



Benemérita Universidad Autónoma de Puebla

Facultad de Ciencia de la Computación

Ingeniería en tecnologías de la
información

Ingeniería de Software II

Golden Mart
Sprint Final

Integrantes:

Javier Aguilar Macias



INDICE

PLANEACION	2
ROLES.....	2
Modelado.....	5
INTERFACES preliminares en BALSAMIQ.....	41
INTERFACES FINALES EN JAVA SWING.....	46
INFORMES JIRA	53
Pruebas.....	54

PLANEACION

Golden Mart es una aplicación para gestionar una tienda de abarrotes. Permite realizar ventas, registrar nuevos productos, mantener un control de inventario y generar tickets para cada venta realizada. Los usuarios pueden agregar productos al carrito de compras, ver el total de la venta, realizar el pago y generar un recibo de compra. Además, el sistema permite a los administradores gestionar el inventario y visualizar reportes de ventas.

ROLES

Rol	Miembro
Scrum Master	Javier Aguilar Macias
Product Owner	Javier Aguilar Macias
Equipo de desarrollo	Javier Aguilar Macias

--	--

HISTORIAS DE USUARIO DESARROLLADAS

Prioridad	#HU	Historia de Usuario	Criterios de Aceptación	Valor
Media	#1	Como: Administrador Quiero: Iniciar Sesión Para: Gestionar el sistema	La interfaz de inicio de sesión del administrador valida las credenciales ingresadas. Si son correctas, se permite el acceso a la gestión; de lo contrario, se muestra un mensaje de error.	5
Media	#2	Como: Administrador Quiero: Registrar Producto Para: Mantener actualizado el catálogo de productos	El sistema registra productos con datos obligatorios y muestra un mensaje de confirmación tras el registro exitoso.	8
Alta	#3	Como: Administrador Quiero: Gestionar inventario Para: Mantener un control detallado del stock y enviar falta de productos	El sistema muestra una lista de productos disponibles en el inventario. Se pueden realizar operaciones de actualización de stock, como agregar o restar unidades de un producto. Se deben registrar los cambios en el inventario correctamente y reflejarlos en la cantidad disponible de cada producto.	20
Alta	#4	Como: Cliente Quiero: Realizar venta Para: Completar mi compra	La interfaz de usuario muestra imágenes de los productos disponibles en el inventario. Se proporciona un campo de búsqueda que permite al usuario buscar productos por nombre o marca. El usuario puede seleccionar un producto de la lista de imágenes para agregarlo a su carrito de compras. El usuario puede ver una lista de productos seleccionados en su carrito de compras, con la opción de quitar productos individualmente si así lo desea. Se recalcula automáticamente el total de la venta y el precio final después de quitar productos del carrito. El usuario puede finalizar la compra una vez que esté satisfecho con su selección de productos.	20
Media	#5	Como: Cajero Quiero: Cancelar Venta Para: Corregir errores, ajustar mi selección de	El cliente no desea la compra de algún producto por lo que el cajero tendrá que cancelar la venta.	13

		productos o cancelar la compra si ya no deseo realizarla.	El cajero tuvo un error y desea empezar nuevamente el proceso.	
Media	#6	Como: Cajero Quiero: Generar el pago del cliente Para: Completar la transacción	El cajero elige el método de pago deseado y completa la transacción. Ingresa el efectivo o la tarjeta para confirmar pago.	13
Media	#7	Como: Cliente Quiero: Recibir mi Ticket Para: Tener confirmación de mi compra	Después de completar la compra, el sistema genera automáticamente un ticket de compra digital que incluye los detalles de la transacción, como la lista de productos comprados, el total de la venta y la fecha de la transacción. El ticket de compra digital se muestra al cliente en pantalla y se puede descargar o imprimir según sea necesario.	8
Alta	#8	Como: Administrador Quiero: Registrar la Venta en la Base de Datos Para: Mantener un registro preciso de todas las transacciones realizadas	Después de que el cajero haya completado una transacción de venta, el sistema debe capturar automáticamente la información relevante de la venta, incluyendo la fecha, hora, productos vendidos y el total de la venta. La información capturada debe ser enviada y almacenada de manera segura en la base de datos. La base de datos debe ser actualizada de manera adecuada para reflejar la venta realizada, incluyendo la reducción de la cantidad disponible de los productos vendidos en el inventario. Se deben implementar medidas de seguridad para garantizar que solo las transacciones exitosas sean registradas en la base de datos. El sistema debe proporcionar retroalimentación visual o un mensaje de confirmación al administrador una vez que la venta haya sido registrada con éxito en la base de datos.	13
Alta	#9	Como: Administrador Quiero: Registrar el Ticket en la Base de Datos Para: Mantener un registro preciso de todas las transacciones realizadas	Para registrar un ticket en la base de datos, el sistema debe recibir un ID de venta válido, verificar su existencia en la base de datos de ventas, generar un ID de ticket único, insertar un nuevo registro en la tabla de tickets que incluya el ID del ticket y el ID de la venta, y devolver una confirmación de éxito con el ID del ticket generado. Si el ID de venta no es válido, o si ocurre un error durante la inserción, el sistema debe manejar estos errores y proporcionar mensajes de error adecuados.	13

Alta	#10	Como: Administrador Quiero: Registrar el Detalle de la Venta en la Base de Datos Para: Mantener un registro preciso de todas las transacciones realizadas	Para registrar el detalle de una venta en la base de datos, el sistema debe recibir un ID de venta válido, un ID de producto y la cantidad del producto agregada, verificar la existencia de ambos IDs, validar que la cantidad sea un número entero positivo, insertar un nuevo registro en la tabla de detalles de ventas con los datos proporcionados y devolver una confirmación de éxito. En caso de IDs inválidos o cantidad no válida, el sistema debe emitir mensajes de error correspondientes, garantizando así un registro preciso del detalle de cada transacción.	13
Media	#11	Como: Administrador Quiero: Registrar Categoría Para: Mantener un catalogo actualizado de productos	El sistema registra categorías con datos obligatorios y muestra un mensaje de confirmación tras el registro exitoso.	8

PLANEACION DE LOS SPRINT

Sprint	1er	2do	3er	4to
Fecha	10/04/2024	17/04/2024	29/04/2024	08/05/2024
Historias	#1, #2	#3, #4	#5, #6	#7, #8, #9 #10,#11
Puntos Sprint	13	33	26	56

