la base de datos. Se deberá hacer un buen uso de las distintas herramientas que se llegarán a ocupar durante el desarrollo del sistema, la finalidad es tener un sistema eficiente, rápido y que sea de calidad.

Las utilidades de este proyecto estarán basadas conforme a las necesidades del usuario, principal-mente tendrá una función que como objetivo hará que el inventario se mantenga actualizado de forma continua. Para esto el usuario tendrá que ingresar los datos de los productos, estos podrían ser cantidad de producto, precio, entre otros datos

acerca del producto y esta función también nos ayudará a revisar cuantos productos hay en existencia o saber cuáles son las necesidades en mercancía en ese momento. Esto garantizará un ahorro de tiempo porque ya no se tendrá que revisar los productos uno por uno de manera física.

También tenemos la función de consultar productos, esta consiste en que se pueda ver las características detalladas de cada producto de una manera agradable para el usuario. Otra función es la de consultar ventas, esta mostrará una ventana emergente la cual ayudará a visualizar las ventas que se realizaron por fecha y el monto de dinero que se pagó en ese momento.

En la parte de la función de ayuda nos mostrará algunos consejos para el usuario y además de ello los colaboradores que realizaron dicho sistema de ventas. Por otra parte, tenemos la función de ventas que consiste en sacar el resultado de la cuenta que se realizó, esto consistirá en sumar la cantidad a pagar por el cliente y se le solicitará al usuario con cuánto dinero pagó el cliente esto de modo que retorne el cambio. Los registros de usuarios consistirán en crear y eliminar usuarios de esta manera se podrán generar nuevos accesos al sistema para controlar las funciones de cada usuario del sistema, ya sea el dueño o los trabajadores de la ferretería.

2.3 Estudio de viabilidad

Se considera que el proyecto es viable ya que se puede hacer un estudio de campo, en el que podemos obtener todos los requerimientos funcionales y los requerimientos no funcionales, lo que nos facilitará desarrollar prototipos antes de implementar el proyecto. para identificar las problemáticas que se requiere que solucione el software enfocándose en un punto de venta específicamente para una ferretería. El proyecto es económicamente viable, ya que el estudio nos demuestra que se cuenta con recursos económicamente y insumos que se requieren para llevar a cabo, estos insumos son proporcionados por la institución.

2.4 Estudio de factibilidad

El proyecto es factible ya que se puede desarrollar debido a que la institución nos proveerá los servicios de cómputo, software necesarios como MySQL, Java, PostgreSQL, asesoría de profesionales, además los requisitos funcionales como la luz, computadoras, internet, limpiezas, laboratorio de desarrollo de software, baños, este proyecto se puede llevar a cabo ya que se cuenta con diferentes conocimiento en lenguajes de programación como java, c, html, javascript, css, lenguaje ensamblador, lenguaje SQL, y de hardware, además de eso se cuenta con el asesoramiento de profesionales del área.

2.5 Requisitos específicos

2.5.1 Requerimientos funcionales

Código del Requerimiento	Rf01
Nombre	Registros de herramientas
Propósito	Dar de alta los productos de la ferretería.
Descripción	Una vez establecida el inicio de sesión con el usuario se tiene un botón que cumpla con la función de agregar nuevas herramientas al inventario.
Entrada	Formulario del registro de productos
Salida	Mensaje de acción satisfactoria
Prioridad	Alta

Figura 1: Requerimientos Funcional Registros de herramientas.

Código del Requerimiento	Rf02
Nombre	Registros de ventas
Propósito	Registrar las ventas que se este llevando
Descripción	Una vez establecida el inicio de sesión con el usuario se tiene un botón que cumpla con la función de registro de venta.
Entrada	Registros de ventas
Salida	Mensaje de acción satisfactoria
Prioridad	Alta

Figura 2: Requerimientos Funcional Registros de ventas.

Código del Requerimiento	Rf03
Nombre	Consulta de productos
Propósito	Consultar el inventario de una manera eficiente para el usuario.
Descripción	Pulsa sobre el botón consultar producto para poder visualizar los productos que se tienen en ese momento.
Entrada	Seleccionar el botón consultas
Salida	Se muestran las consultas
Prioridad	Media

Figura 3: Requerimientos Funcional Consulta de productos.

Código del Requerimiento	Rf04
Nombre	Consulta de ventas
Propósito	Consultar las ventas realizadas por fechas
Descripción	Una vez establecida el inicio de sesión con el usuario se tiene un botón que cumpla con la función de consultar las ventas que han hecho según la fecha.
Entrada	Seleccionar el botón buscar ventas
Salida	Se muestran las consultas
Prioridad	Media

Figura 4: Requerimientos Funcional Consulta de ventas.

Código del Requerimiento	Rf05
Nombre	Registro de usuario
Propósito	Registrar a los usuarios que usarán el programa
Descripción	Una vez establecida el inicio de sesión con el usuario principal se tiene un botón que cumpla con la función de registrar a los demás usuarios y darle los permisos necesarios.
Entrada	Seleccionar el botón alta de usuarios
Salida	Mensaje de acción satisfactoria
Prioridad	Alta

Figura 5: Requerimientos Funcional Registro de usuario.

Código del Requerimiento	Rf06
Nombre	Consulta de usuarios
Propósito	Consultar y modificar los usuarios agregados
Descripción	Una vez establecida el inicio de sesión con el usuario se tiene un botón que cumpla con la función de consultar usuarios donde se podrá actualizar los usuarios
Entrada	Seleccionar el botón consultar usuarios
Salida	Se muestran las consultas
Prioridad	Alta

Figura 6: Requerimientos Funcional Consulta de usuarios

2.5.2 Requerimientos no funcionales

Código del Requerimiento	RNF01
Nombre	Seguridad
Descripción	Se debe asegurar que los datos estén protegidos del acceso no autorizado
Prioridad	Alta

Figura 7: Requerimientos no funcional Seguridad.

Código del Requerimiento	RNF02
Nombre	Mensajes de error
Descripción	El sistema debe mandar mensajes de error que ayuden al usuario
Prioridad	Media

Figura 8: Requerimientos no funcional Mensajes de error.

Código del Requerimiento	RFN03
Nombre	Look and feel
Descripción	El aspecto es de una manera amigable, cumple con los requerimientos del usuario.
Prioridad	Alta

Figura 9: Requerimientos no funcional Look and feel.

Código del Requerimiento	RFN04
Nombre	Restricciones de contenido
Descripción	El acceso a cada función del sistema está determinado por el rol del usuario.
Prioridad	Alta

Figura 10: Requerimientos no funcional Restricciones de contenido.

Código del Requerimiento	RFN05
Nombre	Confidencialidad
Descripción	Toda la información otorgada por los usuarios se manipulará únicamente con fines educativos y de manera limpia.
Prioridad	Alta

Figura 11: Requerimientos no funcional Confidencialidad.

Código del Requerimiento	RFN06
Nombre	Robustez
Descripción	El software debe ser capaz de manejar toda la información recolectada a través del tiempo con fluidez.
Prioridad	Media

Figura 12: Requerimientos no funcional Robustez.

2.5.3 Requerimientos de interfaz

Descripción: La figura 3 permite al usuario acceder a las funciones principales de una manera fácil y rápida. El usuario tiene que colocar su nombre de usuario seguida de su contraseña.



Figura 13: Ventana login.

Descripción: La figura 4 es la principal que mostrará las funciones del software a desarrollar, las funciones están en la parte superior de la ventana que son: inicio, ventas, producto, usuarios y ayuda. Además, en la parte superior se muestra la fecha actual.



Figura 14: Ventana principal.

Descripción: En la figura 5 se realizará la función de venta, la cual incluye el procedimiento de realizar la suma de productos, para aquello el usuario tiene que llenar las casillas en blanco.

The screenshot shows a user interface for a sales application. At the top left is a shopping cart icon with the text "Realizar venta". The main title "Ventas" is centered at the top. To the right of the title is a date field showing "04/02/2022". Below the title is a small shopping cart icon with a plus sign. A table with columns labeled "ID", "Nombre de producto", "Identificador", "Precio", "IdVendedor", "Fecha", and "Cantidad" follows. Underneath the table are three input fields labeled "Cuenta por cobrar", "Recibido", and "Cambio". At the bottom are four buttons with icons: "Generar venta" (monitor with dollar sign), "Eliminar producto" (shopping cart with minus sign), "Cancelar" (shopping cart with red X), and "Salir" (exit door).

Figura 15: Ventana ventas.

Descripción: En la figura 6 se realizará la función de seleccionar producto, la cual incluye el procedimiento para realizar la suma de productos, para aquello el usuario tiene que digitar el producto en la casilla en blanco después tiene que seleccionar la cantidad y presionar el botón agregar producto.

The screenshot shows a user interface for selecting products. At the top left is a logo of a shopping cart with the word "Productos". The main title "Productos" is centered in large, bold letters. Below the title is a form with two input fields: "Nombre del producto" (Product Name) and "Cantidad" (Quantity), which is set to 1 with a spin button. A table below lists product details with columns: ID, Nombre de producto, Cantidad, Marca, Identificador, Cantidad, and Precio. Underneath the table is a section labeled "Datos de producto" (Product Data) with four input fields: "Identificador" (Identifier), "Cantidad" (Quantity), "Nombre producto" (Product Name), and "Precio" (Price). At the bottom are two buttons: "Agregar producto" (Add Product) with a plus sign icon and "Salir" (Exit) with a right-pointing arrow icon.

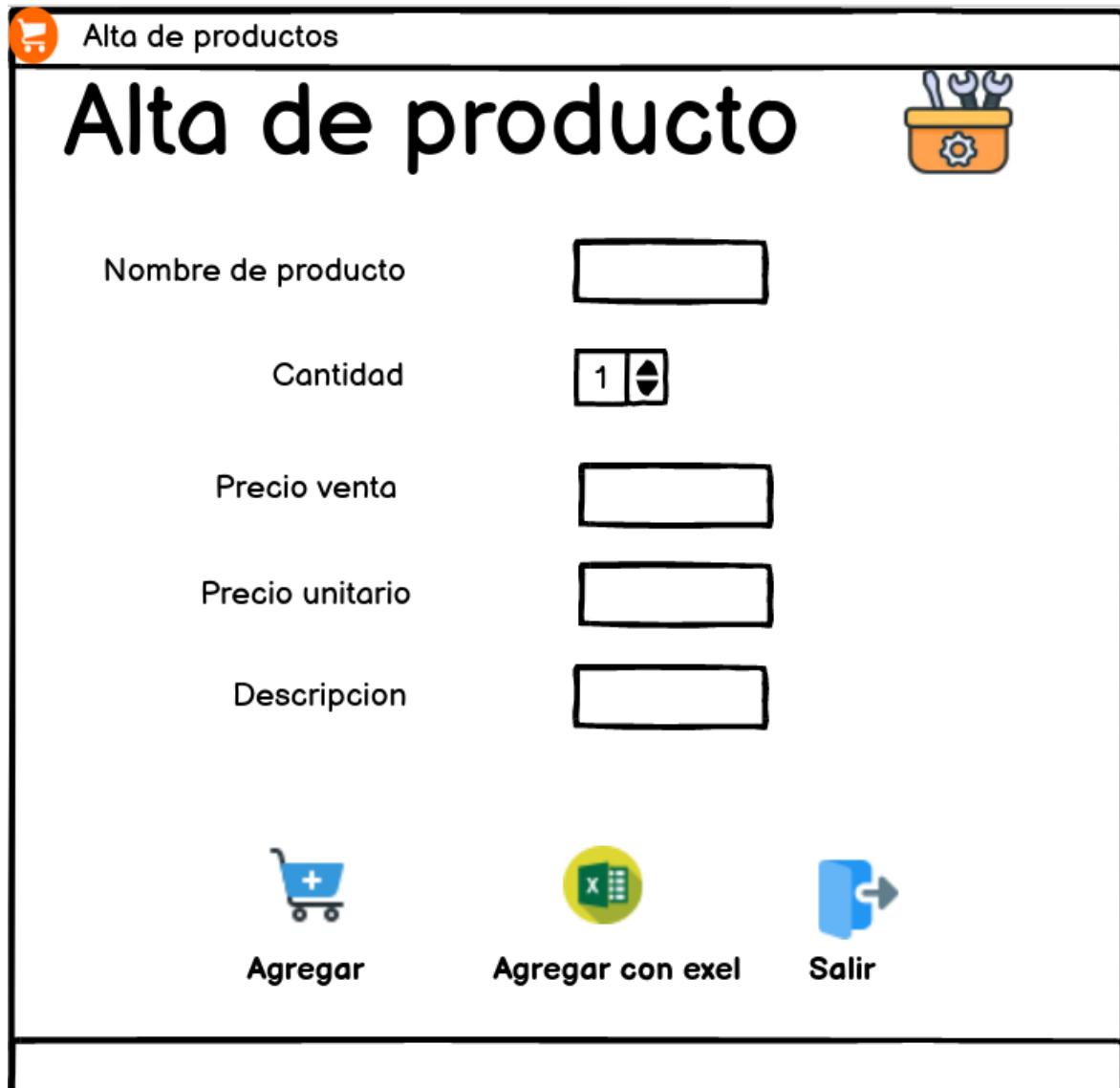
Figura 16: Ventana seleccionar producto.

Descripción: La figura 7 permite buscar las ventas que fueron realizadas con anterioridad, dando como resultado la fecha de la venta, productos comprados y monto total de la venta.

The screenshot shows a software window titled "Consultar ventas". The main title is "Consultar venta". Below it is a search bar labeled "Buscar" with a magnifying glass icon, followed by a date picker icon and a "Buscar" button. A table below the search bar has four columns: "Nombre producto", "Cantidad", "Fecha venta", and "Monto". At the bottom left is a yellow "Generar excel" button with an Excel icon. At the bottom right is a blue "Salir" button with a right-pointing arrow icon.

Figura 17: Ventana consulta ventas.

Descripción: La figura 8 es una ventana que ayudará a poder dar de alta los productos de una manera eficiente, se tiene que agregar algunos datos del producto para poder ya tenerlos en el sistema.



Alta de productos

Alta de producto

Nombre de producto

Cantidad

Precio venta

Precio unitario

Descripcion

Agregar Agregar con excel Salir

Figura 18: Ventana alta de productos.

Descripción: La figura 9 permite buscar algún producto que estén dentro del sistema, también permite actualizar y eliminar el producto.

The screenshot shows a window titled "Consultar productos" (Search products). At the top, there is a header "Consultar producto". Below it, a label "Ingrese el ID" (Enter the ID) is followed by a text input field. A table below the input field has columns labeled "ID", "Nombre de producto", "Cantidad", "Precio venta", "Precio unitario", and "Descripción". At the bottom of the window are four buttons: "Actualizar" (Update) with a circular arrow icon, "Generar excel" (Generate excel) with an Excel icon, "Eliminar" (Delete) with a trash bin icon, and "Salir" (Exit) with a blue document icon.

Figura 19: Ventana consulta de productos.

Descripción: La figura 10 permite actualizar los datos del producto.

Actualizar producto

Actualizar

Nombre de producto

Cantidad

Precio venta

Precio unitario

Descripcion

Actualizar

Cerrar

Figura 20: Ventana actualizar.

Descripción: la figura 11 permite agregar nuevos usuarios al sistema, se tiene que digitar el nombre del usuario y contraseña que se desea asignar, presionar un click en guardar y de esa manera se tendrá un nuevo usuario.

The screenshot shows a user interface titled "Alta de usuarios". The form consists of several input fields and a dropdown menu:

- Nombre de usuario: An input field.
- Contraseña: An input field.
- Confirmar contraseña: An input field.
- Telefono: An input field.
- Correo: An input field.
- Cargo: A dropdown menu currently set to "Administrador".

At the bottom left is a "Guardar" button with a folder icon. At the bottom right is a "Salir" button with an exit icon.

Figura 21: Ventana agregar usuarios.

Descripción: la figura 12 permite buscar a los usuarios que están registrados en el sistema, esto da como resultado NIP, nombre y contraseña.

The screenshot shows a window titled "Consultar usuarios" with a sub-section titled "Consultar usuario". A search input field labeled "Usuario a buscar" is present. Below it is a table with columns: ID, Nombre, Telefono, Correo, and Cargo. At the bottom are three buttons: "Actualizar" (refresh), "Eliminar" (delete), and "Salir" (exit).

ID	Nombre	Telefono	Correo	Cargo

Figura 22: Ventana consulta de usuarios.

2.6 Mapa de navegación

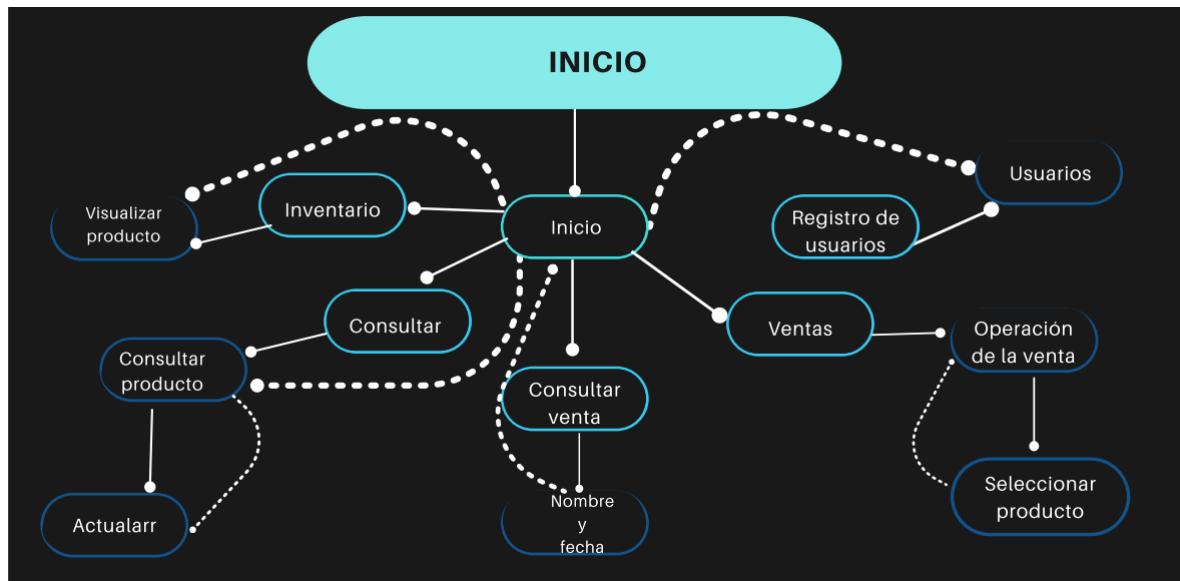


Figura 23: Mapa de navegación

3 Análisis

3.1 Diagramas de estructura UML

3.1.1 Diagrama de flujo de datos

Descripción: la siguiente figura hace referencia al diagrama de flujo de datos para ventas.



Figura 24: Diagrama de flujo de datos para venta

Descripción: la siguiente figura hace referencia al diagrama de flujo de datos para productos.

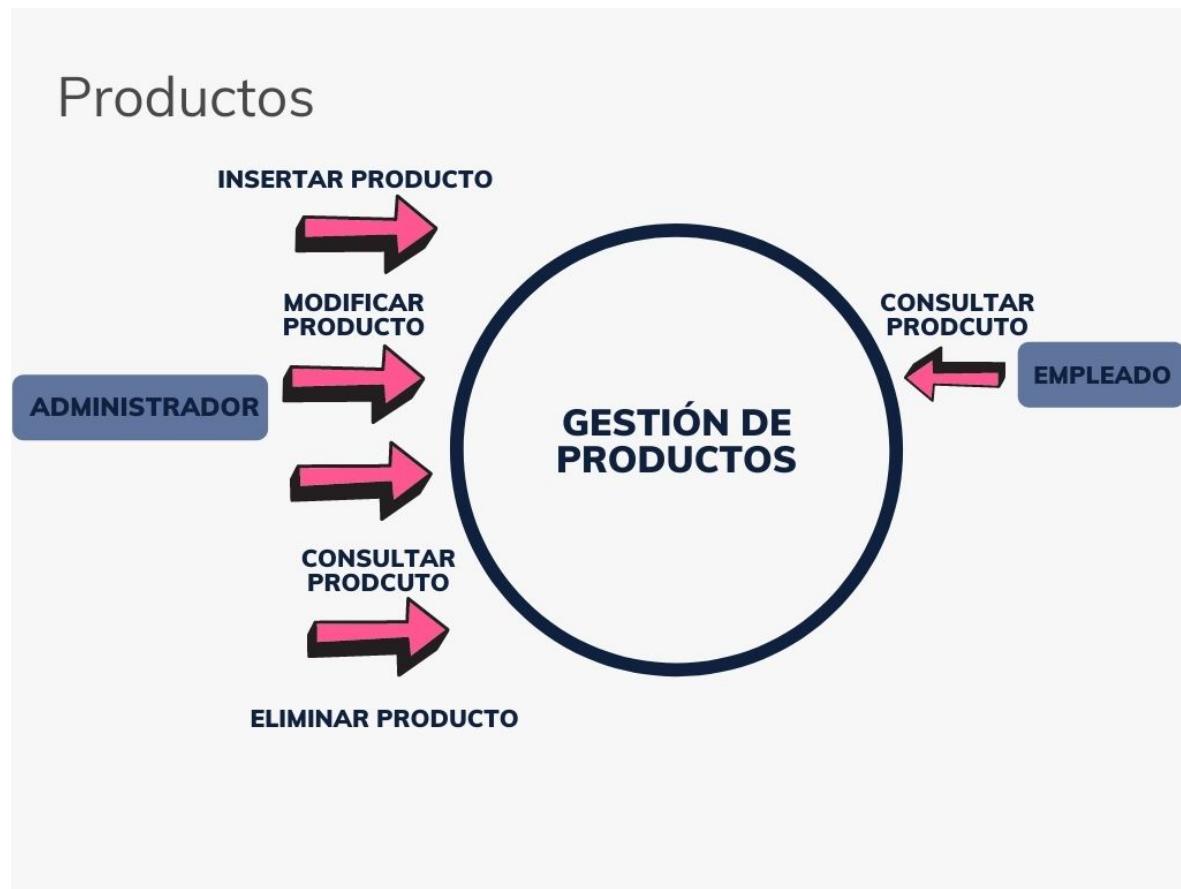


Figura 25: Diagrama de flujo de datos para productos

Descripción: la siguiente figura hace referencia al diagrama de flujo de datos para usuarios.

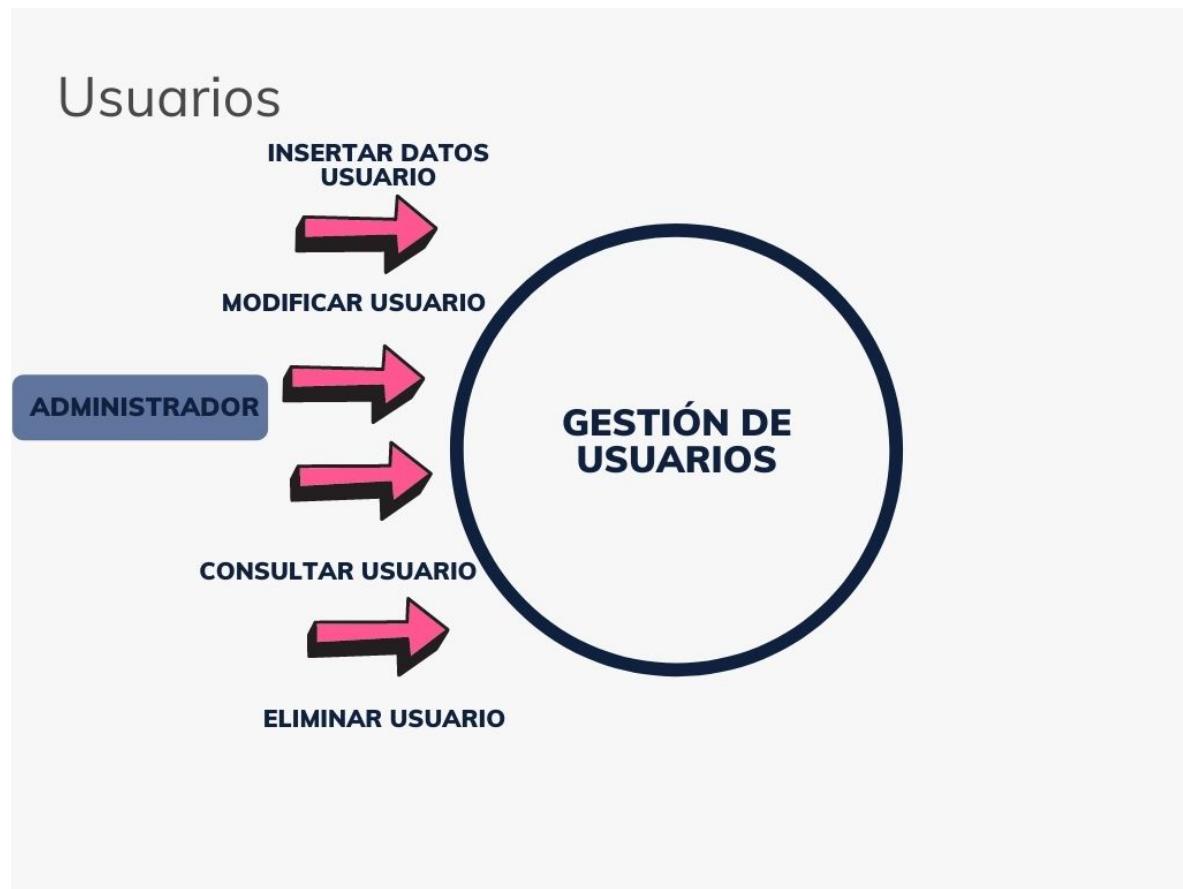


Figura 26: Diagrama de flujo de datos para usuarios

Descripción: la siguiente figura hace referencia al diagrama de flujo de datos para el logueo.



