



UNIVERSIDAD DE LA SIERRA SUR

LICENCIATURA EN INFORMÁTICA

MATERIA: INGENIERÍA DE SOFTWARE

GRUPO: 506-B

PRIMERA ENTREGA

NOMBRE DEL PROFESOR:

M.T.C.A Rolando Pedro Grabiel.

INTEGRANTES:

López López Iris Estrella
Ramírez Rodríguez Cristian Adair
Sainos Hernández Baldomero

08 DE NOVIEMBRE DEL 2021

Índice de contenido

0.1	Introducción	4
0.2	Propósito	5
0.3	Viabilidad	5
0.4	Factibilidad	5
0.5	Ámbito del sistema	6
0.6	Mapa de navegación	7
0.7	Requisito funcional	16
0.8	Requisitos no funcionales	17
0.9	Personal Involucrado	22
1	Scrip BD	43
2	Diagrama de actividades	45
3	Diagrama de paquetes	56
4	Diagrama de Flujo de datos	65
5	Diagrama de Componentes	67
6	Diagrama de Secuencia	69
7	Diagrama de Despliegue	74
8	Diagrama de Clase	76
9	Diagrama de Objeto	80

Índice de Tablas

1	Personal Involucrado	22
---	--------------------------------	----

0.1 Introducción

Las tiendas para mascotas son la mejor opción si quieras emprender en el negocio de los animales, además requiere de poca inversión y es innecesario que seas un profesional en medicina veterinaria. Nuestra sociedad cada momento va evolucionando y por lo mismo una tienda de mascotas no puede quedar atrás por lo mismo el siguiente proyecto que hemos comenzado a elaborar, hemos tomado ciertos puntos importantes para poder lograr su elaboración lo más completa posible. Nuestra temática se centrará en la administración de productos de una tienda de mascotas, donde se le agregarán los productos que tengan a la venta en ese momento lograremos tener un registro de las ventas, de la cantidad de productos que aun estén en la tienda y por último con cierta cantidad comprada se podrá dar ciertos descuentos. Con el aumento de las personas en el mundo va igual en aumento las mascotas que tiene en su casa y nuestro proyecto da una manera mas eficiente de como poder obtener los productos de una manera más eficiente ahorrando el proceso de ir hasta una tienda hábil, preguntar por los precios, estar cargando con efectivo teniendo el riesgo de ser robados y por ultimo mejorar la manera de conseguir los productos para nuestras mascotas. Hay diversos servicios que puedes brindar en tu negocio, pero es necesario que te informes primero y obtengas algún título en lo que vayas a enfocarte, es decir,

- Venta de alimentos o accesorios.
- Venta de animales.
- Higiene y peluquería.
- Cuidado diario.

0.2 Proposito

Las mascotas han influido positivamente en la vida de los seres humanos, por esta razón, es importante asumir el compromiso de cuidar y proteger la vida de un animal. Nuestro propósito va más orientado en ayudar a los clientes para que puedan obtener los productos de una forma más sencilla, eficiente y ahorrando el tiempo que toma el ir hasta una tienda de mascotas donde deben de preguntas por el precio de los productos, estar eligiendo por el alimento adecuado para sus mascotas. Con esta mayor facilidad que se tendría al tener una aplicación como la nuestra, se podría tener mayores extensiones hacia otros clientes los cuales están más conectados a la redes sociales y podrían conocer más rápidamente la tienda(PetMark).

0.3 Viabilidad

Con el aumento de personas en el mundo se ha tenido un aumento constante de los integrantes de las familias las cuales han llegado a sentir por ellos un cariño profundo a las mascotas que viven a su lado por lo mismo se deben alimentar, ser atendidos y tener los cuidados como se lo merecen, por lo mismo nuestra aplicación se centrara en ayudar al cliente en conseguir todos los productos necesario para su cuidado. Técnico Como primer punto la viabilidad técnica se nos presentaran detalles para como poder presentar al usuario una forma mas fácil de comprar los productos correctos, para este detalle hemos pensado en dividirlo por secciones correspondiendo de las mascotas que tengan o lo que gusten comprar. También el tiempo sera muy primordial para poder finalizar el proyecto por lo mismo se aprovechara los momentos necesarios para darle fin. Con el aumento de la tecnología a los usuarios se les hará la facilidad para poder encargar los productos y por lo mismo esperaran que les llegue sus productos, para iniciar tendremos un registro de los usuarios que ingresen guardando lo que compraron Económico Para tomar las riendas de este proyecto así como tomamos unas horas de nuestro tiempo también deben ser remuneradas pero eso ya se tomara cuando tengamos un avance. Por otro lado el saber si es necesario la aplicación o si alguien lo compraría las ofreceríamos a tiendas que estén iniciando ya que son las que necesitan este tipo de impulso para no perderse o estancarse en solo una forma de venta, la ubicación que se esta por el momento es la ciudad de Miahuatlán de Porfirio Díaz .

0.4 Factibilidad

Se hará uso de ciertos materiales, tiempo y conocimiento para poder dar seguimiento de la aplicación que esperamos hacer, todos estos recursos no serán necesarios de efectivo pero debemos tomar las medidas para recaudar algo. Con la facilidad que tendrán para administrar los productos y hacerlo llegar al cliente también debe tener ciertos beneficios los vendedores ahí es donde entrarán las ganancias las cuales se tomará por cada producto que vendan y con esto tomaremos como inicio para que la tienda tenga el aumento que se espera y no solo quedarse estancados.

0.5 Ámbito del sistema

Con el crecimiento de la humanidad se ha dado un gran cariño hacia las mascotas tanto que los productos que ellos necesitan también se deben de comprar en una tienda especialmente para ellos, por lo mismo, se debe tener un control de los productos comprados. Las pocas tiendas para mascotas que existen por lo general no cuentan con un sistema que les permita controlar tanto sus ventas como el inventario con el que cuentan teniendo como consecuencia la perdida de ganancias, también es importante poder administrar los roles que se despeñan dentro de la empresa es decir tener claramente cuáles son las actividades que puede desarrollar un administrador, un vendedor o un cliente. Con el proyecto tienda de mascotas se espera desarrollar e implementar un sistema que permita solucionar esta problemática. Punto de venta Generar ventas de productos, teniendo así un mejor control de inventario, se buscara el producto según su código y el sistema debe mandar las características y precio del producto, se agrega a la tabla y se ira mostrando el total de la compra, después de haber agregado el total de productos pasara a generarse la venta, el sistema imprimirá en pantalla el ticket de la venta donde se mostrara detalladamente los productos el precio de cada uno el total de su compra y datos como el nombre de la tienda y el nombre del vendedor para cualquier aclaración y para un mejor manejo de ventas, una vez realizado esto restaran los productos que ya se vendieron de la base de datos, automáticamente se actualiza el inventario y se registra esa venta en el sistema para poder llevar un control de las ventas que se realizan anualmente y controlar las ganancias. Control de usuarios Otra de las funciones de este sistema será validar el inicio de sesión de los usuarios controlando así los roles de cada usuario donde el administrador tendrá acceso a la base de datos y puede agregar un producto al sistema ingresando el código de cada producto como su llave primaria para poder identificarlos de una forma rápida, agregaremos sus respectivas descripciones, su precio de compra y el precio de venta para que de esa forma podamos obtener las ganancias, teniendo en cuenta que los precios de los productos estarán variando constantemente es necesario poder modificar los precios o eliminar algún producto en caso de que se deje de fabricar o ponerlo como inactivo si por el momento no se trabaja con el , también puede agregar nuevos usuarios. Los vendedores solamente podrán realizar ventas o en su caso consultar el precio de algún producto en el cual el cliente esté interesado, evitando así el mal uso del sistema. Nos permitirá dividir los productos por categorías para tener un orden, se espera conocer cuáles son los productos que más se consumen y así tener el conocimiento en qué áreas tenemos que surtir la tienda. El administrador podrá en cualquier momento eliminar los registros de las ventas pasadas que ya no necesite, esto con la finalidad de ahorrar espacio en memoria evitando de esta manera gastos innecesarios, contará con aviso de alerta cuando el usuario elimine un registro, de esta forma el usuario estará seguro si desea eliminar tal registro, evitando el riesgo de eliminar registros importantes.

0.6 Mapa de navegación

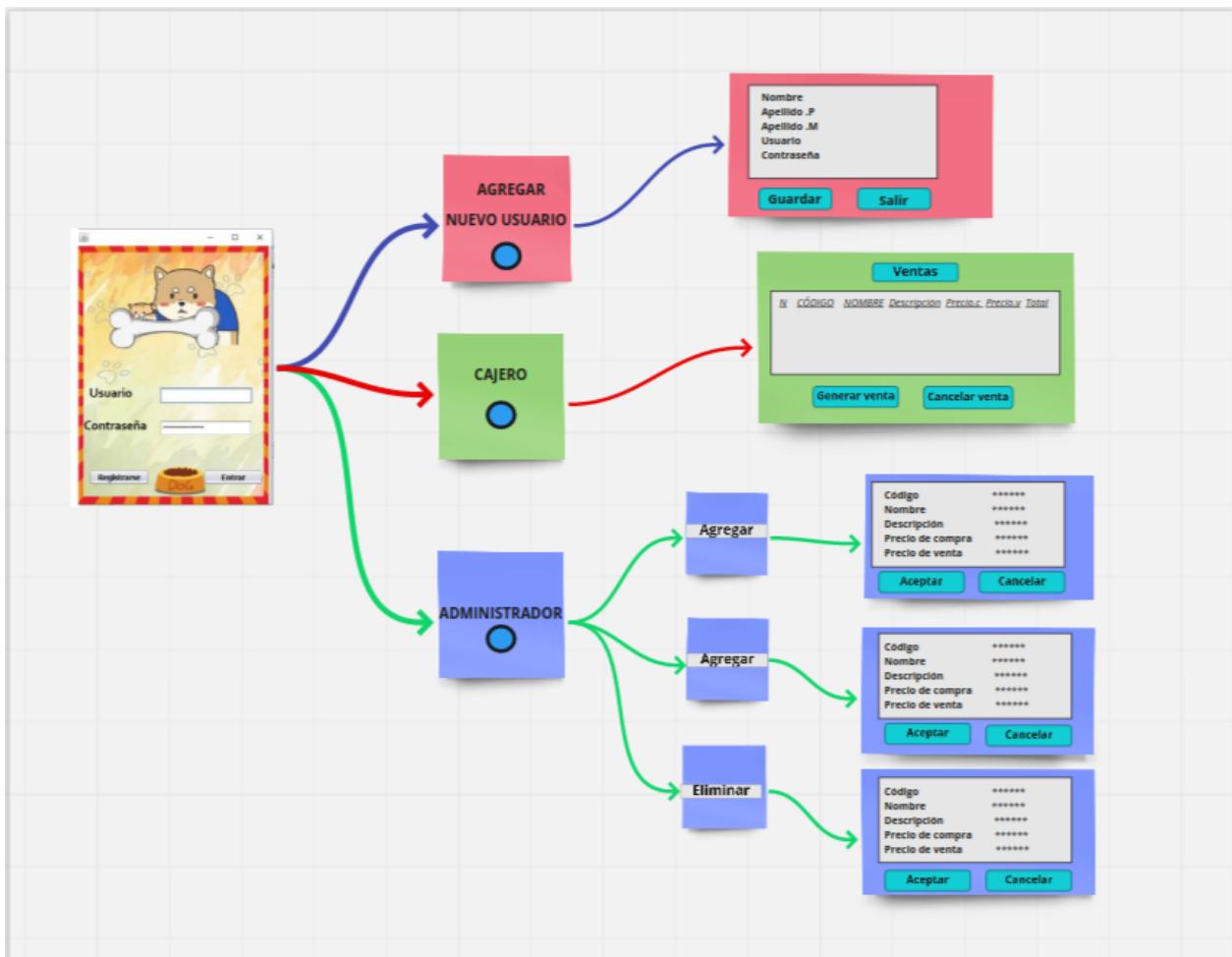


Figure 1: Mapa de navegación

“Interfaces graficas de la aplicación”



Figura 1: Figura 1: Login

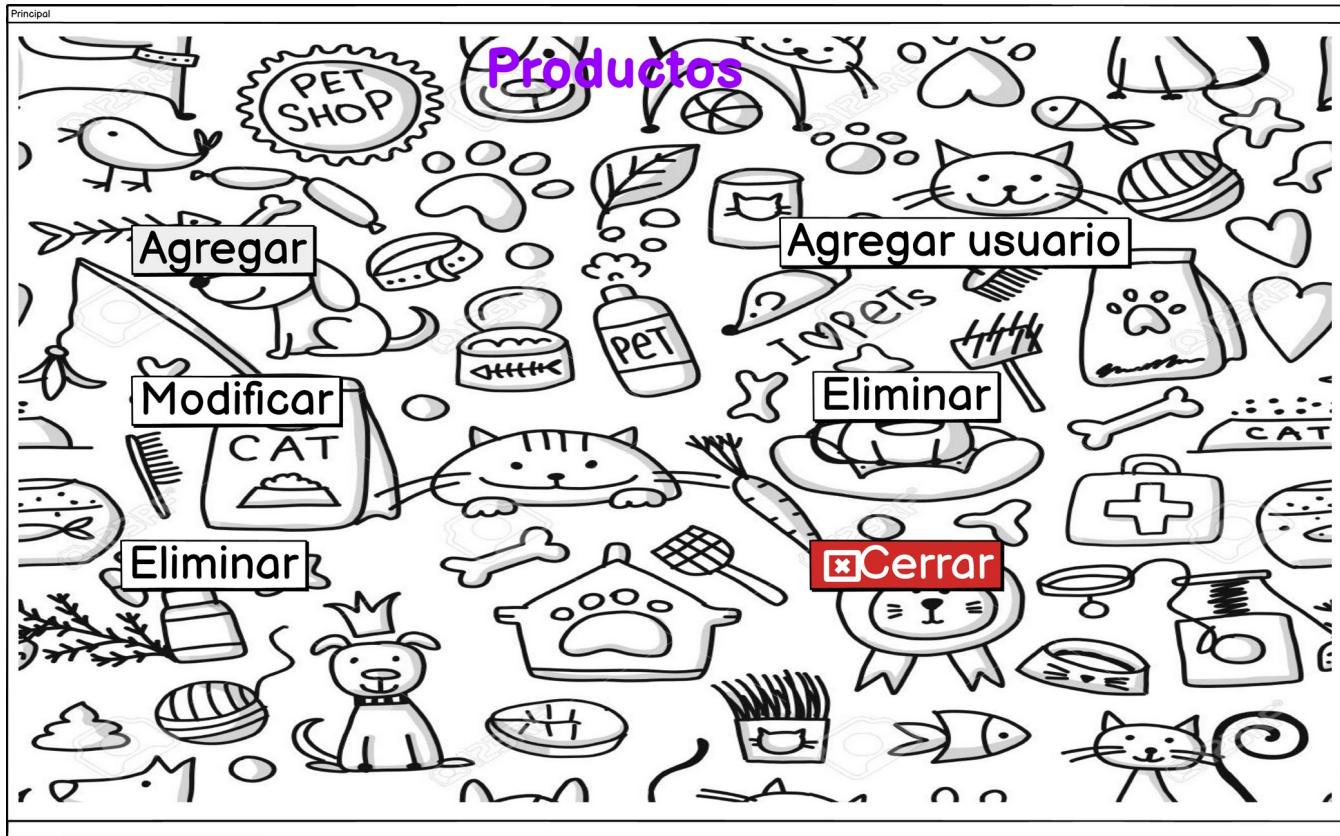


Figura 2: Figura 2: Pagina principal

Agregar Usuario

Agregar nuevo usuario

Nombre

Nombre Usuario

Contraseña

Tipo de usuario

Agregar

Figura 3: Agregar usuario

Agregar producto

Agregar nuevo producto

Codigo

Nombre

Descripcion

Precio

Agregar

Figura 4: Agregar producto

fiel_edit



N	codigo del producto	cantidad	Nombre	Descripcion	precio unitario	total
1	8057698	4	jaula mediana	\$2384		
2	40856971	3	Alimento para cachorro	\$840		
3	0856971	3	pecera	\$840		
4						
5						
6						
total				\$4800		

Figura 5: Figura 5: Tabla de ventas

Editar

Editar

Codigo

Buscar

Figura 6: Figura 6: Editar producto

Eliminar

Eliminar

Codigo

Buscar

Figura 7: Figura 7: Eliminar producto.

tiket

Tienda de mascotas

Atendido por: vendedor

Cantidad :

Producto

Precio:

Total:

Figura 8: Figura 8: Mostrar ticket de venta.

0.7 Requisito funcional

Código el requerimiento	RF_01
Nombre	Registros de productos
Propósito	Crear un nuevo registro para los productos.
Descripción	Estando en la página principal, el usuario debe crear un nuevo registro y llenar los datos que se requiere de dicho producto y al final debe de pulsar el botón guardar registro.
Entrada	Rellenar un formulario de los datos del producto, y guardar los datos.
Salida	El producto aparecerá registrado en la base de datos.
Prioridad	Alta

Figure 2: Requisito funcional

Código el requerimiento	RF_02
Nombre	Registro de usuario
Propósito	Registrar el nombre los usuarios que realizan compras.
Descripción	Se oprimirá el botón “Registrar usuario”, donde se capturara los datos del cliente y posteriormente guardarlos.
Entrada	Rellenar un formulario con los datos necesarios del usuario.
Salida	El nombre del cliente aparecerá en el registro de usuario.
Prioridad	Alta

Figure 3: Requisito funcional

Código el requerimiento	RF_03
Nombre	Modificación de datos del producto.
Propósito	Poder corregir o actualizar los datos de los productos si es necesario.
Descripción	Tras haber pasado el tiempo, si se desea cambiar algún dato del producto el usuario podrá actualizar los datos mediante un botón de “actualizar.”
Entrada	Se ingresara al formulario y se modificara los datos del producto.
Salida	Se mostrara un mensaje de cambio realizado correctamente.
Prioridad	Alta

Figure 4: Requisito funcional 3

0.8 Requisitos no funcionales

Código el requerimiento	RNF_01
Nombre	Seguridad
Descripción	El registro usado para manejar la seguridad en la información del usuario debe ser lo suficientemente confiable. La información sensible, como contraseñas debe manipular bajo algún nivel de en o cifrado.
Prioridad	Alta

Figure 5: Requisito funcional 1

Código el requerimiento	RNF_02
Nombre	Confidencialidad
Descripción	Toda la información otorgada por los usuarios se manipulará únicamente con fines corporativos y de manera limpia.
Prioridad	Alta

Figure 6: Requisito funcional 2

Código el requerimiento	RNF_03
Nombre	Look and feel.
Descripción	En esta sección el aspecto de la aplicación deberá ser presentar intuitivo para el usuario y debe tener consistencia.
Prioridad	Alta

Figure 7: Requisito funcional 3

Código el requerimiento	RNF_04
Nombre	Eficiencia.
Descripción	El software debe ser capaz de manejar toda la información recolectada a través del tiempo con fluidez
Prioridad	Media

Figure 8: Requisito funcional 4

Estándar de codificación

1. Tipos de Codificación

Notación Pascal – El primer carácter de todas las palabras se escribe en Mayúsculas y los otros caracteres en minúsculas.