MODELADO

DIAGRAMA DE BASE DE DATOS

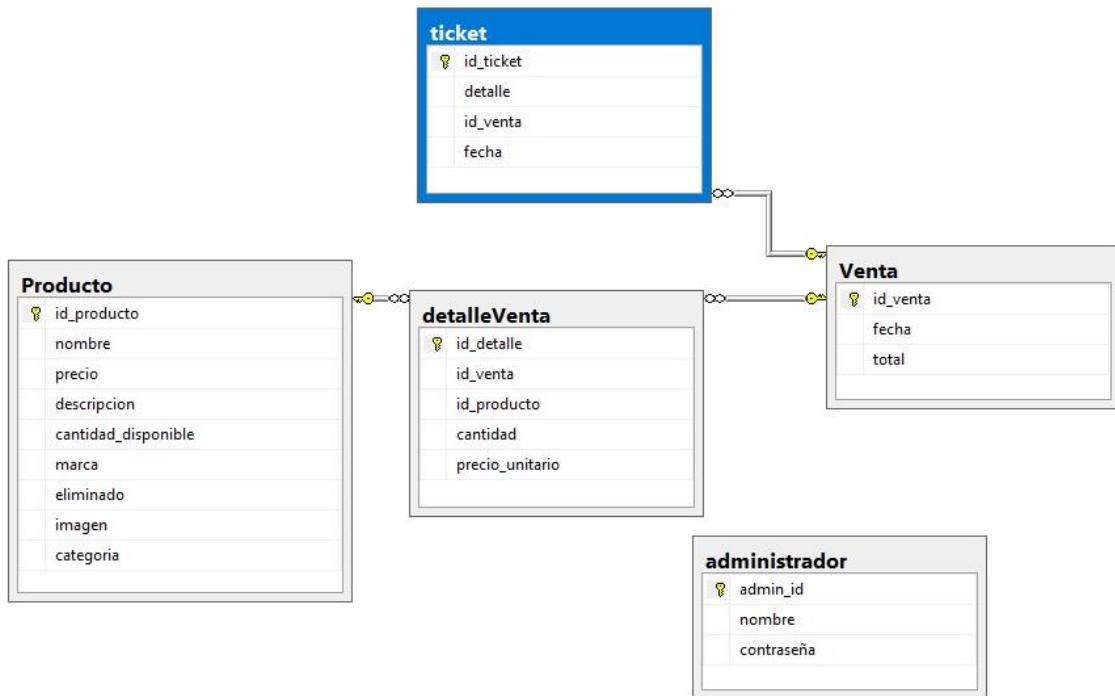


Diagrama de Base de Datos SIN Normalizar

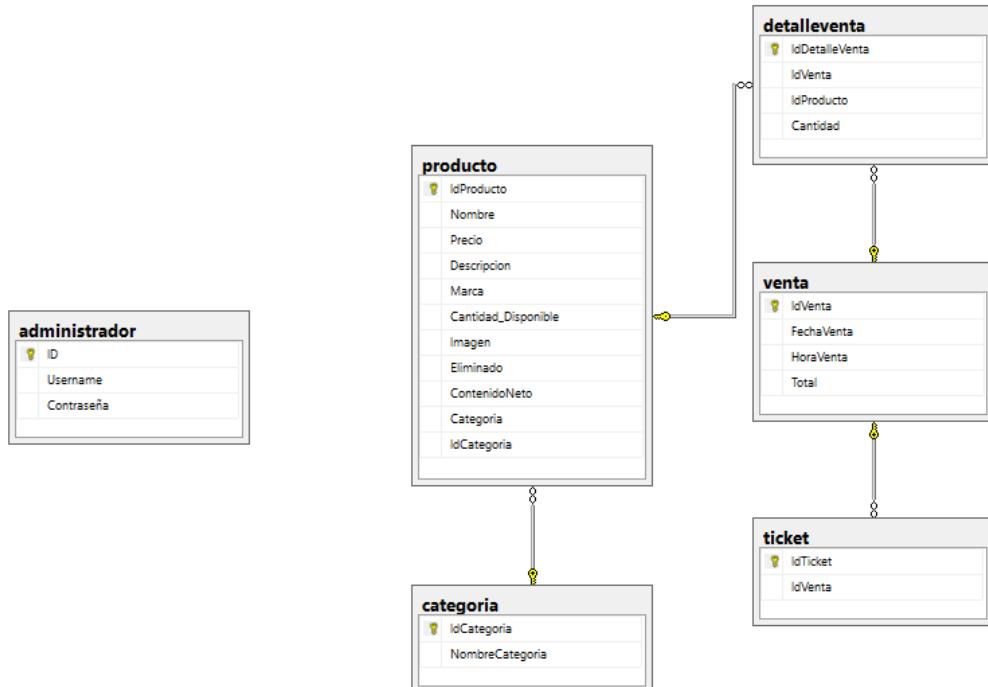


Diagrama de Base de Datos Normalizada

DIAGRAMA DE CLASES GENERAL

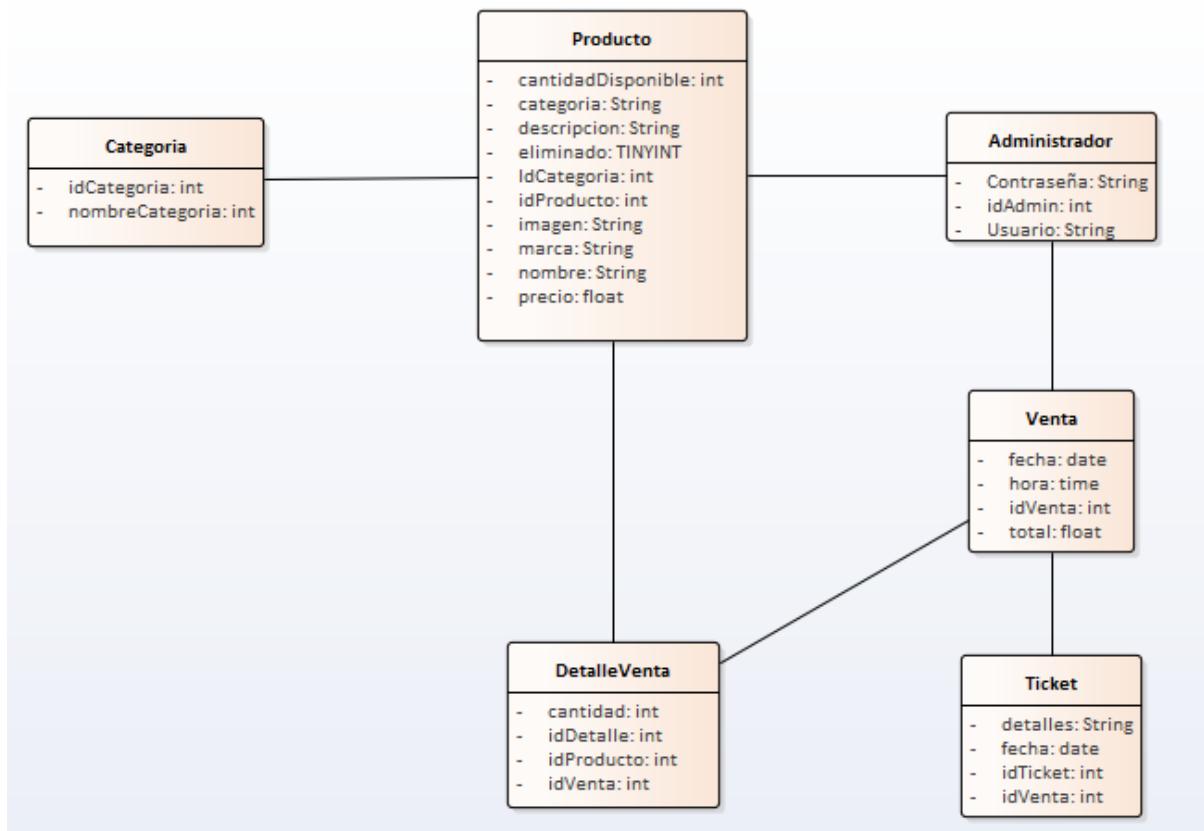