Figura 27: Diagrama de flujo de datos para el logueo

3.1.2 Diagrama de clases y paquetes

Descripción: El siguiente código hace referencia a los diagramas de clases y paquetes.

```
@startuml
class entity.Usuario #green{
    -NombreUsuario:String
    -Contraseña:String
    +String getNombreUsuario()
    +String setNombreUsuario()
    +String getContraseña()
    +String setContraseña()
    +void Usuario()
    +void Usuario(NombreUsuario, Contraseña)
}
class entity.Venta #green{
    -NumerodeVenta:int
    -Pagar:float
    -Cambio:float
    +float getPagar()
    +float setPagar()
    +float getCambio()
    +float setCambio()
    +void Venta()
    +void Venta(NumerodeVenta, Pagar, Cambio)
}
class entity.Producto #green{
    -NombreProducto:String
    -Cantidad:int
    -PrecioCompra:float
    -PrecioVenta:float
    -Descripcion:String
    +String getNombreProductos()
    +String setNombreProductos()
    +String getCantidad()
    +String setCantidad()
    +String getPrecioCompra()
```

```

+String setPrecioCompra()
+String getPrecioVenta()
+String setPrecioVenta()
+String getDescripcion()
+String setDescripcion()
+void Producto()
+void Producto(NombreProducto , Cantidad , PrecioCompra , PrecioVenta ,
               Descripcion)

}

interface model.IUsuarioModel #violet {
+void BuscarUsuario()
+void ConsultarUsuario()
+void EliminarUsuario()
+void ActaulizarUsuario()

}

interface model.IVentaModel #violet{
+void EliminarProducto()
+void CancelarVenta()
+void MuestraProducto()
+void RealizarVenta()
}

interface model.IConsultarVentaModel #violet{
+void BuscarVenta()
+void ConsultarVenta()
+void EliminarVenta()
+void ActaulizarVenta()
}

interface model.IProductoModel #violet{
+void consultarProducto()
+void eliminarProducto()
+void actualizarProducto()
+void CambiarDatos()
+void buscaProducto()
}

```

```

class model . UsuarioModelImpl #skyblue {
    —Conexion :conección
    —Usuario :usuario
    +void BuscarUsuario()
    +void ConsultarUsuario()
    +void EliminarUsuario()
    +void ActaulizarUsuario()

}

model . IUsuarioModel <|— model . UsuarioModelImpl

class model . VentaModelImpl #skyblue{
    —Conexion :conección
    —Usuario :usuario
    +void EliminarProducto()
    +void CancelarVenta()
    +void MuestraProducto()
    +void RealizarVenta()
}
model . IVentaModel <|— model . VentaModelImpl

class model . ConsultarVentaImpl #skyblue{
    —Conexion :conección
    —Usuario :usuario
    +void BuscarVenta()
    +void ConsultarVenta()
    +void EliminarVenta()
    +void ActaulizarVenta()
}
model . IConsultarVentaModel <|— model . ConsultarVentaImpl

class model . ProductoModelImpl #skyblue{
    —Conexion :conección
    —Usuario :usuario
    +void consultarProducto()
    +void eliminarProducto()
    +void actualizarProducto()
    +void CambiarDatos()
    +void buscaProducto()
}

```

```

}

model.IProductoModel <|-- model.ProductoModelImpl

class controlador.usuarioControlador #pink{
    +void BuscarUsuario()
    +void ConsultarUsuario()
    +void EliminarUsuario()
    +void ActaulizarUsuario()

}

class controlador.VentaControlador #pink{
    +void EliminarProducto()
    +void CancelarVenta()
    +void MuestraProducto()
    +void RealizarVenta()
}

class controlador.ProductoControlador #pink{
    +void consultarProducto()
    +void eliminarProducto()
    +void actualizarProducto()
    +void CambiarDatos()
    +void buscaProducto()
}

class vista.Panel.Venta {
    +SeleccioanrProducto()
    +CuentaPorCobrar()
    +recibido()
    +cambio()
    +CalcelarCompra()
    +RealizarCompra()
    +EliminarProducto()
    +JtableVentas()
}

```

```

class vista.Panel.InicioDeSesion {
+Contraseña [JPasswordField]()
+Usuario [JtextField]

}

class vista.Panel.Producto {
+NombreProducto()
+Cantidad()
+PrecioCompra()
+PrecioVenta()
+Descripcion()
}

class vista.Panel.ConsultarVenta {
+buscar()
+actualizar()
+eliminar()
}

class vista.Panel.Administrador {
+Inicio()
+Venta()
+Producto()
+Usuarios()
+Salir()
}

class vista.Panel.AgregarUsuario{
+Guardar()
+Limpiar()
+Salir()
+TxtNombreUsuario()
+TxtContraseña()
}

class vista.Panel.ConsultarUsuario{
+Actualizar()
+Eliminar()
+Salir()
+TxtBuscaUsuario()
}

```

```

+JtableUsuarios()
}

class vista.Panel.ConsultarProducto{
+Actualizar()
+Eliminar()
+Salir()
+TxtBuscaProducto()
+JtableProducto()
}

class vista.Panel.SeleccionarProducto{
+AgregarProducto()
+Salir()
+SpinnerCantidadProducto()
+TxtBuscaProductos()
+JtableProductos()
}

entity . Producto --> model . IProductoModel
entity . Venta --> model . IConsultarVentaModel
entity . Usuario --> model . IUsuarioModel

model --> controlador
controlador --> vista.Panel

```

@enduml

Descripción: La siguiente figura hace referencia al diagrama de clases y de paquetes.

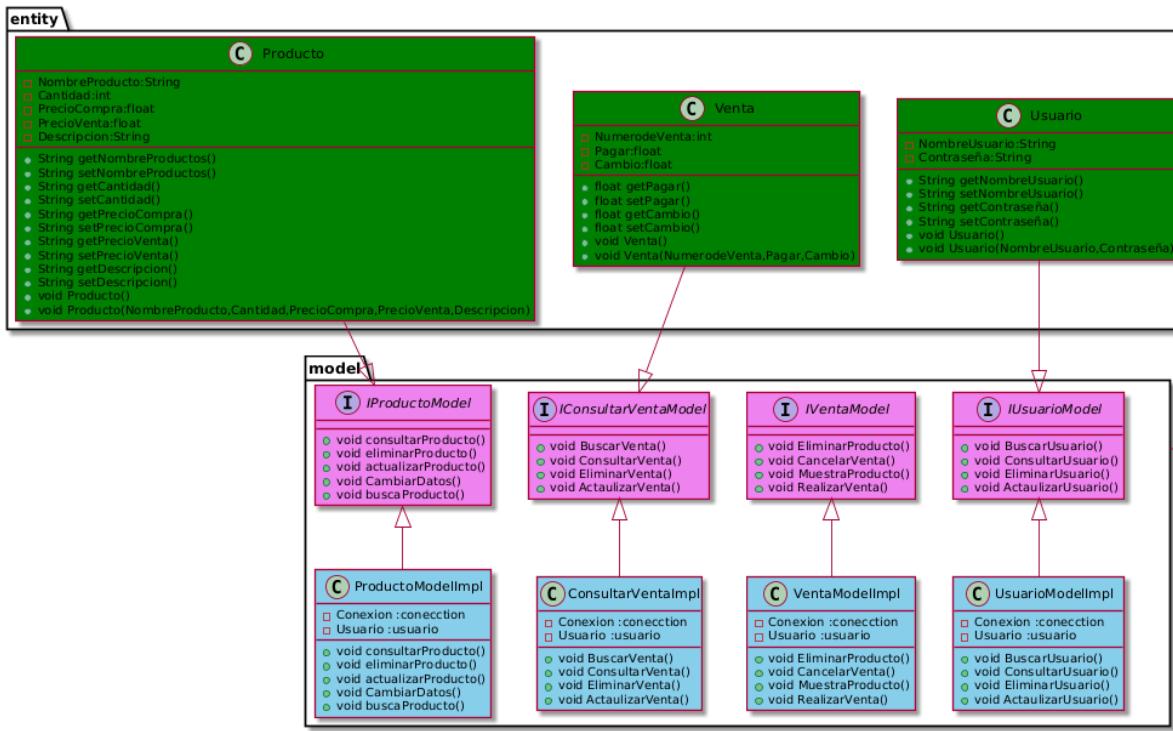


Figura 28: Diagrama de clases primera parte

Descripción: La siguiente figura hace referencia al diagrama de clases.

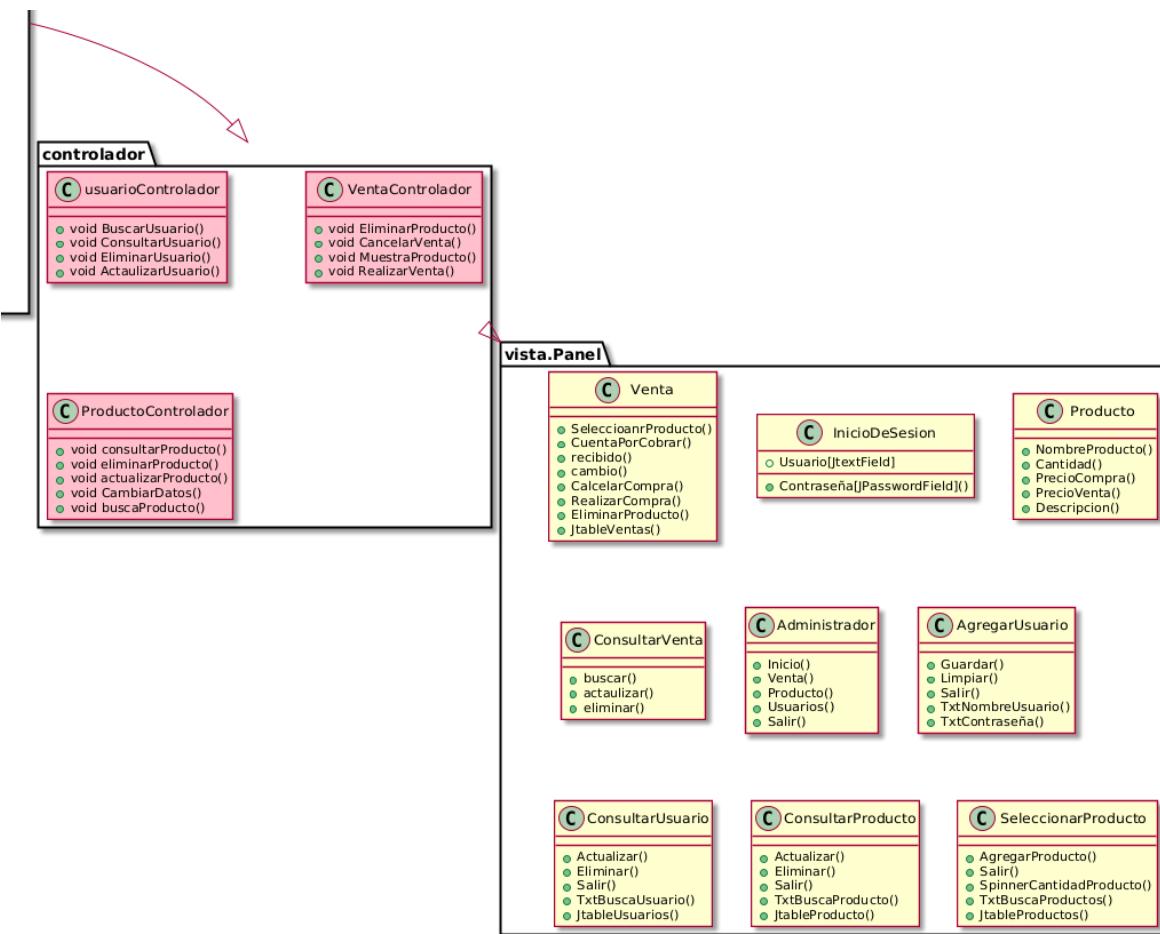


Figura 29: Diagrama de clases segunda parte parte

3.1.3 Diagrama de objetos

Descripción: El siguiente código hace referencia al diagrama de obejetos.

```
@startuml
object Usuario #green{
    -NombreUsuario: Admi
    -Contraseña: Admi
}

object Vendedor #green{
    -NombreUsuario: Leonel
    -Contraseña: 123
    -Contraseña: 123
    -Rol: Vendedor
}
Usuario --> Vendedor
Usuario --> Administrador

object Administrador #green{
    -NombreUsuario: Administrador
    -Contraseña: 12345
    -Rol: SuperUsuario
}

object Venta #green{
    -NumerodeVenta:00001
    -Pagar:$411
    -Cambio:$123
}

object Producto #green{
    -NombreProducto: Pala
    -Cantidad:1
    -PrecioCompra:$45
    -PrecioVenta:$135
    -Descripcion:Pala para arena
}
```

```
Venta --> Producto  
@endum
```

Descripción: La siguiente figura hace referencia al diagrama de objetos.

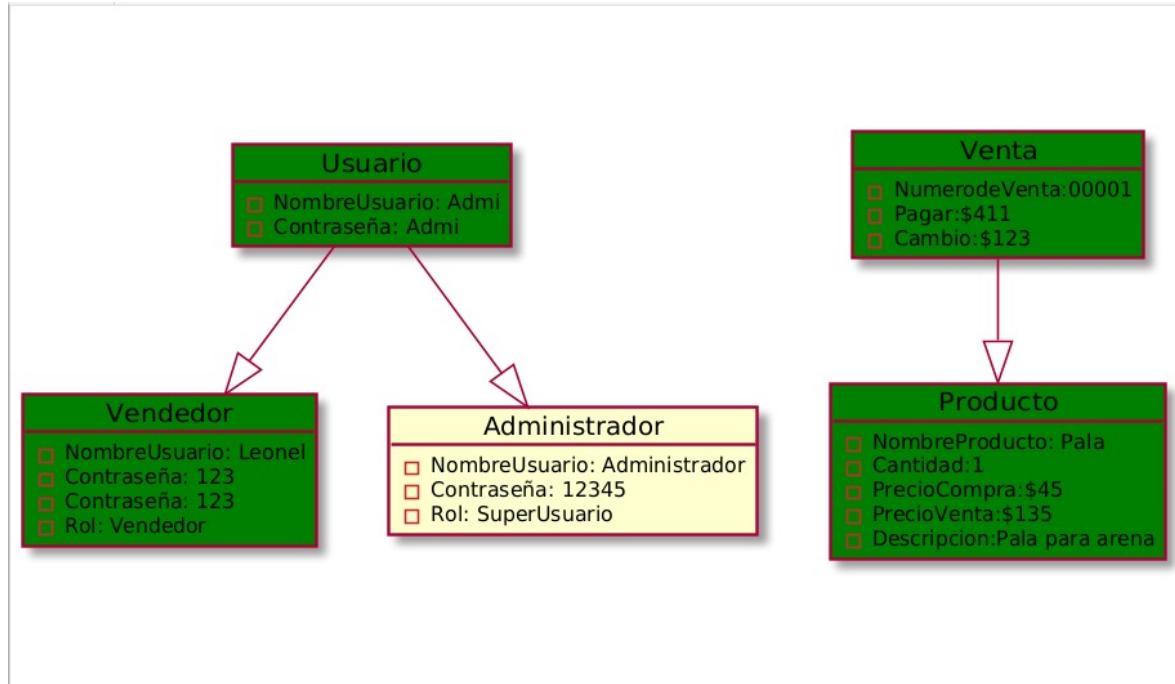


Figura 30: Diagrama de objetos

3.1.4 Diagrama de componentes

Descripción: El siguiente código hace referencia al diagrama de componentes.

```
@startuml
package
node
folder
frame
cloud
database

@startuml
package "Presentacion" {

    Ingreso -- [Intefaz login]
    [Intefaz ventas]
    [Intefaz principal]
    [Intefaz prductos]
    [Intefaz alta productos]
    [Intefaz consultar productos]
    [Intefaz agregar usuarios]
    [Intefaz actualizar usuarios]
}

package "Logica" {
    Ingreso -- [Gestion de ventas]
    [Creacion del total por pagar]
    [Gesti n de productos]
    [Gesti n de empleados]
    [Gesti n de ventas]
    [Gesti n de login]--> [Base de datos]

}

database "Postgrest" {
    folder "Base_de_datos" {
        [Base de datos FERRIMAX]
    }
}
```

```
}
```

```
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de componentes.

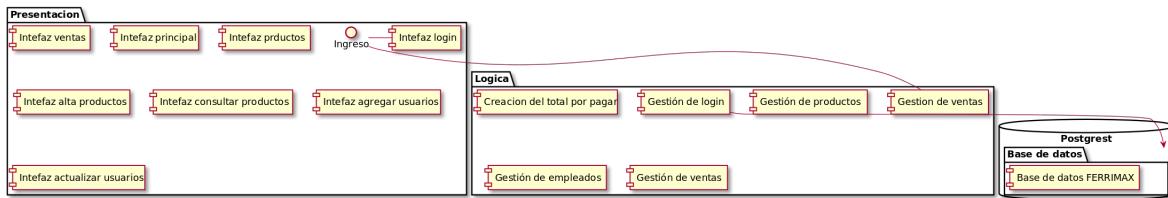


Figura 31: Diagrama de componentes

3.1.5 Diagrama de despliegue

Descripción: El siguiente código hace referencia al diagrama de despliegue

```
@startuml  
node SistemaDeLogeo  
node Principal  
node Ventas  
node Producto  
node Usuario  
node BasedeDatos  
artifact jdbc  
SistemaDeLogeo-->Principal  
Principal-->Ventas  
Principal-->Producto  
Principal-->Usuario  
Ventas<-->jdbc  
Producto<-->jdbc  
Usuario<-->jdbc  
  
jdbc-->BasedeDatos  
  
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de despliegue.

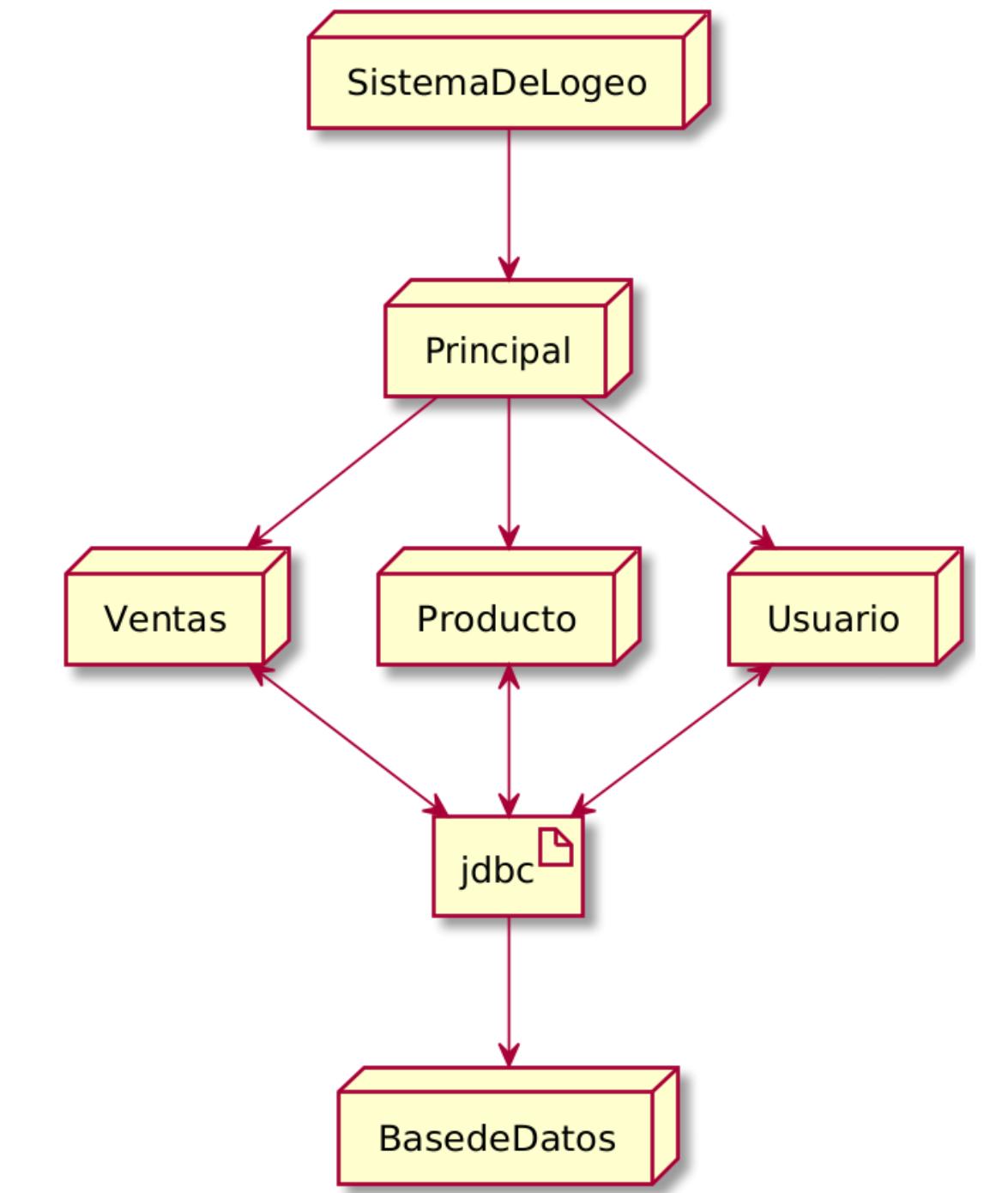


Figura 32: Diagrama de despliegue

3.1.6 Diagrama de actividades

Descripción: El siguiente código hace referencia al diagrama de actividades actualizar producto.

```
@startuml
(*) --> "Actualizar_producto"
--> "Ingresa_los_nuevos_datos"
If " " then
--> [Los datos son correctos] "El_producto_se_actualizo"
--> (*)
else
--> [Los datos son erroneos]"Revisar_los_datos_ingresados"
--> "Ingresa_los_nuevos_datos"
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades de actualizar producto.

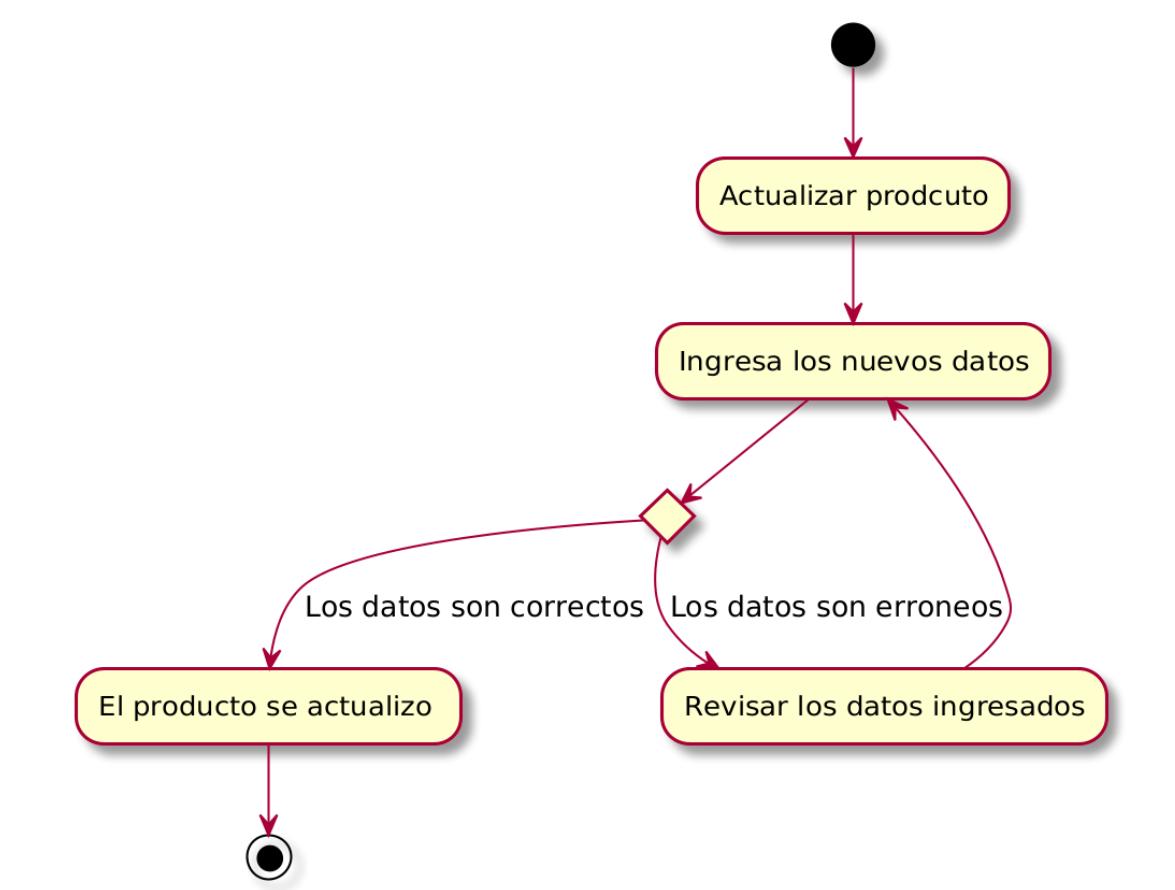


Figura 33: Diagrama de actividades actualizar producto

Descripción: El siguiente código hace referencia al diagrama de actividades actualizar usuario.

```
@startuml
(*) --> "Actualizar_usuario"
--> "Ingresa_los_nuevos_datos"
If " " then
--> [Los datos son correctos] "El_usuario_se_actualizo"
--> (*)
else
--> [Los datos son erroneos] "Revisar_los_datos_ingresados"
--> "Ingresa_los_nuevos_datos"
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades de actualizar usuario.

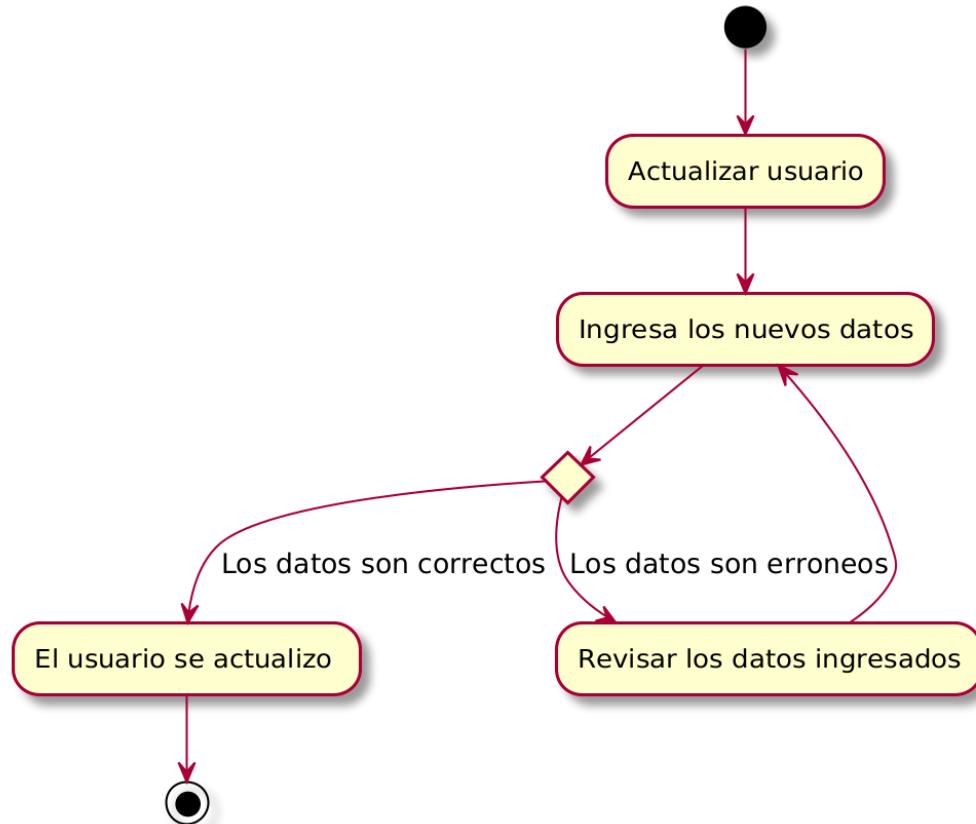


Figura 34: Diagrama de actividades actualizar usuario.

Descripción: El siguiente código hace referencia al diagrama de actividades agregar productos.

```
@startuml
(*) --> "Seleccion_de_producto"
--> "Ingresa_la_cantidad_del_producto"
If " " then
--> [La cantidad es mayor] "El_producto_se_agrega"
-->"Se_realiza_la_cuenta_a_pagar"
--> (*)
else
--> [La cantidad es menor]"Revisar_la_cantidad_solicitada"
--> "Ingresa_la_cantidad_del_producto"
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades de agregar productos.

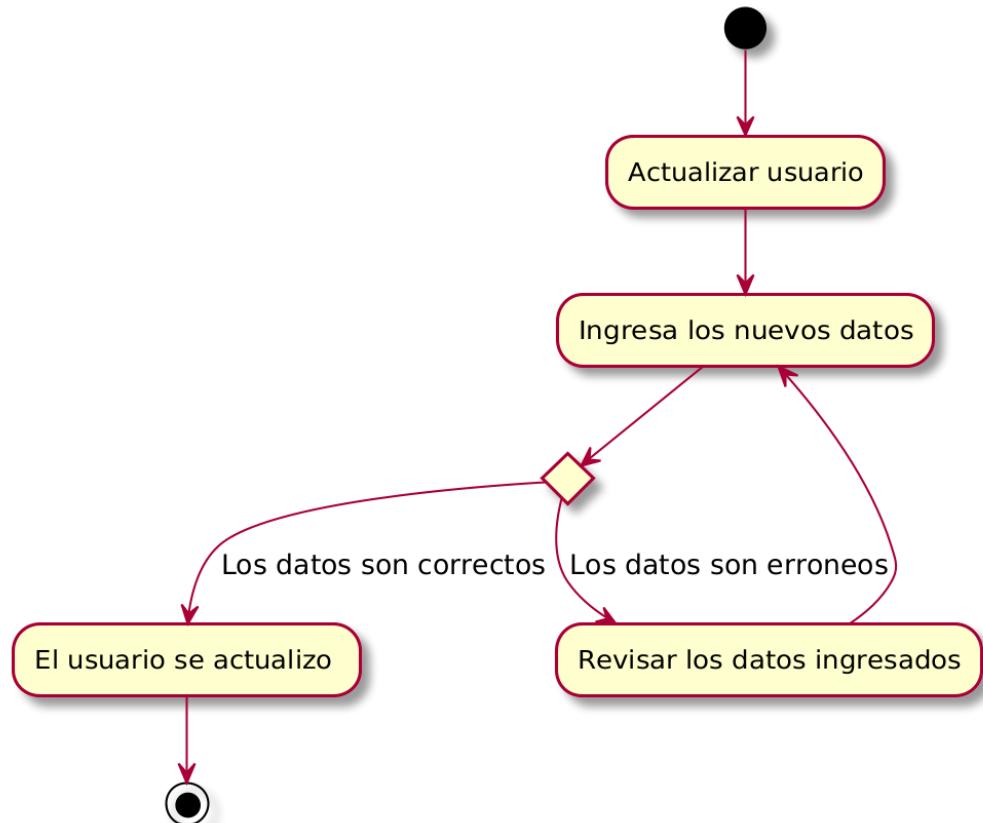


Figura 35: Diagrama de actividades agregar productos.

Descripción: El siguiente código hace referencia al diagrama de actividades consultar productos.