Ejemplo: ColorDeFondo

```
public class HolaMundo(string MensajeCompleto) { ... }
```

2. Uso de nombres entendibles.

Usa palabras entendibles y descriptivas para nombrar a las variables. No usar abreviaciones.

Correcto:

```
string direccion;  
int usuario;
```

Incorrecto:

```
string nom;  
string dir;  
  
int pass;
```

3. Comentarios

Los comentarios deben de estar al mismo nivel que el código.

Correcto:

```
//Este es un comentario de ejemplo que está alineado  
string MensajeCompleto = "Hola" + nombre;  
  
MessageBox.Show(MensajeCompleto);
```

Incorrecto:

```
//Este es un comentario de ejemplo que NO está alineado
```

```
string MensajeCompleto = "Hola" + nombre;  
MessageBox.Show(MensajeCompleto);
```

Las llaves ({}) deben estar en el mismo nivel que el código fuera de las llaves.

```
boolean EsMayorDeEdad = false;  
  
if (Edad > 18)  
{  
    return EsMayorDeEdad = true; //Validación es mayor de edad  
}
```

3. Separador

Usar una línea en blanco como separador para dos grupos lógicos de código y deberá de haber solo una línea en

blanco para separar dos métodos dentro de una clase.

```
public class clase1(string Parametro1)  
{ ...  
}
```

```
public class clase2(string Parametro1)  
{ ...  
}
```

4. Nombres Descriptivos.

El nombre de los métodos deberá de describir de forma clara lo que ejecutara, si el nombre del método es suficientemente claro, no habrá que documentar la explicación de su función.

Correcto:

```
void CrearUsuario ( DataTable UsuarioDetalle )  
{ ...
```

```
}
```

Incorrecto:

```
void SalvaDetalle (DataTable DetEmp)  
{ ...  
}
```

5. Sangría

Usar TAB para la sangría, jamás use espacios, la sangría deberá de estar definida por 2 TABs

6. Encapsular código.

Se debe de separar la aplicación en múltiples ensamblados y métodos, esto permitirá la reutilización de código de forma sencilla, y dejará organizado el sistema por funcionalidades más específicas

7. Uso de sentencia Try, Catch,Finally

Uso de sentencias Try y Catch en las diferentes capas de la arquitectura, esto te permitirá por atrapar todas las excepciones que pueden generarse.

0.9 Personal Involucrado

Nombre	Roles
López López iris Estrella	Desarrollador
Sainos Herández Baldomero	SCRUM
Cristian Adair Ramirez Rodriguez	Auxiliar
Entrada	Formulario de perfil del cliente
M.T.C.A.Rolando-Pedro-Gabriel	Usuario Final
M.T.E.Everardo-de-Jesús-Pacheco Antonio	Programador
M.C.CLirio Ruiz Guerra	Administrador de la Base de Datos

Table 1: Personal Involucrado

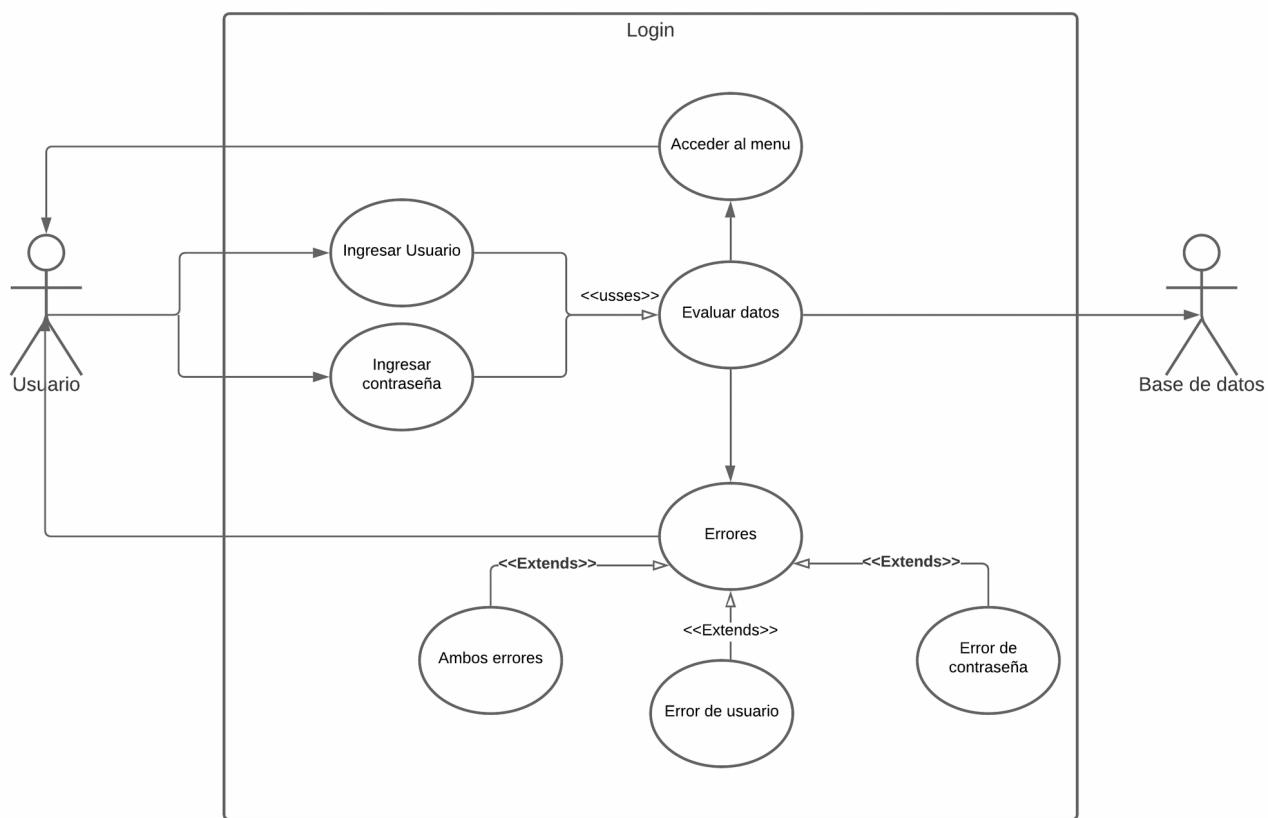


Figura 9: Diagrama de caso login

CASO DE USO			
Id:	CU – 1		
Nombre:	Login.		
Creado por:	S.H.B	Actualizado por:	L.L.I.E
Fecha de creación:	30/11/21	Fecha de última revisión:	30/11/21
Actores:	Base de datos, Administrador, Usuario.		
Descripción:	Ingresar los datos para el iniciar sesión.		
Disparador:	N/A		
Precondiciones:	Se envía un mensaje “Usuario y/o datos incorrectos” en una ventana emergente si los datos son incorrectos.		
Poscondiciones:	Se muestra la pantalla P.2.		
Flujo normal:	1. Ingresar nombre de usuario. 2. Ingresar la contraseña. 3. Oprimir el botón entrar. 3.1 Si el nombre de usuario no existe en la BD se enviará un mensaje en una ventana emergente.”El usuario no existe”. 3.2 Si los datos son correctos permitirá el acceso al sistema. 3.2.1 Si el dato del vendedor no existe en la BD se enviará un mensaje en una ventana emergente.”El vendedor no existe” . 4. El programa de muestra la pantalla P.2. 5. Regresar a Login P.1.		
Flujos alternos:	Oprimir botón cerrar y regresar a Login.		
Includes:	N/A		
Frecuencia de uso:	Media.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notas:	N/A		

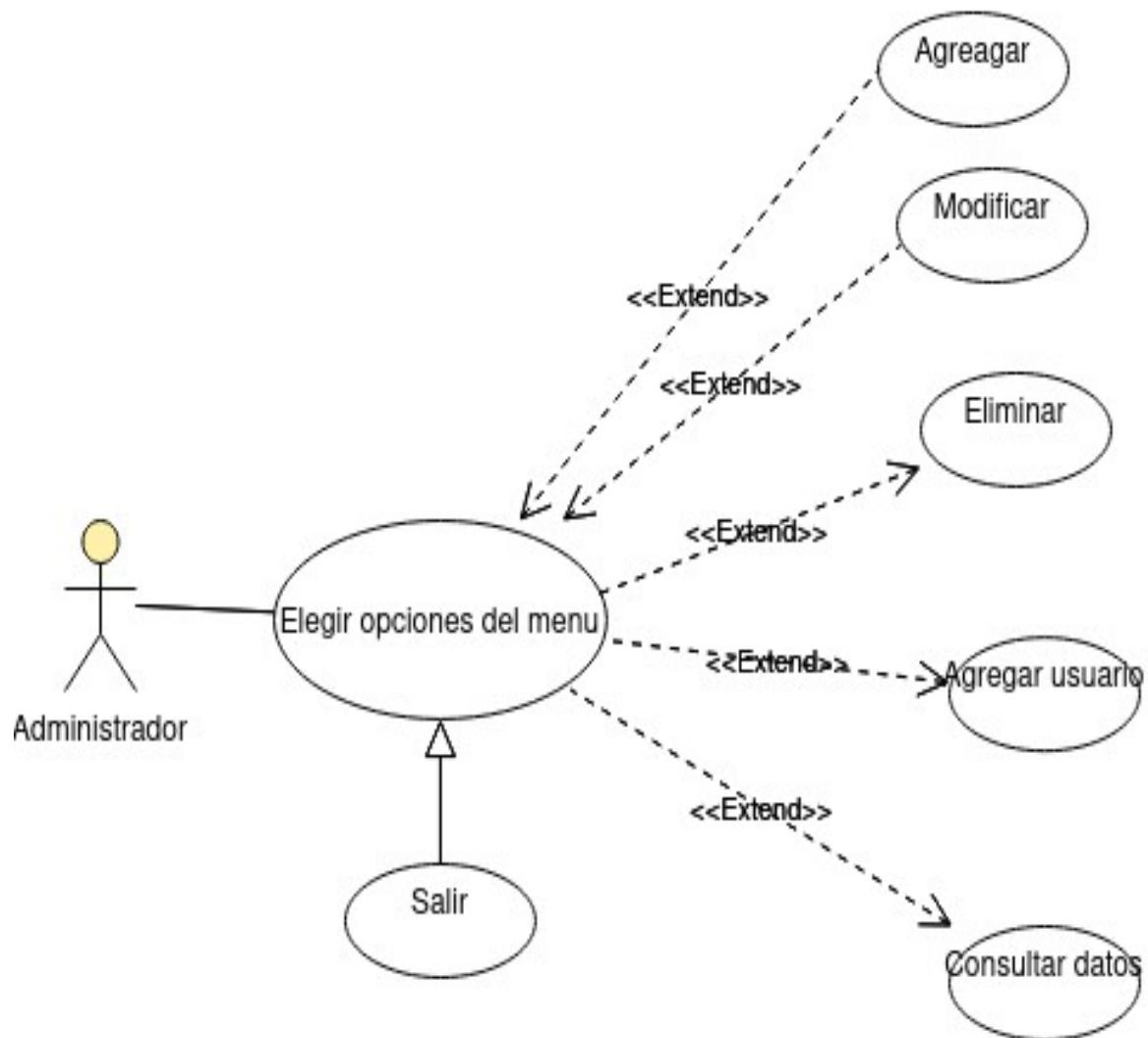


Figura 10: Diagrama de caso Menú

CASO DE USO			
Id:	CU – 2		
Nombre:	Menú Administrador.		
Creado por:	S.H.B	Actualizado por:	L.L.I.E
Fecha de creación:	06/11/21	Fecha de última revisión:	06/11/21
Actores:	Administrador.		
Descripción:	Permite al administrador ingresar a las pantallas P3_Agregar producto, P4_Modificar producto, P5_Eliminar producto, P6_Agregar usuario, P7_Ver inventario.		
Disparador:	N/A		
Precondiciones:	N/A		
Poscondiciones:	Se muestra la pantalla seleccionada, P3_Agregar producto, P4_Modificar producto, P5_Eliminar producto, P6_Agregar usuario, P7_Ver inventario.		
Flujo normal:	<ol style="list-style-type: none"> 1. Seleccionar botón agregar producto y se enviará a la pantalla P3.1. 2. Seleccionar botón modificar producto y se enviará a la pantalla P4.1. 3. Seleccionar botón eliminar producto y se enviará a la pantalla P5.1 4. Seleccionar botón agregar usuario y se enviará a la pantalla P6.1 5. Seleccionar botón ver inventario y se enviará a la pantalla P7.1 6. Seleccionar botón cerrar. <ol style="list-style-type: none"> a) Se enviará el mensaje, “Esta seguro de cerrar sesión como administrador”. b) Si el usuario selecciona la opción si, se enviará a la pantalla P1. c) Si el usuario selecciona la opción no, se regresará a la pantalla P2. 		
Flujos alternos:	Oprimir botón cerrar y regresar a Login.		
Includes:	N/A		
Frecuencia de uso:	Media.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notas:	N/A		

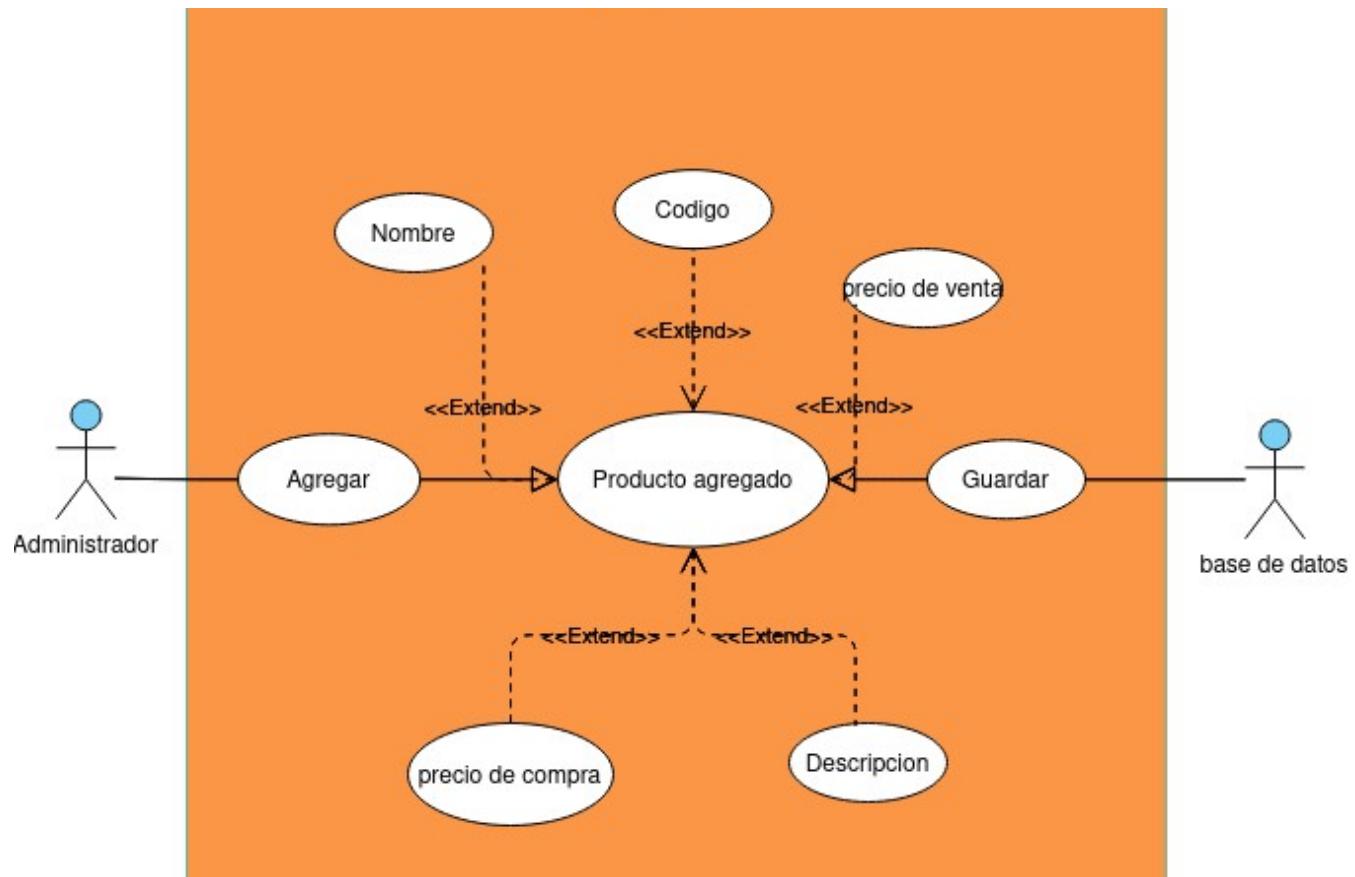


Figura 11: Diagrama de caso Agregar producto

CASO DE USO			
Id:	CU – 3		
Nombre:	Aregar producto.		
Creado por:	L.L.I.E	Actualizado por:	S.H.B
Fecha de creación:	1/12/21	Fecha de última revisión:	01/12/21
Actores:	Base de datos y Administrador.		
Descripción:	Aregar nuevos productos		
Disparador:	En el menú de la pantalla 2, presionar el botón Agregar producto.		
Predi-condiciones:			
Pos-condiciones:	Se guardan los datos y se limpian los campos de la ventana 3		
Flujo normal:	<p>1.- Ingresar el código del producto.</p> <p>2. Ingresar el nombre del producto.</p> <p>2.1 Si ingresa un numero le mostrará una ventana emergente con un mensaje “El dato debe ser ingresado con letras”.</p> <p>3. Agregar una breve descripción del producto.</p> <p>3.1 Si ingresa un numero le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con letras”.</p> <p>4. Agregar el precio de compra.</p> <p>4.1 Si ingresa una letra le mostrará una ventana emergente con un mensaje “El dato debe ser ingresado con números”.</p> <p>5. Agregar el precio de venta.</p> <p>5.1 Si ingresa una letra le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con números”.</p> <p>6. Ingresar la cantidad de productos que se agregarán .</p> <p>7. Oprimir el botón Agregar.</p> <p>7.1 Si alguno de los campos no está llenado se enviará un mensaje, “Rellenar todos los campos”.</p> <p>7.1.1-Oprimir el botón OK para salir de la ventana emergente y regresa al paso 7.</p> <p>7.2- Si los datos son correctos aparecerá el mensaje “Se agregó correctamente el producto”.</p>		
Flujos alternos:	Se cierra la ventana P3 y regresa a la pantalla P2.		
Includes:	N/A		
Frecuencia de uso:	Frecuente.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notes:	N/A		