DIAGRAMA DE CLASES PARTICIPANTES #HU1

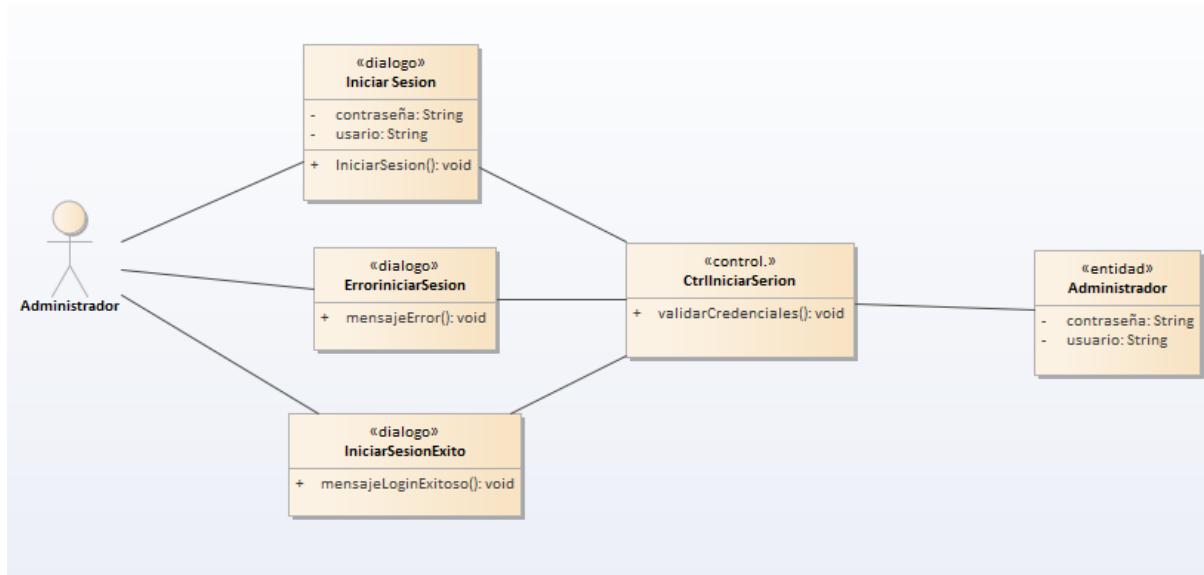


DIAGRAMA DE SECUENCIA DE ANALISIS #HU1

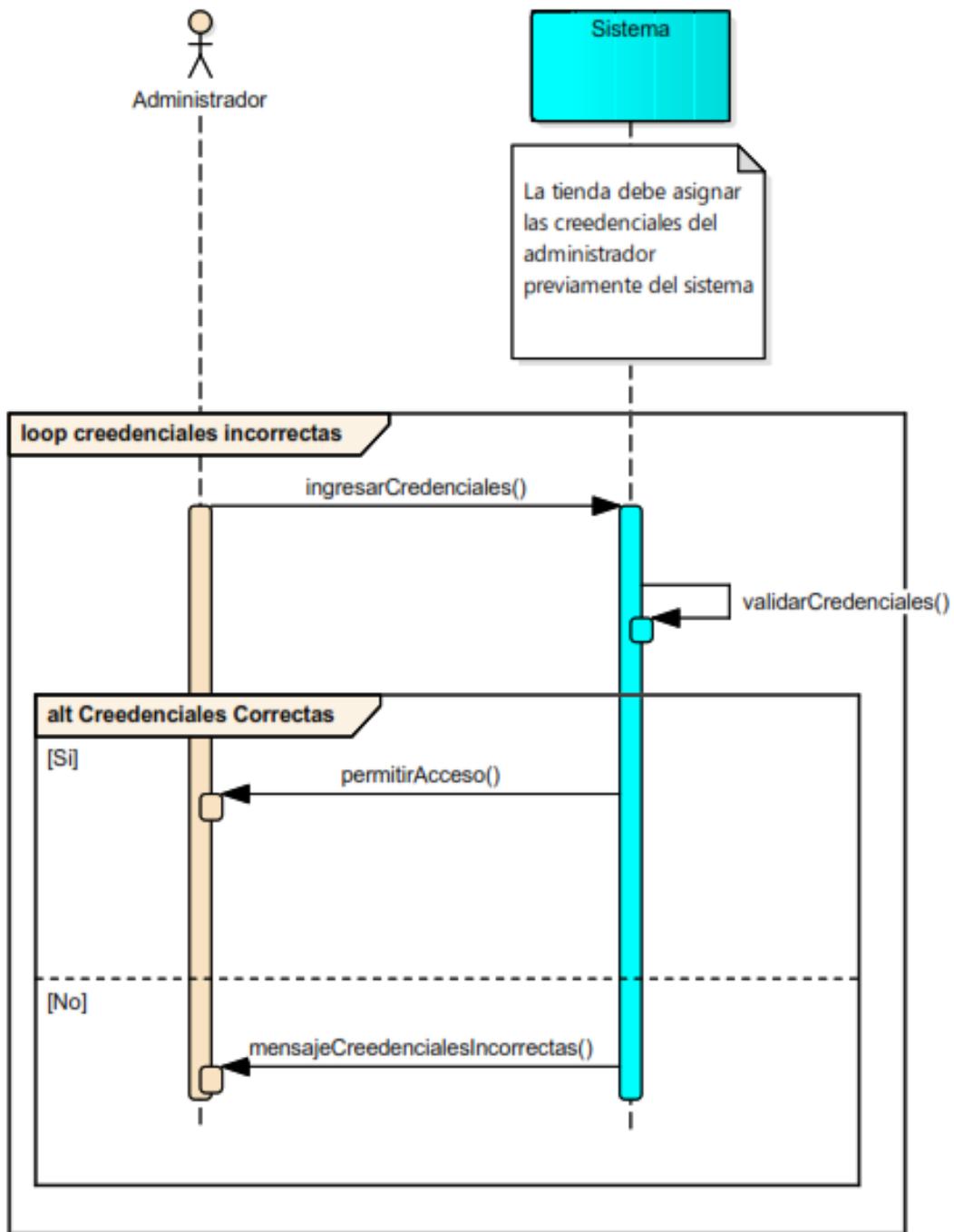


DIAGRAMA DE SECUENCIA DE DISEÑO #HU1

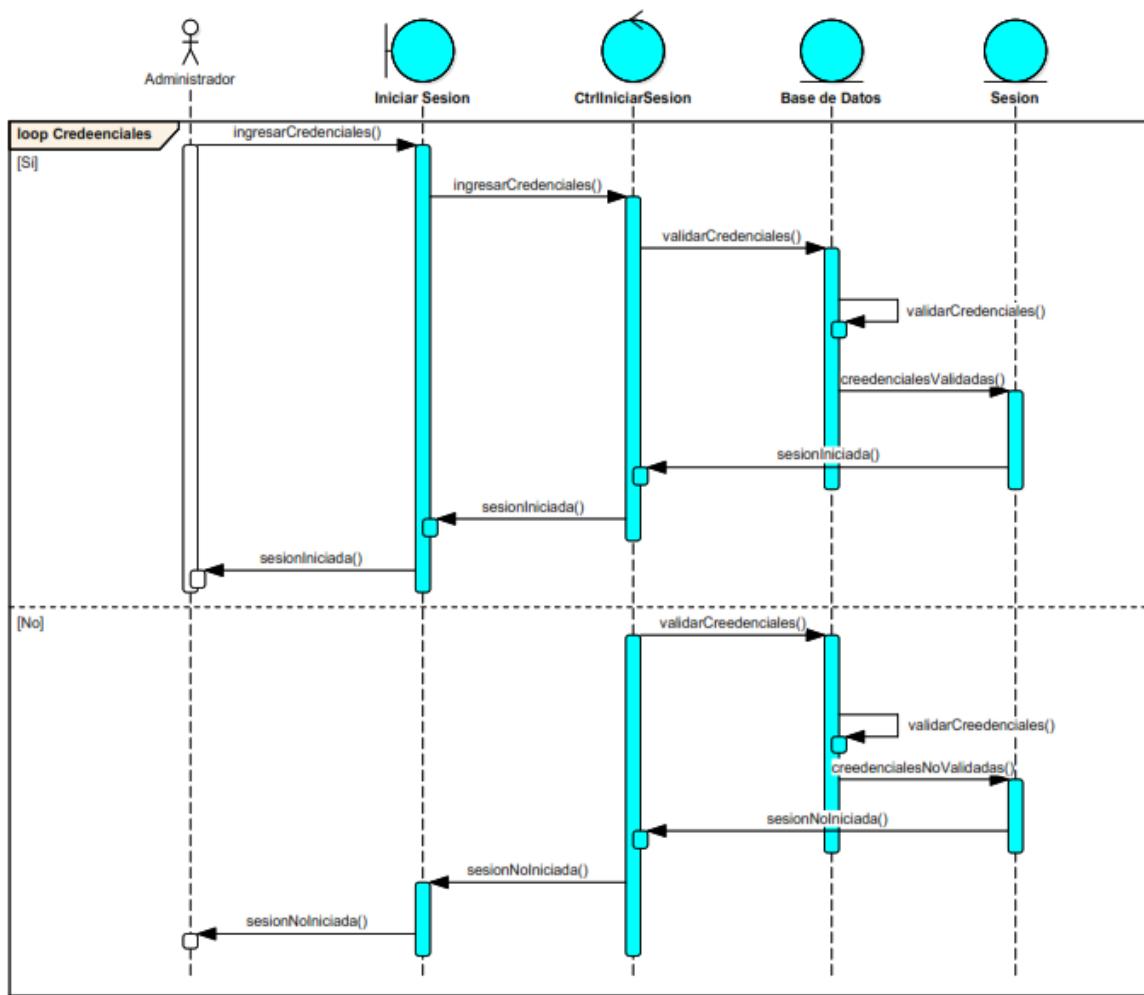