```
@startuml
(*) --> "Consultar_producto"
--> "Ingresa_nombre_de_producto"
If " " then
--> [ Existe el producto ] "Muestra_producto"
-->" Eliminar_producto"
" Muestra_producto"-->" Modificar_producto"
" Modificar_producto"--> (*)
" Eliminar_producto"--> (*)
else
--> [No existe el producto]"Revisar_inventario"
--> "Ingresa_nombre_de_producto"
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades de consultar productos.

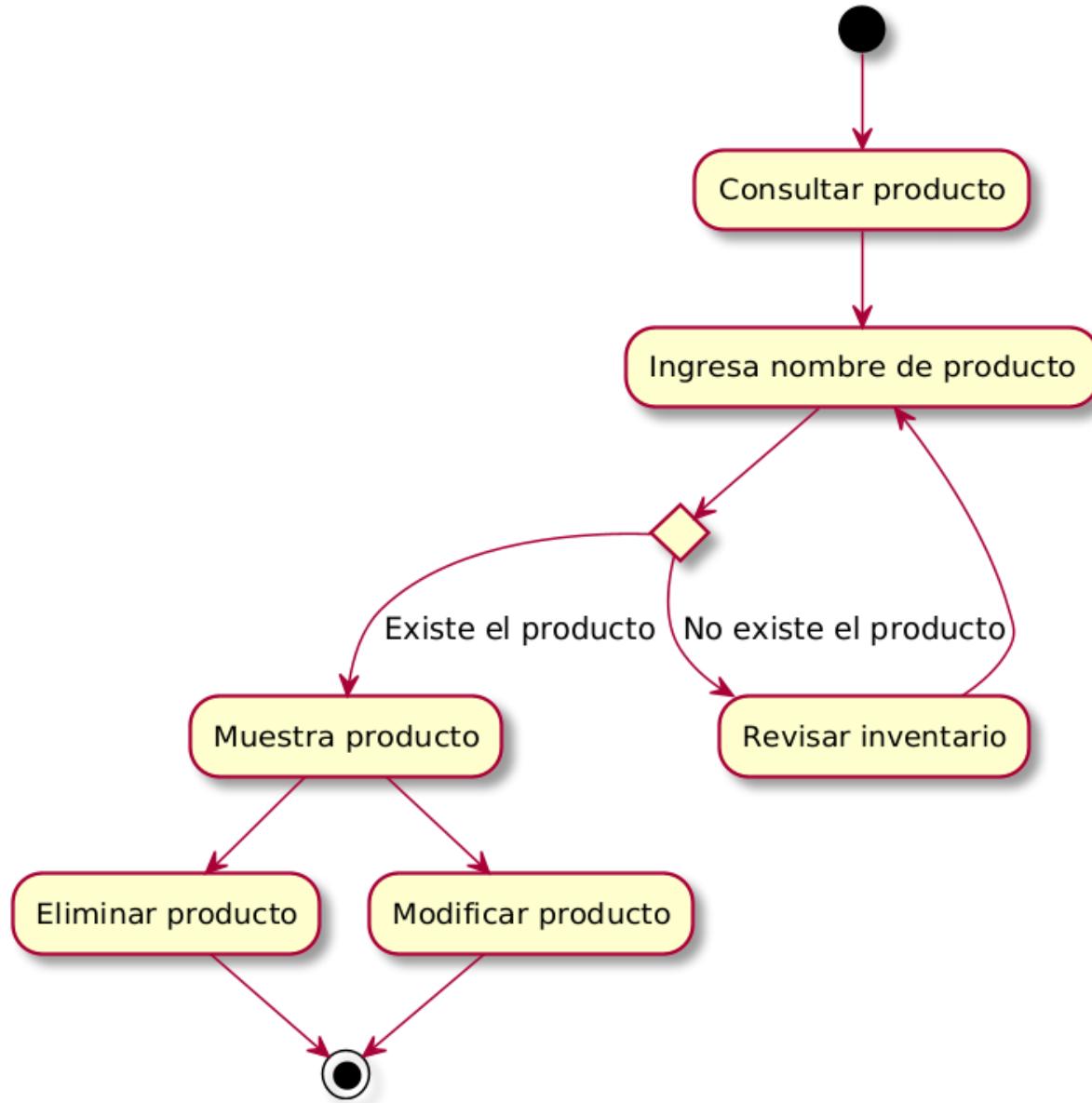


Figura 36: Diagrama de actividades consultar productos.

Descripción: El siguiente código hace referencia al diagrama de actividades consultar usuario.

```
@startuml
(*) --> "Consultar_usuario"
--> "Ingresa_nombre_de_usuario"
If " " then
--> [ Existe nombre de usuario ] "Muestra_datos_usuario"
-->" Eliminar_usuario"
" Muestra_datos_usuario"-->" Modificar_usuario"
" Modificar_usuario"--> (*)
" Eliminar_usuario"--> (*)
else
--> [No existe el usuario]"Revisar_el_nombre_de_usuario_digitado"
--> "Ingresa_nombre_de_usuario"
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades de consultar usuario.

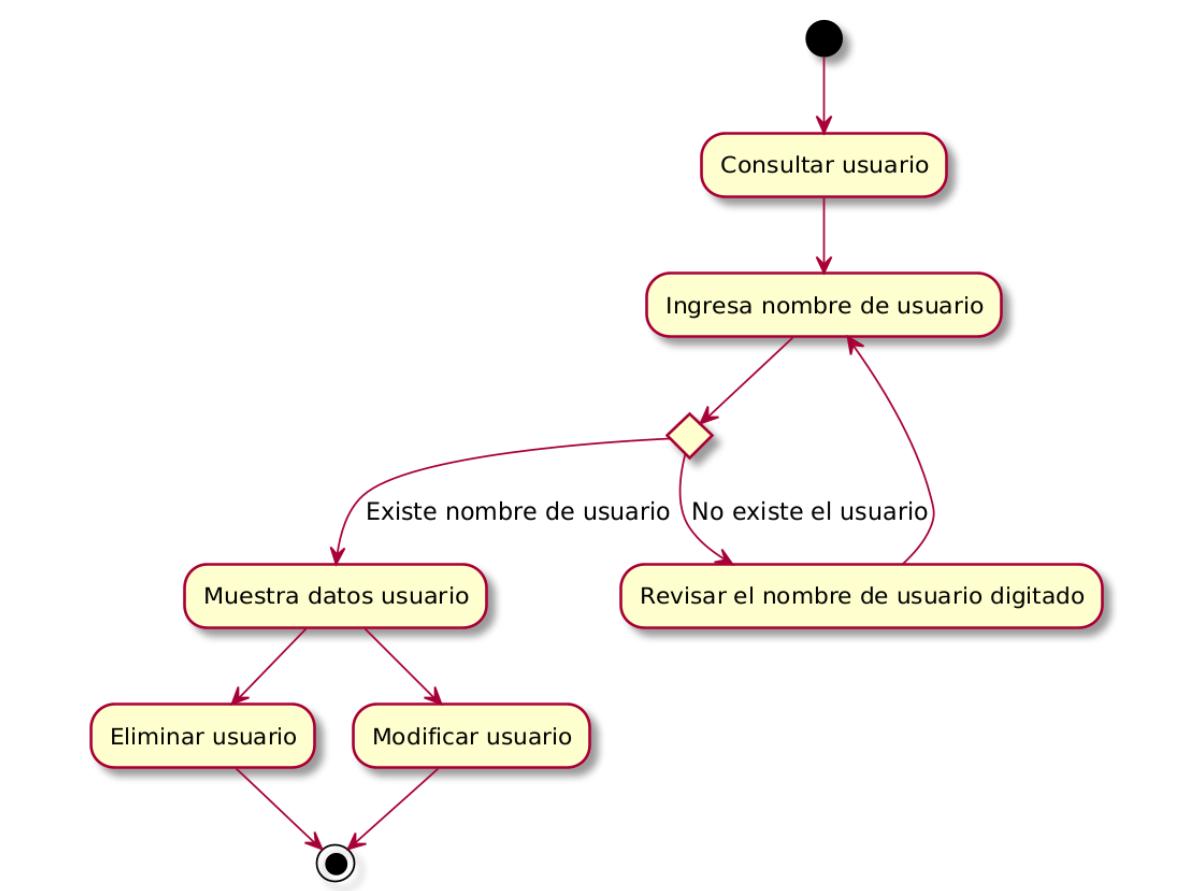


Figura 37: Diagrama de actividades consultar usuario.

Descripción: El siguiente código hace referencia al diagrama de actividades inicio de sesión.

```
@startuml
(*) --> "Inicio_de_sesion"
--> "Ingresar_datos_de_usuario"
--> "Verificar_datos"
If " " then
--> [ Datos_correctos ] "Acceso_al_sistema"
--> (*)
else
--> [ Datos_erroneos ] "Ingresar_datos_de_usuario"
--> "Verificar_datos"
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades de inicio de sesión.

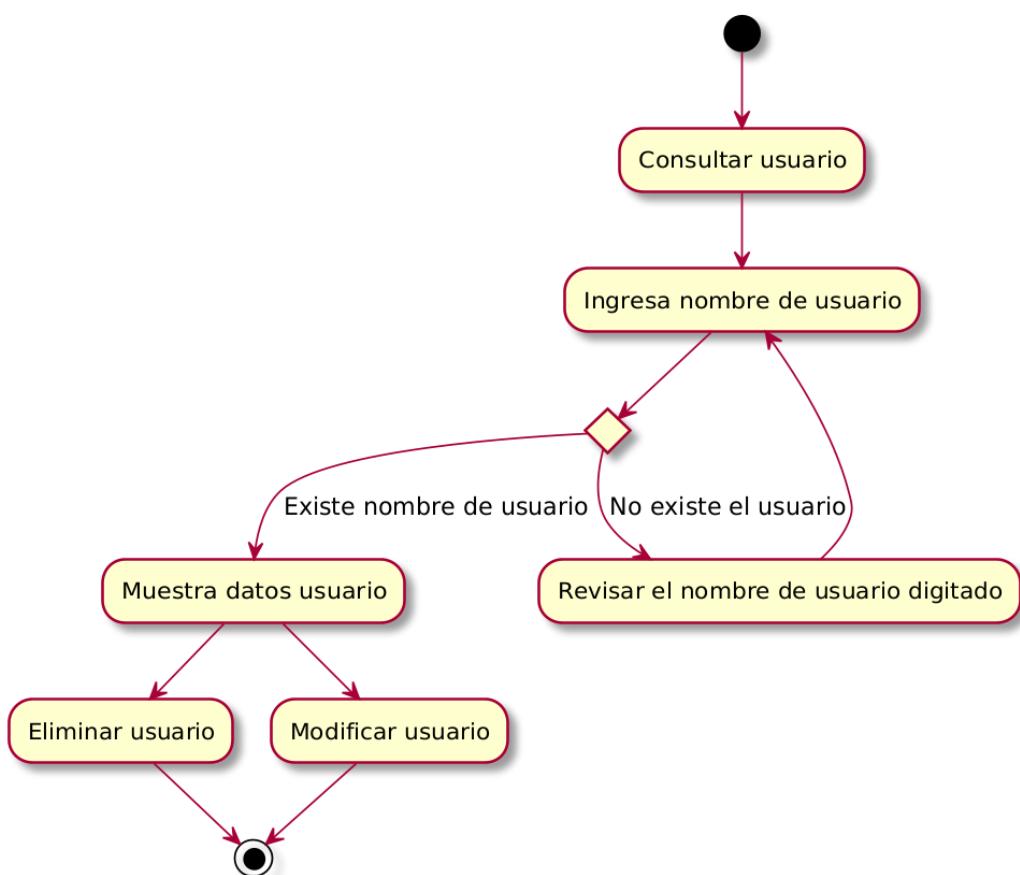


Figura 38: Diagrama de actividades inicio de sesión.

Descripción: El siguiente código hace referencia al diagrama de actividades alta de productos.

```
@startuml  
(*) --> "Registra_productos"  
--> "Ingresar_datos_de_producto"  
--> "Verificar_datos"  
If " " then  
--> [ Datos_correctos ] "Se realiza el registro del producto"  
--> (*)  
else  
--> [ Datos_erroneos ] "Ingresar_datos_de_producto"  
-->"Verificar_datos"  
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades de alta de productos.

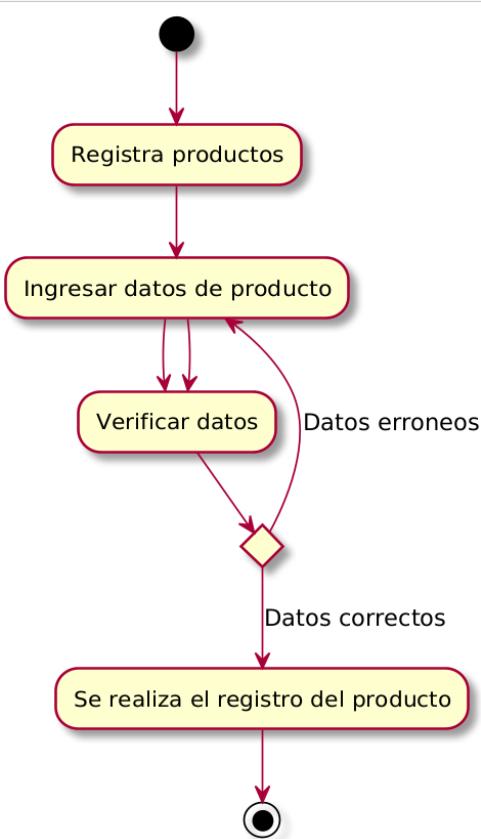


Figura 39: Diagrama de actividades alta de productos .

Descripción: El siguiente código hace referencia al diagrama de actividades alta de usuarios.

```
@startuml
(*) --> "Registra_de_usuarios"
--> "Ingresar_datos_de_usuario"
--> "Verificar_datos"
If " " then
--> [ Datos_correctos ] "Se_realiza_el_registro_del_usuario"
--> (*)
else
--> [ Datos_erroneos ] "Ingresar_datos_de_usuario"
-->"Verificar_datos"
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades de alta de usuarios.

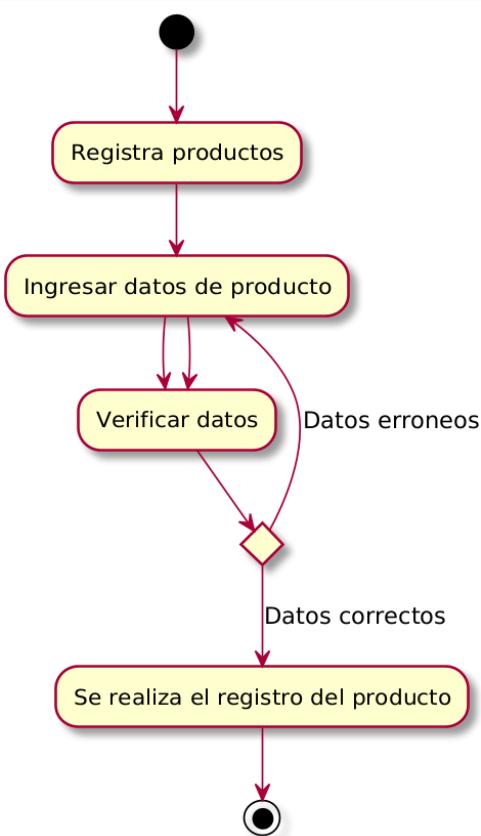


Figura 40: Diagrama de actividades alta de usuarios.

Descripción: El siguiente código hace referencia al diagrama de actividades realizar venta.

```
@startuml
(*) --> "Solicitar_producto"
--> "Procesar_producto"
--> "Extraer_producto"
If "Existencia_producto" then
--> [ Si ] "Se_realiza_la_compra"
--> "Se_realizo_la_compra_con_exito"
--> (*)
else
--> [No existe producto] "Buscar_producto"
-->"Extraer_producto"
@enduml
```

Descripción: La siguiente figura hace referencia al diagrama de actividades realizar venta.

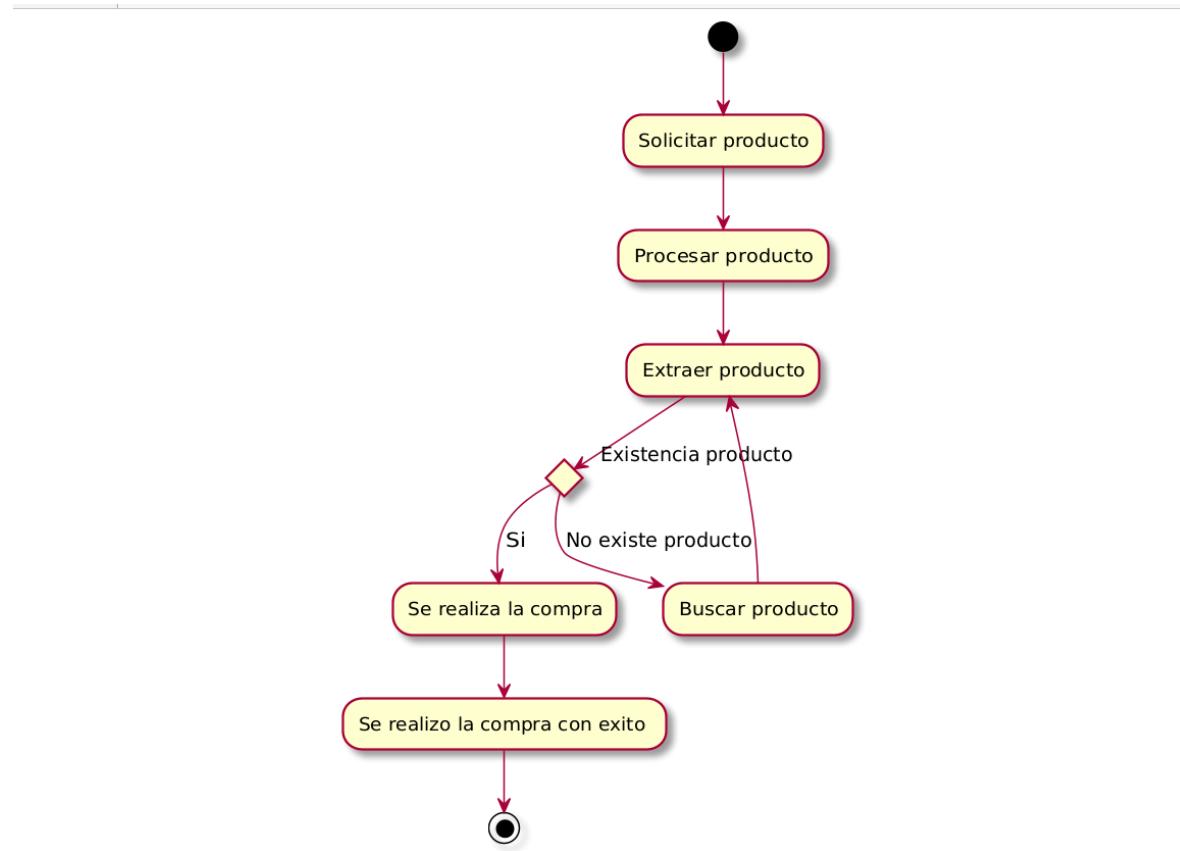


Figura 41: Diagrama de actividades realizar venta.

3.2 Diagramas de comportamiento UML

3.2.1 Diagramas de caso de uso

Descripción: La siguiente figura hace referencia al diagrama de caso de uso del inicio de sesión

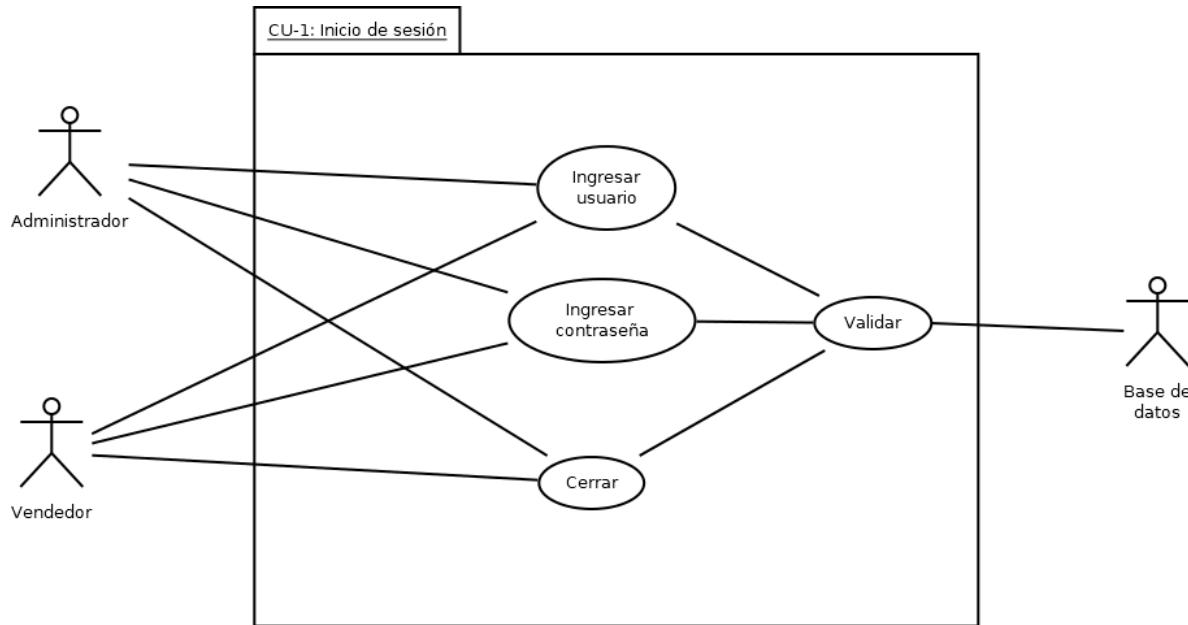


Figura 42: Caso de uso 1 inicio de sesión.

Id	CU-1		
Nombre	Inicio de sesión		
Creado por:	VSG	Actualizado por:	VSG
Fecha de creación:	30/11/2021	Fecha de última revisión	1/12/2021
Actores:	Administrador, vendedor.		
Descripción:	Inicio de sesión.		
Disparador:	N/A		
Precondiciones:	N/A		
Postcondición:	Ir a la pantalla principal		
Flujo normal:	1.- Ingresar nombre de usuarios. 2.-Ingresar contraseña. 3.-Presionar el botón “ENTRAR”. 3.1.- El sistema valida los datos ingresados en el 1 y 2. 3.1.1.- Si los datos son correctos accede a la pantalla CU-2. 3.1.2.- Sí los datos no coinciden en la base de datos, enviar un mensaje “Los datos no coinciden con la base de datos”, regresar a 1 y 2. 4.- Presionar el botón “CERRAR”. 4.1.- Salir del sistema.		
Flujos alternos:			
Include:			
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales:	N/A		
Supuesto:			
ISSUES y notas:	N/A		

Figura 43: Documentación del caso de uso 1 de inicio de sesión.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso de la ventana principal

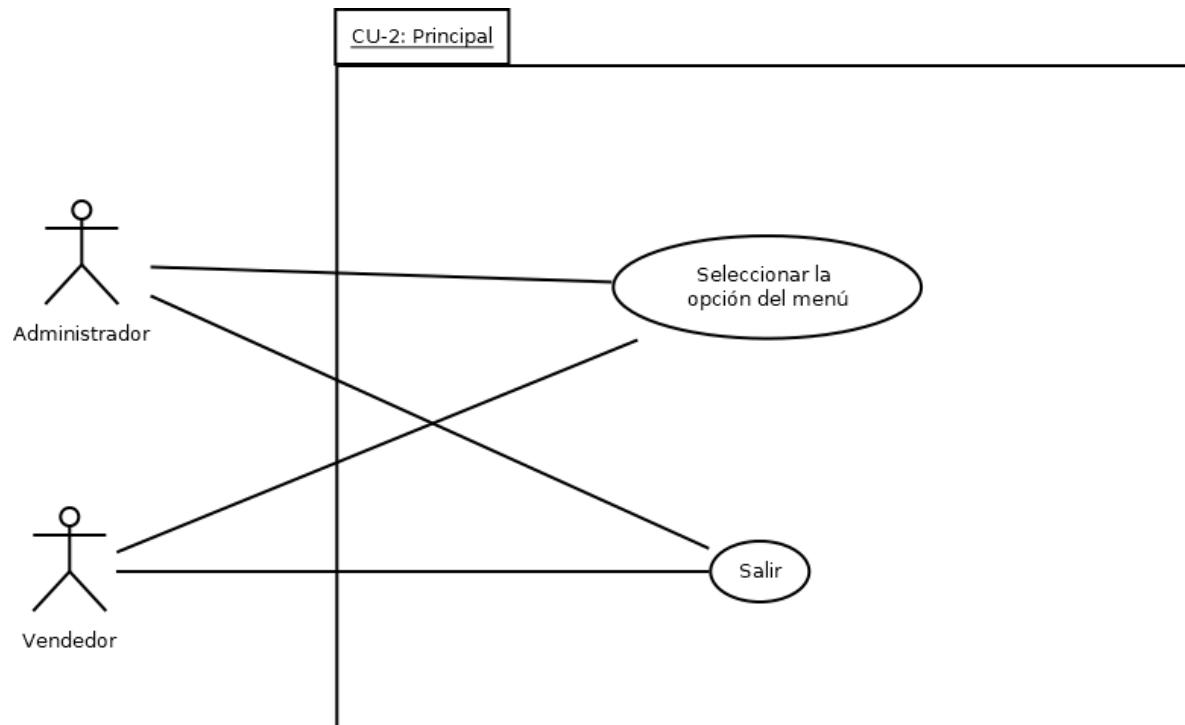


Figura 44: Caso de uso 2 principal.

Id	CU-2		
Nombre	Principal		
Creado por:	VSG	Actualizado por:	VSG
Fecha de creación:	30/11/2021	Fecha de última revisión	06/12/2021
Actores:	Administrador, vendedor.		
Descripción:	Muestra las opciones del menú.		
Disparador:	Al presionar el botón entrar de la venta de inicio de sesión.		
Precondiciones:	N/A		
Postcondición:	N/A		
Flujo normal:	1.-Al presionar el botón de menú “ Iniciar ”, el sistema desplegará un submenu “ Cerrar sesión ”. 1.1.- Se sale de la sesión 2.-Al presionar el botón de menú “ Venta ”, el sistema desplegará un submenu “ Consultar ventas ” y “ Ventas ”. 3.- Al presionar el botón de menú “ Productos ”, el sistema desplegará un submenu “ Dar de alta ” y “ consultar ”. 4.-Al presionar el botón de menú “ Usuarios ”, el sistema desplegará un submenu “ Agregar ” y “ Consultar ”.		
Flujos alternos:			
Include:			
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales	N/A		
Supuesto:			
ISSUES y notas:	N/A		

Figura 45: Documentación del caso de uso 2 principal.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso realiza venta.