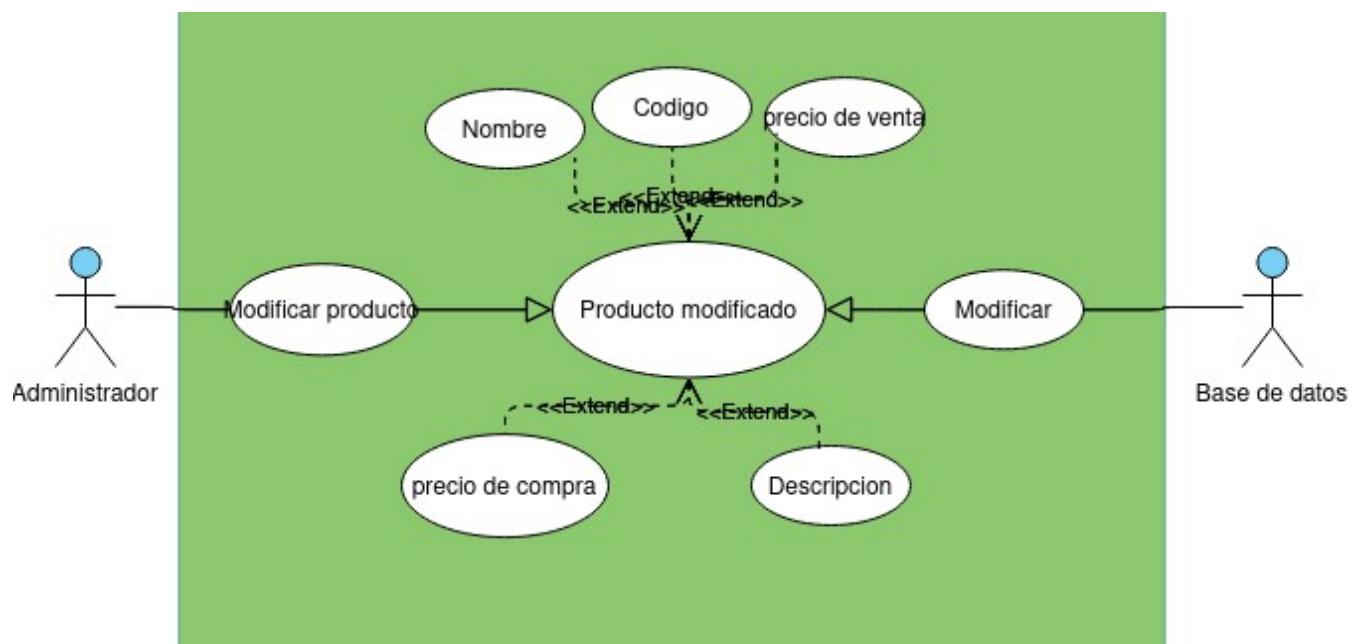


Figura 12: Diagrama de caso de Modificar

CASO DE USO			
Id:	CU – 4		
Nombre:	Modificar producto.		
Creado por:	S.H.B	Actualizado por:	L.L.I.E
Fecha de creación:	02/12/21	Fecha de última revisión:	02/12/21
Actores:	Base de datos, Administrador.		
Descripción:	Permite al administrador modificar el registro del producto.		
Disparador:	Seleccionar el botón modificar de la pantalla P2.		
Pre-condiciones:	N/A		
Pos-condiciones:	Se regresa a la pantalla P2.		
Flujo normal:	1._Ingresar el código del producto. 2._Presionar el botón buscar. 2.1._ Si el código ingresado no existe en la base de datos se mostrará el siguiente mensaje, “El código no existe ”. 2.2._Si existe el código del producto, enviar a la pantalla P4.1. 3._En la pantalla P4.1 se muestra los campos, Código, Nombre, Descripción, Precio de compra, Precio de venta, Cantidad con los datos existentes. 3.1._Se ingresan los datos nuevos a modificar. a) Si deja un campo vacío, enviara el siguiente mensaje, “Rellenar todos los campos”. b) Si el campo nombre, se ingresa con números se enviará el mensaje, “El campo debe ser llenado con letras”. c) Si En el campo Código, Precio de compra, Precio de venta, se ingresan letras se enviara el mensaje, “Los campos deben ser llenados con números”. 6.-Se confirma y se guarda la modificación.		
Flujos alternos:	Cerrar la venta P4, y regresar a la pantalla P1.		
Includes:	N/A		
Frecuencia de uso:	Frecuente.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notas:	N/A		

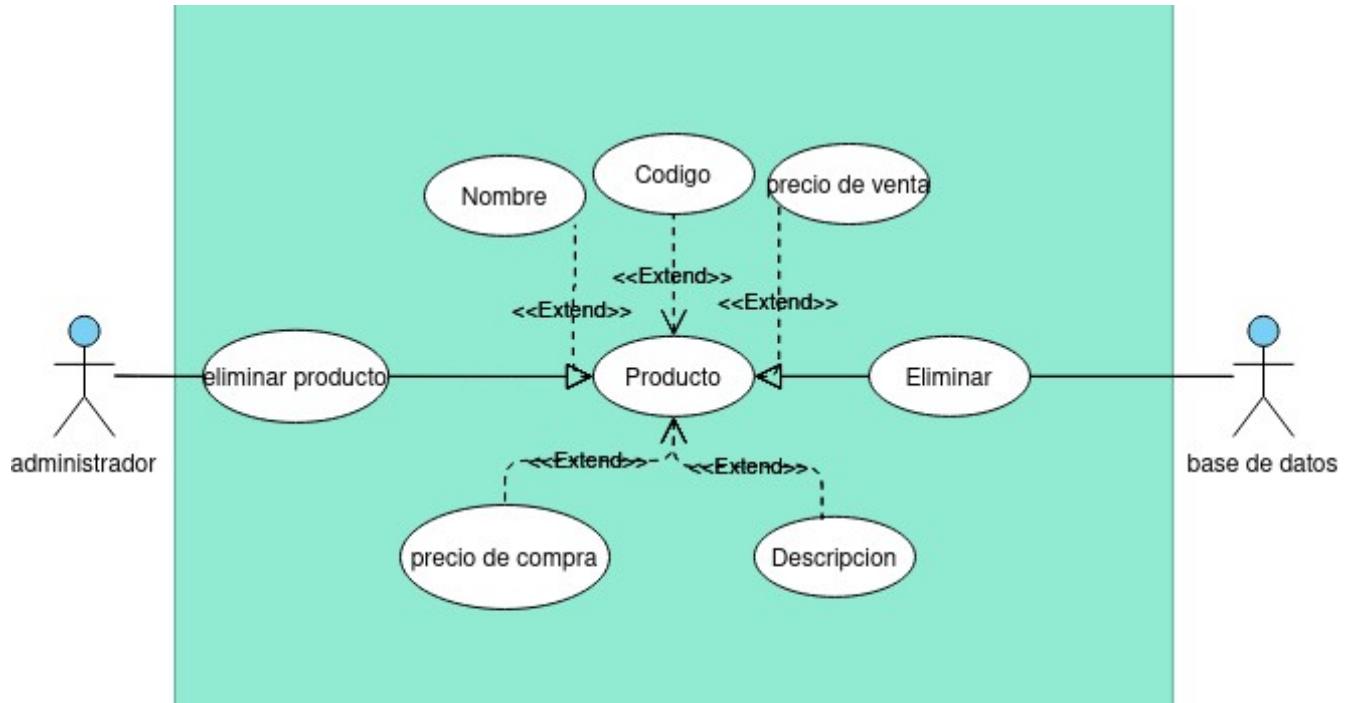


Figura 13: Diagrama de caso de eliminar

CASO DE USO			
Id:	CU – 5		
Nombre:	Eliminar producto.		
Creado por:	S.H.B	Actualizado por:	L.L.I.E
Fecha de creación:	06/12/21	Fecha de última revisión:	06/11/21
Actores:	Base de datos, Administrador.		
Descripción:	Permite eliminar un producto existente en el sistema.		
Disparador:	Seleccionar el botón modificar de la pantalla P2.		
Pre-condiciones:	N/A, El administrador debe estar autenticado.		
Pos-condiciones:	Se regresa a la pantalla P2.		
Flujo normal:	1._Ingresar el código del producto. 2._Presionar el botón buscar. 2.1._ Si el código ingresado no existe en la base de datos se mostrará el siguiente mensaje, “El código no existe ”. 2.2._Si existe el código del producto, enviar a la pantalla P5.1. 3._En la pantalla P5.1 Se muestran los campos, Código, Nombre, Descripción, Precio de compra, Precio de venta, Cantidad, con los datos existentes. 4._Se seleccionar el producto a eliminar. 5._ Oprimir el botón eliminar. 5.1_ Se enviará un mensaje de de confirmación, “Esta seguro de eliminar el producto *****”. a) Si el administrador confirma que si, el producto sera eliminado del inventario y se regresará a la pantalla p5. b) Si el administrador confirma que no, se regresa a la pantalla P5.		
Flujos alternos:	Cerrar la venta P5, y regresar a la pantalla P2.		
Includes:	N/A		
Frecuencia de uso:	Muy baja.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notas:	N/A		

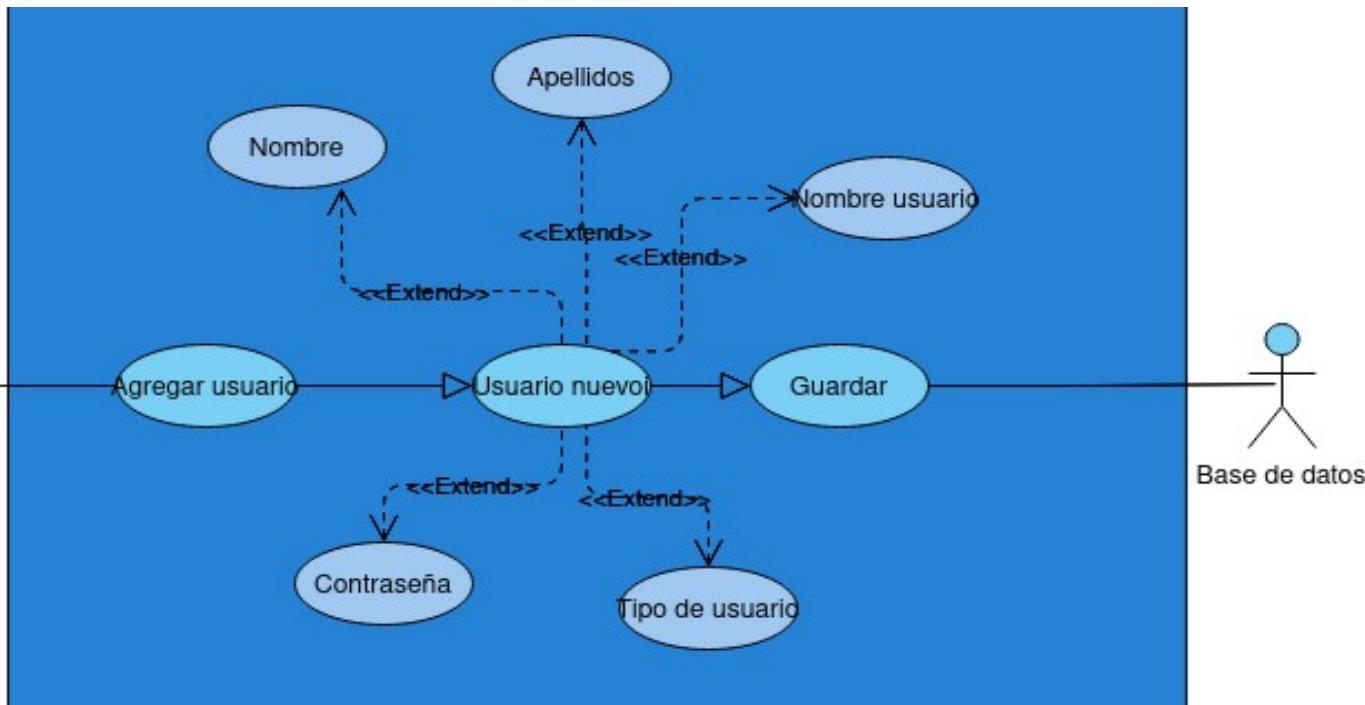


Figura 14: Diagrama de casos de Agregar usuario

CASO DE USO		
Id:	CU – 6	
Nombre:	Aregar usuario	
Creado por:	L.L.I.E	Actualizado por: S.H.B
Fecha de creación:	2/12/21	Fecha de última revisión:
Actores:	Base de datos, Administrador.	
Descripción:	Aregar nuevo usuario	
Disparador:	En el menú de la pantalla 2, oprimir el botón Agregar usuario	
Predi-condiciones:		
Pos-condiciones:	Se guardan los datos y se limpian los campos de la ventana 6	
Flujo normal:	<ol style="list-style-type: none"> 1. Ingresar el nombre del usuario. 2. Ingresar su apellido. 2.1 Si ingresa un numero le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con letras”. 3. Ingresar el nombre de usuario. 3.1 Si ingresa un numero le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con letras”. 4. Ingresar la contraseña. 4.1 Si ingresa un numero le mostrara una ventana emergente con un mensaje “El dato debe ser ingresado con letras”. 5. Elige el tipo de usuario (administrador o vendedor). 6. Oprimir el botón Agregar . 6.1 Si alguno de los campos no esta rellenado se enviara un mensaje “Rellenar todos los campos”. 6.2 Oprimir el botón OK para salir de la ventana emergente y regresa al paso 6. 6.3 Si los datos son correctos aparecerá el mensaje “se agregó correctamente”. 6.4. presionar OK para salir de la ventana se limpian los campos y regresa al paso 1. 7. Si oprime el botón salir regresa a la pantalla P2. 	
Flujos alternos:	Se cierra la pantalla P6 y regresa a la pantalla 2.	
Includes:	N/A	
Frecuencia de uso:	Frecuente.	
Requerimientos especiales:	N/A	
Supuestos:	N/A	

Issues y Notes:

N/A

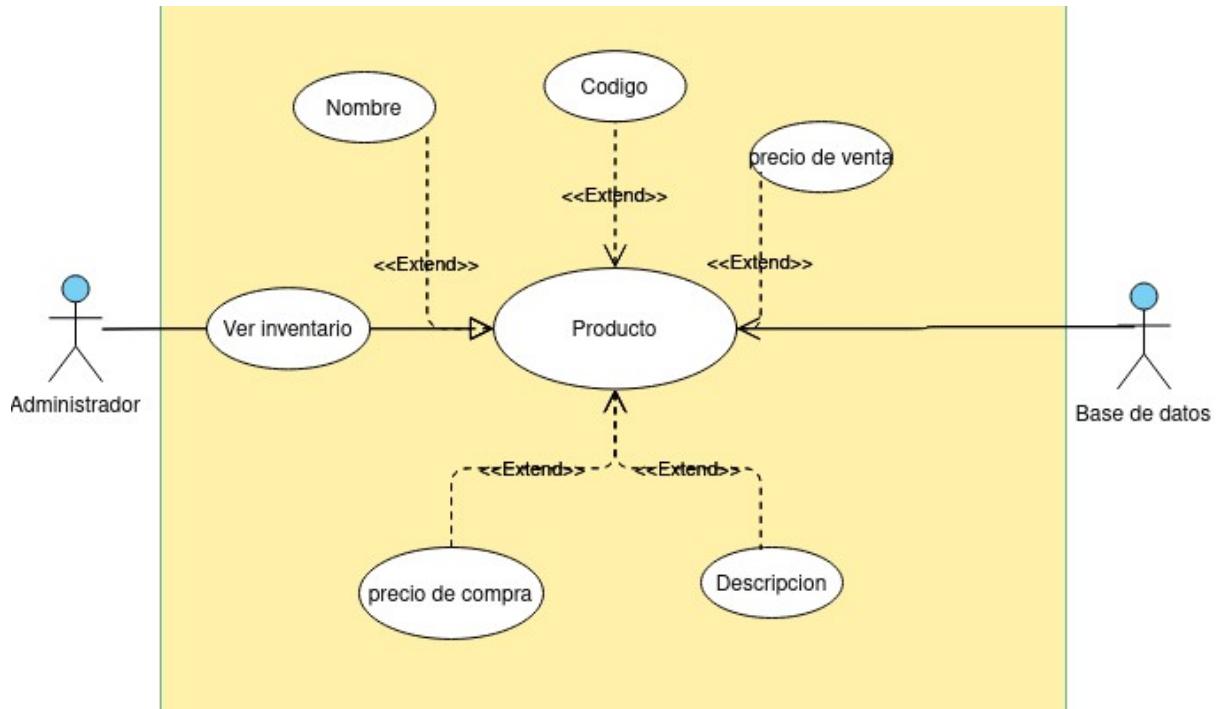


Figura 15: Diagrama de caso de inventario

CASO DE USO			
Id:	CU – 7		
Nombre:	Inventario		
Creado por:	L.L.I.E	Actualizado por:	S.H.B
Fecha de creación:	3/12/21	Fecha de última revisión:	06/12/21
Actores:	Base de datos, Administrador.		
Descripción:	Muestra el inventario de la tienda.		
Disparador:	En el menú de la pantalla 2, oprimir el botón inventario.		