DIAGRAMA DE CLASES PARTICIPANTES #HU2

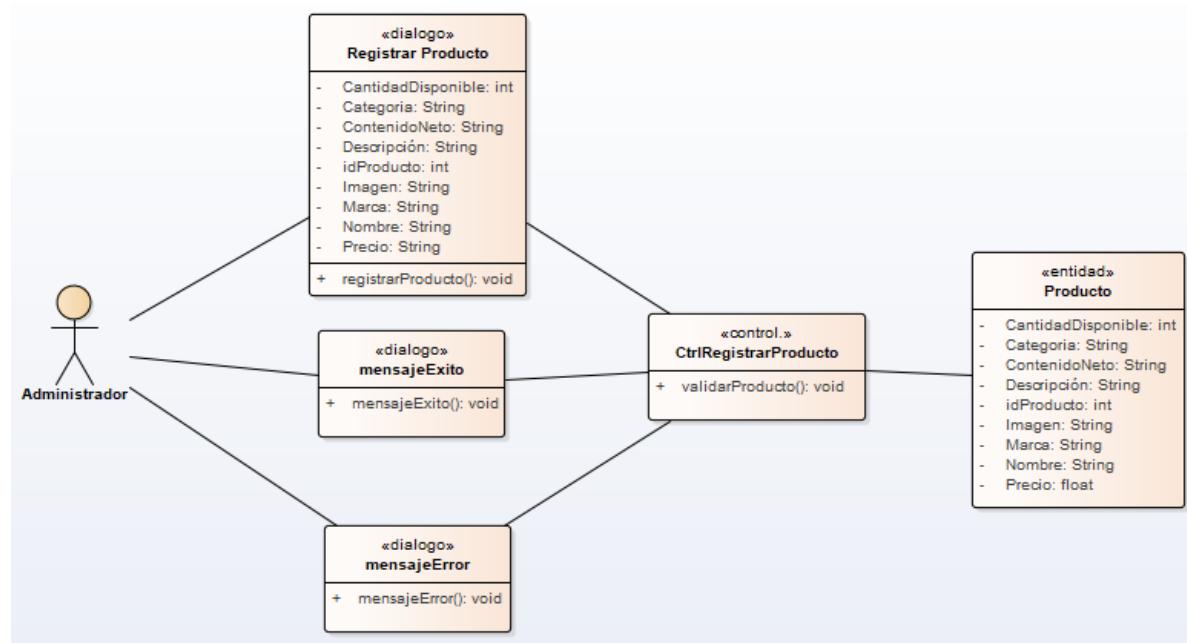


DIAGRAMA DE SECUENCIA DE ANALISIS #HU2

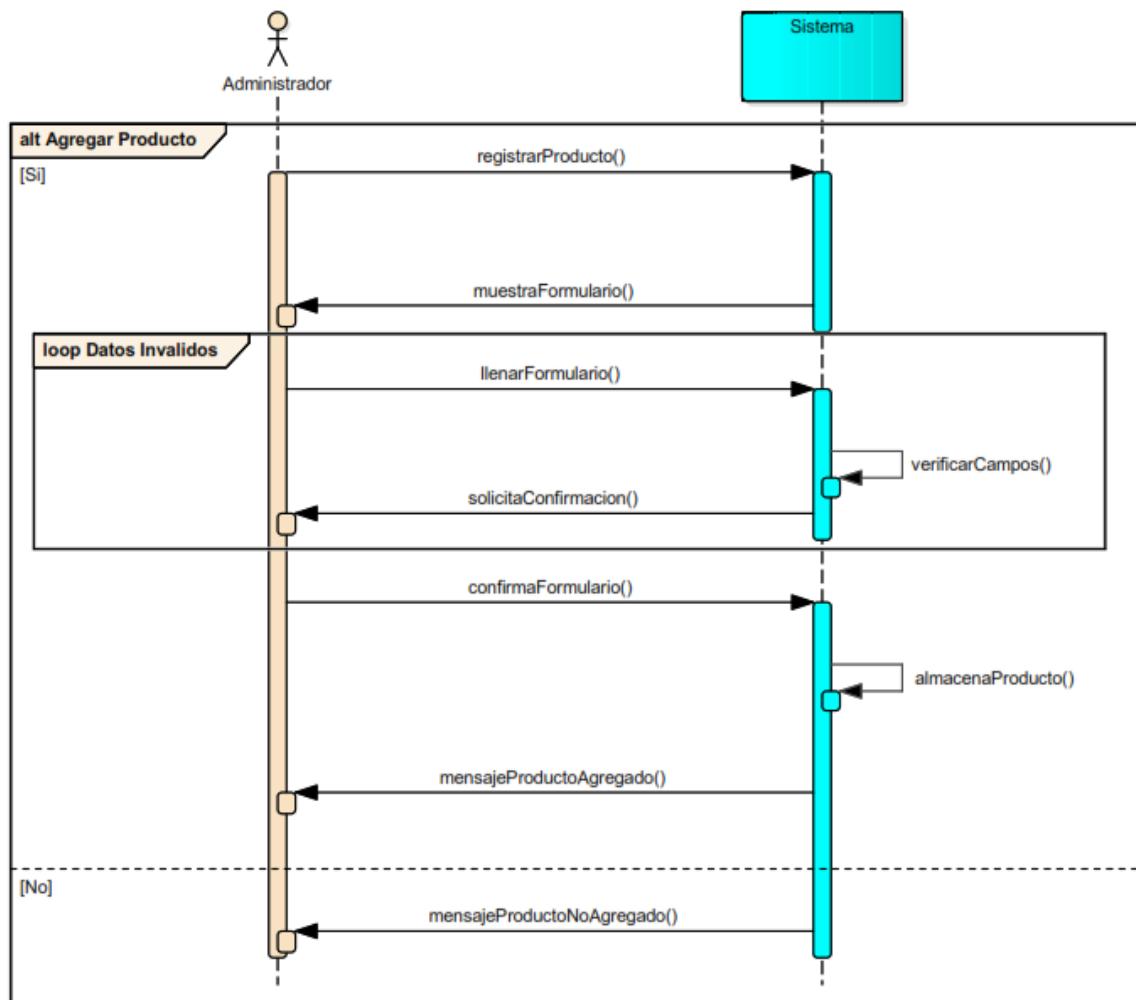


DIAGRAMA DE SECUENCIA DE DISEÑO #HU2

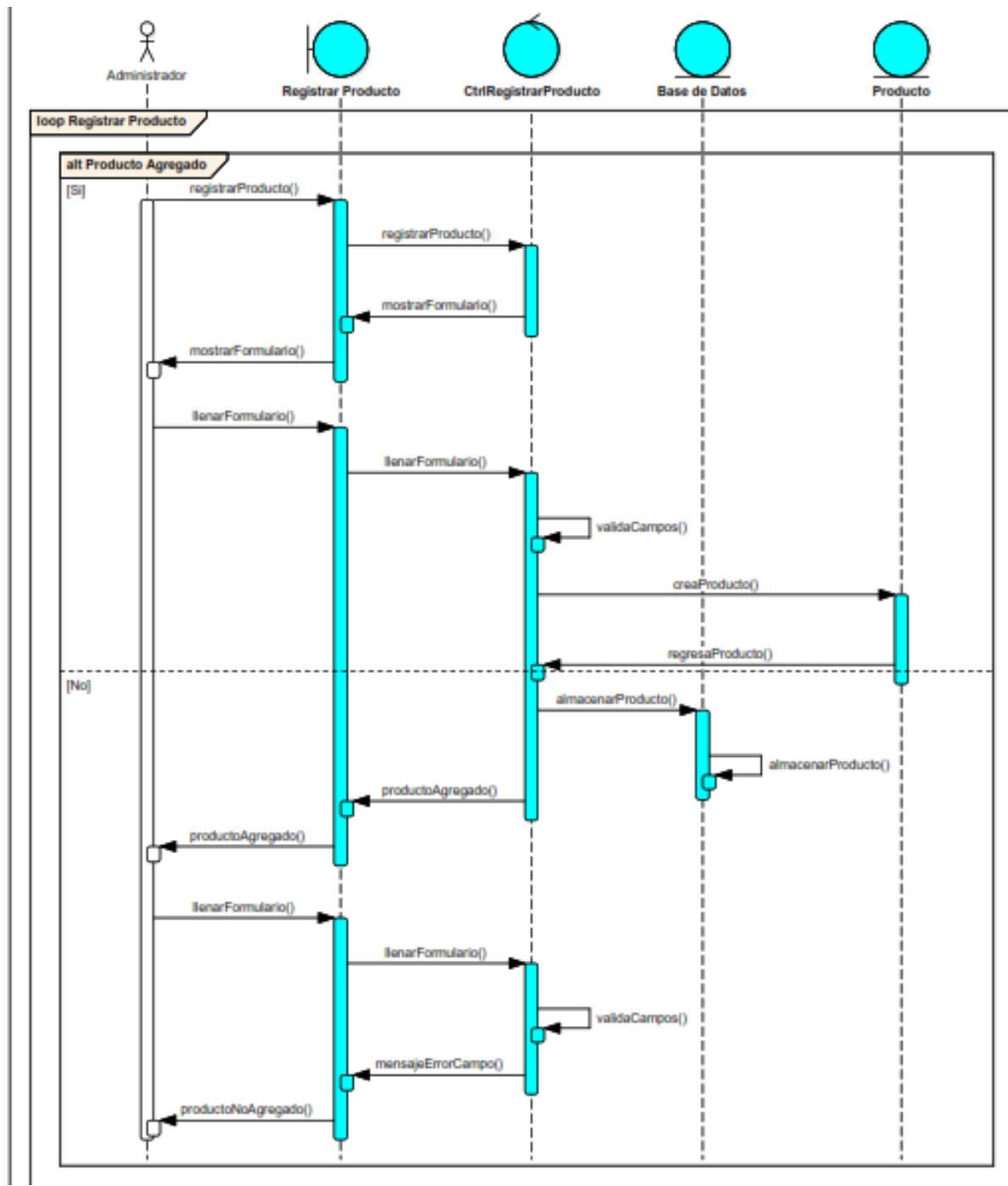