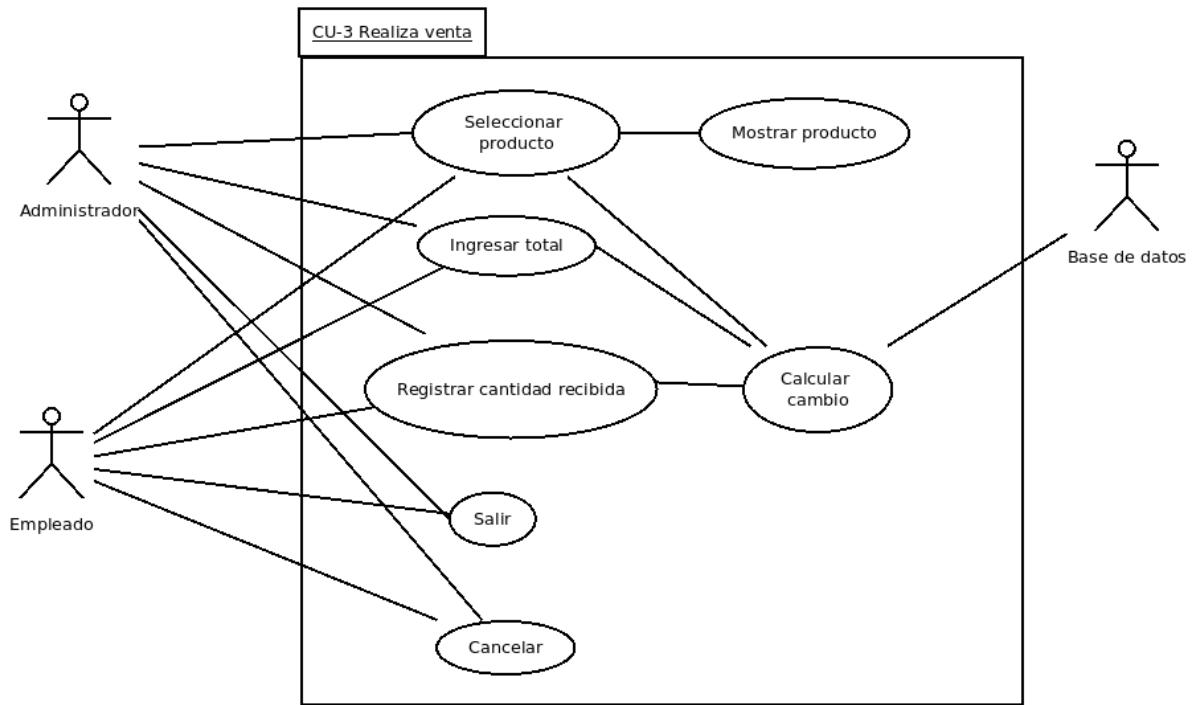


Figura 46: Caso de uso 3 realiza venta.

Id	CU-3		
Nombre	Realiza venta		
Creado por:	VHVL	Actualizado por:	VHVL
Fecha de creación:	30\11\21	Fecha de última versión:	1\12\21
Actores:	Administrador, empleado, Base de datos		
Descripción:	El vendedor o administrador hace la operación de venta		
Disparador:	Se selecciona la opción ventas, del menú ventas.		
Precondiciones:	N\A		
Postcondición:	Se muestra la información de la pantalla 3.1		
Flujo normal:	1. Selecciona el botón “Seleccionar el producto” 1.1 Ir al caso de uso 3.1 2. El sistema llenará una tabla de productos obtenidos en el punto 1. 3. El sistema calculará la cuenta por cobrar. 4. El vendedor ingresa la cantidad recibida. 4.1 si es que la cantidad recibida. 4.1.1 el sistema calcula el resto de y muestra en pantalla. 5. Se puede eliminar un producto de la lista si así se desea con el botón “eliminar producto”. 6. Damos clic en el botón “aceptar” para realizar la venta o el botón cancelar si así es necesario.		
Flujos alternos:			
Include:			
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales	N\A		
Supuesto:	N\A		
ISSUES y notas:			

Figura 47: Documentación de caso de uso 3 realiza venta.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso seleccionar producto.

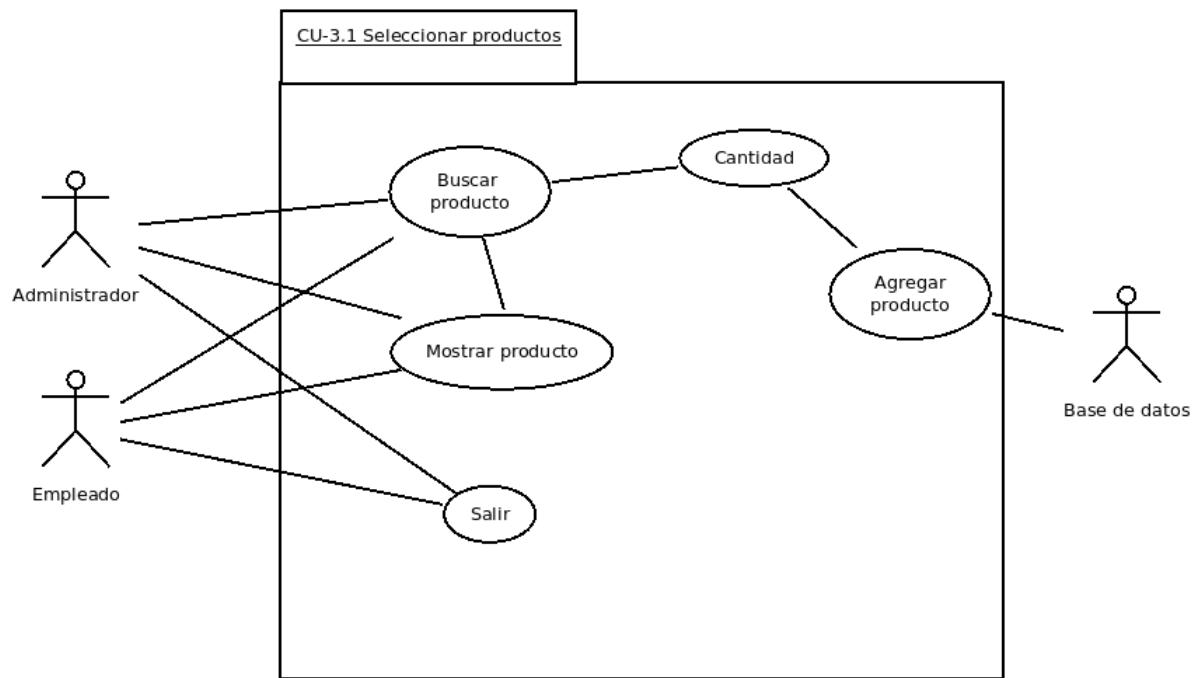


Figura 48: Caso de uso 3.1 seleccionar producto.

Id	CU-3.1		
Nombre	Seleccionar productos		
Creado por:	VHVL	Actualizado por:	VHVL
Fecha de creación:	02\12\21	Fecha de última versión:	04\12\21
Actores:	Administrador, empleado, Base de datos		
Descripción:	El vendedor o administrador selecciona los productos a vender		
Disparador:	Seleccionar la opción “seleccionar el producto” de la pantalla 3.		
Precondiciones:	N\A		
Postcondición:	Se muestra la información en la ventana 3		
Flujo normal:	1. Se ingresa el nombre del producto a buscar. 2. Se selecciona a la cantidad del producto elegido. 3. Se mostrará los productos seleccionados en la tabla. 4. Se agregan los productos en con el botón “agregar producto”. 5. Damos clic en el botón “salir”.		
Flujos alternos:			
Include:			
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales	N\A		
Supuesto:	N\A		
ISSUES y notas:	N\A		

Figura 49: Documentación de caso de uso 3.1 seleccionar producto.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso consultar ventas.

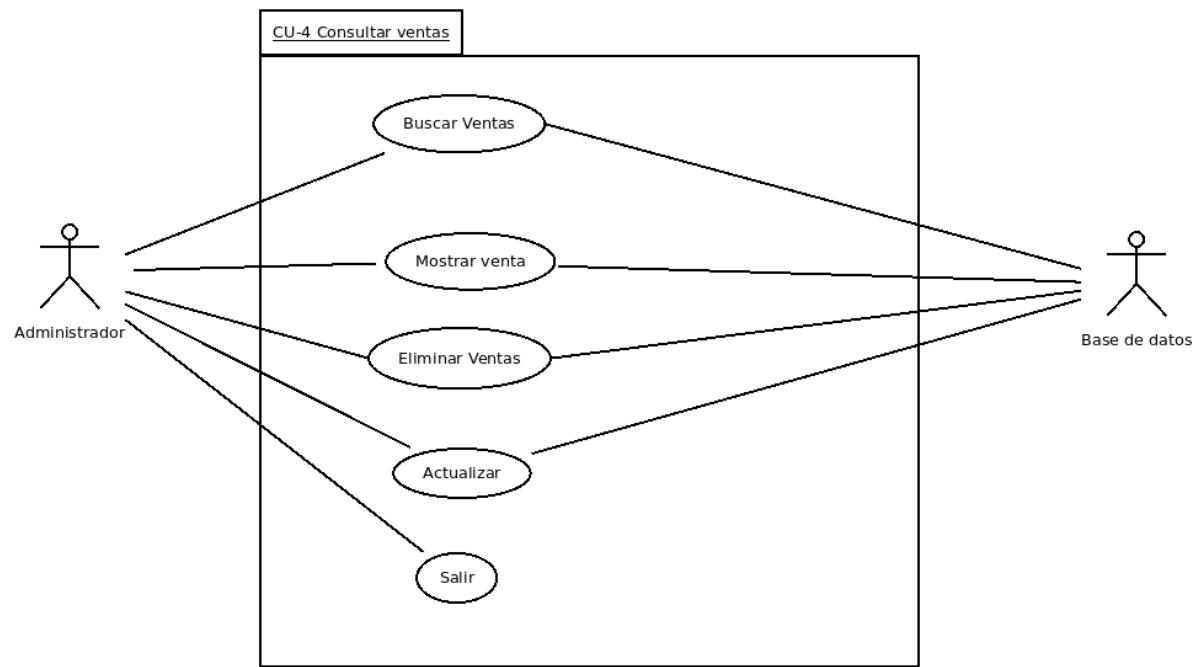


Figura 50: Caso de uso 4 consultar ventas.

Id	CU-4		
Nombre	Consultar ventas		
Creado por:	VHVL	Actualizado por:	VHVL
Fecha de creación:	02\12\21	Fecha de última versión:	4\12\21
Actores:	Administrador, Base de datos		
Descripción:	El administrador realiza la consulta de las ventas		
Disparador:	Se selecciona la opción “ consulta de ventas ”, del menú ventas que se encuentra en la pantalla principal.		
Precondiciones:	NVA		
Postcondición:	NVA		
Flujo normal:	1. Se ingresa la venta a buscar en el botón “ Buscar ”. 1.1 Se muestra la venta en la tabla. 2. Se selecciona el botón “ actualizar ” si así se desea. 3. Se selecciona la opción “ eliminar ” si es necesario. 4. Se da clic en el botón “ salir ”.		
Flujos alternos:			
Include:			
Frecuencia de uso:	Frecuente		
Requerimientos especiales	NVA		
Supuesto:	NVA		
ISSUES y notas:			

Figura 51: Documentación de Caso de uso 4 Consultar ventas.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso dar alta de productos.

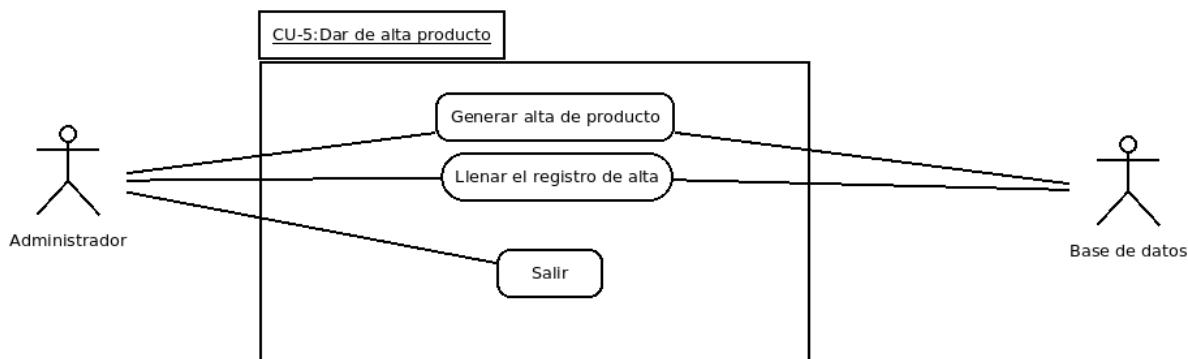


Figura 52: Caso de uso 5 alta de productos .

ID	CU-5		
Nombre:	Dar de alta producto.		
Creado por:	SLG	Actualizado:	SLG
Fecha de creación:	01/11/2021	Fecha de última revisión:	08/12/2021
Actores:	Administrador, base de datos.		
Descripción:	Se le muestra un formulario para que lo rellene en (Pantalla 5).		
Disparador:	Selecciona la opción “ Producto ” del menú principal (pantalla 2) y después selecciona la opción del submenu “ Dar alta ”.		
Precondiciones:	Ir a pantalla(2)		
Flujo normal	1. Presionar el botón “ Guardar ”. 1.1 El sistema agregara los datos en la tabla que se muestra en la (pantalla 6) y en la base de datos. 1.2 El sistema limpiara los campos que se fueron llenados.		
Flujo de alternos			
Include			
Frecuencia de uso	Muy frecuente.		
Requerimientos especiales	N/A		
Supuestos			
Issues y notas	N/A		

Figura 53: Documentación de caso de uso 5 dar alta de productos.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso consultar productos.

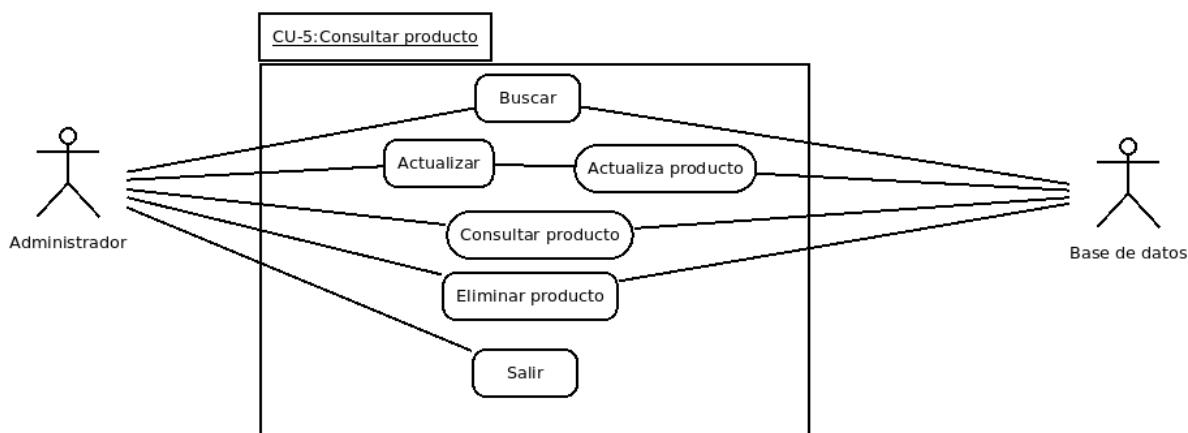


Figura 54: Caso de uso 6 consultar productos.

ID	CU-6		
Nombre:	Consultar producto		
Creado por:	SLG	Actualizado:	SLG
Fecha de creación:	01/11/2021	Fecha de última revisión:	08/12/2021
Actores:	Administrador, base de datos		
Descripción:	Se le muestra la consulta de los productos en una tabla la cual contiene ID, nombre de producto, cantidad, precio venta, precio unitario, descripción en la(pantalla 6)		
Disparador:	Selecciona la opción “ Producto ” del menú principal (pantalla 2) y después selecciona del submenú “ Consulta ” en la misma pantalla.		
Precondiciones:	Ir a (pantalla 2)		
Flujo normal	<p>1. El sistema mostrara una tabla de datos de los productos consultados de la base de datos.</p> <p>2. Ingresar código de producto.</p> <p>3. Presionar el botón “Buscar”.</p> <p>3.1 El sistema validará el código del producto ingresado en el paso 2 con la base de datos, si existe el producto se mostrará únicamente los datos de ese producto.</p> <p>4. Presiona el botón “Actualizar”.</p> <p>4.1. Se muestra un formulario de la pantalla emergente 6 donde se actualizara los datos del producto.</p> <p>5. Presiona el botón “Eliminar”.</p> <p>5.1. El sistema arrojara un mensaje de que “No hay fila seleccionada”.</p> <p>5.1.1 Si selecciono la fila, se elimina la fila seleccionada y los datos de la base de datos.</p> <p>6. Presiona el botón “Salir”</p> <p>6.1 Regresa a la (pantalla 2)</p>		
Flujo de alternos	N/A		
Include			
Frecuencia de uso	Muy frecuente		
Requerimientos especiales	N/A		
Supuestos			
Inssues y notas	N/A		

Figura 55: Documentación de caso de uso 6 consultar productos.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso actualizar.

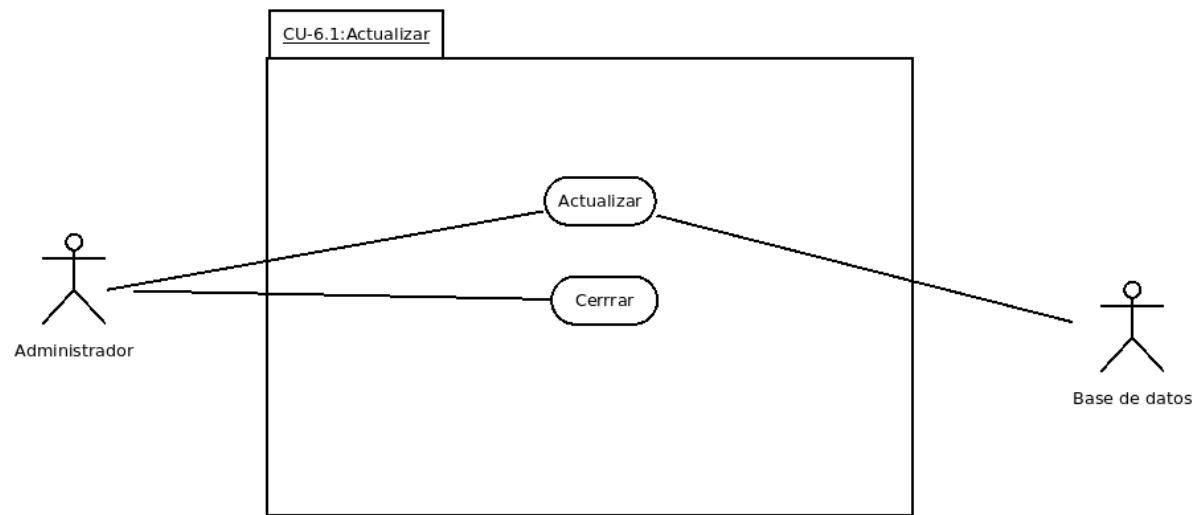


Figura 56: Caso de uso 6.1 actualizar.

ID	CU-6.1		
Nombre:	Actualizar		
Creado por:	SLG	Actualizado:	SLG
Fecha de creación:	01/11/2021	Fecha de última revisión:	08/12/2021
Actores:	Administrador, base de datos.		
Descripción:	Se le muestra un formulario con los datos ya existentes del producto (la pantalla 6.1) para modificar los datos del producto.		
Disparador:	Selecciona el botón “Actualizar” de la (pantalla6)		
Precondiciones:	Ir a pantalla(6)		
Flujo normal	1. Presionar el botón “Actualizar” 1.1 Se muestra un formulario de la (pantalla 6.1) donde se actualizara los datos. 2. Presiona el botón “Actualizar”. 2.1. Se actualizan los datos y se modifican los datos en la base de datos. 3. Presiona el botón “Salir” 3.1. Regresa a la (pantalla 5).		
Flujo de alternos	N/A		
Include			
Frecuencia de uso	Muy frecuente		
Requerimientos especiales	N/A		
Supuestos			
Inssues y notas	N/A		

Figura 57: Documentación de caso de uso 6.1 actualizar.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso agregar usuario.

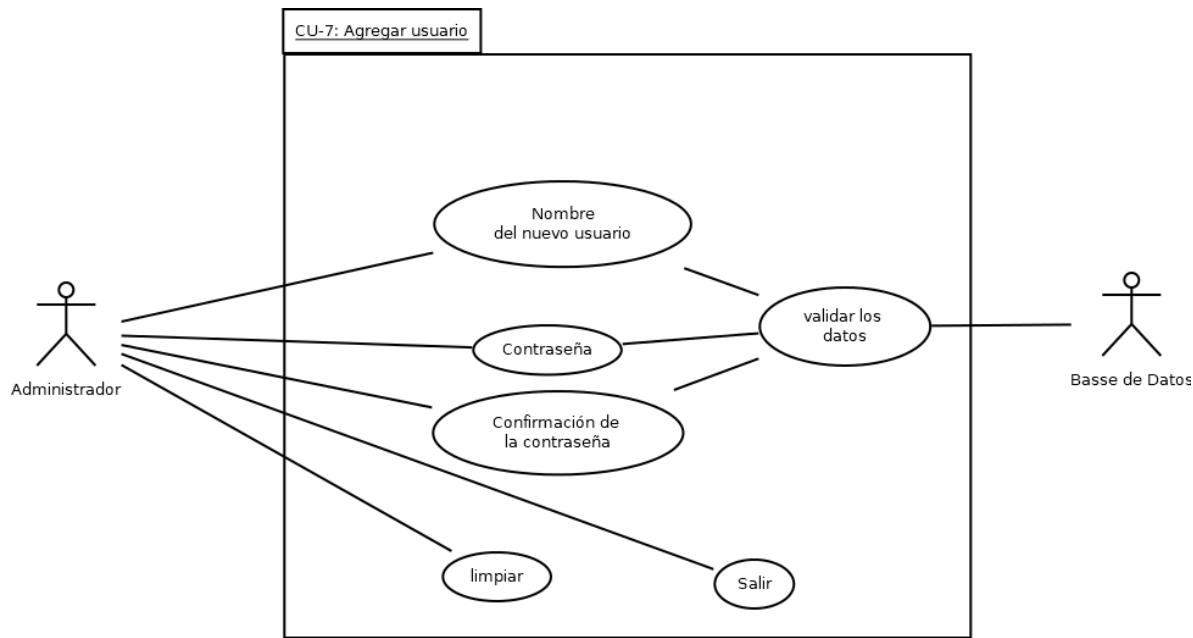


Figura 58: Caso de uso 7 de agregar usuario.

Id	CU-7				
Nombre	Aregar usuario				
Creado por:	VSG	Actualizado por:	VSG		
Fecha de creación:	2/12/2021	Fecha de última revisión	08/12/2021		
Actores:	Administrador				
Descripción:	Agrega usuarios				
Disparador:	Al presionar el botón de “Aregar Usuario”				
Precondiciones:					
Postcondición:	N/A				
Flujo normal:	1.- Ingresar nombre de usuarios. 2.-Ingresar contraseña. 3.-Ingresa la confirmación de la contraseña. 3.-Presionar el botón “Guardar”. 3.1.- El sistema valida los datos ingresados en el 1, 2 y 3. 3.1.1.- Si los datos son correctos limpia los campos de la pantalla. 3.1.2.- Sí los datos ya existe en la base de datos, enviará un mensaje “El usuario ya existe ”, regresar a 1, 2 y 3. 3.1.3.- Sí las contraseñas no coinciden, enviará un mensaje “Las contraseñas no coinciden”, regresar a 1, 2 y 3. 4.- Presionar el botón “Limpiar” 4.1.- limpia la los campos de CU-7. 5- Presionar el botón “Salir”. 5.1.- Salir del sistema.				
Flujos alternos:					
Include:					
Frecuencia de uso:	No muy frecuente				
Requerimientos especiales					
Supuesto:					
ISSUES y notas:					

Figura 59: Documentación de caso de uso 7 de agregar usuario.

Descripción: La siguiente figura hace referencia al diagrama de caso de uso consultar usuario.

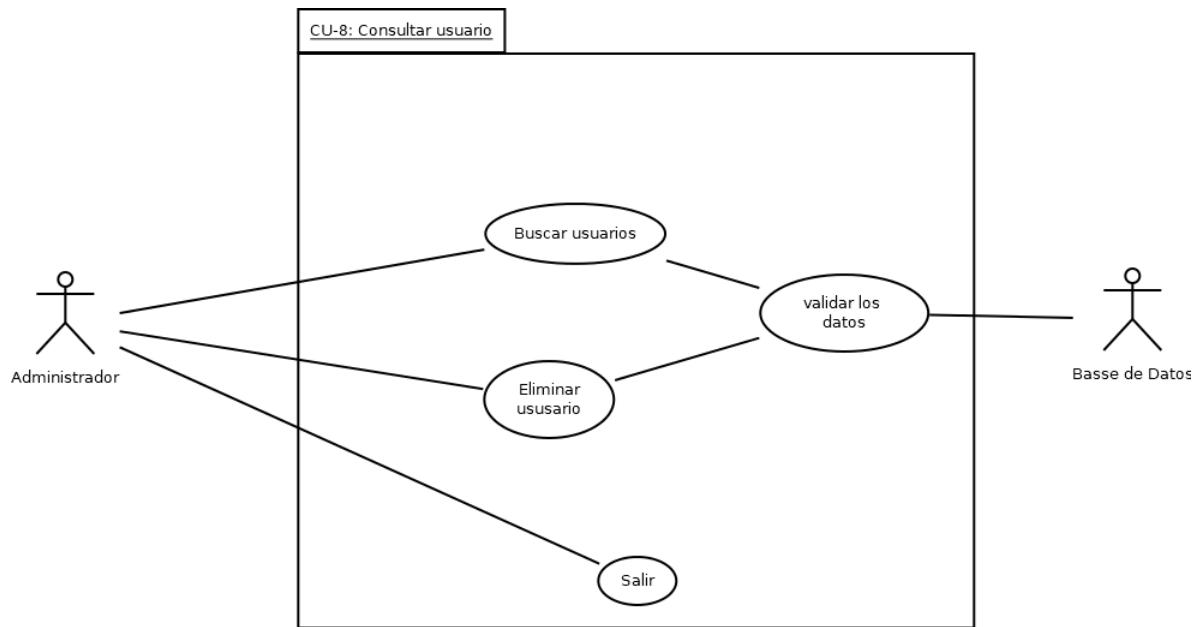


Figura 60: Caso de uso 8 de consultar usuario.

Id	CU-8				
Nombre	Consultar usuario				
Creado por:	VSG	Actualizado por:	VSG		
Fecha de creación:	2/12/2021	Fecha de última revisión	8/12/2021		
Actores:	Administrador				
Descripción:	Consulta usuario				
Disparador:	Al presionar el botón “ Consultar usuario ”;				
Precondiciones:					
Postcondición:	N/A				
Flujo normal:	1.- Ingresar el nombre del usuario en la barra de búsqueda. 2.-Automáticamente se rellena las tablas. 2.1.1.- Para eliminar una fila de la información se debe seleccionar y presionar el botón “ Eliminar ”. 2.1.2.- para actualizar la tabla se debe presionar el botón “ Actualizar ”. 5- Presionar el botón “ Salir ”. 5.1.- Salir del sistema.				
Flujos alternos:					
Include:					
Frecuencia de uso:	No muy frecuente				
Requerimientos especiales					
Supuesto:					
ISSUES y notas:					

Figura 61: Documentación de caso de uso 8 de consultar ususrio.