Predi-condiciones:	N/A
Pos-condiciones:	N/A
Flujo normal:	<p>1.- Se mostrará el inventario en una tabla .</p> <p>2.- Si desea buscar algún producto en particular ingrese su código o nombre en la caja de texto.</p> <p>3.- Presionar el botón salir para regresar a la ventana P2 .</p>
Flujos alternos:	Se cierra la pantalla P7 y regresa a la pantalla P2.
Includes:	N/A
Frecuencia de uso:	Frecuente.
Requerimientos especiales:	N/A
Supuestos:	N/A
Issues y Notes:	N/A

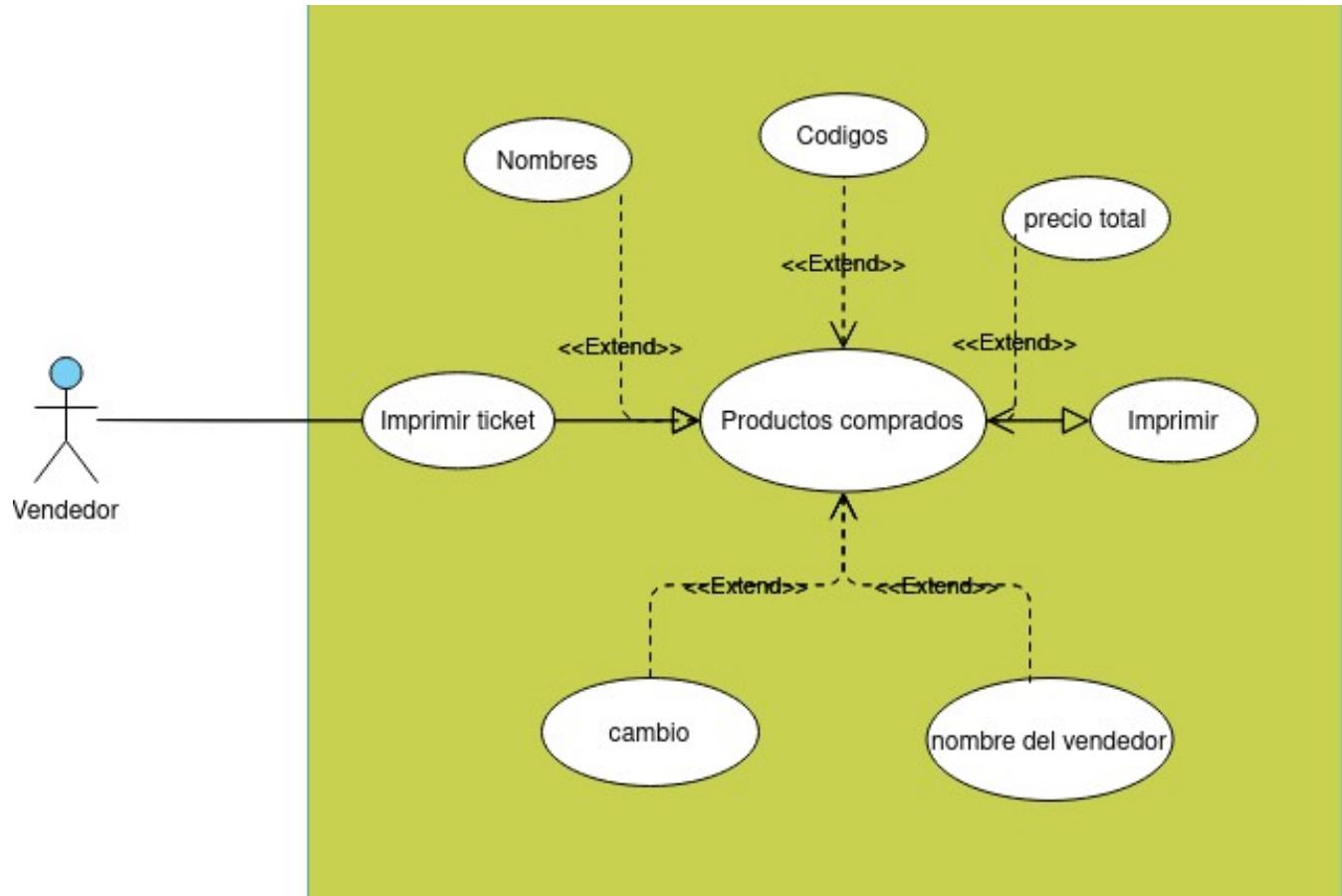


Figura 16: Diagrama de caso de ticket

CASO DE USO			
Id:	CU – 8		
Nombre:	Ticket.		
Creado por:	S.H.B	Actualizado por:	L.L.I.E
Fecha de creación:	3/12/21	Fecha de última revisión:	06/12/21
Actores:	Vendedor.		
Descripción:	Muestra el ticket de la compra de los productos.		
Disparador:	En el punto de venta, oprimir el botón aceptar.		
Predi-condiciones:	N/A		
Pos-condiciones:	N/A		
Flujo normal:	1.- Se muestra los detalles de la venta.		
Flujos alternos:	Se cierra la pantalla P8 y regresa a la pantalla P7.		
Includes:	N/A		
Frecuencia de uso:	Muy frecuente.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notes:	N/A		

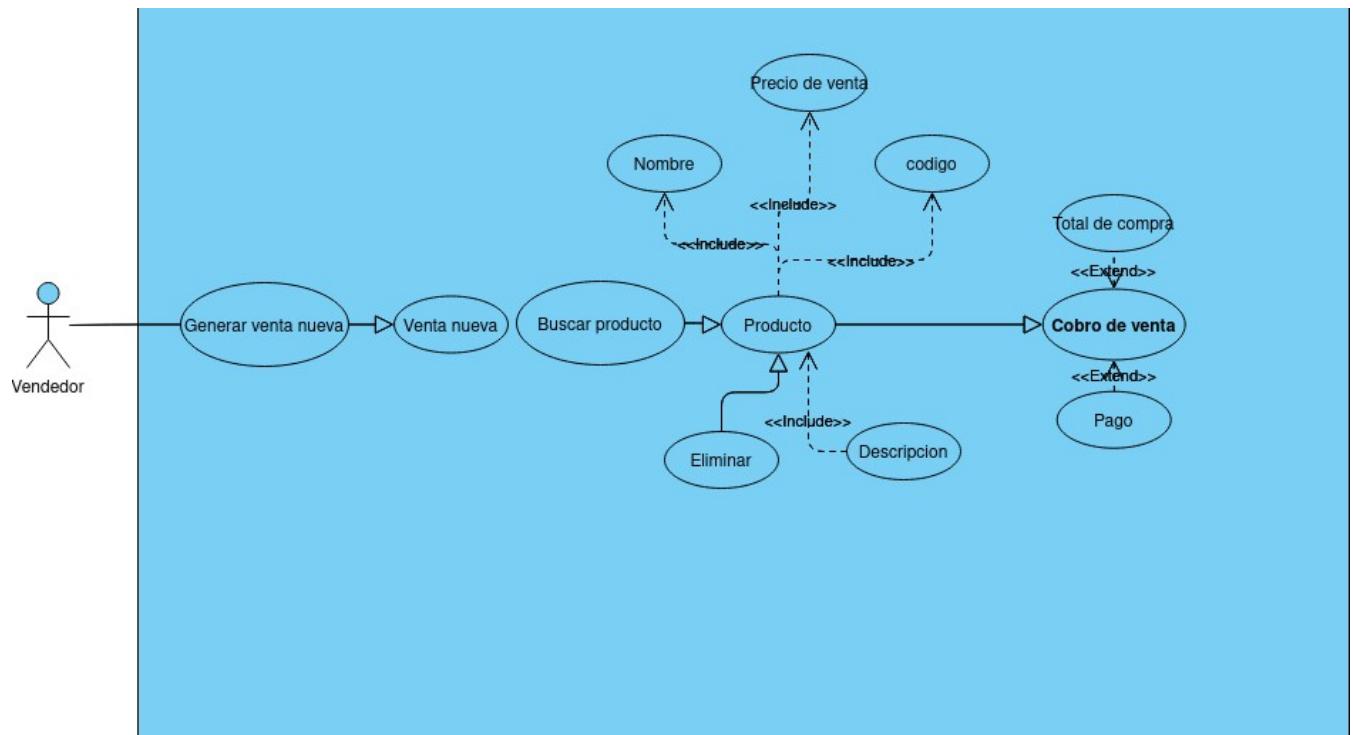


Figura 17: Diagrama de caso del punto de venta

CASO DE USO			
Id:	CU – 9		
Nombre:	Punto de venta.		
Creado por:	L.L.I.E	Actualizado por:	S.H.B
Fecha de creación:	06/12/21	Fecha de última revisión:	06/12/21
Actores:	Base de datos y vendedor.		
Descripción:	Ejecuta venta de productos.		
Disparador:	Inicio de sesión.		
Predi-condiciones:	N/A		
Pos-condiciones:	N/A		
Flujo normal:	1.- En el menú principal elegir la opción “Archivo” , en el submenu seleccionar “Nueva venta”. 2.- Aparecerá una tabla. 2.-Para agregar un producto a la venta escribir el código en la caja de texto. 4.- En la tabla aparecerán los datos del producto al que corresponde ese código. 5.- En la parte superior aparecerá el subtotal y posteriormente el total en caso de aplicar algún descuento. 6.- Oprimir el botón aceptar. 6.1.- Guarda los datos de la venta. 7.- El botón cancelar cancelará la venta y limpiara la tabla.		
Flujos alternos:	Se cierra la pantalla P9 y regresa a la pantalla P2.		
Includes:	N/A		
Frecuencia de uso:	Frecuente.		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y Notes:	N/A		

Diccionario de datos

Tabla de Usuarios

La tabla usuarios tiene 6 campos el primero es el id_usuario, es una llave primaria que es de tipo **varchar** y como máximo tendrá 10 caracteres para identificar a cada usuario.

El campo nombre es de tipo **varchar** como máximo tendrá 30 caracteres para agregar nombres de usuario.

El campo Apellido es de tipo **varchar** como máximo tendrá 30 caracteres para agregar los apellidos del usuario.

El campo Nombre_usuario es de tipo **varchar** como máximo tendrá 30 caracteres para agregar el nombre que desea usar el usuario.

El campo contraseña es de tipo **varchar** como máximo tendrá 30 caracteres para agregar la contraseña del usuario.

El campo nombre es de tipo **varchar** como máximo tendrá 30 caracteres para agregar el Tipo de usuario.

Columna	Tipo de dato	Descripción
PK	Id_usuario	Varchar(10)
	Nombre	Varchar (30)
	Apellido	Varchar (30)
	Nombre_usuario	Varchar (30)
	contraseña	Varchar (30)
	Tipo	Varchar (30)

Tabla de Productos

Columna	Tipo de dato	Descripción
PK	código	Varchar(10)
	Existencia	integer
	Nombre	Varchar (30)
	Descripción	Varchar (30)
	PrecioC	float
	PrecioV	float

Tabla de Ventas

Tabla de ventas_detalles

	Columna	Tipo de dato	Descripción
PK	Folio	Varchar (10)	Folio de venta
	Fecha	Varchar (10)	Fecha de venta

	Columna	Tipo de dato	Descripción
FK	Folio	Varchar (10)	Folio de la venta
FK	Id_producto	Varchar (10)	Código del producto
	cantidad	integer	cantidad de productos vendidos

Chapter 1

Scrip BD

Scrip BD

```
create database tienda_mascotas;
create table producto (codigo varchar (10) primary key, existencia integer,
Nombre varchar (30), descripcion varchar (30), precioc float, preciov float);

create table usuarios (id varchar (10) primary key, nombre varchar (30),
apellido varchar (30), nombre_user varchar (20), contraseña varchar (20),
tipo_user varchar (20));

create table ventas_fecha (folio varchar (20) primary key, fecha varchar (30));

create table venta_detalles (folio varchar (20) primary key,
codigo varchar (10) primary key, cantidad integer,
foreign key (folio) references ventas_fecha(folio),
foreign key (codigo) references producto(codigo) );

-- Revocando privilegios del rol 'public'
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON DATABASE mascotasbd FROM PUBLIC;

-- Rol solo lectura
-- Estos usuarios solo podran consultar los datos pero no pueden agregar ni eliminar de la base de datos
CREATE ROLE vendedor;
GRANT CONNECT ON DATABASE mascotasbd TO vendedor;
GRANT USAGE ON SCHEMA tienda TO vendedor;
GRANT SELECT ON ALL TABLES IN SCHEMA tienda TO vendedor;
ALTER DEFAULT PRIVILEGES IN SCHEMA tienda GRANT SELECT ON TABLES TO vendedor;

-- Creacion de usuarios
CREATE USER estrella WITH PASSWORD 'lopez';
CREATE USER sainos WITH PASSWORD 'baldomero';
```

```
-- Rol lectura-escritura
-- Este rol permite la lectura como la escritura en la base de datos
CREATE ROLE administrador;
GRANT CONNECT ON DATABASE mascotasbd TO administrador;
GRANT USAGE, CREATE ON SCHEMA tienda TO administrador;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA tienda TO administrador;
ALTER DEFAULT PRIVILEGES IN SCHEMA tienda GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO administrador;
GRANT USAGE ON ALL SEQUENCES IN SCHEMA tienda TO administrador;
ALTER DEFAULT PRIVILEGES IN SCHEMA tienda GRANT USAGE ON SEQUENCES TO administrador;

-- Creacion de usuarios
CREATE USER cristian WITH PASSWORD 'adair123';
CREATE USER baldo WITH PASSWORD 'sainos';

-- Concedemos permisos a cada usuario
GRANT estrella TO administrador;
GRANT sainos TO administrador;
GRANT cristian TO vendedor;
GRANT baldo TO vendedor;
```

Chapter 2

Diagrama de actividades

Login

```
@startuml
start
#palegreen:Login;
:Ingresamos usuario;
if ( Verifica si el ususario existe) then
    :Error el usuario no existe;
    stop
endif
:Limpia los campos;
stop
@enduml
```

Menú Administrador

```
@startuml
start
#palegreen:Menú Administrador;
if (agregar producto y) then (Yes)
    :Confirmar;
elseif (Modificar producto) then (Yes)
    :Confirmar;
elseif (Eliminar producto ) then (Yes)
    :Confirmar;
elseif (Agrega usuario) then (Yes)
    :Confirmar;
elseif (Ver inventario) then (Yes)
    :Confirmar;
else (Salir)
    :Regresa a menu;
endif
stop
@enduml
```

Agregar usuario

```
@startuml
start
#palegreen: Agregar usuario;

:Ingresar el nombre del usuario;
:Ingresar su apellido;
:Cerrar la venta P2, y regresar a la pantalla P1;
:Ingresar la contraseña;
:Elige el tipo de usuario (administrador o vendedor);
:Oprimir el botón Agregar;
:ok;
stop
@enduml
```

Agregar producto

```
@startuml

start
#palegreen: Agregar producto;

:Se Muestra los campos para agrgar;
:Se guardan los datos;
:Cerrar la venta P3, y regresar a la pantalla P2;
stop
@enduml
```

Modificar producto

```
@startuml
start
#palegreen: Modificar producto;
if (Si el código existe) then
    :Si no existe el código, error;
    stop
endif
:Se Muestra los datos a modificar;
:Se guardan los datos;
:Cerrar la venta P2, y regresar a la pantalla P1;
stop
@enduml
```

Mostrar inventario

```
@startuml
start
#palegreen: Mostrar inventario;
:Oprimir el boton mostrar inventario;
if (Ingresamos código del producto) then
    :Error el usuario no existe;
```

```
    stop
endif
:Se muestra la tabla;
:Presionar el botón salir para regresar a la ventana P2 .;
stop
@enduml
```

 Eliminar producto

```
@startuml
start
#palegreen:Eliminar producto;
if (Si el código existe) then
    :Si no existe el código, error;
    stop
endif
:Se Muestra el producto a eliminar;
:Cerrar la venta P5, y regresar a la pantalla P2;
stop
@enduml
```

 Ticket

```
@startuml
start
#palegreen:Ticket;
:Oprimimos el boton mostrar ticket;
:Se muestra los detalles de la venta;
:salir;
stop
@enduml
```