DIAGRAMA DE CLASES PARTICIPANTES #HU3

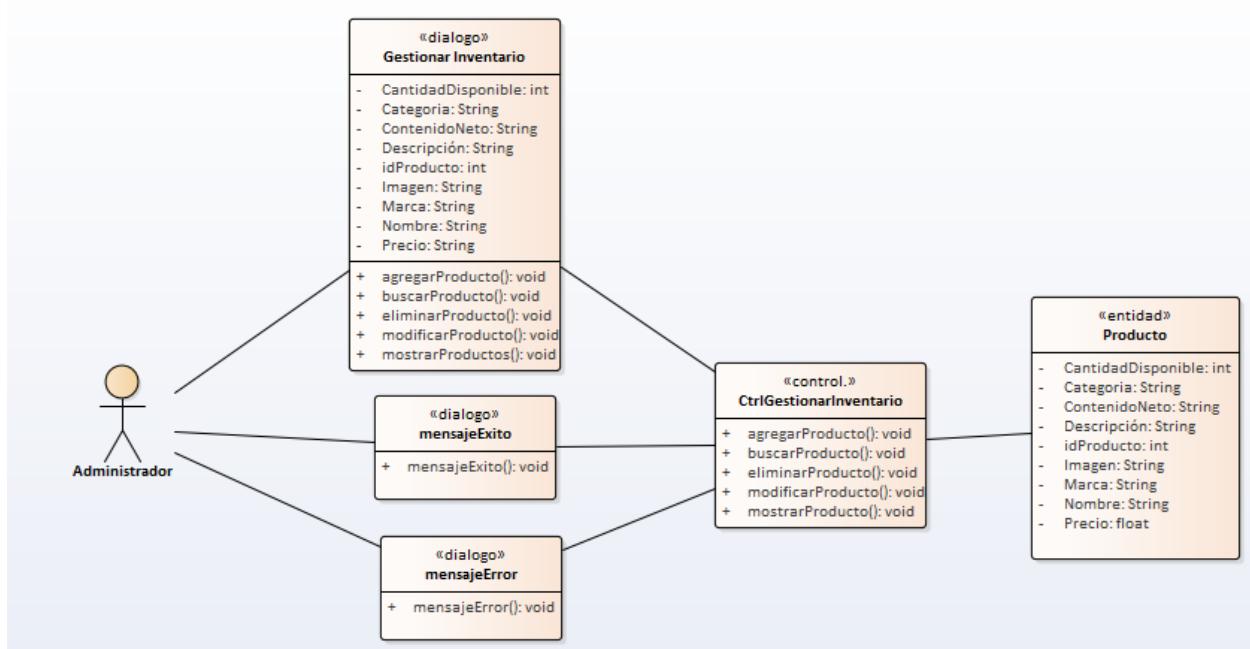


DIAGRAMA DE SECUENCIA DE ANALISIS #HU3.1 MODICAR PRODUCTO

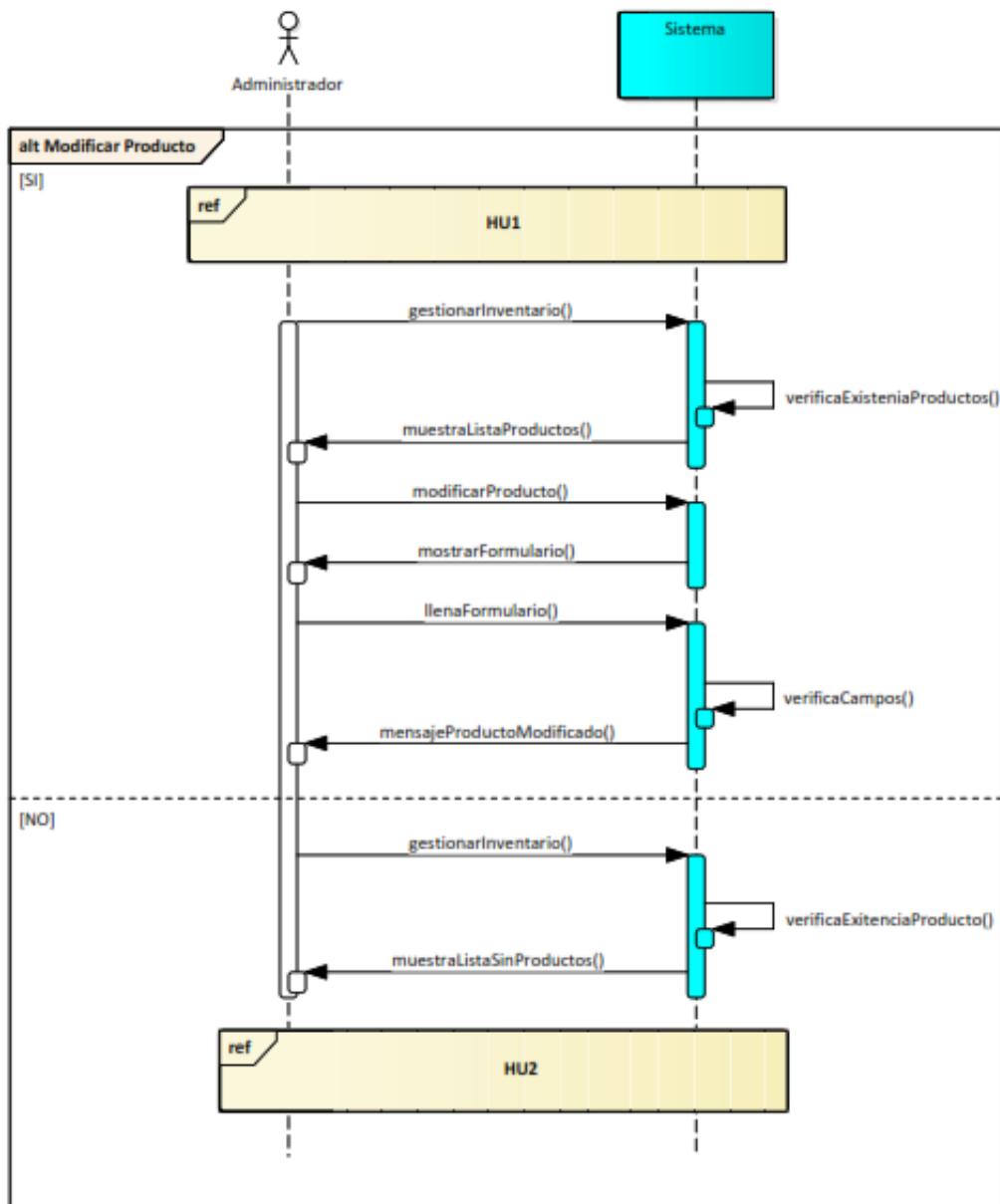


DIAGRAMA DE SECUENCIA DE DISEÑO #HU3.1 MODICAR PRODUCTO