3.2.2 Diagramas de estado

Descripción: El siguiente código hace referencia al diagrama de estados del login.

```
@startuml
[*] --> Login
    Login: Iniciar Ventana De Logeo
    Datos: pedir usuario y contraseña
    Ingresar : Ingresar los datos
    Login --> Datos
    Datos --> Ingresar

    Ingresar-->Validacion
    Validacion-->IngresPrincipal:Datos correctos
    Validacion-->Error : Datos Incorrectos
    Error-->Ingresar
    IngresPrincipal-->[*]

@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados del login.

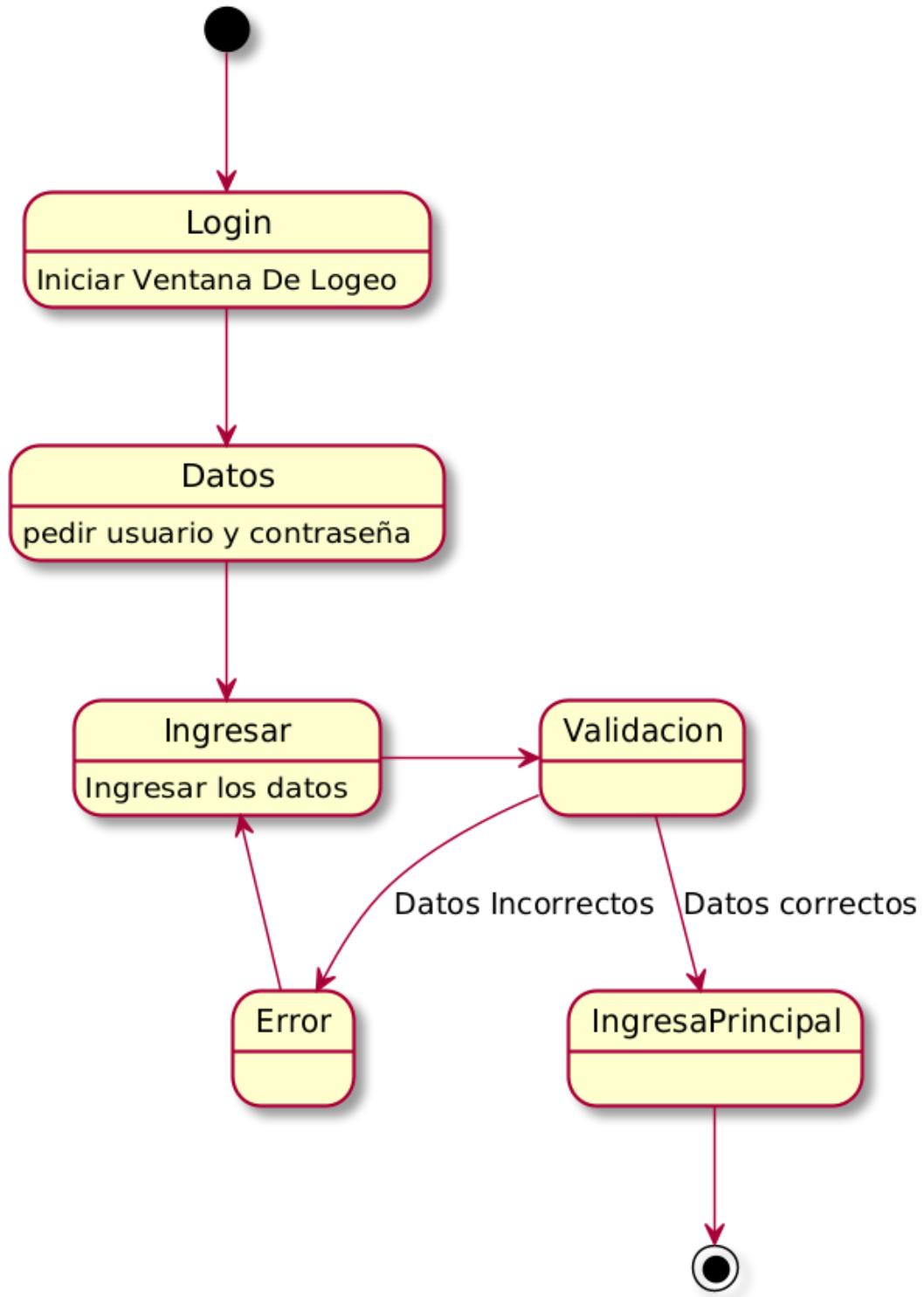


Figura 62: Diagrama de estados del login

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana principal.

```
@startuml

[*] --> VentanaPrincipal

VentanaPrincipal--> Inicio : ir a
Inicio-->CerrarSesion
VentanaPrincipal-->Ventas: ir a
Ventas-->RealizaVenta
Ventas-->ConsultaVenta
VentanaPrincipal-->Producto: ir a
Producto-->AltaProducto
Producto-->ConsultaProducto
VentanaPrincipal-->Usuario: ir a
Usuario-->AltaUsuario
Usuario-->ConsultaUsuario

CerrarSesion-->[*]
RealizaVenta --> [*]
ConsultaVenta --> [*]
AltaProducto --> [*]
ConsultaProducto --> [*]
AltaUsuario--> [*]
ConsultaUsuario--> [*]

@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados de la ventana principal.

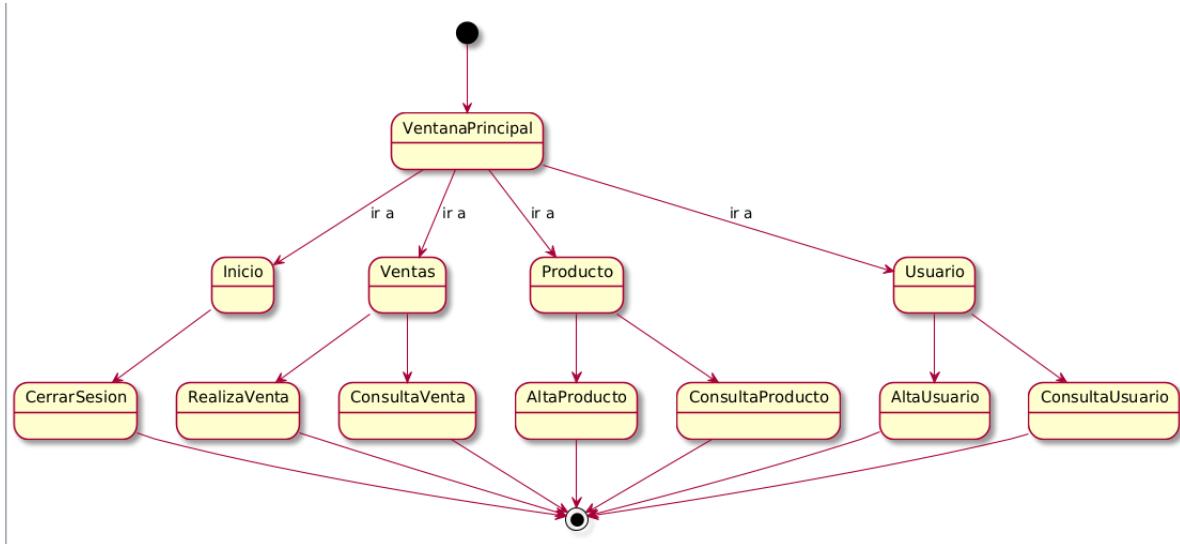


Figura 63: Diagrama de estados de la interfaz principal

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana realiza ventas.

```
@startuml  
[*] --> SeleccionarProducto  
  
SeleccionarProducto --> MostrarProductos: ir a  
MostrarProductos --> CobrarCuenta  
MostrarProductos --> EliminaProductos  
CobrarCuenta --> RealizaVenta  
CobrarCuenta --> CancelarVenta  
RealizaVenta --> [*]  
CancelarVenta --> [*]  
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados de la ventana realiza ventas.

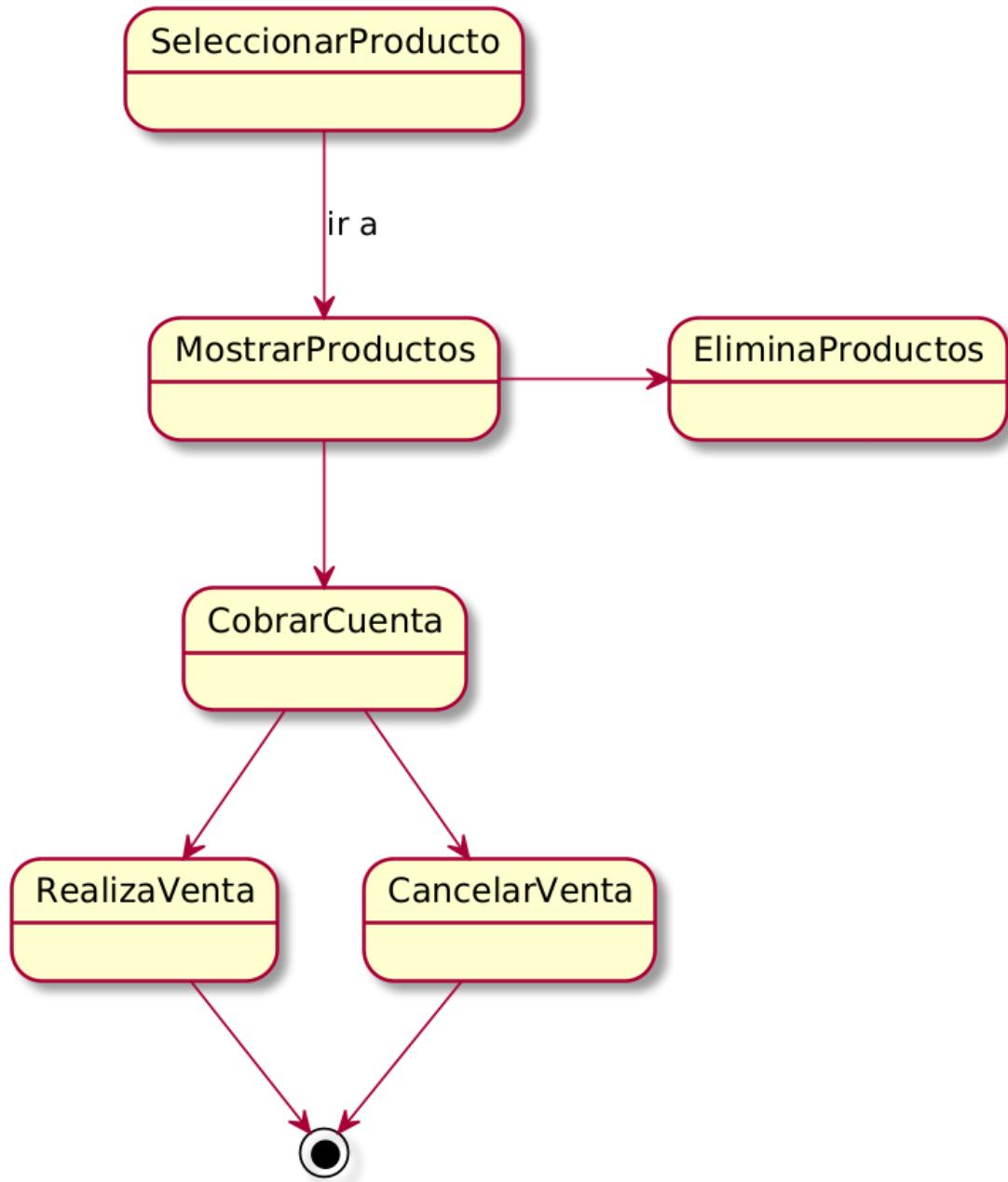


Figura 64: Diagrama de estados de la interfaz realiza venta

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana agregar producto de venta.

```
@startuml  
[*] --> BuscarProducto  
  
BuscarProducto--> MostrarProducto: ir a  
MostrarProducto-->AregarProducto  
AregarProducto --> [*]  
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados de la ventana agregar producto de venta

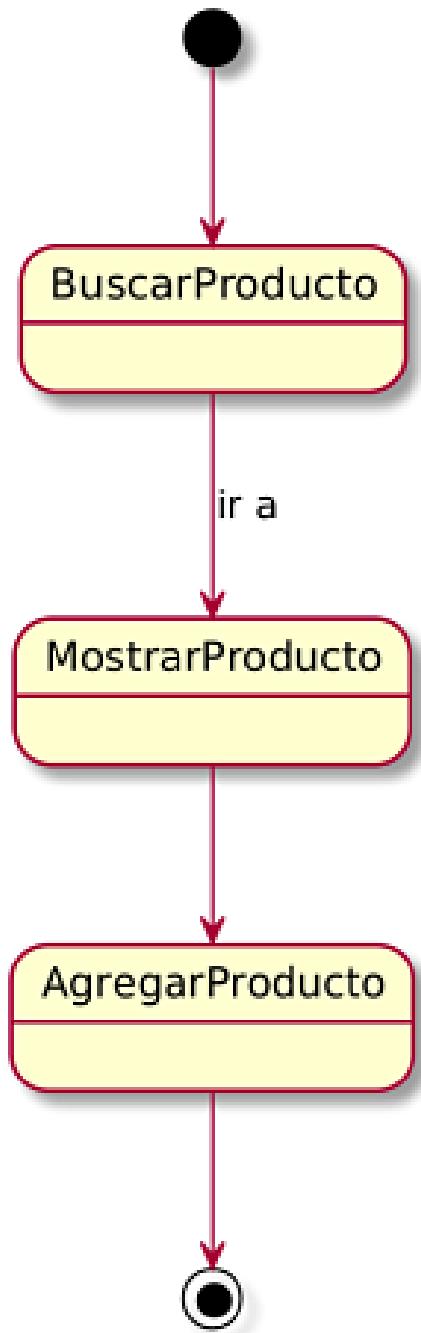


Figura 65: Diagrama de estados de la interfaz agragar producto de venta

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana consulta ventas.

```
@startuml

[*] --> BuscarVenta

BuscarVenta--> MostrarVenta: ir a
MostrarVenta-->Actualizar
MostrarVenta-->Eliminar
MostrarVenta-->GenerarExcel
Actualizar--> [*]
Eliminar--> [*]
GenerarExcel-->[*]

@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados de la ventana consulta ventas.

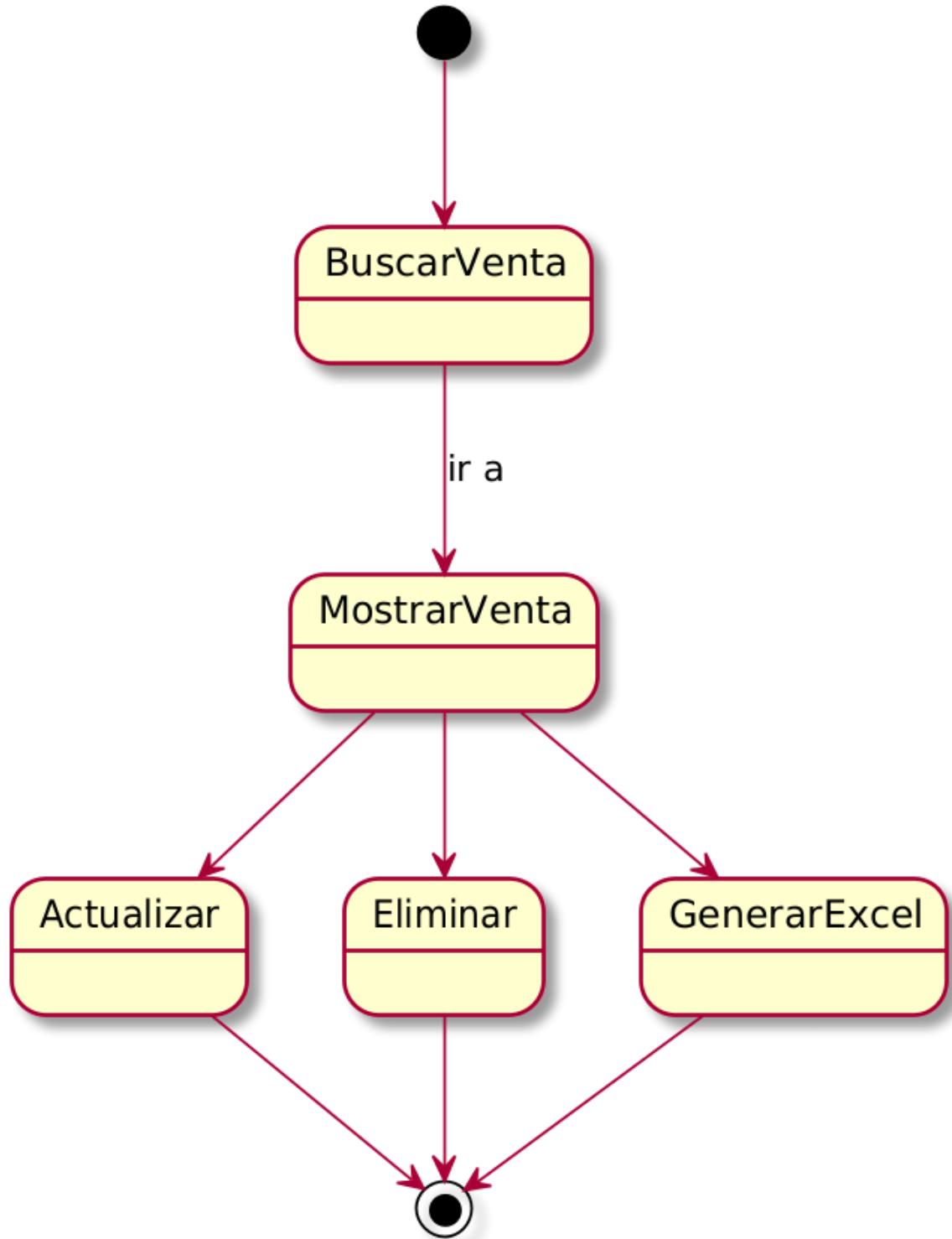


Figura 66: Diagrama de estados de la interfaz consulta ventas

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana de alta de productos.

```
@startuml  
  
[*] --> AltaProducto  
  
AltaProducto--> IngresaDatos  
IngresaDatos-->Validar  
Validar-->Guardar : Datos correctos  
Validar-->Error : Datos incorrectos  
Error-->IngresaDatos  
Guardar--> [*]  
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados de la ventana de alta de productos.

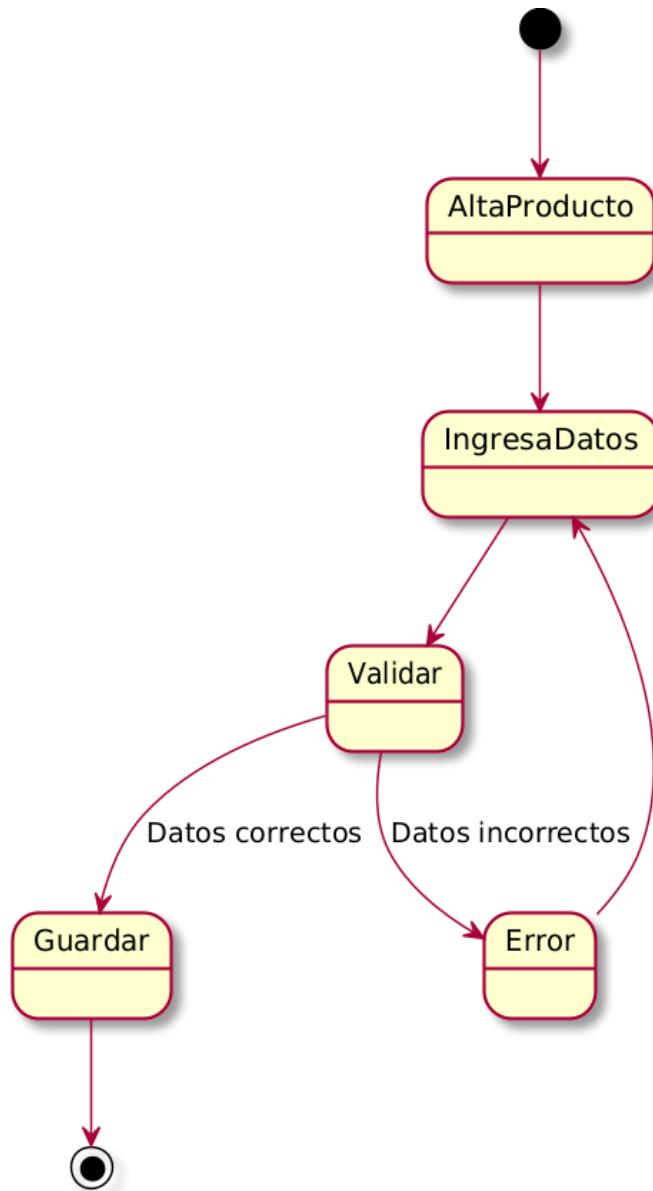


Figura 67: Diagrama de estados de la interfaz alta de productos

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana consulta productos.

```
@startuml  
[*] --> BuscarProducto  
  
BuscarProducto--> MostrarProducto  
MostrarProducto-->Actualizar  
MostrarProducto-->Eliminar  
MostrarProducto-->GenerarExcel  
Actualizar--> [*]  
Eliminar--> [*]  
GenerarExcel--> [*]  
  
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados de la ventana consulta productos.

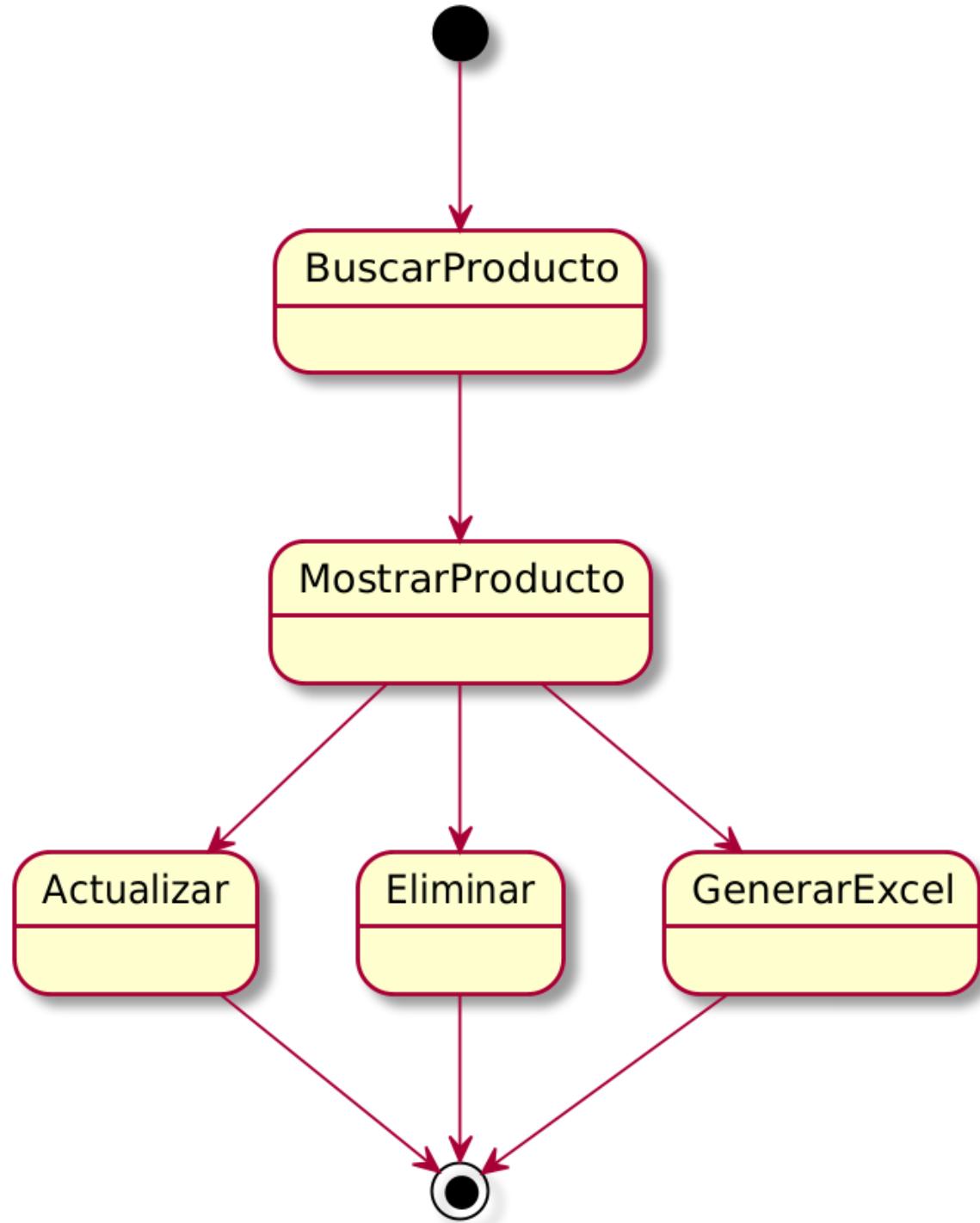


Figura 68: Diagrama de estados de la interfaz consulta productos

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana actualiza productos.

```
@startuml  
[*] --> ActualizarProducto  
  
ActualizarProducto --> IngresaDatos  
IngresaDatos --> Validar  
Validar --> Actualizar : Datos correctos  
Validar --> Error : Datos incorrectos  
Error --> IngresaDatos  
Actualizar --> [*]  
  
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados de la ventana actualiza productos.

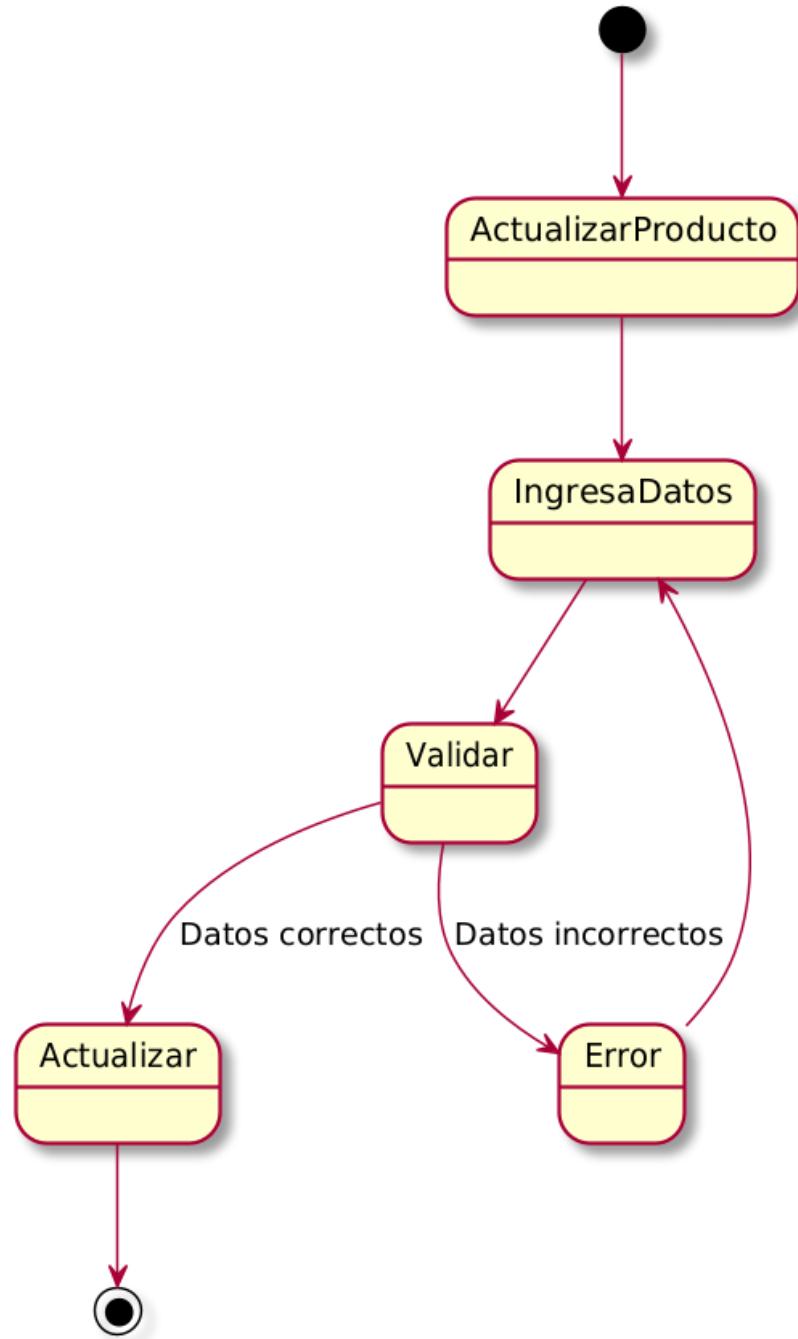


Figura 69: Diagrama de estados de la interfaz actualiza productos

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana alta usuarios.

```
@startuml  
[*] --> AltaUsuario  
  
AltaUsuario--> DatosUsuario: ir a  
DatosUsuario-->Validar  
Validar-->Guardar  
Validar-->Error  
Error--> DatosUsuario  
Guardar-->[*]  
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de estados de la ventana alta usuarios.

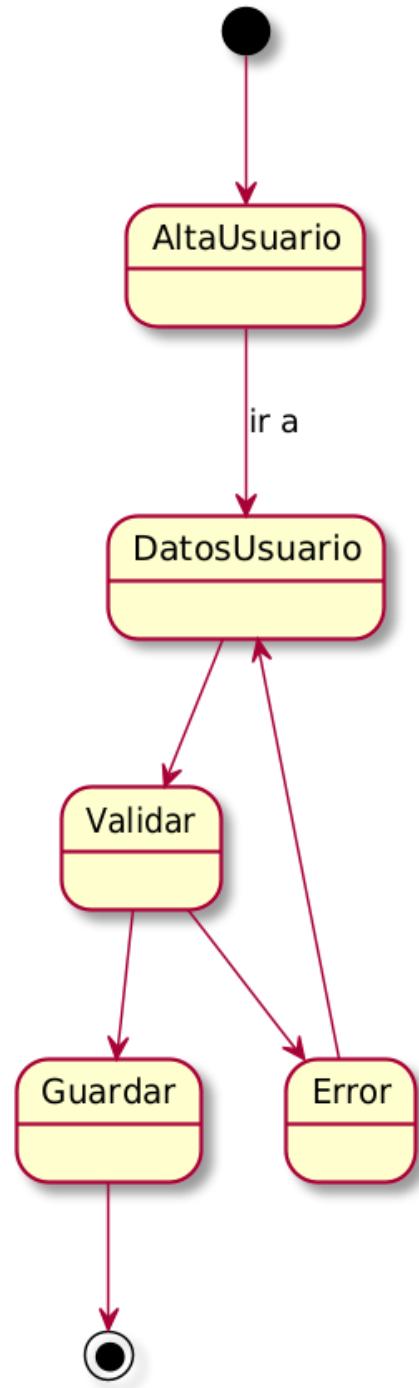


Figura 70: Diagrama de estados de la interfaz alta usuarios

Descripción: El siguiente código hace referencia al diagrama de estados de la ventana consulta usuarios.