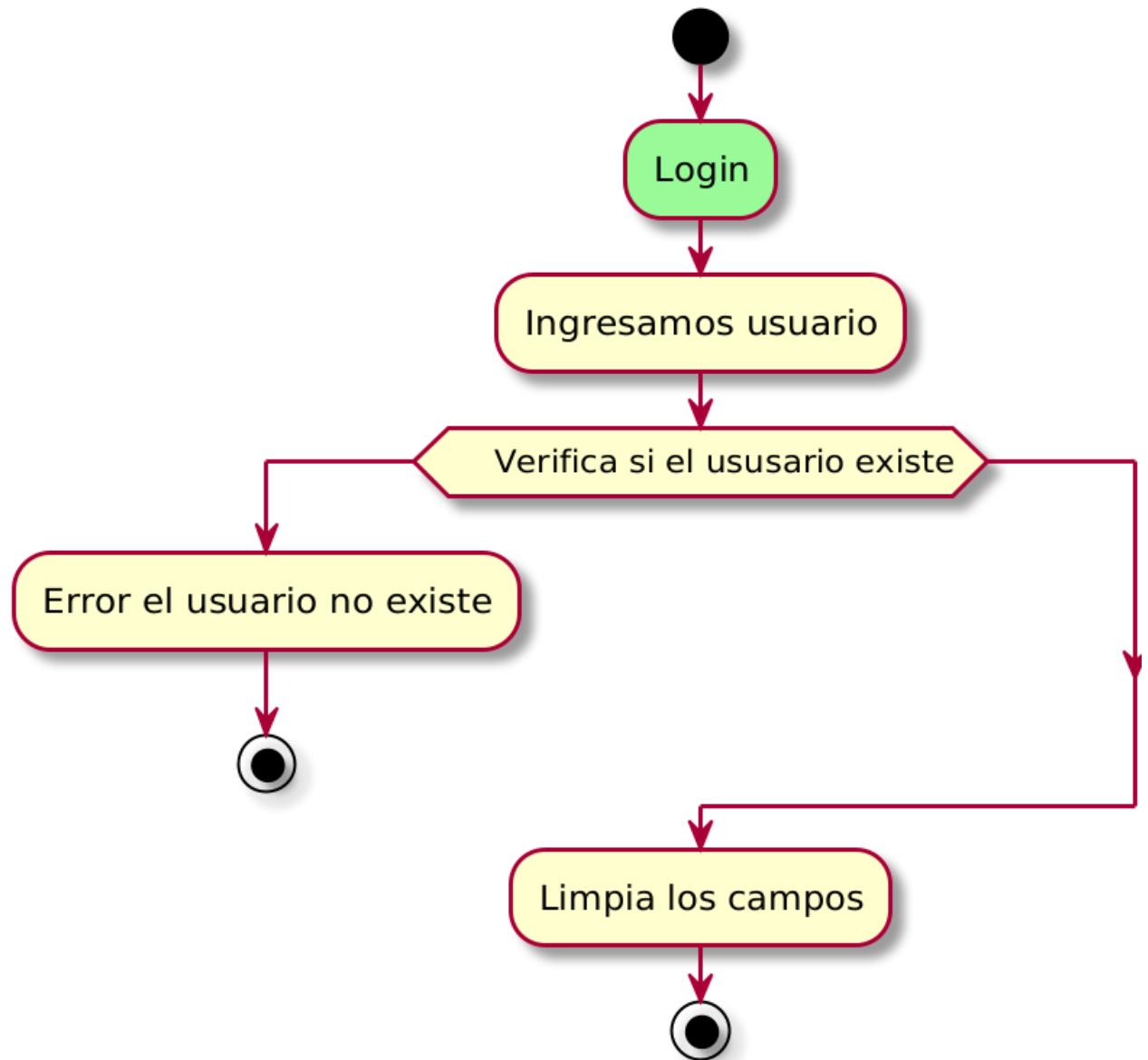


Figure 2.1: Login

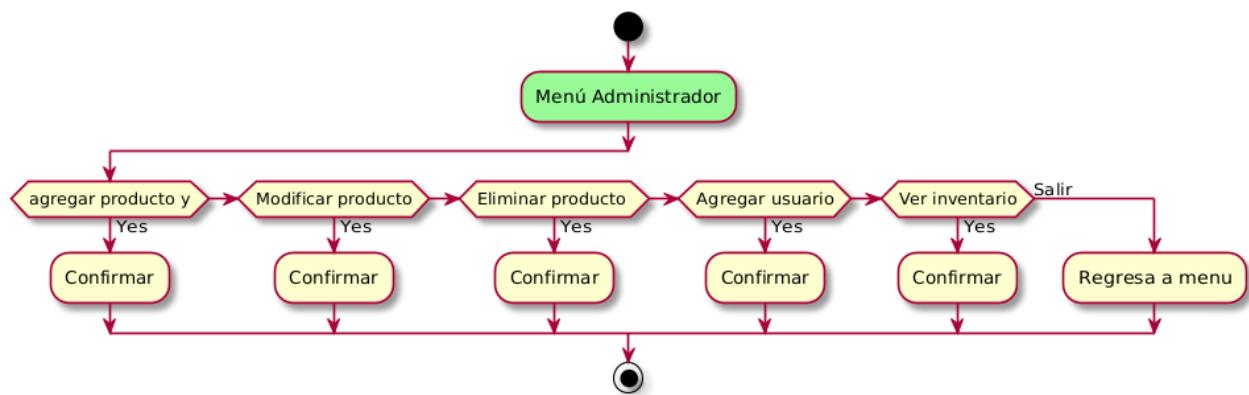


Figure 2.2: Menú administrador

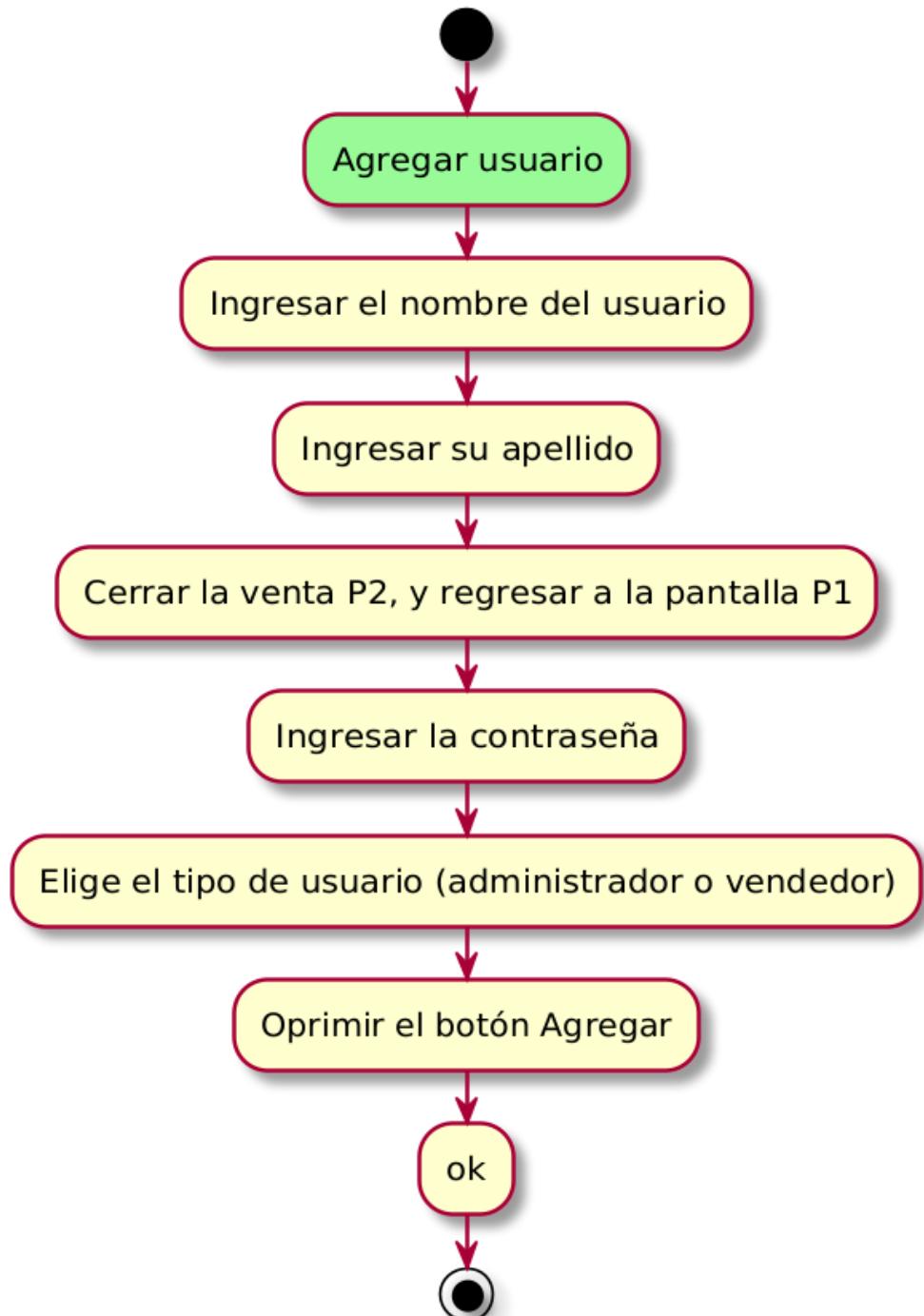


Figure 2.3: Agregar Usuario

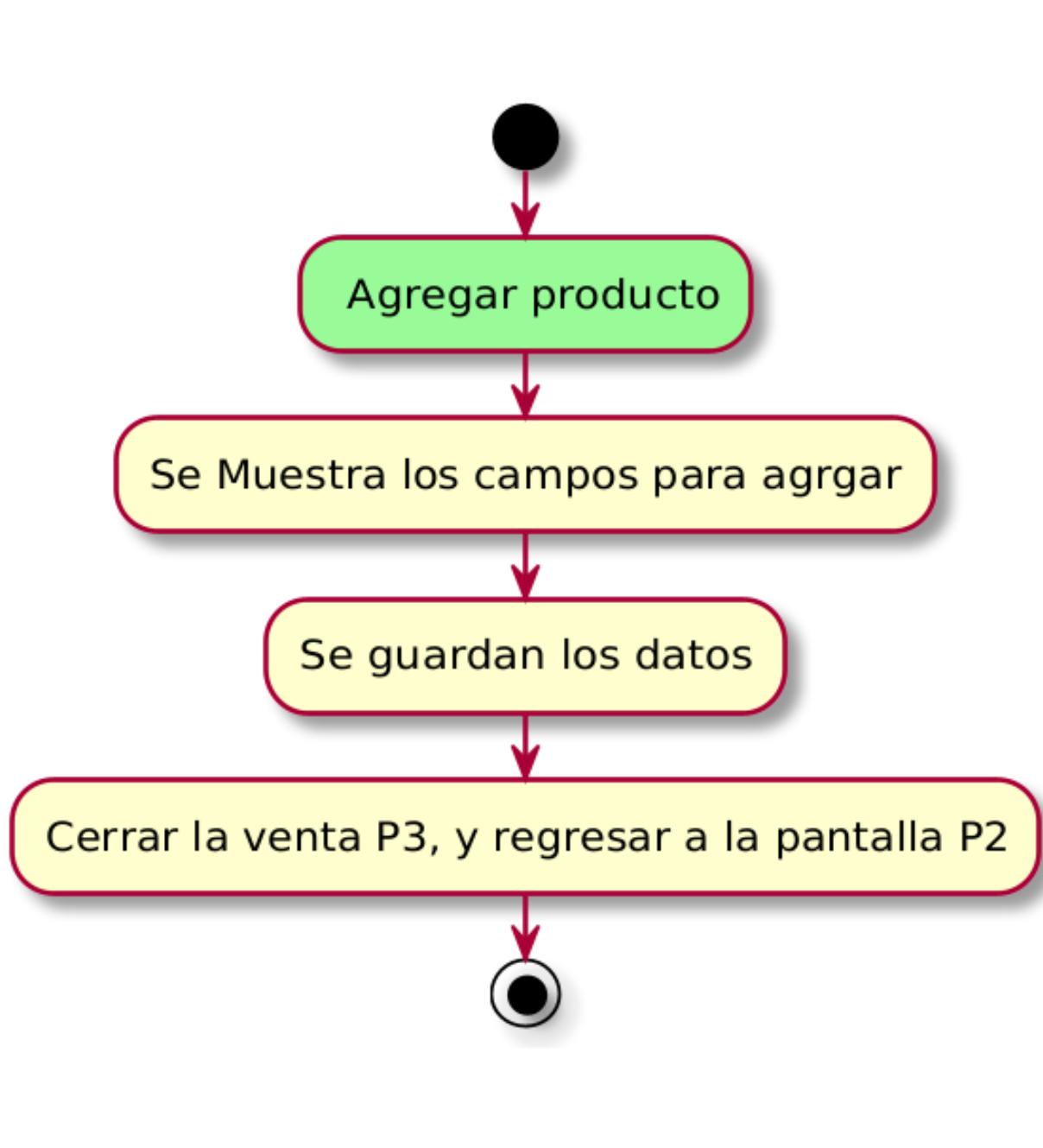


Figure 2.4: Agregar Producto

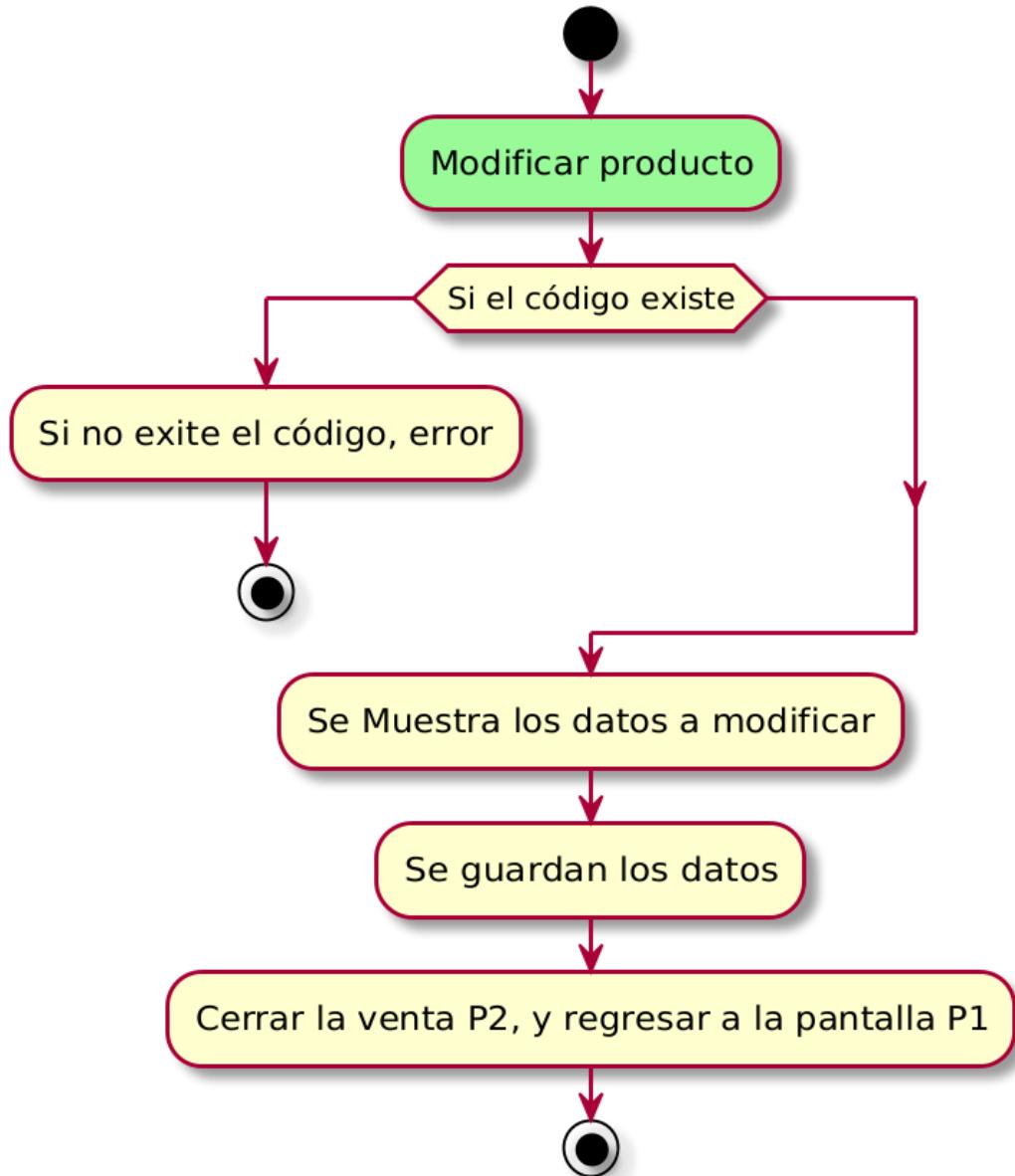


Figure 2.5: Modificar Producto

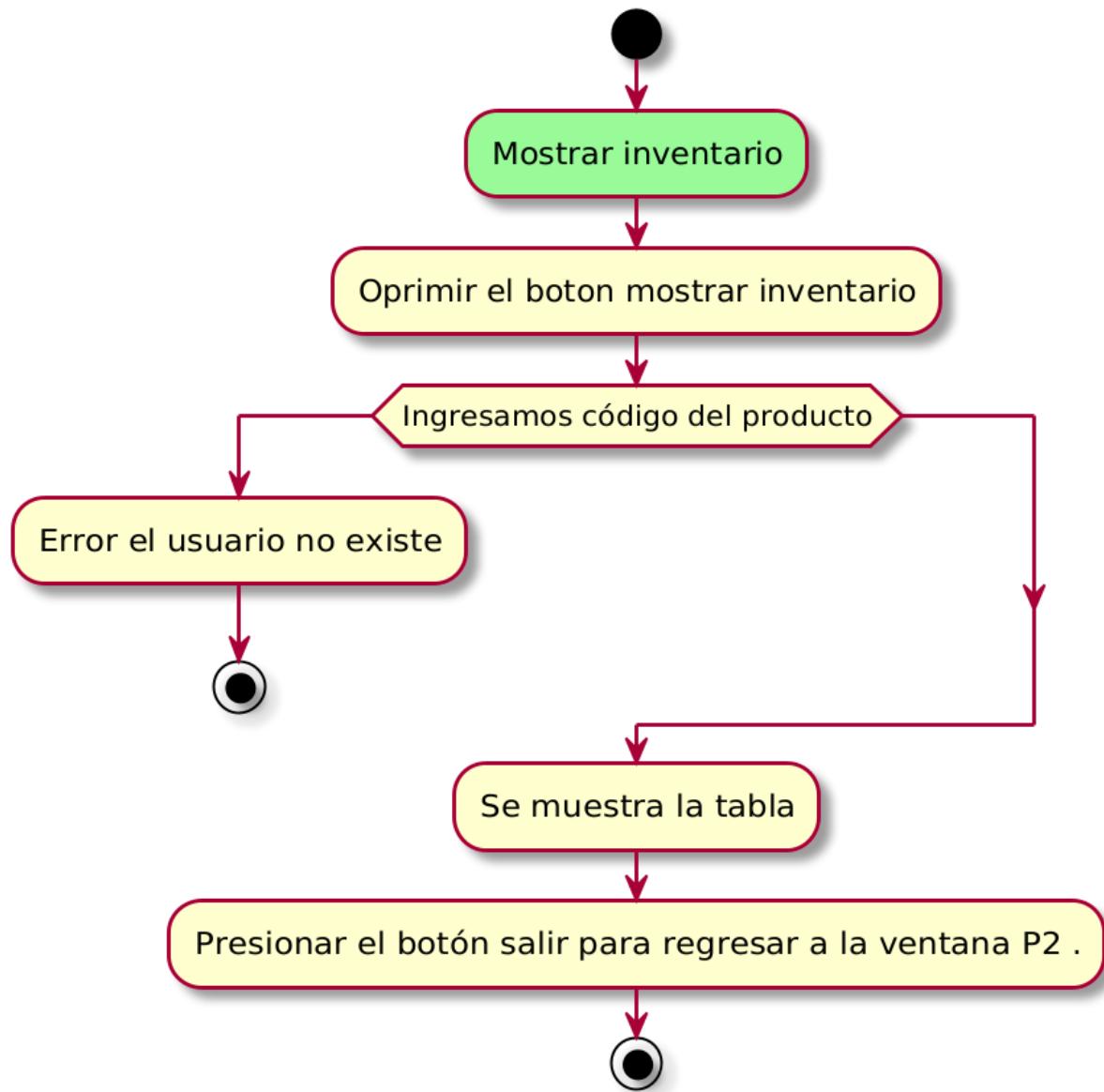


Figure 2.6: Modificar Inventario

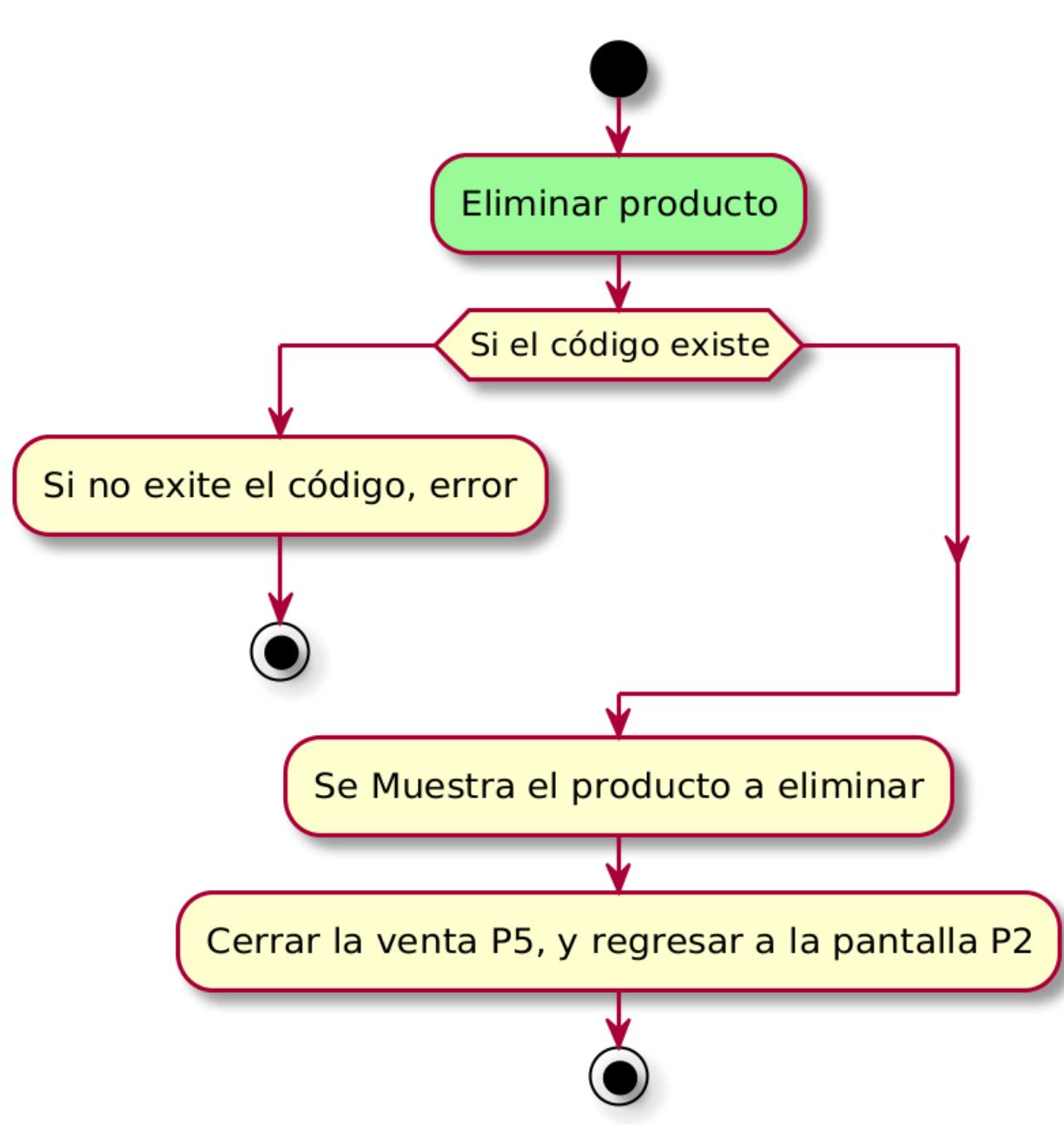


Figure 2.7: Eliminar producto

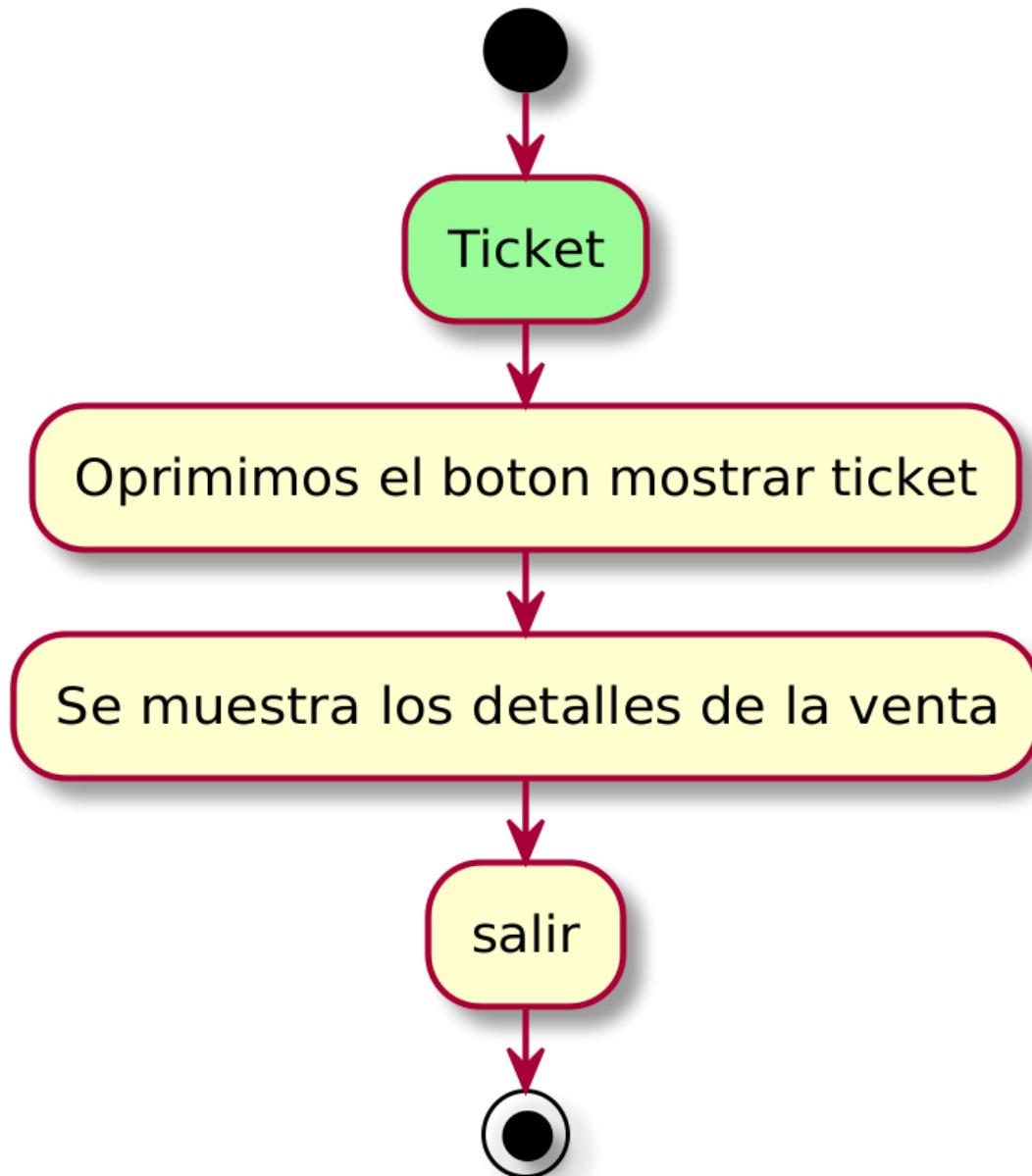


Figure 2.8: Ticket

Chapter 3

Diagrama de paquetes

Base de Datos

```
@startuml

class bd.conexion #sky{
    -host:String
    -bd:String
    -usr:SString
    -pasw:String

    +ConnectioFlujo de Datosn conectar_base()
}

@enduml
```

Entity

```
@startuml

class entity.Usuario #green{
    -nombre_usuario:String
    -contraseña:String
    -tipo_usuario
    +getnombre_usuario()
    +getcontraseña()
    +gett tipo_usuario()

}
class entity.Ventas #green{

    -fecha:String
    -producto:Producto
    -cantidad:int
    +getfecha()
    +getproducto()
    +getcantidad()

}
```

```
}

class entity.Producto #green{
    codigo:String
    -nombre:String
    -existencia:int
    -descripcion:String
    -precioc: double
    -preciov: double

    +getnombre()
    +getcodigo()
    +getexistencia()
    +getdescripcion()
    +getprecioc()
    +getpreciov
}

@enduml

Model

@startuml

interface model.IUsuarioModel #skyblue{
    -usuario: Usuario
    +List<Usuario> obtenerUsuarios()
    +Usuario obtenerUsuario()
    +void actualizarUsuario(Usuario)
    +void eliminarUsuario(int id)
    +void crearUsuario(Usuario)
}