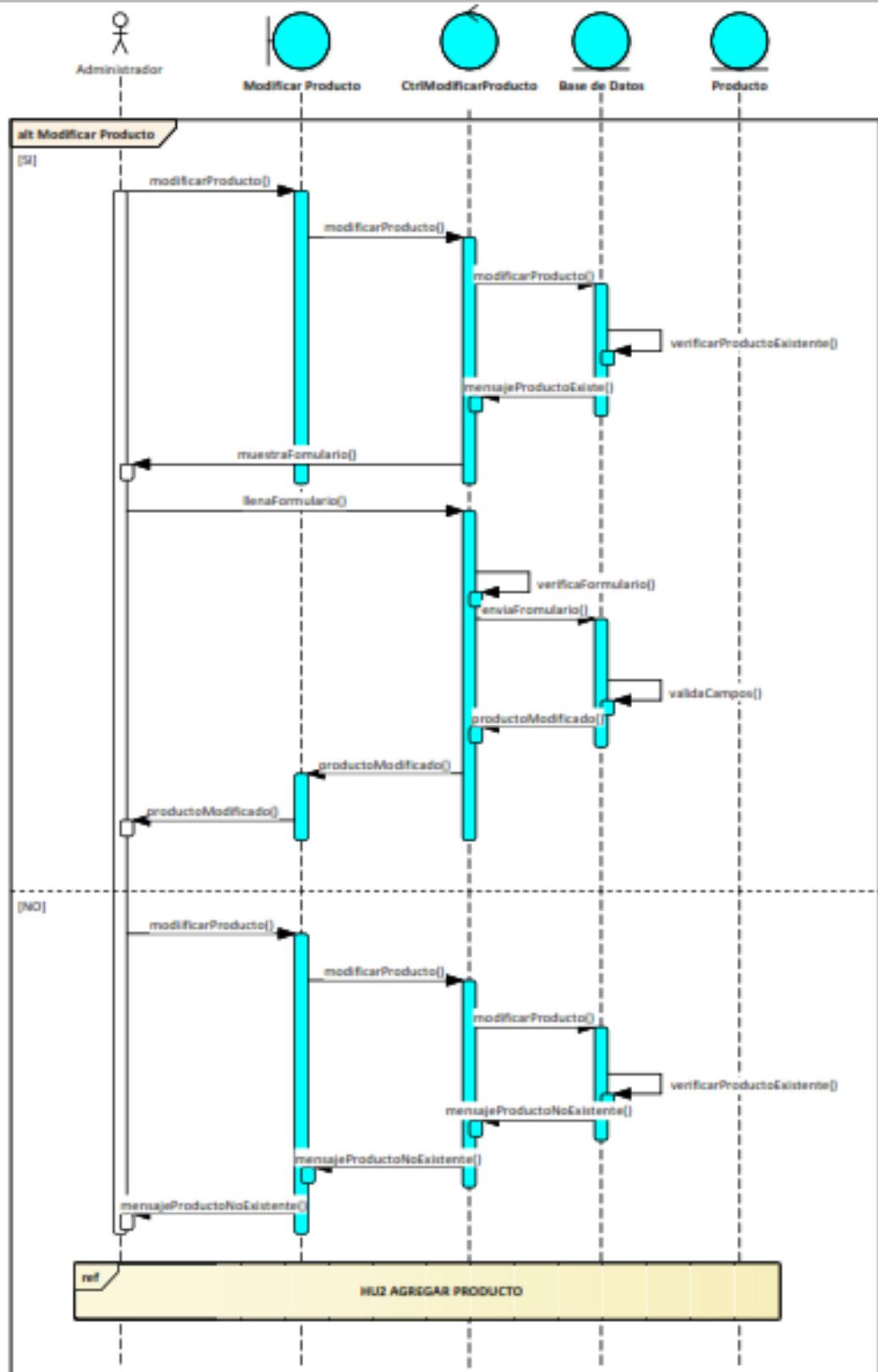


DIAGRAMA DE SECUENCIA DE ANALISIS #HU3.2 ELIMINAR PRODUCTO

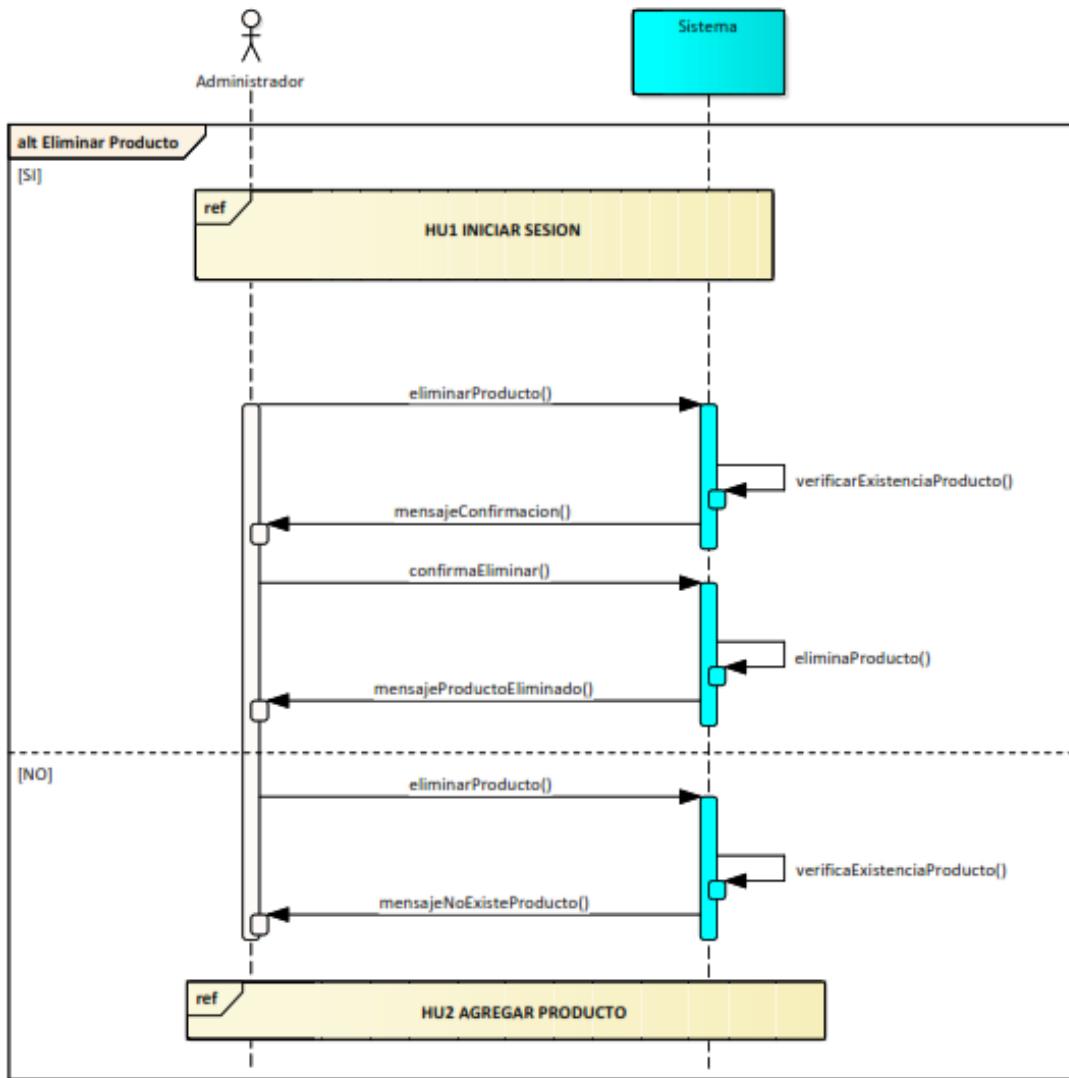


DIAGRAMA DE SECUENCIA DE DISEÑO #HU3.2 ELIMINAR PRODUCTO

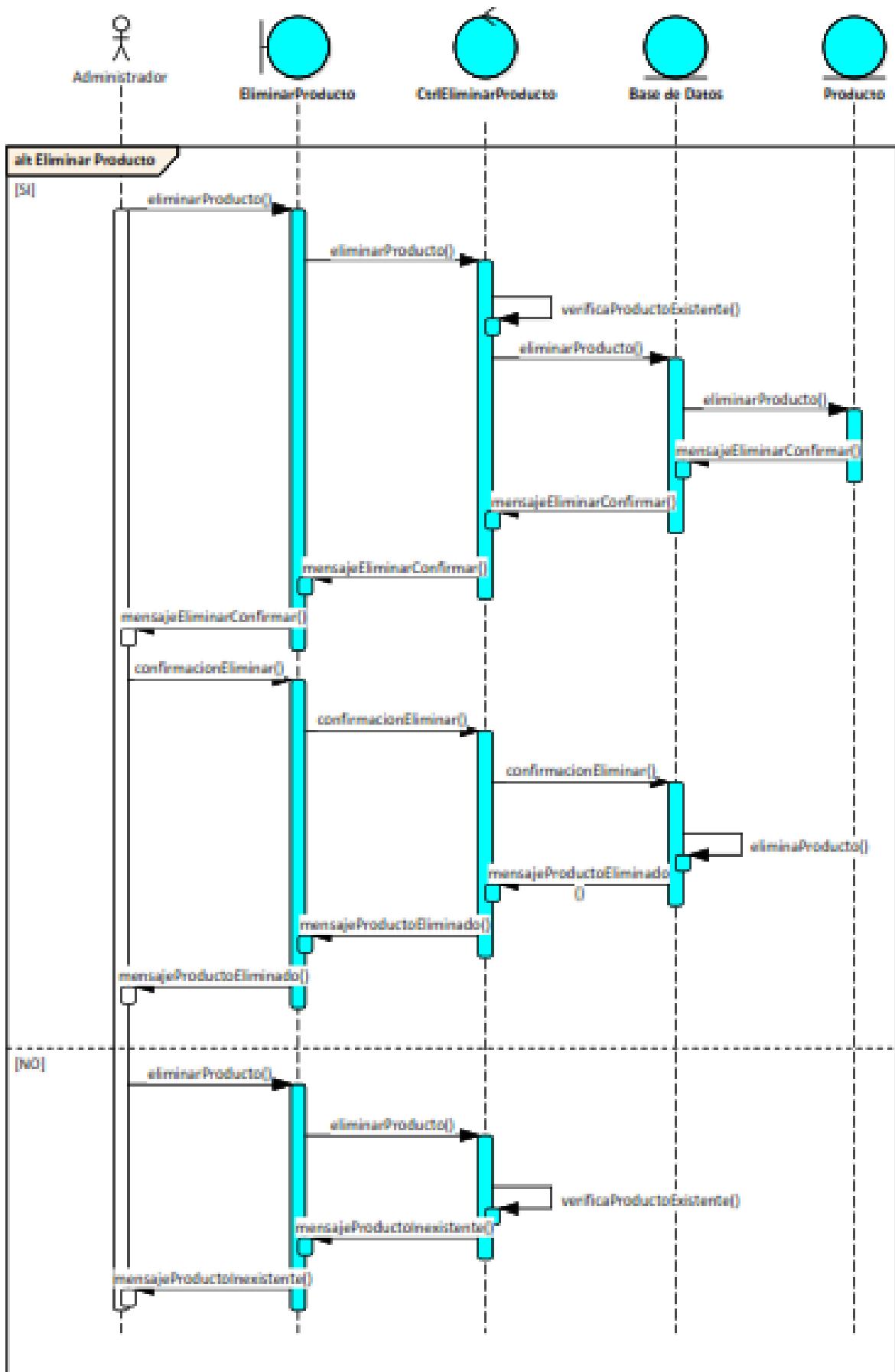