```
@startuml  
[*] --> BuscarUsuario  
  
BuscarUsuario--> MostrarUsuario: ir a  
MostrarUsuario-->ActualizarUsuario  
MostrarUsuario-->eliminarUsuario  
ActualizarUsuario-->[*]  
eliminarUsuario-->[*]  
@enduml
```

Descripción: la siguiente figura Hace referencia al diagrama de estados de la ventana consulta usuarios.

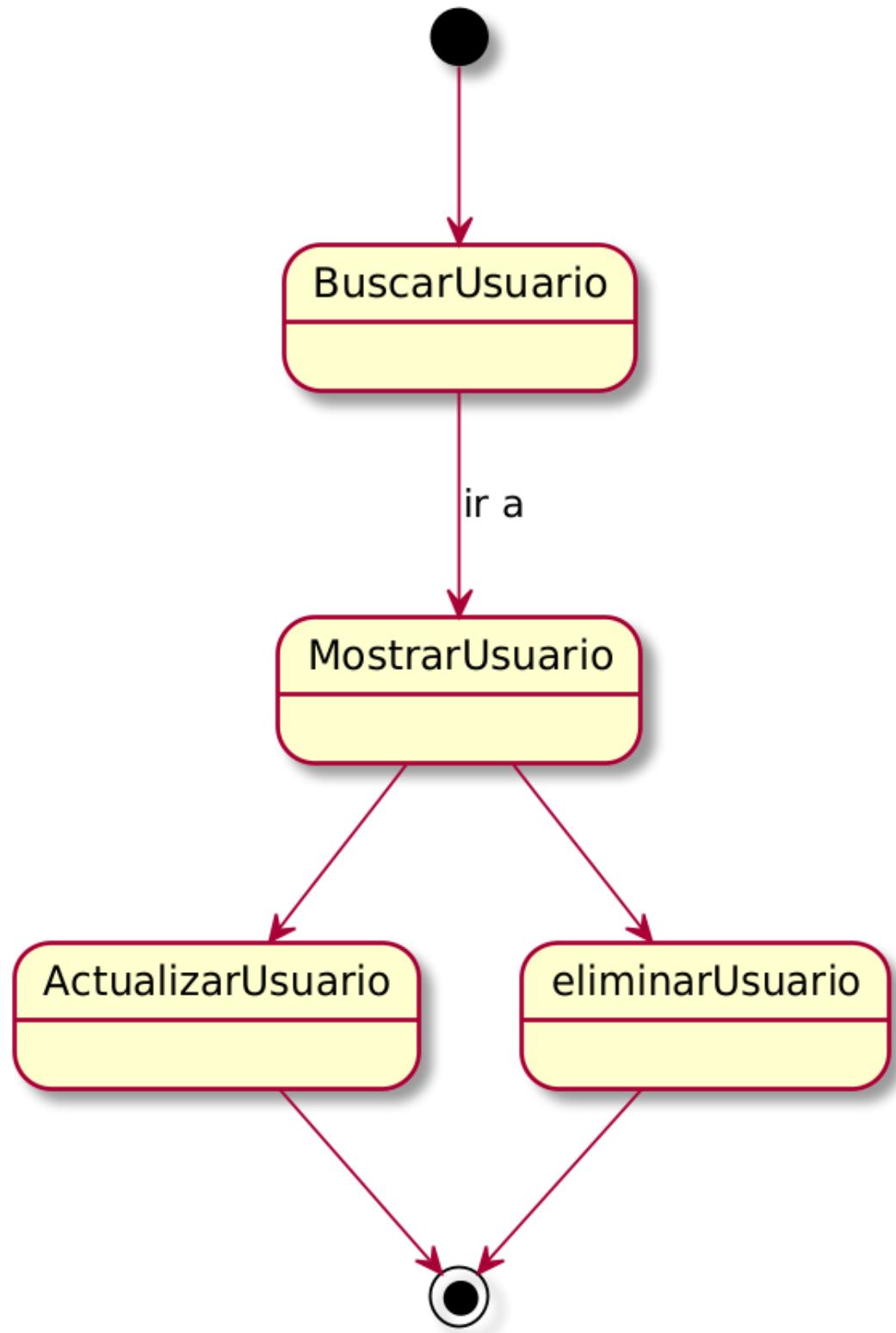


Figura 71: Diagrama de estados de la interfaz consulta usuarios

3.2.3 Diagramas de secuencia

Descripción: El siguiente código hace referencia al diagrama de secuencia consulta venta.

```
@startuml
Actor Administrador
database Base_de_datos
Administrador->Base_de_datos:1.- Consultar las ventas realizadas
Activate Base_de_datos
Base_de_datos->Base_de_datos:2.- Digitar la fecha de venta
Base_de_datos-->Administrador:3.- Reporte de la venta buscada
deactivate Base_de_datos

@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de secuencia consultar venta.

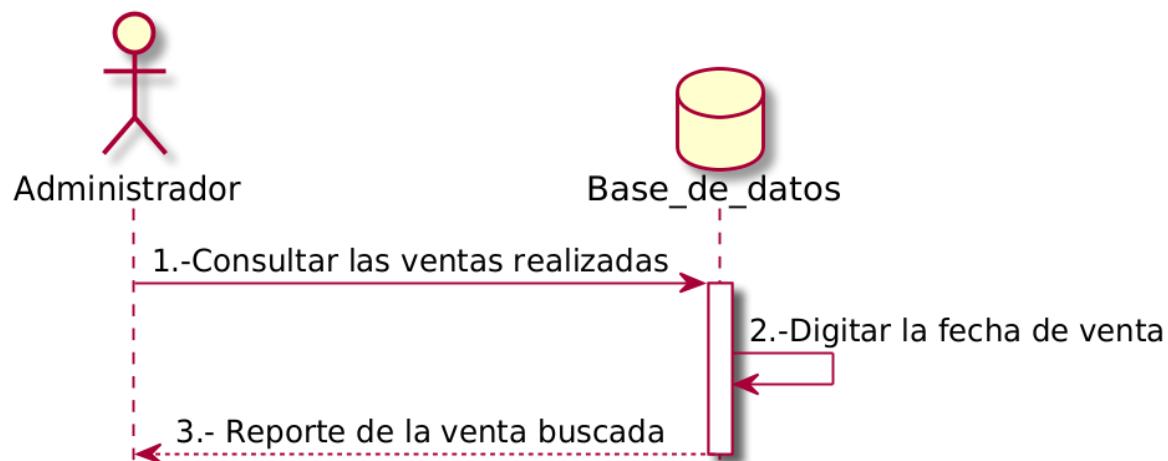


Figura 72: Diagrama de secuencia consultar venta

Descripción: El siguiente código hace referencia al diagrama de secuencia inicio de sesión .

```
@startuml
Actor Administrador_Usuario
Administrador_Usuario --> Administrar_datos:1. – Ingresar datos
Activate Administrar_datos
Administrar_datos --> Administrar_datos:2. – Ingresa usuario
Administrar_datos --> Administrar_datos:3. – Ingresa contraseña a
database Base_de_datos
Administrar_datos --> Base_de_datos:4. – Validar información
deactivate Administrar_datos
Activate Base_de_datos
Base_de_datos --> Administrador_Usuario:5. – Ingreso al sistema
deactivate Base_de_datos
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de secuencia inicio de sesión.

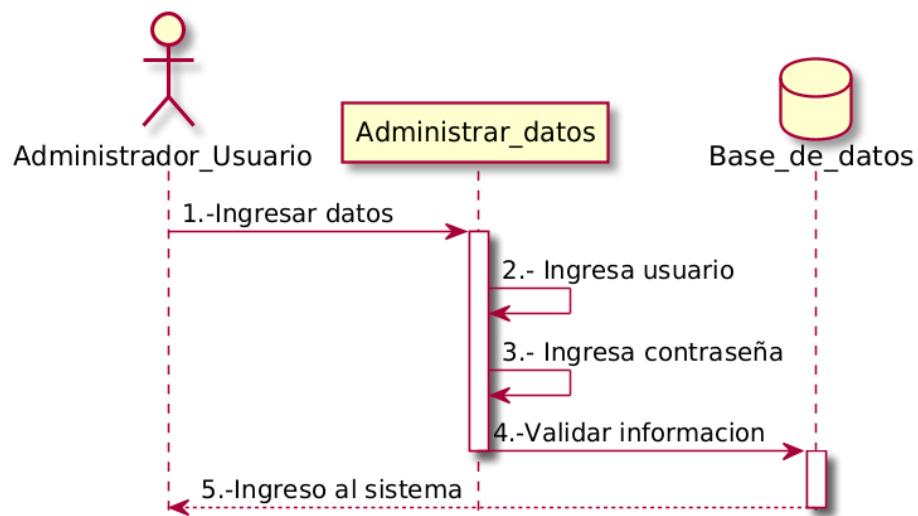


Figura 73: Diagrama de secuencia inicio de sesion

Descripción: El siguiente código hace referencia al diagrama de secuencia inventario. .

```
@startuml
Actor Administrador
Administrador->Opcion:1.- Buscar producto
Activate Opcion
Opcion->Administrador_datos:2.- Insertar nombre de producto
deactivate Opcion
Activate Administrador_datos
database Base_de_datos
Administrador_datos ->Base_de_datos:3.- Verificar busqueda
deactivate Administrador_datos
Activate Base_de_datos
Base_de_datos-->Administrador:4.- Producto buscado
deactivate Base_de_datos

Administrador->Opcion:5.- Agregar producto
Activate Opcion
Opcion->Administrador_datos:6.- Proporcionar datos de producto
deactivate Opcion
Activate Administrador_datos
Administrador_datos->Administrador_datos:7.- Digitar datos del producto
Administrador_datos ->Base_de_datos:8.- Validar datos de producto
deactivate Administrador_datos
Activate Base_de_datos
Base_de_datos-->Administrador:9.- Producto agregado al inventario
deactivate Base_de_datos

Administrador->Opcion:10.- Cambiar datos del producto
Activate Opcion
Opcion->Administrador_datos:11.- Digtar datos del producto
deactivate Opcion
Activate Administrador_datos
Administrador_datos->Administrador_datos:12.- Digitar datos del producto
Administrador_datos ->Base_de_datos:13.- Validar datos ingresados
deactivate Administrador_datos
Activate Base_de_datos
```

```
Base_de_datos-->Administrador:14.- La actualizacion se realizo con exito
deactivate Base_de_datos

Administrador-->Opcion:15.- Eliminar producto del inventario
Activate Opcion
Opcion-->Administrar_datos:16.- Seleccionar producto a eliminar
deactivate Opcion
Activate Administrar_datos

Administrar_datos-->Base_de_datos:17.- Confirmar eliminacion del producto
deactivate Administrar_datos
Activate Base_de_datos
Base_de_datos-->Administrador:18.- Producto eliminado
deactivate Base_de_datos
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de secuencia inventario.

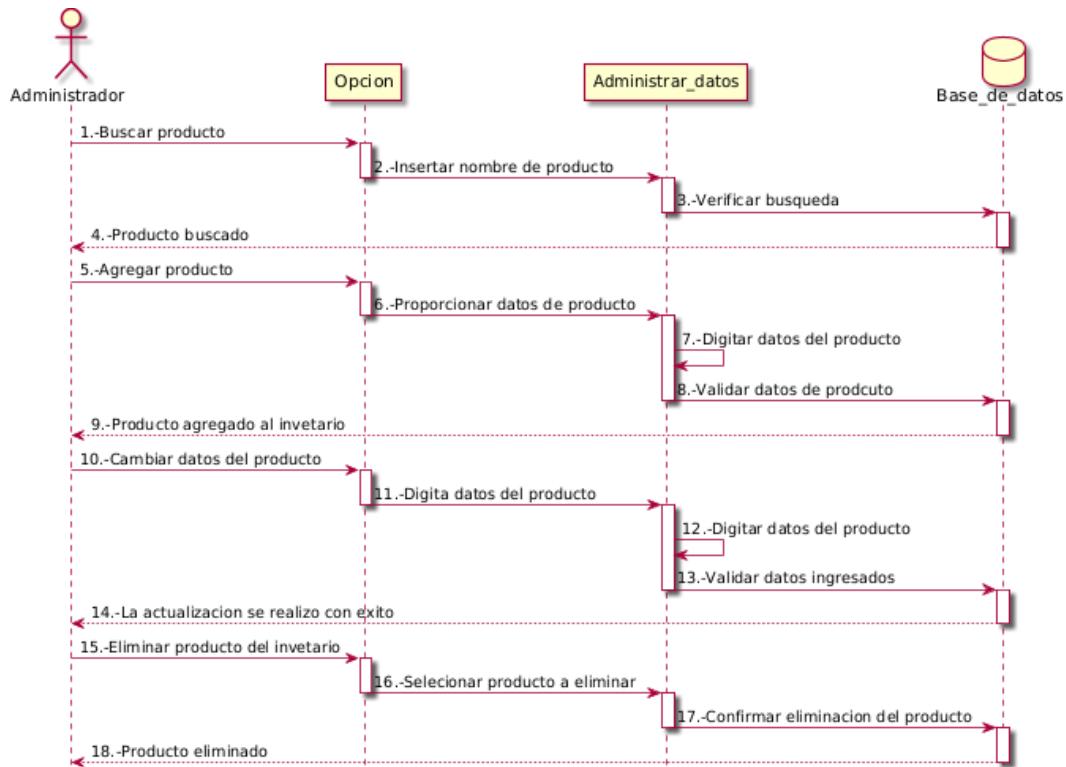


Figura 74: Diagrama de secuencia inventario

Descripción: El siguiente código hace referencia al diagrama de registro de usuarios.

```
@startuml
Actor Usuario

Usuario --> Administrar_contenido:1. - Registrar nuevo usuario
Activate Administrar_contenido
Administrar_contenido --> Administrar_contenido:2. - Digitar el nombre
Administrar_contenido --> Administrar_contenido:3. - Digitar el correo
Administrar_contenido --> Administrar_contenido:4. - Digitar contraseña a
database Base_de_datos
Administrar_contenido --> Base_de_datos:5. - Validar información
    proporcionada
deactivate Administrar_contenido
Activate Base_de_datos
Base_de_datos --> Usuario:6. - Se registró un nuevo usuario
deactivate Base_de_datos
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de secuencia registro de usuarios.

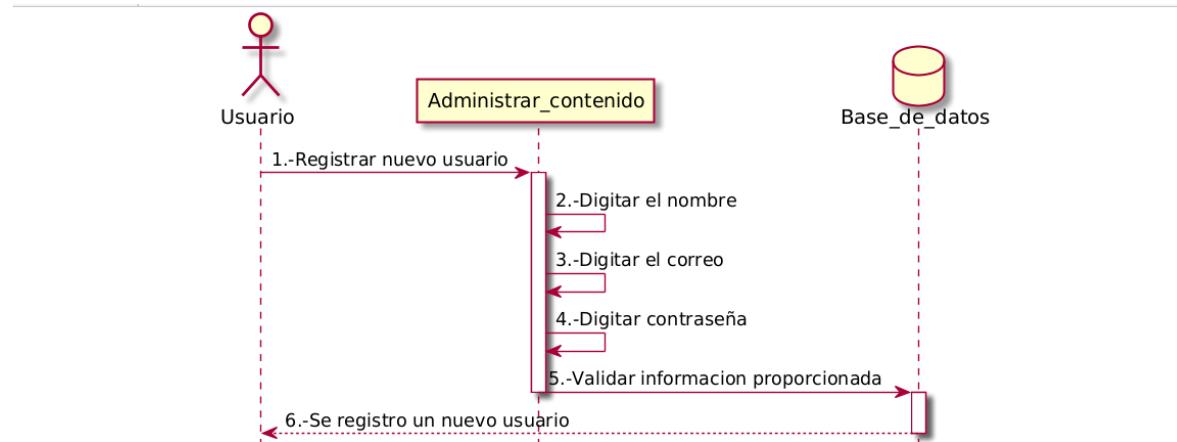


Figura 75: Diagrama de secuencia registro de usuarios

Descripción: El siguiente código hace referencia al diagrama de secuencia usuario.

```
@startuml
Actor Administrador
Administrador->Opcion:1. – Busqueda de usuario
Activate Opcion
Opcion->Administrar_contenido:2. – Digite el nombre de usuario a buscar
deactivate Opcion
Activate Administrar_contenido
Administrar_contenido->Administrar_contenido:3. – Nombre ususario
database Base_de_datos
Administrar_contenido->Base_de_datos:4. – Validar informacion
Activate Base_de_datos
deactivate Administrar_contenido
Base_de_datos-->Administrador:5. – Busqueda exitosa
deactivate Base_de_datos

Administrador->Opcion:6. – Modificar usuario
Activate Opcion
Opcion->Administrar_contenido:7. – Informacion del usuario
Activate Administrar_contenido
Administrar_contenido->Administrar_contenido :Nombre
Administrar_contenido->Administrar_contenido :Direccion
Administrar_contenido->Administrar_contenido :Telefono
Administrar_contenido->Administrar_contenido :

Administrar_contenido->Base_de_datos:8. – Validaci n de informacion
deactivate Opcion
deactivate Administrar_contenido
Activate Base_de_datos
Base_de_datos--> Administrador:9. – Datos modificados
deactivate Base_de_datos

Administrador->Opcion:10. – Eliminar usuario
Activate Opcion
Opcion->Base_de_datos:11. – Confirmar la eliminacion de usuario
```

```
deactivate Opcion
Activate Base_de_datos

Base_de_datos-->Administrador:Usuario eliminado
deactivate Base_de_datos
@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de secuencia usuario.

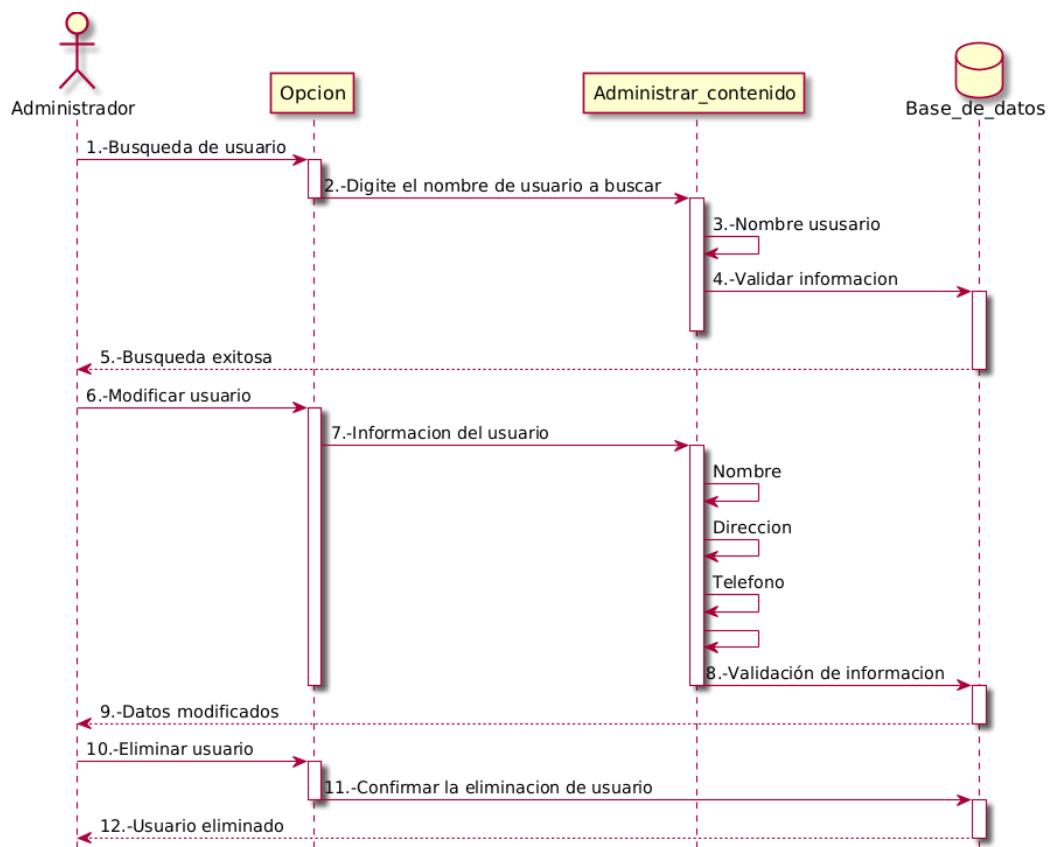


Figura 76: Diagrama de secuencia usuario

Descripción: El siguiente código hace referencia al diagrama de secuencia venta.

```
@startuml
Actor Administrador_o_Usuario
Administrador_o_Usuario-> Administrar_datos:1. – Recibir el monto total de
la venta
Activate Administrar_datos
deactivate Administrar_datos
Administrador_o_Usuario-> Administrar_datos:2. – Ingresar la cantidad a
cobrar
Activate Administrar_datos
deactivate Administrar_datos
Administrador_o_Usuario-> Administrar_datos:3. – Cobrar monto
Activate Administrar_datos
database Base_de_datos
Administrar_datos-> Base_de_datos
deactivate Administrar_datos
Activate Base_de_datos
Base_de_datos--> Administrador_o_Usuario:4. – Venta realizada con éxito
deactivate Base_de_datos

@enduml
```

Descripción: la siguiente figura hace referencia al diagrama de secuencia venta.

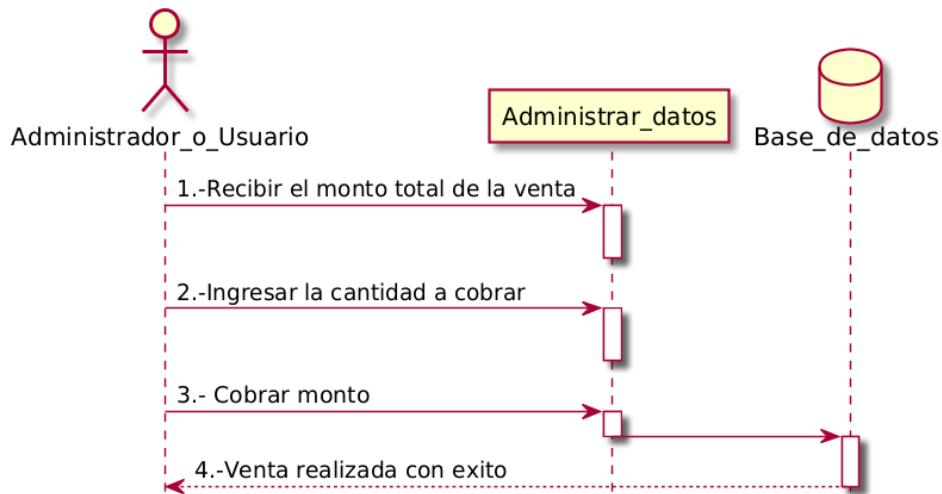


Figura 77: Diagrama de secuencia venta

3.2.4 Diagramas de colaboración

Descripción: la siguiente figura hace referencia al diagrama de colaboración.

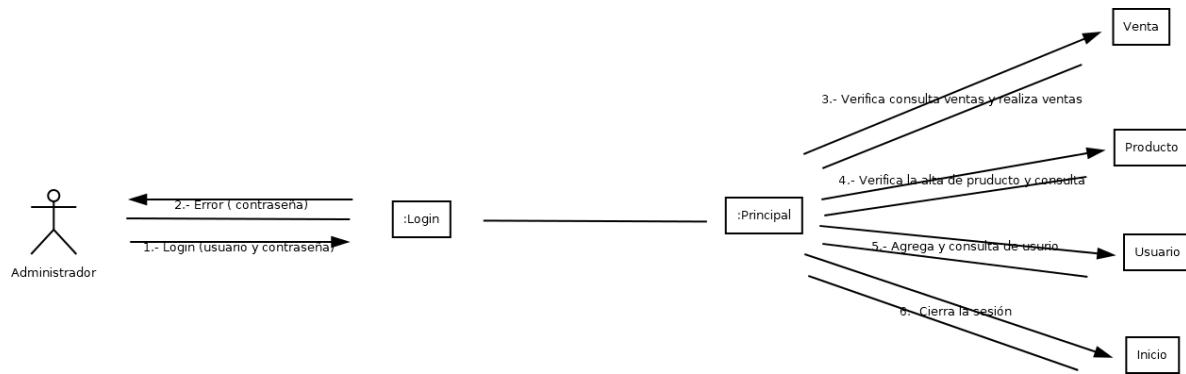


Figura 78: Diagrama de colaboración del administrador

Descripción: la siguiente figura hace referencia al diagrama de colaboración.

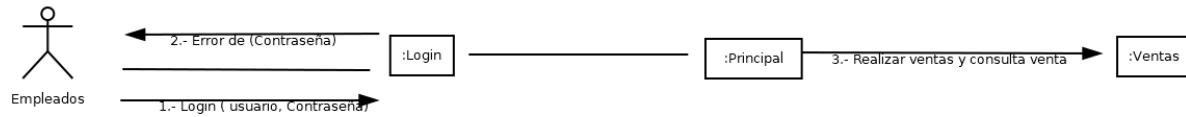


Figura 79: Diagrama de colaboración del empleado

4 Diseño

4.1 Introducción

A continuación, se describe todos los recursos que se utilizaron en el desarrollo del software, los cuales fueron el sistema de cómputo, la arquitectura como física y lógica, el diseño de datos y dirección de datos.

4.2 Descripción del sistema

El sistema Ferrimax se encuentra integrado por tres apartados, el primero es la sección de ventas, ahí se encuentra dos ventanas, la primera realiza las ventas que sean necesarias en la ferreterías, y la segunda es la consulta de ventas, donde puede consultar que ventas se han realizado a través de los días. Por otra parte se encuentra la sección de productos, con dos ventanas, una para la alta de productos y la segunda para la consulta, modificación de productos y eliminación de los mismos. La ultima sección es la de usuarios, ahí se pueden dar de alta a los usuarios, además de consultarlos, todas estas partes forman el sistema punto de venta Ferrimax.

4.3 Arquitectura del sistema

4.3.1 Física

Para poder realizar el punto de ventas FERRIMAX se ocuparon diferentes equipos los cuales contaban con hardware y software capaz de cumplir las necesidades del proyecto.

Descripción: La siguiente figura hace referencia a las características con las que contaba la computadora que se ocupó para desarrollar FERRIMAX.

Nº	Memoria de almacenamiento	Memoria RAM	Procesador	Sistema operativo
1	500.1 GB	7.7 GiB	Intel® Core™ i5-4570 CPU @ 3.20GHz × 4	Ubuntu 20.04.3 LTS
2	762.2 GB	13.7 GiB	AMD® Ryzen 5 3400g with radeon vega graphics × 8	Ubuntu 20.04.3 LTS

Figura 80: Componentes de los equipos utilizados

Descripción: La siguiente figura hace referencia a los materiales físicos que se ocuparon para desarrollar FERRIMAX.