interface model.IUsuarioModel<|--interface model.UsuarioModelImplements

interface model.IVentasModel #skyblue{

    +void generarventa()
}

interface model.IProductoModel #skyblue{
    -codigo:String
    -nombre:String
    -existencia:int
    -descripcion:String
    -precioc: double
    -preciov: double

    +void agregarProducto()
    +void modificarProducto()
    + void eliminarProducto()
}
```

```
}
```

```
interface model.UsuarioModelImplements #green{
+List<Usuario> obtenerUsuarios()
+Usuario obtenerUsuario()
+void actualizarUsuario(Usuario)
+void eliminarUsuario(int id)
+void crearUsuario(Usuario)
}

interface model.VentasModelImplements #green{
+void generarventa()
}

interface model.ProductosModelImplements #green{
+void agregarProducto()
+void modificarProducto()
+ void eliminarProducto()
}
@enduml
```

Controller

```
@startuml
```

```
class controller.Usuariocontroller #yellow{
}

class controller.Ventascontroller #yellow{
}

class controller.Productoscontroller #yellow{
}

class controller.ReporteProductoscontroller #yellow{
}

class controller.ReporteVentascontroller #yellow{
}

class view.Principal #pink {
}

class view.Login #pink {
-Usuario:string
-Contraseña:string

+verificar_usuario()
}

class view.Punto_venta #pink {
```

```
-codigo:String  
+void busca_codigo()  
+void muestra_datos()  
}  
@enduml
```

View

```
@startuml  
class view.Insertar_Producto #pink {  
-codigo:String  
-nombre:String  
-existencia:int  
-descripcion:String  
-precioc: double  
-preciov: double  
  
+Limpiarcampos()  
}  
  
class view.Consultar #pink {  
-codigo:String  
  
}  
class view.Modificar #pink {  
-codigo:String  
}  
  
@enduml
```