DIAGRAMA DE SECUENCIA DE ANALISIS #HU3.3 BUSCAR PRODUCTO

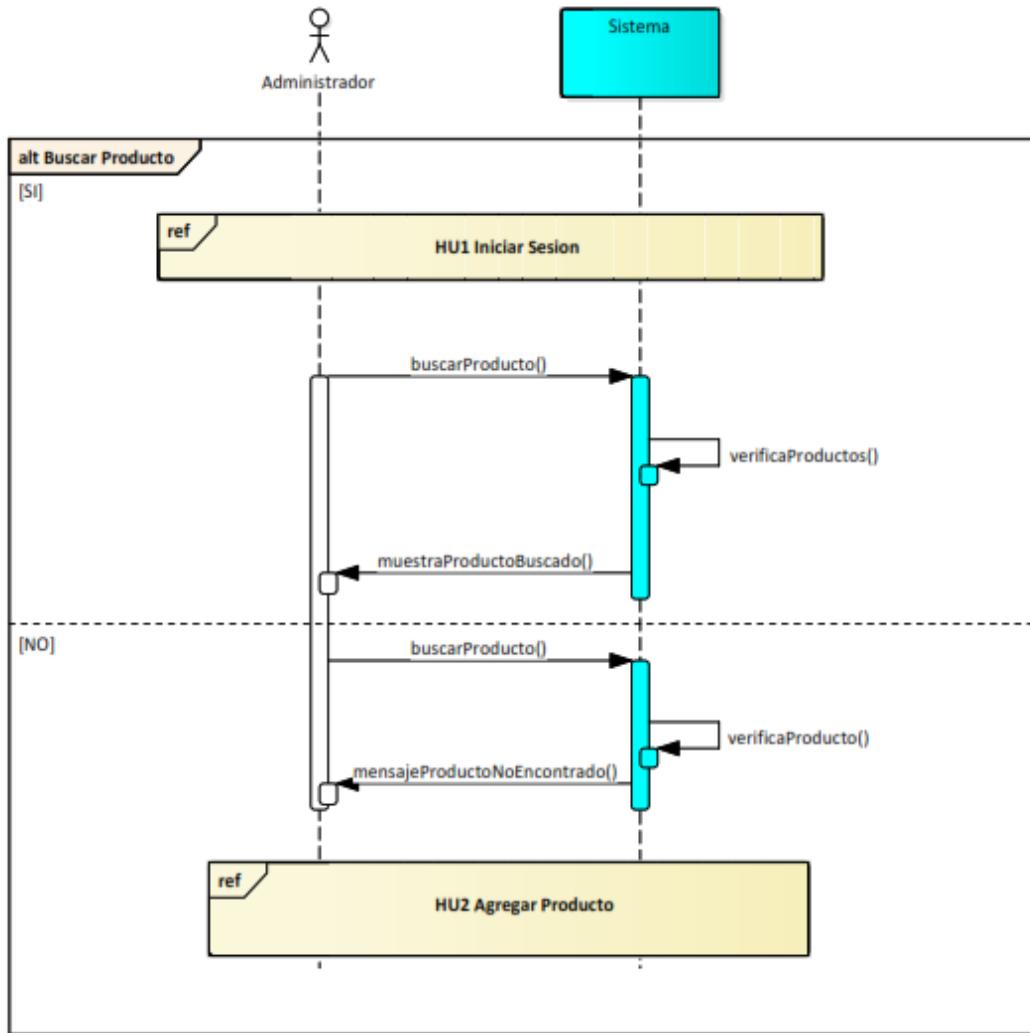


DIAGRAMA DE SECUENCIA DE DISEÑO #HU3.3 BUSCAR PRODUCTO

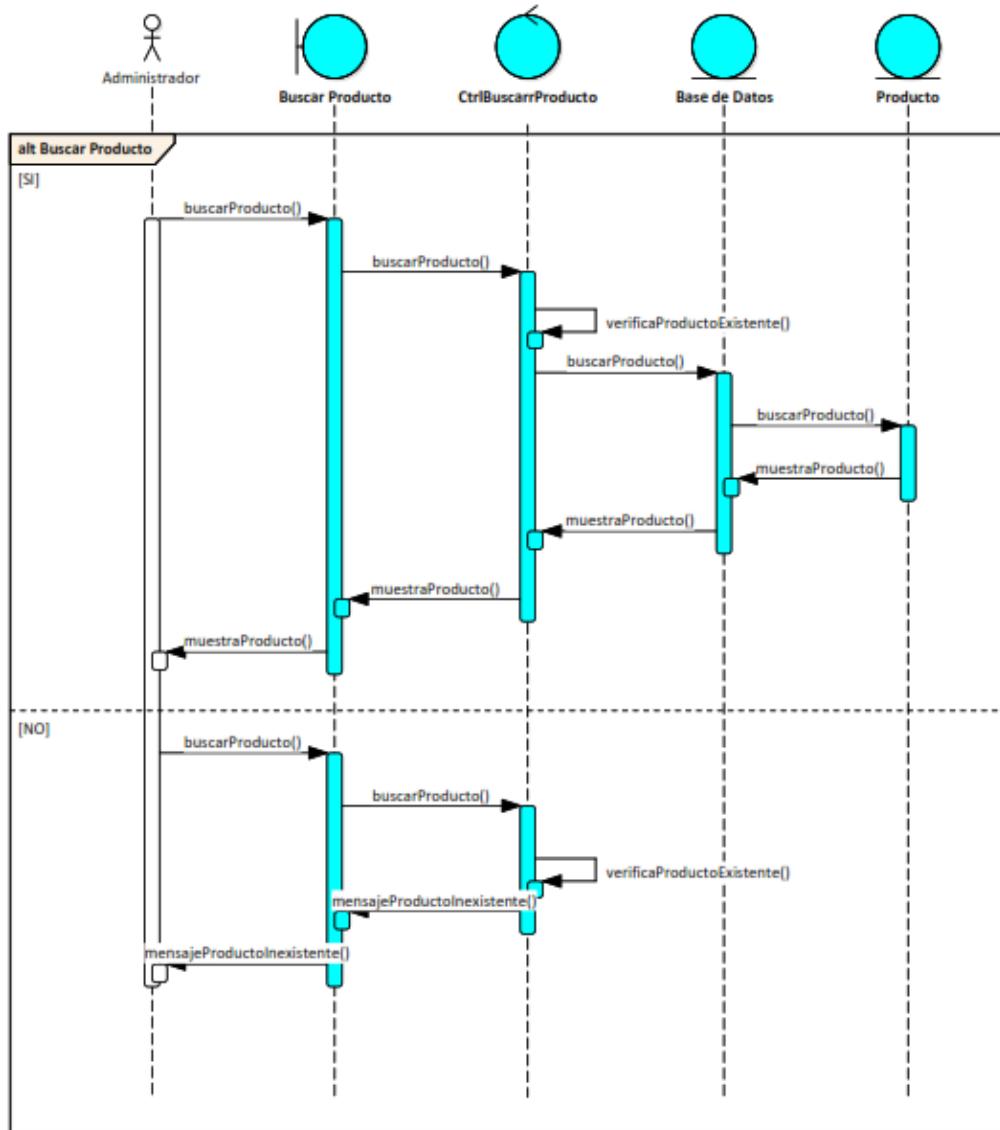


DIAGRAMA DE SECUENCIA DE ANALISIS #HU3.4 MOSTRAR PRODUCTO