Nº	Nombre	Cantidad	Precio	Descripción
1	Hojas de libreta	75	\$20	Conjunto de hojas de papel, impresas o en blanco, unidas con una espiral o dobladas, encajadas o cosidas, que forman un libro delgado.
2	Lapiceros	5	\$10	Es un instrumento de escritura o de dibujo consistente en una mina o barrita de pigmento
3	Lápiz	2	\$10	Utensilio para escribir o dibujar.
Total			\$40	

Figura 81: Componentes de los equipos utilizados

4.3.2 Lógica

Para poder realizar el punto de ventas FERRIMAX se ocuparon diferentes softwares los cuales contaban con diferentes características. A continuación se muestra los software utilizados durante el desarrollo del proyecto Ferirmax

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.



Figura 82: Entorno de desarrollo

PostgreSQL Es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL.



Figura 83: Componentes de los equipos utilizados

Pgadmin4 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.

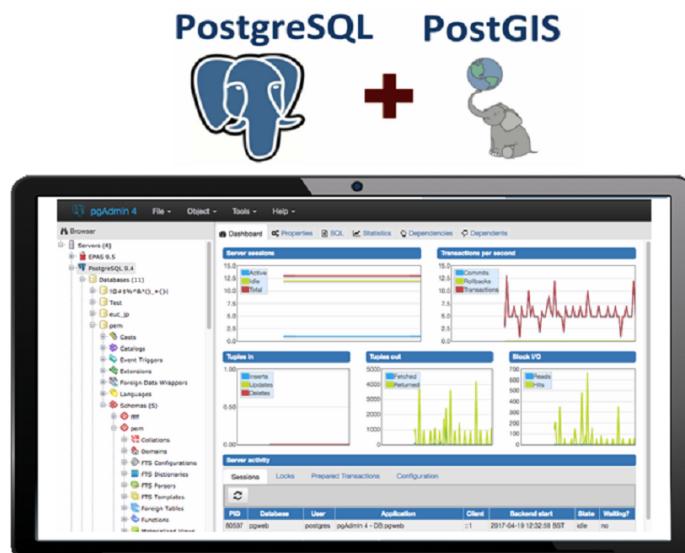


Figura 84: Software pgadmin4

Photoshop es un programa de edición de imágenes desarrollado por Adobe Systems Incorporated, que fue creado en 1987 y sirve para editar o crear imágenes con el uso de varias herramientas como pinceles, lápices, rellenos, formas, textos, relieves o efectos, entre otros.



Figura 85: Logotipo de photoshop

Dia es una aplicación informática de propósito general para la creación de diagramas, creada originalmente por Alexander Larsson, y desarrollada como parte del proyecto GNOME. Está concebido de forma modular, con diferentes paquetes de formas para diferentes necesidades.

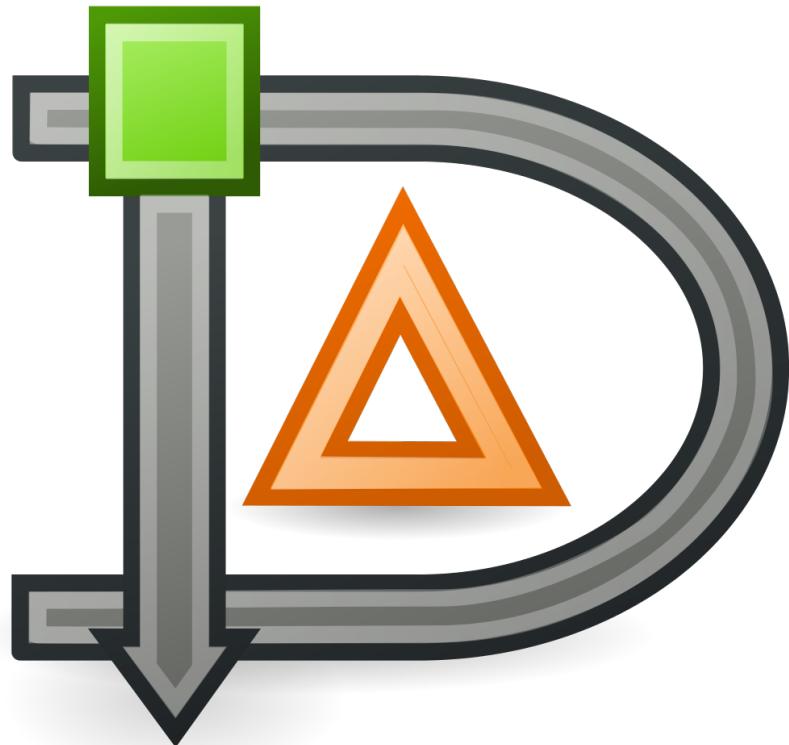


Figura 86: Logotipo de dia

Process Dashboard es una iniciativa de código abierto para crear una herramienta de soporte de PSP(SM) / TSP(SM).

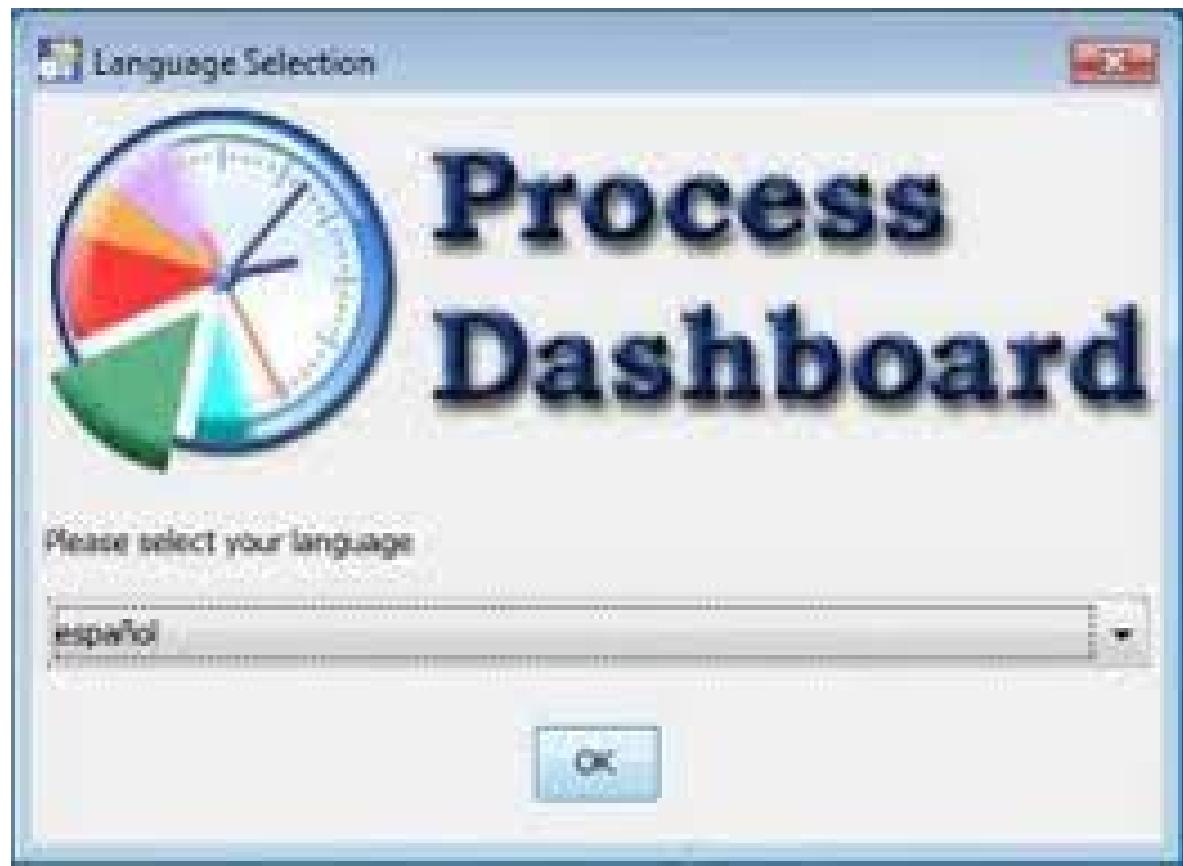


Figura 87: Logotipo de Process Dashboard

Github es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails.



Figura 88: Logotipo de Github

4.4 Estándar de codificación

4.4.1 Nombre de variable

El tipo de variable a utilizar es el UpperCamelCase, consiste en que la primera letra de cada una de las palabras será en mayúscula.

Ejemplo:

VentanaNumero

4.4.2 Sangría

La sangría será una implementada por defecto de Apache NetBeans IDE.

4.4.3 Comentario

Comentarios de varias líneas

Para describir un método completo en un código o un fragmento, se utilizará el comentario de varias líneas.

Ejemplo:

Sintaxis:

/*El comentario comienza

continúa

continúa

.

.

El comentario termina*/

Comentarios de comienzo

Todos los ficheros fuente deben comenzar con un comentario en el que se lista el nombre de la clase, información de la versión, fecha, y copyright:

```
/*
 * Nombre de la clase
 *
 * Informacion de la version
 *
 * Fecha
 *
 * Copyright
 */
```

4.4.4 Indentación

Deben tomarse 4 espacios como unidad de indentación. La construcción exacta de la indentación (espacios o tabuladores) no es crítica. Los tabuladores deben estar fijados exactamente cada 8 espacios (no cada 4).

4.4.5 Métodos

Los métodos deben ser verbos escritos en minúsculas. Cuando estos estén compuestos por varias palabras se tendrá la primera palabra en mayúscula.

Ejemplo:

```
public void insertaUnidad(Unidad unidad)
public void eliminaAgenda(Agenda agenda)
public void actualizaTramite(Tramite tramite)
```

4.4.6 Espacios

Se deben usar espacios en blanco en las siguientes circunstancias: Una palabra clave del lenguaje seguida por un paréntesis debe separarse por un espacio.

Hay que notar que no se debe usar un espacio en blanco entre el nombre de un método y su paréntesis de apertura. Esto ayuda a distinguir palabras claves de llamadas a métodos.

Debe aparecer un espacio en blanco después de cada coma en las listas de argumentos. Todos los operadores binarios excepto se deben separar de sus operandos con espacios en blanco. Los espacios en blanco no deben separar los operadores unarios, incremento ("++") y decremento ("–") de sus operandos. Ejemplo:

```
a += c + d;  
a = (a + b) / (c * d);
```

Las expresiones en una sentencia for se deben separar con espacios en blanco.

Ejemplo:

```
for (expr1; expr2; expr3)
```

4.4.7 Encabezado

En el uso de encabezados se utiliza el siguiente formato:

```
/*  
*****  
*Autor: Vicente Leonel Vásquez  
*Fecha creación: 15 de Enero 2022  
*Fecha actualización 17 de Enero 2022  
*Descripción: La siguiente clase realiza a través de un  
* método el proceso de venta de nuestro sistema  
*  
*  
*****  
:*
```

Figura 89: Formato de encabezado.

4.4.8 Número de caracteres por línea

Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

Nota: Ejemplos para uso en la documentación deben tener una longitud inferior, generalmente no más de 70 caracteres.

4.4.9 Nombre de interfaces

Partes de la declaración de una clase o interfaces	Notas
Comentario de implementación de la clase o interfaz si fuera necesario (*...*)	Este comentario debe contener cualquier información aplicable a toda la clase o interfaz que no era apropiada para estar en los comentarios de documentación de la clase o interfaz.
Variables de clase (static)	Primero las variables de clase public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
Variables de instancia	Primero las public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
Métodos	Estos métodos se deben agrupar por funcionalidad más que por visión o accesibilidad. Por ejemplo , un método de clase privado puede estar entre dos métodos públicos de instancia. El objetivo es hacer el código más legible y comprensible.

Figura 90: Partes de la declaración de una clase o interfaces.

4.4.10 Nombre de clases

Los nombres de las clases deben ser sustantivos, en mayúsculas y minúsculas, con **la primera letra de cada palabra interna en mayúscula**. El nombre de las interfaces también debe estar en mayúscula (la primera) al igual que los nombres de las clases. Use palabras completas y se debe evitar acrónimos y abreviaturas.

Por ejemplo:

Interface Bicycle

Class MountainBike implements Bicycle

Interface Sport

Class Football implements Sport

4.4.11 Nombre de Objetos

Para el uso de ellos objetos la declaración se realiza de la siguiente manera:

MiClase miObjeto = new MiClase(); //Declaramos y creamos el objeto en una línea

Ejemplo:

```
Conexion met = new Conexion();
```

4.4.12 Constantes

Debería estar todo en mayúsculas con palabras separadas por guiones bajos (“_”).

- Hay varias constantes utilizadas en clases predefinidas como Float, Long, String, etc.

Por ejemplo:

```
static final int MIN_WIDTH = 4;  
// Algunas variables constantes utilizadas en la clase Float predefinida  
public static final float POSITIVE_INFINITY = 1.0f / 0.0f;  
public static final float NEGATIVE_INFINITY = -1.0f / 0.0f;  
public static final float NaN = 0.0f / 0.0f;
```

4.4.13 Listas

Para la declaración de listas en Java se realiza de la siguiente manera:

```
ArrayList<String> nombreArrayList=new ArrayList<String>();
```

4.5 Diseño de datos

4.5.1 Diseño conceptual de la base de datos

Para la creación del proyecto en proceso el cual lleva por nombre ferrimax, se requiere de una base de datos la cual se creo con el nombre de ferrimax. Esta base de datos cuenta con tres tablas hasta el momento, las cuales son una tabla de empleado, otra tabla de nombre venta y una tabla que se nombró como producto. En la tabla de Venta se cuenta con distintos campos que son el idventa que será nuestra llave primaria, un NombreProducto que es de tipo varchar, en esa tabla se cuenta con un campo cantidad de tipo integer que se refiere a la cantidad de productos que se van a comprar, también tenemos un campo fechaVenta que es de tipo Date y por último un idProducto que es una llave foránea.

Por otra parte se cuenta con una tabla de nombre producto que si llave primaria se llama IdProducto, además, tenemos un campo llamado nombre de tipo varchar, el cual almacena el nombre del producto. Hay un campo Cantidad de tipo Integer, en esa columna se guardará la cantidad de productos. Se cuenta con do campos de precio, uno es precioVenta y precioCompra ambos de tipo float y almacenarán los precios de venta y compra, por último tenemos un campo descripcion que es el encargado de almacenar la descripción de los productos. La última tabla que tenemos es una tabla de empleado, la cual tiene siete campos, un IdVendedor, nombre de tipo varchar, un campo teléfono de tipo Int, domicilio y correo de tipo varchar y un cargo que se refiere al cargo del empleado y también es de tipo varchar.

4.5.2 Diseño lógico entidad relación

Descripción: La siguiente figura hace referencia al diseño lógico conceptual de la base de datos.

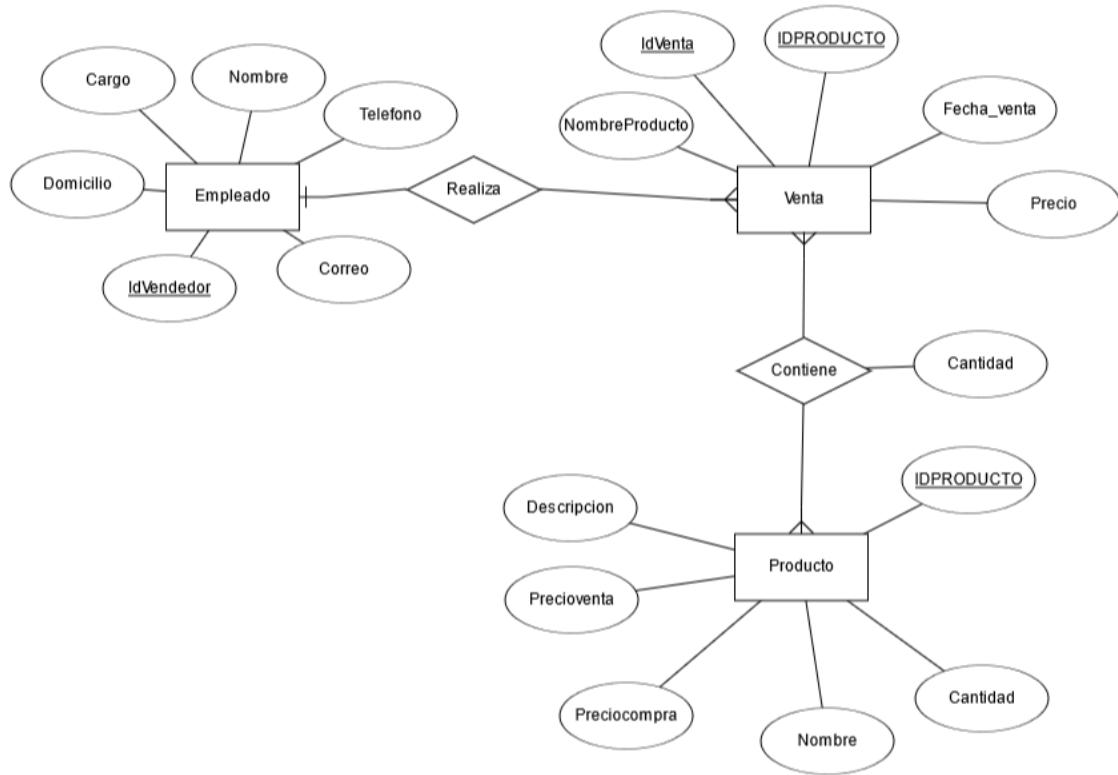


Figura 91: Mapa entidad relación

4.5.3 Esquema relacional

Descripción: La siguiente figura hace referencia al esquema relacional de la base de datos.



Figura 92: Esquema relacional

4.5.4 Diseño Físico

```
CREATE TABLE Producto
(
    IDPRODUCTO INT NOT NULL,
    Nombre varchar(20) NOT NULL,
    Precio_compra real ,
    Precio_venta real ,
    Descripcion varchar(20) ,
    Cantidad INT NOT NULL,
    PRIMARY KEY (IDPRODUCTO)
);

CREATE TABLE Empleado
(
    Nombre varchar(20) ,
    Telefono INT NOT NULL,
    Domicilio varchar(20) ,
    Correo varchar(20) ,
    IdVendedor INT NOT NULL,
    Cargo varchar(20) ,
    PRIMARY KEY (IdVendedor)
);

CREATE TABLE Venta
(
    IdVenta INT NOT NULL,
    Nombre_Producto varchar(20) ,
    IDPRODUCTO INT NOT NULL,
    Precio real ,
    IdVendedor INT NOT NULL,
    Fecha_venta varchar(20) ,
    — Opcional
    Cantidad INT,
    PRIMARY KEY (IdVenta),
    FOREIGN KEY (IdVendedor) REFERENCES Empleado(IdVendedor),
    UNIQUE (IDPRODUCTO)
);
```

```
CREATE TABLE Contiene
(
    IdVenta INT NOT NULL,
    IDPRODUCTO INT NOT NULL,
    — Opcional
    Cantidad int ,
    Fecha_venta varchar(20) ,
    Monto real ,
    PRIMARY KEY (IdVenta , IDPRODUCTO) ,
    FOREIGN KEY (IdVenta) REFERENCES Venta(IdVenta) ,
    FOREIGN KEY (IDPRODUCTO) REFERENCES Producto(IDPRODUCTO)
);
```

4.5.5 Diccionario de datos

Descripción: La siguiente figura hace referencia al diccionario de datos de contiene.

	Columna	Tipo de texto	Descripción
PK, FK	IdVenta	integer	Número de identificación para Las ventas.
PK,FK	IDPRODUCTO	integer	Número de identificación para producto.
	Cantidad	integer	Cantidad de producto

Figura 93: Diccionario de datos

Descripción: La siguiente figura hace referencia al diccionario de datos de empleado.

	Columna	Tipo de texto	Descripción
	Nombre	Varchar(20)	Nombre del empleado
	Telefono	integer	Número de teléfono del empleado
	Domicilio	Varchar(20)	Domicilio del empleado
	Correo	Varchar(20)	Correo electrónico del empleado
PK	Idvendedor	integer	Número de identificación para el vendedor.
	Cargo	Varchar(20)	Cargo del empleado

Figura 94: Diccionario de datos

Descripción: La siguiente figura hace referencia al diccionario de datos de producto.

	Columna	Tipo de texto	Descripción
PK	IDPRODUCTO	integer	Número de identificación para producto realizadas.
	Nombre	varchar(20)	Nombre de producto
	Cantidad	Integer	Cantidad de producto
	Precioventa	real	Precio de venta
	PrecioCompra	real	Precio de compra
	Descripcion	Varchar(20)	Descripción de producto

Figura 95: Diccionario de datos

Descripción: La siguiente figura hace referencia al diccionario de datos de venta.

	Columna	Tipo de texto	Descripción
PK	IdVenta	Integrer	Número de identificación para las ventas realizadas.
	NombreProducto	varcha(20)	Nombre del producto
FK	idVendedor	Integer	Número de identificación para vendedor
	Precio	real	Precio de producto
	Fecha_venta	Date	Fecha de venta
	IDPRODUCTO	integer	Número de identificación de producto

Figura 96: Diccionario de datos

5 Código fuente

Descripción: El siguiente código crea una conexión y retorna Connection.

```
public Connection conectaBase(String nombrebd) { //Metodo que crea una  
conxion y retorna Connection  
Connection con = null;  
try {  
    Class.forName("org.postgresql.Driver");  
    con = DriverManager.getConnection("jdbc:postgresql://" + host  
        + "/" + nombrebd, user, pass);  
//        System.out.println("conexion establecida");  
  
} catch (Exception ex) {  
    System.out.println("Error al conectar:" + ex.getMessage());  
}  
return con;  
}
```

Descripción: El siguiente método usuarios sirve para validar el inicio de sesión

```
public void usuarios() { //EL metodo usuarios sirve para validar el  
inicio de sesion  
String nombreBD = Venta.Usuario.getText();  
String ContraseñaBD = new String(Venta.Contraseña.getPassword());  
String contraseña = "", nombre = "", idvendedor = "";  
int ban;  
  
try {  
  
    rs = ejecutaQuery("select * from empleado where cargo='"  
        + Administrador"'); //Se realiza una consulta  
  
    while (rs.next()) {  
        //Se guardan los datos en variables  
        idvendedor = rs.getString(1);  
    }  
}
```

```

nombre = rs.getString(2);
contraseña = rs.getString(6);

System.out.println(idvendedor);
System.out.println(nombre);
System.out.println(contraseña);

}

if (nombre.equals(nombreBD)) { //Se comprueba el nombre con
    nombreBD
    if (!ContraseñaBD.equals("")) {
        if (nombre.equals(nombreBD) && contraseña.equals(
            ContraseñaBD)) {

            Venta.Login.setVisible(false); //Hace invisible
            el Login
            Venta.Administrador.setBounds(0, 0, 720, 390); //
            Se le asigna un tamaño
            Venta.Administrador.setLocationRelativeTo(null);
            //Centra el jdialog
            Venta.Administrador.setVisible(true); //Es visible
            el administrador
            desoculta();
    } else {

        JOptionPane.showMessageDialog(null, "Los datos no
            coinciden con la base de datos, vuelva a
            ingresarlos");
        Venta.Usuario.setText(null);
        Venta.Contraseña.setText(null);
    }
}
else {

    JOptionPane.showMessageDialog(null, "Faltan datos por
        ingresar!");
}

```

```

} else {
    if (!ContraseñaBD.equals("")){
        if (nombreBD.equals("gama") && ContraseñaBD.equals("123")){
            Venta.Login.setVisible(false);
            Venta.Administrador.setBounds(0, 0, 720, 390);
            Venta.Administrador.setLocationRelativeTo(null);
            Venta.Administrador.setVisible(true);
            ocultaus();
        }
    } else {
        JOptionPane.showMessageDialog(null, "Los datos no coinciden con la base de datos, vuelva a ingresarlos");
        Venta.Usuario.setText(null);
        Venta.Contraseña.setText(null);
    }
} else {
    JOptionPane.showMessageDialog(null, "Faltan datos por ingresar!");
}
conexion.close();
}

} catch (Exception e) {
    System.out.println(e.getMessage());
}
}

```

Descripción: El siguiente método ayuda a realizar la consulta del producto

```
public void consultar (DefaultTableModel x) { //M todo que ayuda a
    realizar la consulta del producto
    Object [] ob = new Object [6]; //Se define un arreglo de objetos
    try {
        rs = ejecutaQuery ("select *from producto ORDER BY id"); //Se
        reaalizar una consulta de producto
        while (rs.next ()) { //Se le asignan los valor a un arreglo de
            objetos
            ob [0] = rs.getObject ("id");
            ob [1] = rs.getObject ("nombre");
            ob [2] = rs.getObject ("cantidad");
            ob [3] = rs.getObject ("precioventa");
            ob [4] = rs.getObject ("preciounitario");
            ob [5] = rs.getObject ("Descripcion");
            x.addRow (ob); //Se agrega los campos en un JTable
        }
        conexion.close (); //Se cierra conexion
    } catch (Exception e) {
        System.out.println (e.getMessage ());
    }
    Venta.TablaConsultaProd.setModel (x);
    Venta.TablaConsultaProd1.setModel (x);
}
```

Descripción: los siguiente métodos ayuda a limpiar las tablas y limpia los textfield que se muestran en este método

```
public void limpiaTabla(DefaultTableModel x) { //m todo que ayuda a  
limpiar las tablas  
    int a = x.getRowCount();  
  
    while (a != 0) {  
        if (a != 0) {  
            x.removeRow(0);  
        }  
        a = x.getRowCount();  
    }  
  
    public void limpiar() { //Limpia los textfield que se muestran en  
este metodo  
        Venta.TextoNombreProducto.setText(null);  
        Venta.TextoPrecioProducto.setText(null);  
        Venta.TextoIdentificadorProducto.setText(null);  
        Venta.txtDescripcion.setText(null);  
        Venta.Cantidad1.setValue(1);  
    }  
}
```

Descripción: los siguientes métodos realizan la alta de los productos e insertan los datos del usuario.

```
public void insertarDatos() { //M todo que ayuda a dar de alta los
    productos
    try {
        if (Venta.TextoNombreProducto.getText().isEmpty() || Venta.
            TextoIdentificadorProducto.getText().isEmpty() || Venta.
            TextoPrecioProducto.getText().isEmpty() || Venta.
            txtDescripcion.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Ingrese los datos
                solicitados", "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            //Se le asigna a la variable sql la actualización
            String sql = "INSERT INTO producto (nombre, cantidad,
                precioventa, preciounitario, descripcion) values (?, ?, ?,
                ?, ?, ?);";
            PreparedStatement ps = conexion.prepareStatement(sql);
            //Se obtienen los datos de los campos
            ps.setString(1, Venta.TextoNombreProducto.getText());
            ps.setInt(2, Integer.parseInt(Venta.Cantidad1.getValue().
                toString()));
            ps.setFloat(3, Float.parseFloat(Venta.TextoPrecioProducto.
                getText()));
            ps.setFloat(4, Float.parseFloat(Venta.
                TextoIdentificadorProducto.getText()));
            ps.setString(5, Venta.txtDescripcion.getText());
            ps.executeUpdate(); //Se insertan los datos en la base de
                datos
            JOptionPane.showMessageDialog(null, "datos insertados
                correctamente");
            conexion.close();
        }
    } catch (Exception ex) {
        System.out.println(ex);
    }
}
```

```

public void InsertarDatosUsuarios() { //M todo que inserta los datos
    del empleado
    try {
        //Se comprueba que los campos no esten vacios
        if (Venta.NombreU.getText().isEmpty() || Venta.NumeroUsuario .
            getText().isEmpty() || Venta.CorreoUsuario.getText() .
            isEmpty() || Venta.PasswordUsusario1.getText().isEmpty()
            || Venta.PasswordUsusario2.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Hace falta datos.");
        } else {
            //Se insertan los datos digitados por el empleado hacia
            la base de datos
            String sql = "INSERT INTO empleado(_nombre,_telefono,
                correo,_contraseña,_cargo)_values(?,?,?,?,?)";
            //Se prepara la sentencia sql
            PreparedStatement ps = conexion.prepareStatement(sql);
            ps.setString(1, Venta.NombreU.getText());
            ps.setString(2, Venta.NumeroUsuario.getText());
            ps.setString(3, Venta.CorreoUsuario.getText());
            ps.setString(4, Venta.PasswordUsusario1.getText());
            ps.setString(5, Venta.Cargo.getSelectedItem().toString());
            ;
            ps.executeUpdate(); //Se insertan los datos en la base de
            datos
            //Se muestra un mensaje de que los usuarios fueron
            insertados
            JOptionPane.showMessageDialog(null, "datos insertados_
                correctamente");
            JOptionPane.showMessageDialog(null, "Usuario ingresado_
                con éxito");
        }
    } catch (Exception ex) {
        System.out.println(ex);
    }
}