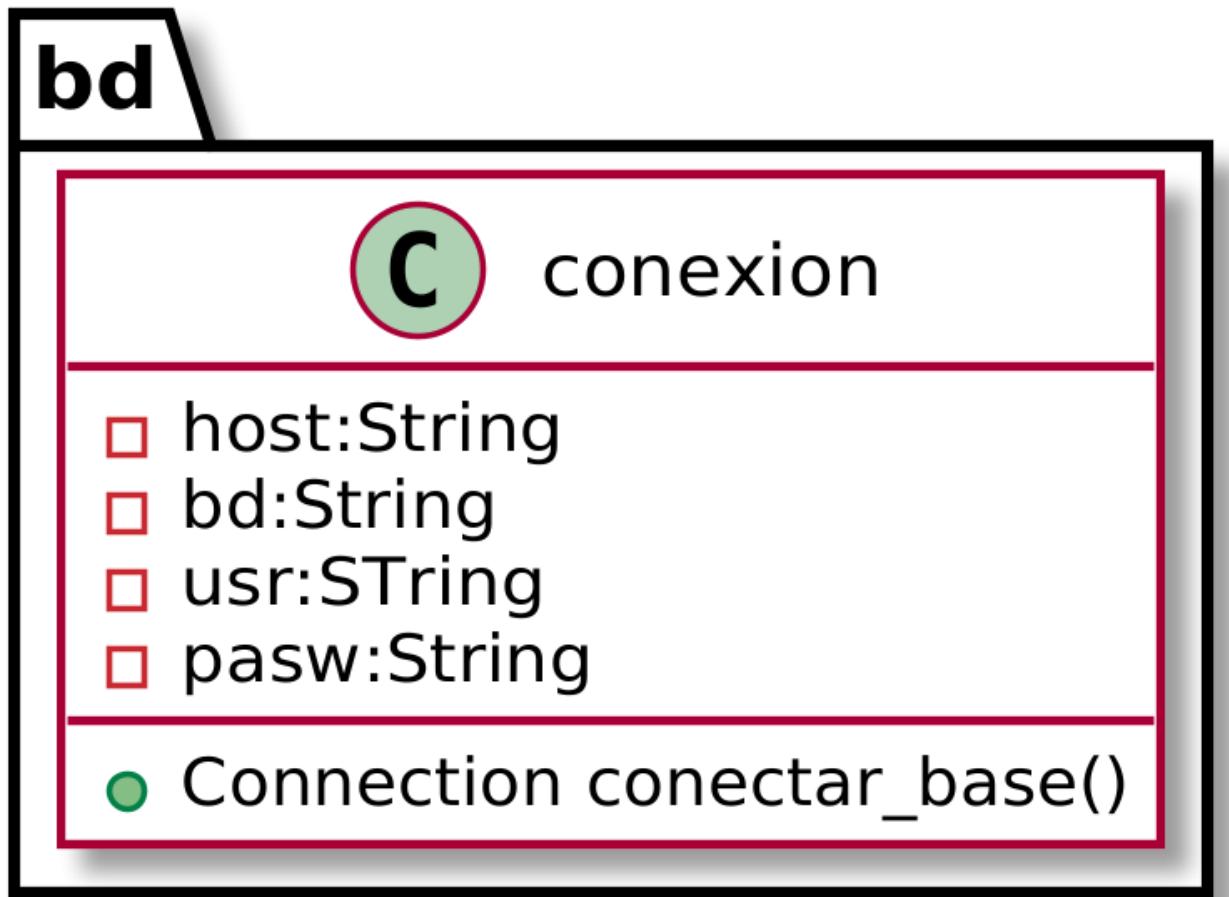


Figure 3.1: Base de Datos



Figure 3.2: Entity

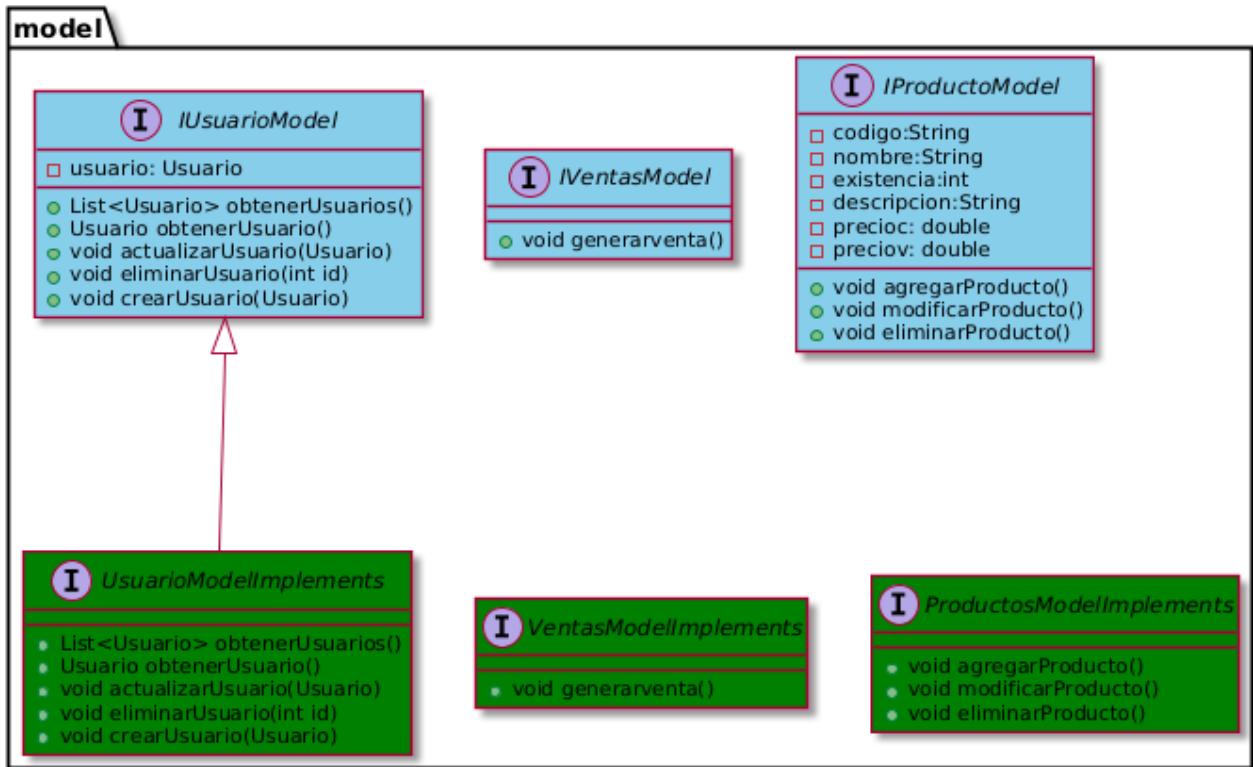


Figure 3.3: Model

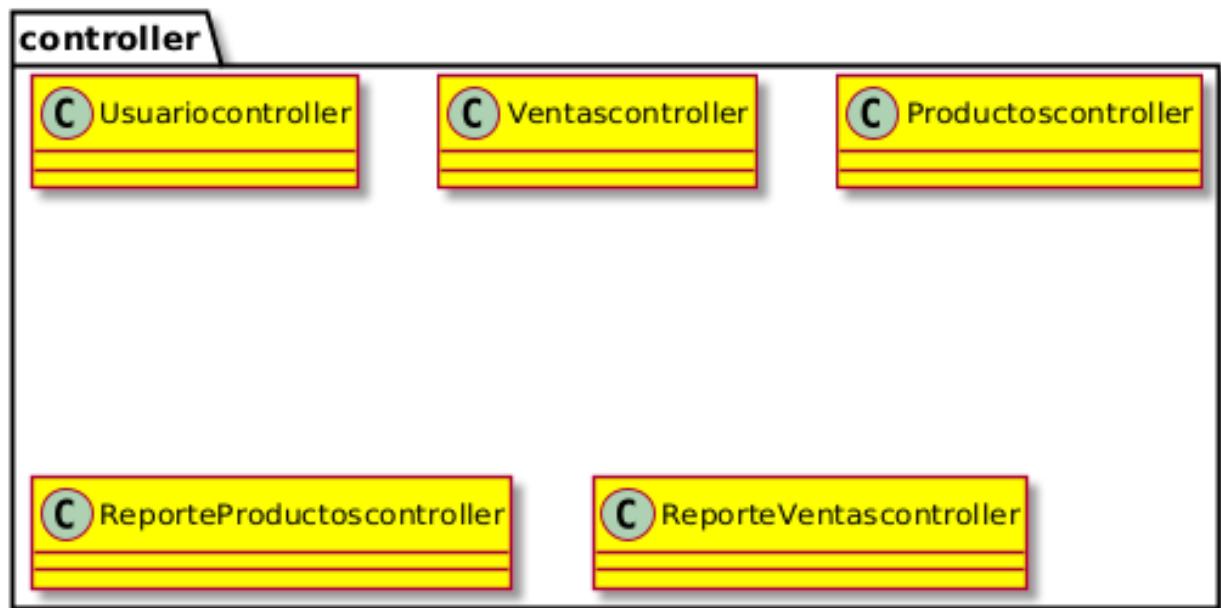


Figure 3.4: Controller

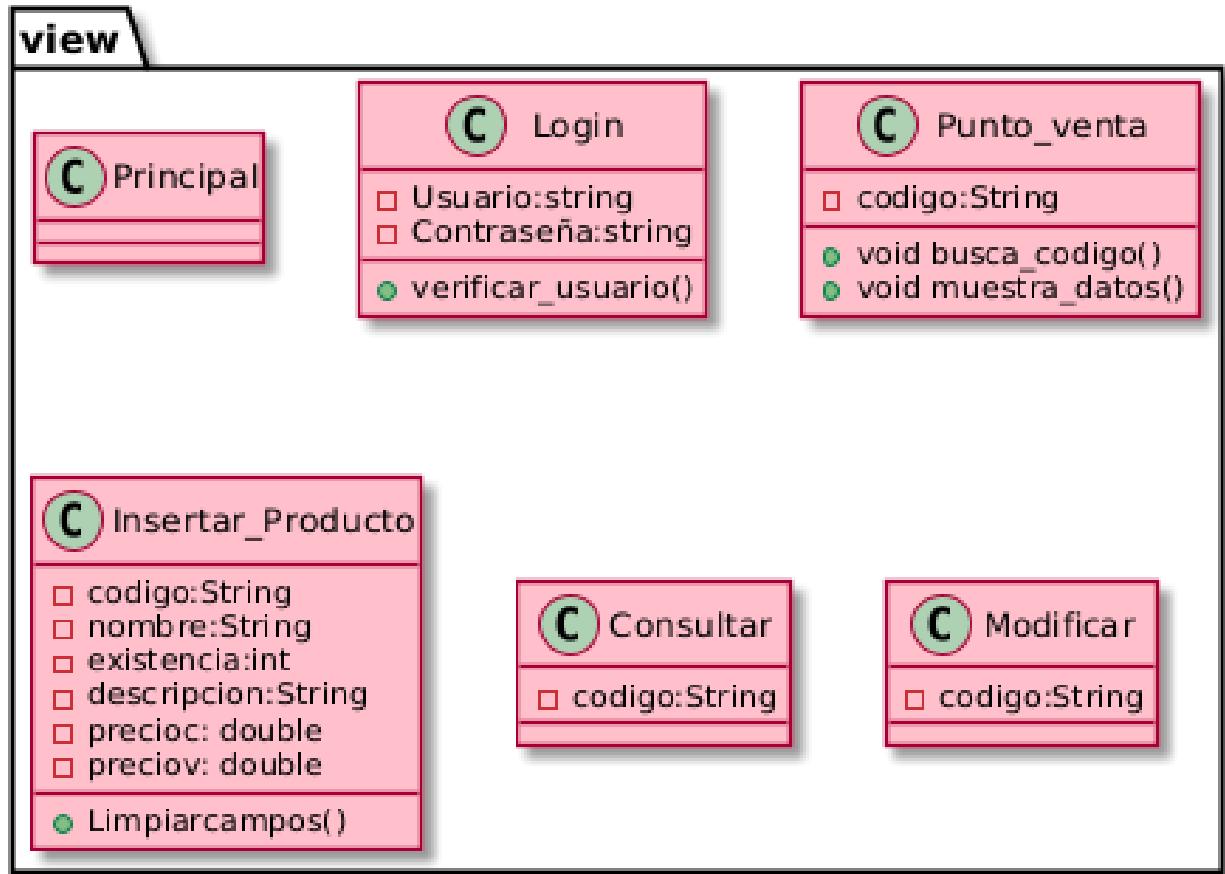


Figure 3.5: View

Chapter 4

Diagrama de Flujo de datos

Nivel 0

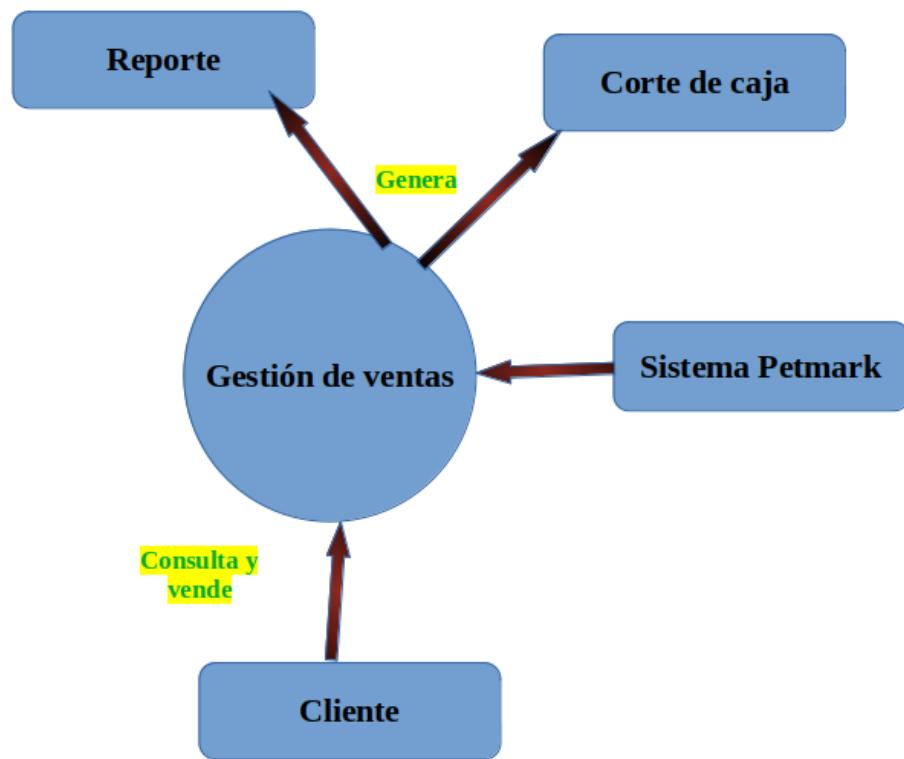


Figure 4.1: Nivel 0

Nivel 1

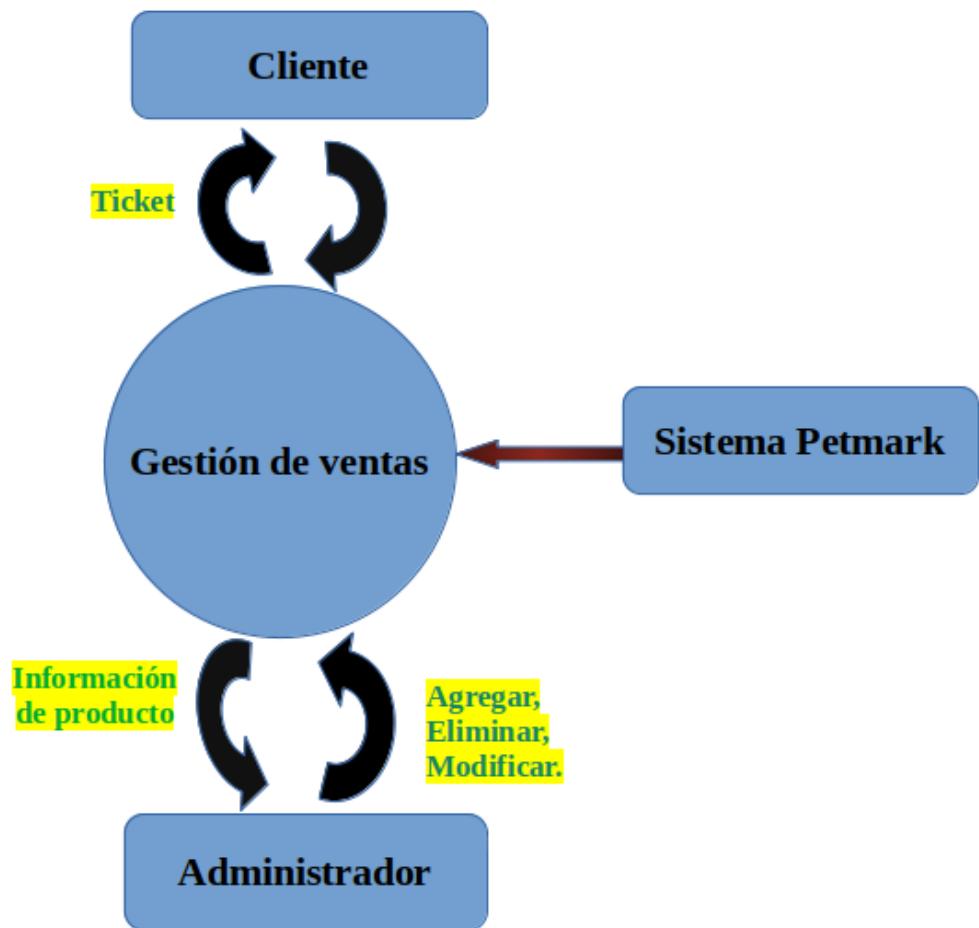


Figure 4.2: Nivel 1

Chapter 5

Diagrama de Componentes

```
@startuml

package "Operaciones" {
    Selección - [Aregar producto]
    Selección- [Aregar usuario]
    Selección - [Eliminar usuario]
    Selección - [consultar producto]
    Selección - [Modificar producto]
    Selección - [Producto]
    Operaciones - Principal
}

package "Lógica" {
    Operación - [Generar reporte]
    Operación - [Generar venta]
    Operación - [Reporte de ventas]
}

node "Logueo" #sky{
    TIPO - [Administrador]
    TIPO - [Usuario]
    TIPO - [User]
    Logueo - Principal
}

node "Principal" {
    [Panel Principal] - TIPO
    [Panel Principal] - Lógica
    [Panel Principal] - Selección
    [Panel Principal] - MySql
}

database "MySql" {
```

```

folder "Producto" {
    [Operaciones]
}
frame "User" {
    [Administrar]
}
    MySql - Principal
}

@enduml

```

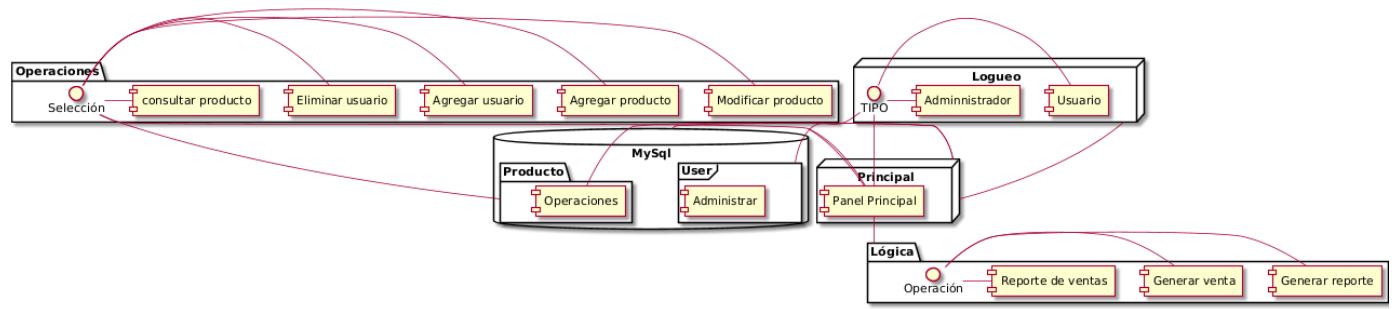


Figure 5.1: Diagrama de componentes

Chapter 6

Diagrama de Secuencia

Login

```
@startuml
```

actor Autor
entity login
control control
database BD
boundary principal
Autor-->login : ingresa usuario
BD-->control: obtener usuario
login-->control :verifica usuario
control-->Autor : usuario incorrecto
Autor-->login : ingresa contraseña
BD-->control : obtener contraseña
login-->control :verifica contraseña
control-->Autor : contraseña incorrecta
control-->BD :reporte de ventas
control--> principal :usuario y cointraseña correctos

```
@enduml
```

Insertar

```
@startuml
```

actor Autor
entity administrar
control control
database BD
Autor --> administrar :Agrega productos
administrar-->BD : guarda datos
Autor--> administrar: modificar datos
administrar-->BD : guarda cambios
Autor-->BD : eliminar datos
administrar-->BD :actualiza informacion

```
BD--> Autor :consultar datos  
@enduml  
Ventas  
@startuml  
  
actor Autor  
entity ventas  
database BD  
control interfaz_cobro  
Autor --> ventas :Agrega codigo  
ventas-->BD : conexion  
ventas<--BD :Resultado de busqueda  
Autor-->ventas: agregar producto  
Autor<--ventas: producto cancelado  
Autor-->BD :guardar venta  
Autor--> interfaz_cobro: cobrar producto  
  
BD--> Autor :consultar inventario  
@enduml
```