```

Descripción: Los siguientes métodos consultan usuarios y pasan los datos del usuario a los campos.

```
public void ConsultarUsuarios(DefaultTableModel x) { //Metodo que ayuda  
    a consultar los usuarios  
    Object [] ob = new Object [5];  
    try {  
        rs = ejecutaQuery("select *from empleado");  
        while (rs.next()) {  
            //Se insertan los datos en un arreglo de objetos  
            ob[0] = rs.getObject("IdVendedor");  
            ob[1] = rs.getObject("nombre");  
            ob[2] = rs.getObject("telefono");  
            ob[3] = rs.getObject("correo");  
            ob[4] = rs.getObject("cargo");  
            //Se agregan los datos a las diferentes filas  
            x.addRow(ob);  
        }  
        conexion.close(); //Se cierra la conexión  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
    }  
    Venta.TablaUsuario.setModel(x); //Se actualiza el JTable  
  
}  
  
  
public void ActualizarUsuario() { //Metodo que pasa los datos del  
    usuario a los diferentes campos  
  
    if (Venta.TablaUsuario.getSelectedRow() < 0) { //Se verifica si  
        el usuario eligio alguna fila para modificar  
        JOptionPane.showMessageDialog(null, "Seleccione el dato a  
        editar"); //En caso de no haber seleccionada alguna se le  
        manda este mensaje  
    } else {
```

```

//Se le pasan los datos a la JDialog Editar para poder
modificar los datos

Venta . ActualizaNombreU . setText (( String ) Venta . TablaUsuario .
    getValueAt (Venta . TablaUsuario . getSelectedRow () , 1));
Venta . ActualizaTelU . setText (String . valueOf (Venta . TablaUsuario
    . getValueAt (Venta . TablaUsuario . getSelectedRow () , 2)));
Venta . ActualizaCoU . setText (( String ) Venta . TablaUsuario .
    getValueAt (Venta . TablaUsuario . getSelectedRow () , 3));
Venta . ActualizarUsuario . setBounds (300 , 300 , 392 , 330);
Venta . ConsultarUsuario . setVisible (false);
Venta . ActualizarUsuario . setBounds (0 , 0 , 396 , 496);
Venta . ActualizarUsuario . setLocationRelativeTo (null);
Venta . ActualizarUsuario . setModal (true);
Venta . ActualizarUsuario . setVisible (true);

}

}

```

Descripción: Los siguientes métodos actualizan los datos del usuario y eliminan productos

```
public void CambiarDatosUsuario() {//M todo que actualiza los datos del usuario
    int fila = Venta.TablaUsuario.getSelectedRow(); //Se le asigna a una variable la fila
    int id = Integer.parseInt(Venta.TablaUsuario.getValueAt(fila, 0).toString());
    try {
        //Definimos la sentencia sql
        String actualizacion = "UPDATE_empleado SET nombre=?," +
            "Telefono=?," + "Correo=?," + "Cargo=?," + "Contraseña=?," +
            "where idvendedor=" + id + ";";
        //Se toman los datos de los campos y se envian a la base de datos
        PreparedStatement actua = conexion.prepareStatement(
            actualizacion);
        actua.setString(1, Venta.ActualizaNombreU.getText());
        actua.setString(2, Venta.ActualizaTelU.getText());
        actua.setString(3, Venta.ActualizaCoU.getText());
        actua.setString(4, Venta.ComboUser.getSelectedItem().toString());
        actua.setString(5, Venta.jPasswordField1.getText());

        actua.executeUpdate(); //Se actualizan los nuevos datos
        Venta.ActualizarUsuario.setVisible(false); //Se hace no visible la ventana de actualizacion de usuarios
        Venta.ConsultarUsuario.setVisible(true);

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

public void eliminarUnProducto(DefaultTableModel dtm) { //Metodo que
```

```

ayuda a eliminar productos
try {
    if (Venta.TablaConsultaProd.getSelectedRow() < 0) { //Se
        verifica si el usuario eligio alguna fila para modificar
        JOptionPane.showMessageDialog(null, "Seleccione el
            producto para eliminar");//Mensaje para notificar que
        tiene que seleccionar un producto
    } else {
        int fila = Venta.TablaConsultaProd.getSelectedRow(); //Se
        asigna la fila seleccionada
        int id = Integer.parseInt(Venta.TablaConsultaProd.
            getValueAt(fila, 0).toString());

        int cantidadBorrar = Integer.parseInt(JOptionPane.
            showInputDialog(null, "Cuantos productos desea
            eliminar:"));
        // cantidadBorrar=Integer.parseInt(JOptionPane.
        // showInputDialog(null,"Cuantos productos desea eliminar
        :"));

        int cantidad = Integer.parseInt(Venta.TablaConsultaProd.
            getValueAt(fila, 2).toString());
        System.out.println(cantidad);
        if (cantidad > cantidadBorrar) { //Se verifica que la
            cantidad sea mayor que cantidad recibida
            rs = ejecutaQuery("update_producto set cantidad=
                cantidad-" + cantidadBorrar + " where id=" + id +
                ";");
            JOptionPane.showMessageDialog(null, "Productos
                eliminados");
            conexion.close();
            dtm.removeRow(TablaConsultaProd.getSelectedRow()); //
                Se elimina la fila seleccionada
        } else if (cantidad == cantidadBorrar) {
            rs = ejecutaQuery("delete from producto where id=" +
                id + ";");
            JOptionPane.showMessageDialog(null, "Producto
                eliminado");
            conexion.close();
    }
}

```

```
        dtm.removeRow( TablaConsultaProd.getSelectedRow() );  
  
    } else { //No se cumple las sentencias anteriores  
        entonces se manda un mensaje  
        JOptionPane.showMessageDialog( null , "El n mero a  
        eliminar es mayor al de existencia" );  
    }  
}  
  
}  
}  
}  
}  
}
```

Descripción: Los siguientes métodos eliminan un usuario y busqueda de empleados

```
public void EliminarUsuario(DefaultTableModel dtm) { //Metodo que
ayuda a eliminar un usuario
try {
    if (Venta.TablaUsuario.getSelectedRow() < 0) { //Se verifica
        si el usuario eligio alguna fila para modificar
        JOptionPane.showMessageDialog(null, "Seleccione el dato
        para eliminar");
    } else {
        int fila = Venta.TablaUsuario.getSelectedRow();
        int id = Integer.parseInt(Venta.TablaUsuario.getValueAt(
            fila, 0).toString());
        rs = ejecutaQuery("delete from empleado where idvendedor=
            " + id + ";"); //Se borra el empleado a nivel de base
        de datos
        JOptionPane.showMessageDialog(null, "Usuario eliminado
        con exito");
        conexion.close();
        dtm.removeRow(Venta.TablaUsuario.getSelectedRow()); //Se
        elimina la fila seleccionada
    }
} catch (Exception e) {
    System.out.println(e.getMessage());
}

}

public void BuscarUsuarario(DefaultTableModel dtm) { //Metodo que no
retorna nada, pero ayuda en la busqueda de empleados
ResultSet con;

limpiaTabla(dtm); //Se limpia la tabla
Object[] ob = new Object[5]; //Creacion de arreglo de objetos
try {
```

```

if (Venta.TextoUsuarioBuscar.getText().length() != 0) {

    String busca = Venta.TextoUsuarioBuscar.getText();
    //Sentencia que ayuda a buscar el empleado deseado
    rs = ejecutaQuery("SELECT *FROM empleado WHERE nombre="
        + busca + " ;");

    // con=s.executeQuery("SELECT *FROM producto where
    cod_barras "+busca);

    while (rs.next()) { //Se le pasan todos los datos al
        arreglo
        ob[0] = rs.getObject("IdVendedor");
        ob[1] = rs.getObject("nombre");
        ob[2] = rs.getObject("telefono");
        ob[3] = rs.getObject("correo");
        ob[4] = rs.getObject("cargo");
        dtm.addRow(ob); //Se agregan las filas
    }

}

conexion.close();
} catch (Exception e) {
    System.out.println("Error al cargar productos");
}
}

```

Descripción: Los siguientes métodos verifican si el usuario eligió alguna fila para modificar y para poder actualizar los datos de un producto

```
public void actualizar() { //Metodo que actualiza los productos

    if (Venta.TablaConsultaProd.getSelectedRow() < 0) { //Se verifica
        si el usuario eligio alguna fila para modificar
        JOptionPane.showMessageDialog(null, "Seleccione el dato a
            editar"); //En caso de no haber seleccionada alguna se le
        manda este mensaje
    } else {
        //Se le pasan los datos a la JDialog Editar para poder
        modificar los datos

        Venta.Actualizar1.setText((String) Venta.TablaConsultaProd.
            getValueAt(Venta.TablaConsultaProd.getSelectedRow(), 1));
        Venta.Actualizar2.setText(String.valueOf(Venta.
            TablaConsultaProd.getValueAt(Venta.TablaConsultaProd.
            getSelectedRow(), 3)));
        Venta.Actualizar3.setText(String.valueOf(Venta.
            TablaConsultaProd.getValueAt(Venta.TablaConsultaProd.
            getSelectedRow(), 4)));
        Venta.Actualizar4.setText(String.valueOf(Venta.
            TablaConsultaProd.getValueAt(Venta.TablaConsultaProd.
            getSelectedRow(), 5)));
        int valor = (int) Venta.TablaConsultaProd.getValueAt(Venta.
            TablaConsultaProd.getSelectedRow(), 2);
        Venta.Actualizar5.setValue(valor);

        Venta.Actualizar.setBounds(300, 300, 392, 330);
        Venta.ConsultarProducto.setVisible(false);

        Venta.Actualizar.setBounds(0, 0, 396, 496);
        Venta.Actualizar.setLocationRelativeTo(null);
        Venta.Actualizar.setModal(true);
        Venta.Actualizar.setVisible(true);

    }
}
```

```

}

public void CambiarDatos() { //Metodo para poder actualizar los datos
    de un producto
    int fila = Venta.TablaConsultaProd.getSelectedRow();
    int id = Integer.parseInt(Venta.TablaConsultaProd.getValueAt(fila
        , 0).toString());

    try {
        //Se verifica si algun campo esta vacio
        if (Venta.Actualizar1.getText().isEmpty() || Venta.
            Actualizar2.getText().isEmpty() || Venta.Actualizar3.
            getText().isEmpty() || Venta.Actualizar4.getText().isEmpty()
        ) {
            JOptionPane.showMessageDialog(null, "Ingrese los datos
                solicitados", "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            //Se crea una sentencia sql para poder actualizar
            String actualizacion = "UPDATE_producto SET nombre=?,
                cantidad=?, precioventa=?, preciounitario=?,
                descripcion=? where id=" + id + ";";
            PreparedStatement actua = conexion.prepareStatement(
                actualizacion);
            //Se le pasan los diferentes datos ingresados a la tabla
            //producto
            actua.setString(1, Venta.Actualizar1.getText());
            actua.setInt(2, Integer.parseInt(Venta.Actualizar5.
                getValue().toString()));
            actua.setFloat(3, Float.parseFloat(Venta.Actualizar2.
                getText()));
            actua.setFloat(4, Float.parseFloat(Venta.Actualizar3.
                getText()));
            actua.setString(5, Venta.Actualizar4.getText());
            actua.executeUpdate(); //Se actualizan los datos
            Venta.Actualizar.setVisible(false);
            Venta.ConsultarProducto.setVisible(true);
        }
    }
}

```

```

        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

```

Descripción: El siguiente método guarda los datos en la tabla de realizar venta.

```

public void RealizarVenta(DefaultTableModel x) { //Metodo que guarda los
    datos en la tabla de realizar venta

    ArrayList lista = new ArrayList(); //Se crea un nuevo ArrayList
    //Se definen variables a utilizar
    int id = 1;
    float numeros = 0;
    id = id + 1;
    String idVendedor = "1234";
    //Se obtienen datos de los campos donde se ingresan los datos
    String NombreP = Venta.jTextNombreProducto.getText();
    int Idproducto = Integer.parseInt(Venta.jTextIdentificador.
        getText());
    int Cantidad = Integer.parseInt(Venta.jSpinnerVentaCantidad.
        getValue().toString());
    float Cantidad2 = Float.parseFloat(Venta.jTextCantidad.getText());
    ;
    float precios = Float.parseFloat((Venta.jTextPrecio.getText()));
    String fecha = Venta.jTextFecha.getText();
    //Se comprueba que la cantidad sea mayor a 0
    if (Cantidad2 > 0) {
        float CantidadT = Cantidad2 - Cantidad;

        String CantidadR = "" + CantidadT;

        Venta.ProductosExistentes.setText(CantidadR);

        if (CantidadT >= 0) {
            float total = Cantidad * precios;

```

```

//Se le mandan los datos que se agregaran a la tabla
lista.add(id);
lista.add(NombreP);
lista.add(Idproducto);
lista.add(total);
lista.add(idVendedor);
lista.add(fecha);
lista.add(Cantidad);

Object [] ob = new Object [7];
//Se agregan los datos a la tabla
ob[0] = lista.get(0);
ob[1] = lista.get(1);
ob[2] = lista.get(2);
ob[3] = lista.get(3);
ob[4] = lista.get(4);
ob[5] = lista.get(5);
ob[6] = lista.get(6);

x.addRow(ob); //Se agregan las nuevas filas

Venta.JTableVenta.setModel(x); //Se actualiza la tabla
CalcularToltal(); //Se llama a la funcion donde se
calculara el monto a pagar
Venta.TablaConsultaProd1.setValueAt(ProductosExistentes.
getText(), Venta.TablaConsultaProd1.getSelectedRow(),
2); //Se actualiza la cantidad del producto
JOptionPane.showMessageDialog(null, "Se agrego-
correctamente el producto");

} else {
//En caso que no se cumpla se manda el mensaje
JOptionPane.showMessageDialog(null, "Es mayor el numero-
de herramienta solicitada");

String CantidadR1 = "";
Venta.ProductosExistentes.setText(CantidadR1); //Se manda
el contenido de la variable CantidadR1

```

```

        }
    } else {
        JOptionPane.showMessageDialog(null, "Ya no existen productos de ese tipo");
    }

}

```

Descripción: Los siguientes métodos insertan datos en la tabla venta en base de datos, el otro método genera insercion de datos en la tabla contiene en la base de datos y el último consulta la venta que se realizó

```

public void GenerarVenta() { //Metodo que inserta datos en la tabla venta en base de datos
    try {
        //Se insertan los datos en la tabla de venta en la base de datos
        String sql = "INSERT INTO venta (IdVenta, NombreProducto, IDPRODUCTO, Precio, IdVendedor, Fecha_venta, cantidad) values (?, ?, ?, ?, ?, ?, ?);";
        PreparedStatement ps = conexion.prepareStatement(sql);
        //Se recorre la Tabla para poder obtener los diferentes valores
        for (int i = 0; i < Venta.JTableVenta.getRowCount(); i++) {

            ps.setInt(1, Integer.parseInt(Venta.JTableVenta.getValueAt(i, 0).toString()));
            ps.setString(2, Venta.JTableVenta.getValueAt(i, 1).toString());
            ps.setFloat(3, Float.parseFloat(Venta.JTableVenta.getValueAt(i, 2).toString()));
            ps.setFloat(4, Float.parseFloat(Venta.JTableVenta.getValueAt(i, 3).toString()));
            ps.setInt(5, Integer.parseInt(Venta.JTableVenta.getValueAt(i, 4).toString()));
            ps.setString(6, Venta.JTableVenta.getValueAt(i, 5).toString());
            ps.setInt(7, Integer.parseInt(Venta.JTableVenta.

```

```

        getValueAt(i, 6).toString())));
    }
    ps.executeUpdate(); //Se agregan los datos a la tabla venta
    JOptionPane.showMessageDialog(null, "Datos insertados_
        correctamente");
    conexion.close();
}

} catch (Exception ex) {
    System.out.println(ex);
}
}

public void GenerarConsultaVenta() { //Genera insercion de datos en
    la tabla contiene en la base de datos
try {
    String sql = "INSERT INTO Contiene_(IdVenta ,_IDPRODUCTO_,_
        NombreProducto ,_Cantidad ,_Fecha_venta ,_Monto)_values (?,_?
        ?,_?,_?,_?,_?)";
    PreparedStatement ps = conexion.prepareStatement(sql);
    //Recorre la tabla de venta para poder obtener los datos
    correspondientes
    for (int i = 0; i < Venta.JTableVenta.getRowCount(); i++) {

        ps.setInt(1, Integer.parseInt(Venta.JTableVenta.
            getValueAt(i, 0).toString()));
        ps.setFloat(2, Float.parseFloat(Venta.JTableVenta.
            getValueAt(i, 2).toString()));
        ps.setString(3, Venta.JTableVenta.getValueAt(i, 1).
            toString());
        ps.setInt(4, Integer.parseInt(Venta.JTableVenta.
            getValueAt(i, 6).toString()));
        ps.setString(5, Venta.JTableVenta.getValueAt(i, 5).
            toString());
        ps.setFloat(6, Float.parseFloat(Venta.JTableVenta.
            getValueAt(i, 3).toString()));

    }
    ps.executeUpdate(); //Se insertan los datos
}

```

```

        conexion.close(); //Se cierra la conexion

    } catch (Exception ex) {
        System.out.println(ex);
    }
}

public void ConsultarVenta(DefaultTableModel x) { //Metodo que
    consulta la venta que se realizo
    Object[] ob = new Object[5]; //Se crea un arreglo de objetos
    try {
        rs = ejecutaQuery("select * from _contiene"); //Con el result
            set obtenemos los valores que estan en la base de datos
        while (rs.next()) {
            ob[0] = rs.getObject("nombreproducto");
            ob[1] = rs.getObject("cantidad");
            ob[2] = rs.getObject("fecha_venta");
            ob[3] = rs.getObject("monto");
            x.addRow(ob); //Se agregan en las filas de la tabla
        }
        conexion.close();
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    Venta.jTableConsultarVenta.setModel(x); //Se actualiza la tabla de
        consultar venta
}

```

6 Conclusiones

Después del desarrollo del proyecto de punto de venta Ferrimax que comenzó hace unos meses, hasta la entrega final se lograron realizar todos los requerimientos funcionales que fueron propuestos al inicio del semestre. Cómo lo son el alta, modificaciones y eliminación de productos, además se logró finalizar la parte de ventas, como lo es la realización de ventas y la consulta de las mismas. Por ultimo también se terminó la parte de los usuarios, las cuales son el alta y consulta de usuarios. Este proyecto se realizó por cerca de 4 meses. El sistema cuenta con un progreso aceptable a nivel de programación como también de base de datos, pero aún quedan puntos por mejorar, además, aún se pueden agregar nuevas funcionalidades que puedan facilitar el trabajo de los clientes.

Por otra parte la realización de este documentos es un gran aporte al sistema desarrollado, porque será de gran ayuda a la hora de darle mantenimiento al proyecto, o a la hora de realizarle nuevas modificaciones. Es importante mencionar que la ingeniería de software ha tomado un papel importante en el desarrollo de software ya que se encarga de coordinar el desarrollo de los sistemas, también se encarga del ciclo de vida de los productos. En cuanto a la materia de ingeniería de software aplicamos varios puntos en el documento y en el desarrollo del software, como es la especificación de requerimientos. Además del desarrollo de diagramas de estructura UML donde destacan los diagramas de clases, también se desarrollaron diagramas de comportamiento UML donde resalta los diagramas de casos de uso. Estos dos últimos son los más utilizados en el desarrollo de software. Cabe mencionar que se buscó trabajar con un modelo de desarrollo de software, tal vez no logramos apegarnos a uno como tal, pero el más cercano fue el Scrum, con realización de reuniones y corrección de errores constantemente.

Para finalizar es importante destacar el aprendizaje personal de cada uno de los desarrolladores, ya que al principio del proyecto no se tenía una noción exacta de lo que se iba a realizar, pero con el paso de los días y el desarrollo de sistema, se aprendieron nuevas cosas, tanto en la materia de ingeniería de software como en la de paradigmas de programación y base de datos.

7 Definición, acrónimos y abreviaturas

Descripción: En la siguiente imagen se muestra la definición, acrónimos y abreviaturas.

Tipo de texto	Descripción
SQL	Es un lenguaje de computación para trabajar con conjuntos de datos y las relaciones entre ellos. Los programas de bases de datos relacionales, como Microsoft Office Access, usan SQL para trabajar con datos.
C	Es un lenguaje para programadores en el sentido de que proporciona una gran flexibilidad de programación y una muy baja comprobación de incorrecciones, de forma que el lenguaje deja bajo la responsabilidad del programador acciones que otros lenguajes realizan por sí mismos.
java	Es un lenguaje de programación
HTML	Es el código que se utiliza para estructurar y desplegar una página web y sus contenidos.
CSS	Es lo que se denomina lenguaje de hojas de estilo en cascada y se usa para estilizar elementos escritos en un lenguaje de marcado como HTML.
javascript	Es un lenguaje de programación ligero, interpretado, o compilado justo-a tiempo (just-in-time) con funciones de primera clase.
lenguaje ensamblador	Es el lenguaje de programación utilizado para escribir programas informáticos de bajo nivel.
c++	Es un lenguaje de programación diseñado en 1979 por Bjarne Stroustrup.

Figura 97: Definición, acrónimos y abreviaturas

Descripción: En la siguiente imagen se muestra la definición, acrónimos y abreviaturas.

python	Es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo.
FERRI-MAX	Es el nombre del software desarrollada por los integrantes del equipo.
MySQL	Es un sistema de base de datos relacional o RDBMS (Relational Database Management System) que emplea un modelo cliente-servidor.
PostgreSQL	Es un sistema para gestionar bases de datos de muy alto nivel, completamente de software libre y con una licencia BSD, compatible con cualquier uso, ya sea personal o comercial.
Rf	Requerimientos funcionales
CU	Caso de uso
RNF01	Requerimientos no funcionales
Look and feel	Una página es la apariencia del sitio y lo que siente el usuario cuando interactúa con él.
UML	Lenguaje Unificado de Modelado (UML) Un lenguaje visual para especificar, construir y documentar los artefactos de los sistemas.

Figura 98: Definición, acrónimos y abreviaturas

Descripción: En la siguiente imagen se muestra la definición, acrónimos y abreviaturas.

String	El objeto String se utiliza para representar y manipular una secuencia de caracteres.
void	Se usa void para indicar que una función no devuelve un valor o que no tiene parámetros o ambos
int	El tipo Int representa valores de tipo entero.
float	El float financiero guarda relación con las operaciones comerciales y es el espacio de tiempo que acontece desde el momento en que la empresa compradora realiza el ingreso en el banco y el momento en que la empresa vendedora verifica que ha recibido el pago y puede disponer del dinero.
node	Es un entorno en tiempo de ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en JavaScript.
UpperCamelCase	Cuando la primera letra de cada una de las palabras es mayúscula.
NetBeans IDE	Es un entorno de desarrollo integrado de código abierto y gratuito para el desarrollo de aplicaciones en los sistemas operativos Windows, Mac, Linux y Solaris.
RAM	Es la memoria principal de un dispositivo, esa donde se almacenan de forma temporal los datos de los programas que estás utilizando en este momento.
Intel	Los procesadores Intel® le ofrecen desempeño de primer nivel para uso personal y empresarial. Encuentra una amplia variedad de procesadores por tipo de dispositivo: computadoras portátiles, desktops, estaciones de trabajo y servidores.
Ubuntu	Es una distribución GNU/Linux que ofrece un interesante sistema operativo para equipos de escritorio y servidores en el ámbito educativo.
AMD	Advanced Micro Devices, Inc. Es un procesador.
PSP(SM)	Es un conjunto de prácticas disciplinadas para la gestión del tiempo y mejora de la productividad personal de los programadores o ingenieros de software, en tareas de desarrollo y mantenimiento de sistemas, mediante el seguimiento del desempeño predicho frente al desempeño real.
TSO(SM)	Un Transmission System Operator es una sociedad mercantil autorizada para la construcción, operación y mantenimiento de instalaciones de la red troncal y certificadas de acuerdo con el procedimiento.

Figura 99: Definición, acrónimos y abreviaturas

8 Anexo

8.1 Evidencias fotográficas

Descripción: En la siguientes imágenes se muestra parte de la presentación del proyecto final.



Figura 100: presentación proyecto



Figura 101: presentación proyecto



Figura 102: presentación proyecto



Figura 103: Trabajando en el proyecto de Ferrimax

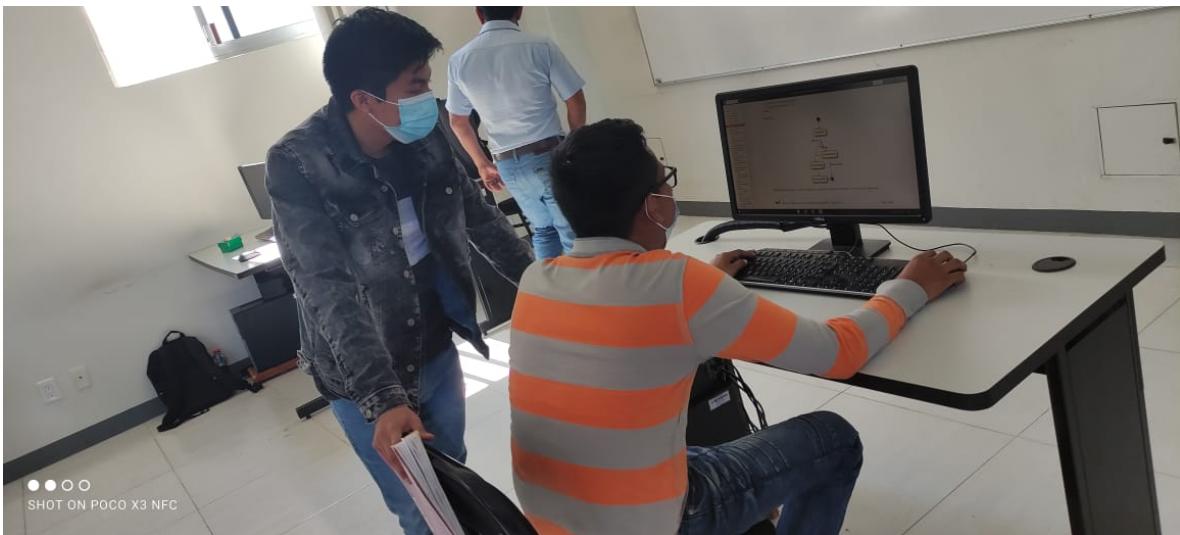


Figura 104: Trabajando en el proyecto de Ferrimax

9 Referencias

Java Tutorial [Internet]. W3schools.com. 2022 [cited 1 February 2022]. Available from: <https://www.w3schools.com/java/default.asp>

Deitel P, Deitel H, Romero Elizondo A. Como programar en Java (9a. ed.). Distrito Federal: Pearson Educacion; 2012.

Sommerville I, Campos Olguin V, Fuenlabrada Velazquez S. Ingenieria de software. Mexico: Addison Wesley; 2011.

Schach S, Fernandez E, Guerrero E, Trejo Ramirez R, Juarez Betancourt S. Ingenieria de software clasica y orientada a objetos. Mexico: McGraw-Hill; 2006.

Podeswa H. UML. Madrid: Anaya Multimedia; 2010.

Le Q, Diaz M. Developing Modern Database Applications with PostgreSQL.

Le D. Developing Modern Database Applications with PostgreSQL. Packt Publishing; 2021.

Cachero Castro C, Ponce de León Amador P, Saquete Boró E. Introducción a la programación orientada a objetos. San Vicente del Raspeig: Publicaciones de la Universidad de Alicante; 2006.

[Internet]. Plantuml.com. 2022 [citado el 5 de febrero de 2022]. Disponible en: <https://plantuml.com/es/class-diagram>

Joyanes Aguilar L. Fundamentos de programación. España: McGraw-Hill/Interamericana; 2008.