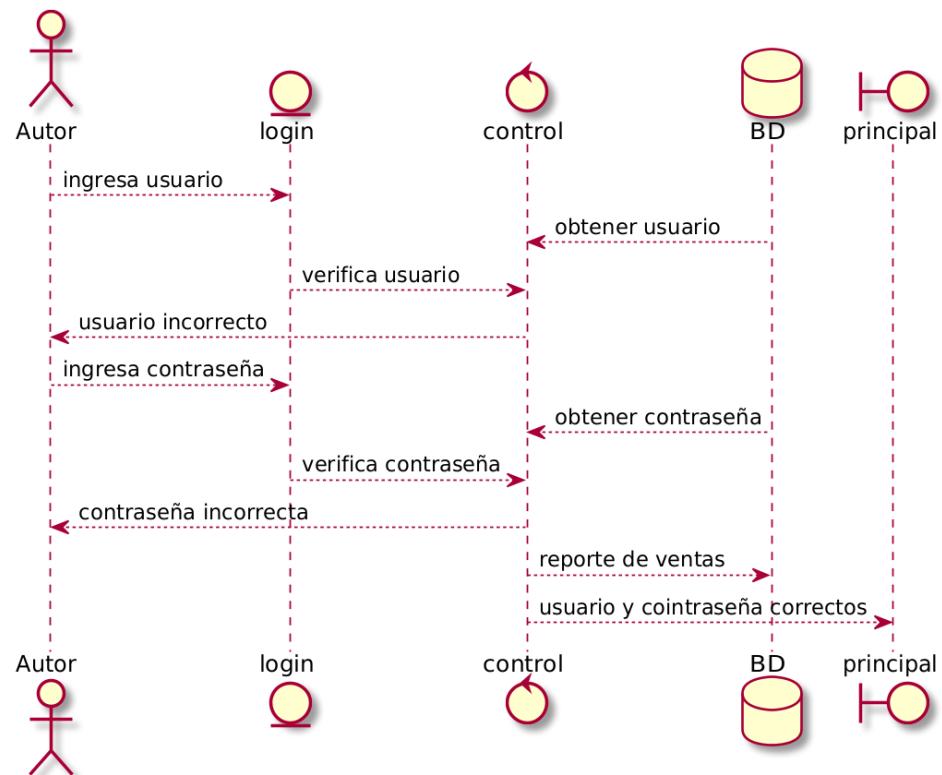


Figure 6.1: logueo

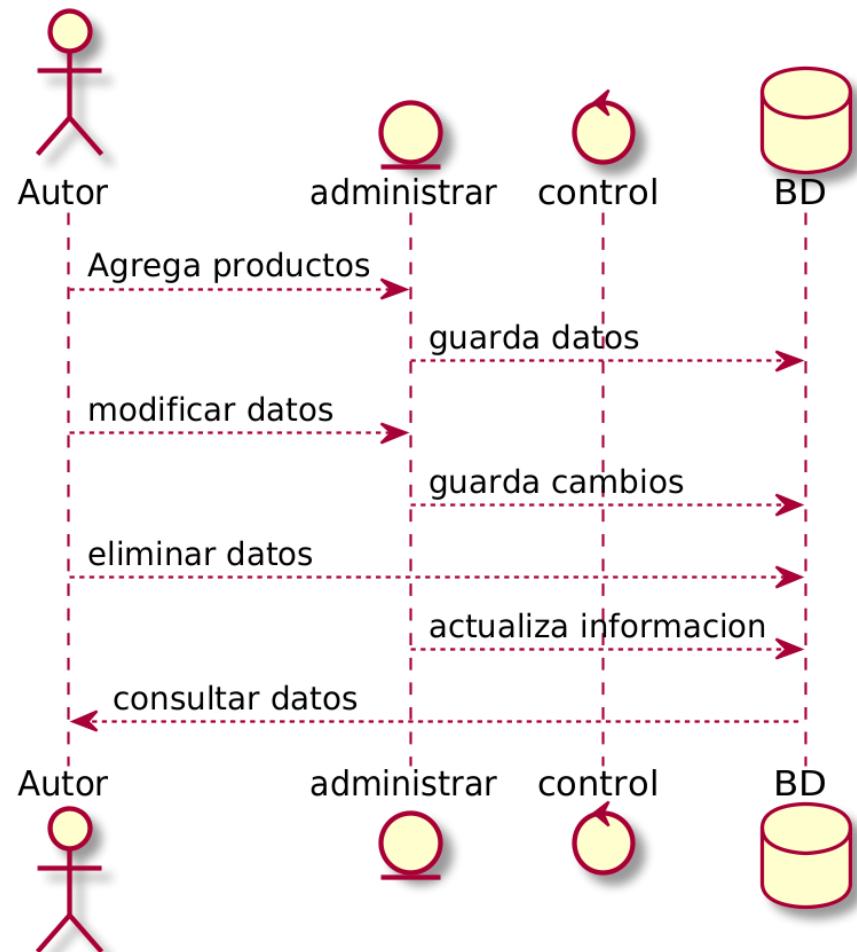


Figure 6.2: Insertar

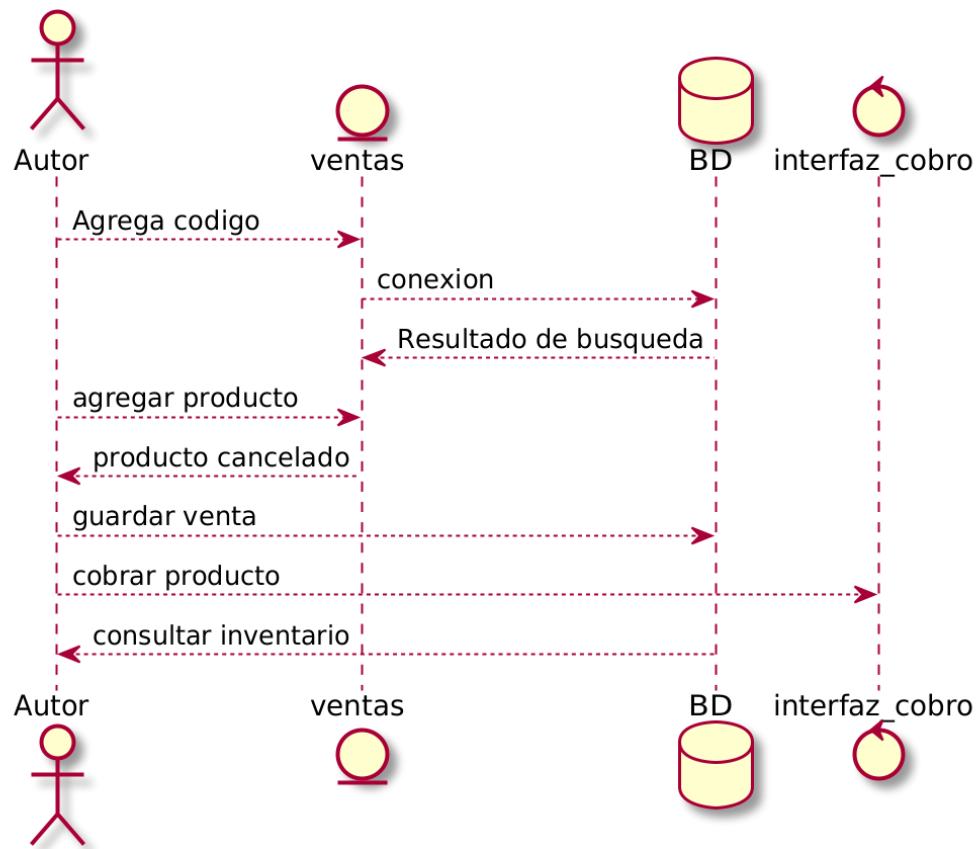


Figure 6.3: Ventas

Chapter 7

Diagrama de Despliegue

Despliegue

@startuml

node Logueo
node Usuario
node Administrador
node Principal_usuario
node Principal_Administrador

node Agregar_Producto
node Agregar_Usuario
node Modificar_producto
node Eliminar_producto

node Venta
node Consulta
node Reporte_de_venta

node BD
node jdbc

Logueo -- Usuario
Logueo -- Administrador

Administrador -- Principal_Administrador

Principal_Administrador -- Agregar_Producto
Principal_Administrador -- Agregar_Usuario
Principal_Administrador -- Modificar_producto
Principal_Administrador -- Eliminar_producto

Usuario -- Principal_usuario
Principal_usuario -- Venta

```
Principal_usuario -- Consulta
Principal_usuario -- Reporte_de_venta
```

```
Venta -- jdbc
Consulta -- jdbc
Reporte_de_venta -- jdbc
```

```
Agregar_Producto -- jdbc
Agregar_Usuario -- jdbc
Modificar_producto -- jdbc
Eliminar_producto -- jdbc
```

```
jdbc -- BD
```

@enduml

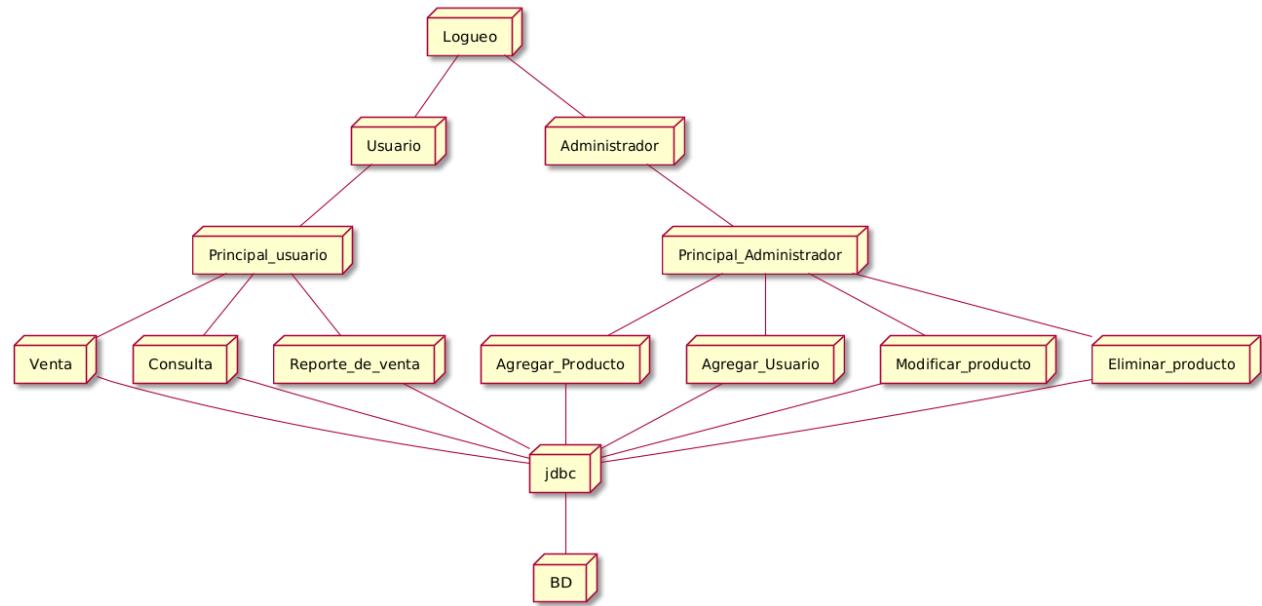


Figure 7.1: Despliegue

Chapter 8

Diagrama de Clase

```
@startuml

class conexion #sky{
    -host:String
    -bd:String
    -usr:SString
    -pasw:String

    +Connection conectar_base()
}

class Usuario #green{
    -nombre_usuario:String
    -contraseña:String
    -tipo_usuario
    +getnombre_usuario()
    +getcontraseña()
    +gettipo_usuario()
}
class Ventas #green{

    -fecha:String
    -producto:Producto
    -cantidad:int
    +getfecha()
    +getproducto()
    +getcantidad()

}
class Producto #green{
    codigo:String
    -nombre:String
    -existencia:int
    -descripcion:String
    -precioc: double
```

```
-preciov: double

+getnombre()
+getcodigo()
+getexistencia()
+getdescripcion()
+getprecioc()
+getreciov

}

interface IUsuarioModel #skyblue{
-usuario: Usuario
+List<Usuario> obtenerUsuarios()
+Usuario obtenerUsuario()
+void actualizarUsuario(Usuario)
+void eliminarUsuario(int id)
+void crearUsuario(Usuario)

}

interface IUsuarioModel<|--interface UsuarioModelImplements

interface IVentasModel #skyblue{

+void generarventa()

}

interface IProductoModel #skyblue{
-codigo:String
-nombre:String
-existencia:int
-descripcion:String
-precioc: double
-reciov: double

+void agregarProducto()
+void modificarProducto()
+ void eliminarProducto()

}

interface UsuarioModelImplements #green{
+List<Usuario> obtenerUsuarios()
+Usuario obtenerUsuario()
+void actualizarUsuario(Usuario)
```

```
+void eliminarUsuario(int id)
+void crearUsuario(Usuario)
}

interface VentasModelImplements #green{
+void generarventa()
}
interface ProductosModelImplements #green{
+void agregarProducto()
+void modificarProducto()
+ void eliminarProducto()
}

class Usuariocontroller #yellow{
}
class Ventascontroller #yellow{
}
class Productoscontroller #yellow{
}
class ReporteProductoscontroller #yellow{
}
class ReporteVentascontroller #yellow{
}

class Principal #pink {
}

class Login #pink {
-Usuario:string
-Contraseña:string

+verificar_usuario()
}
class Punto_venta #pink {
-codigo:String
+void busca_codigo()
+void muestra_datos()
}
class Insertar_Producto #pink {
-codigo:String
-nombre:String
-existencia:int
```

```

-direccion:String
-precioC: double
-precioV: double

+LimpiarCampos()
}

class Consultar #pink {
-codigo:String

}
class Modificar #pink {
-codigo:String
}

@enduml

```

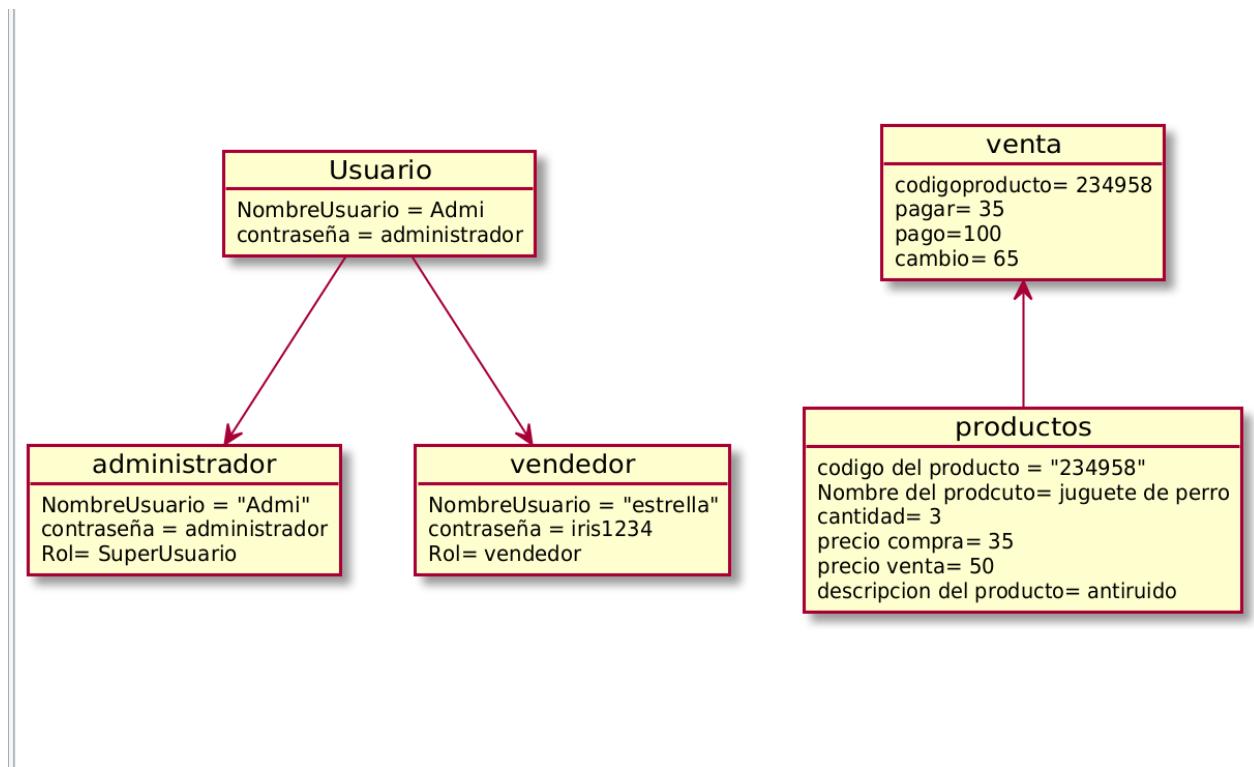


Figure 8.1: Objeto

Chapter 9

Diagrama de Objeto

```
@startuml
object administrador
object vendedor
object venta
object Usuario

venta<--productos
Usuario--> administrador
Usuario--> vendedor

object Usuario {
    NombreUsuario = Admi
    contraseña = administrador
}
object administrador {
    NombreUsuario = "Admi"
    contraseña = administrador
    Rol= SuperUsuario
}
object vendedor {
    NombreUsuario = "estrella"
    contraseña = iris1234
    Rol= vendedor
}

object productos {
    codigo del producto = "234958"
    Nombre del producto= juguete de perro
    cantidad= 3
    precio compra= 35
    precio venta= 50
    descripcion del producto= antiruido
}
object venta {
```

```

codigoproducto= 234958
pagar= 35
pago=100
cambio= 65
}
@enduml

```

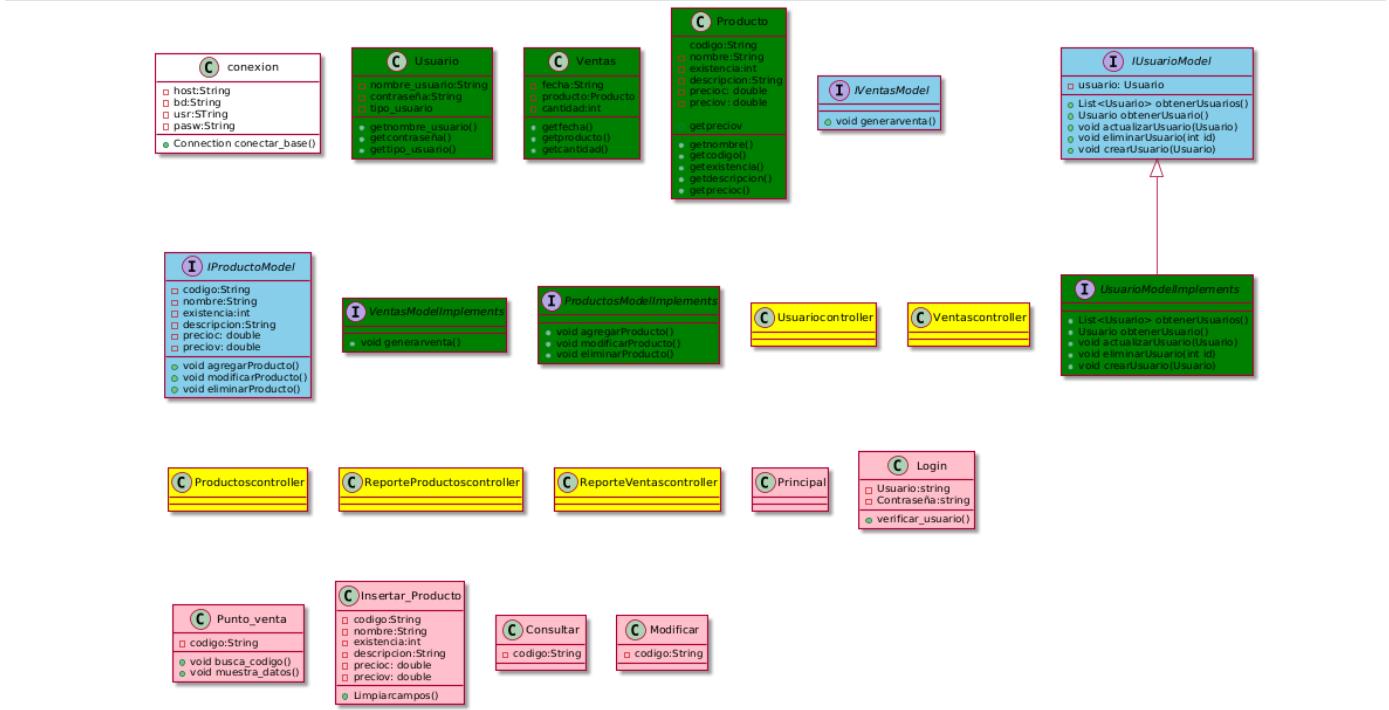


Figure 9.1: Clases