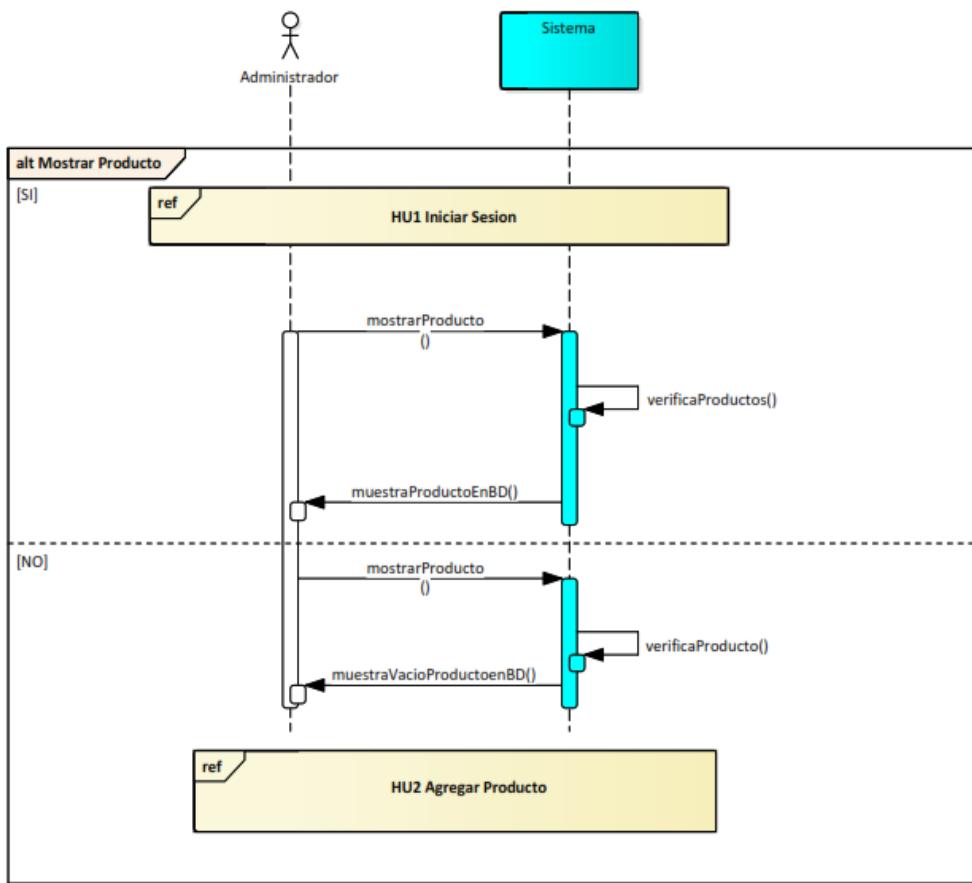


DIAGRAMA DE SECUENCIA DE DISEÑO #HU3.4 MOSTRAR PRODUCTO

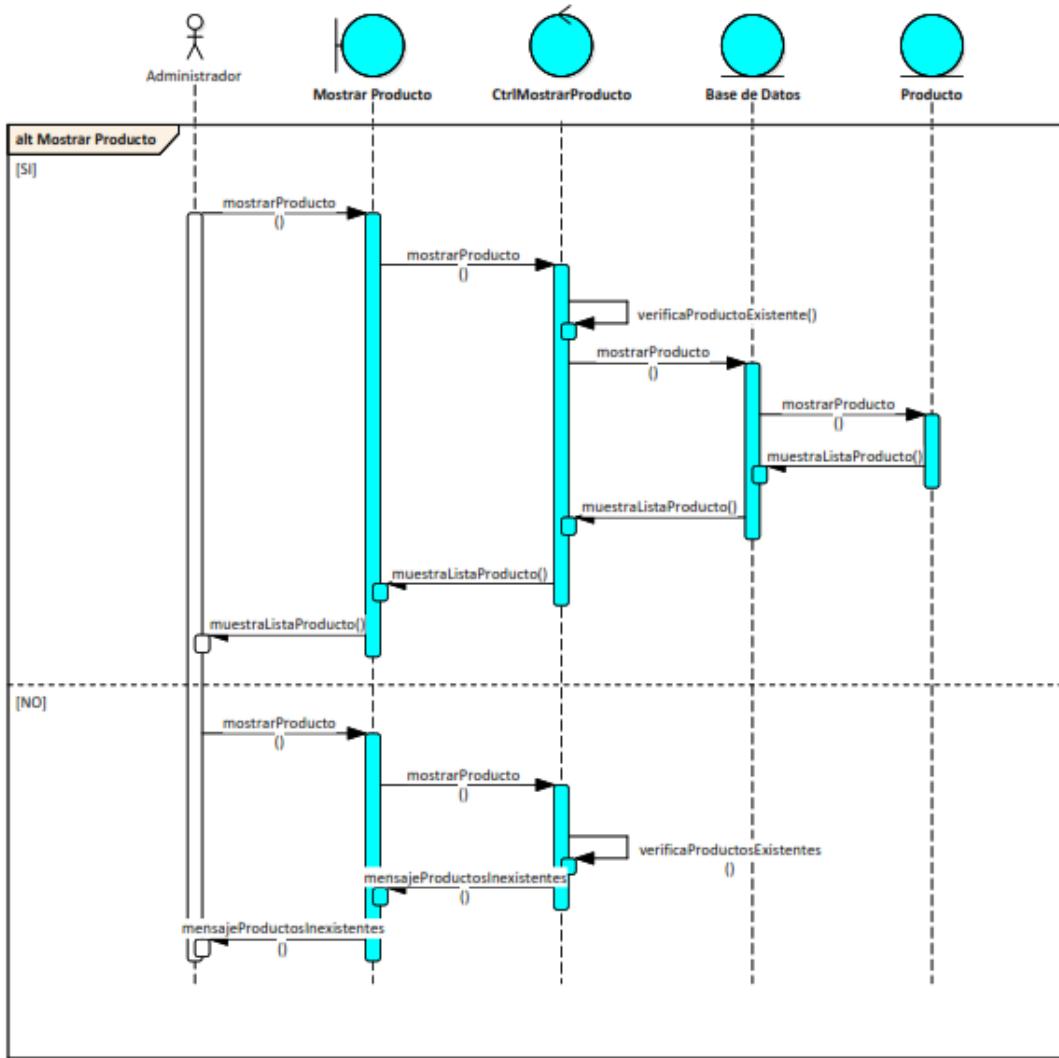


DIAGRAMA DE CLASES PARTICIPANTES #HU4

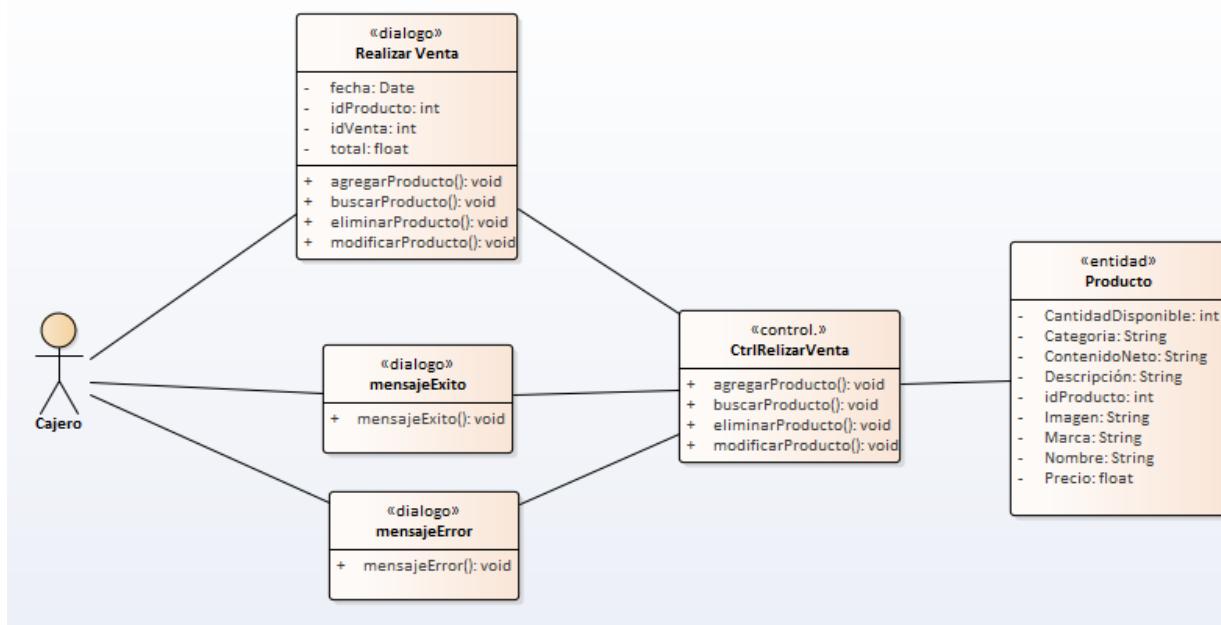


DIAGRAMA DE SECUENCIA DE ANALISIS #HU4

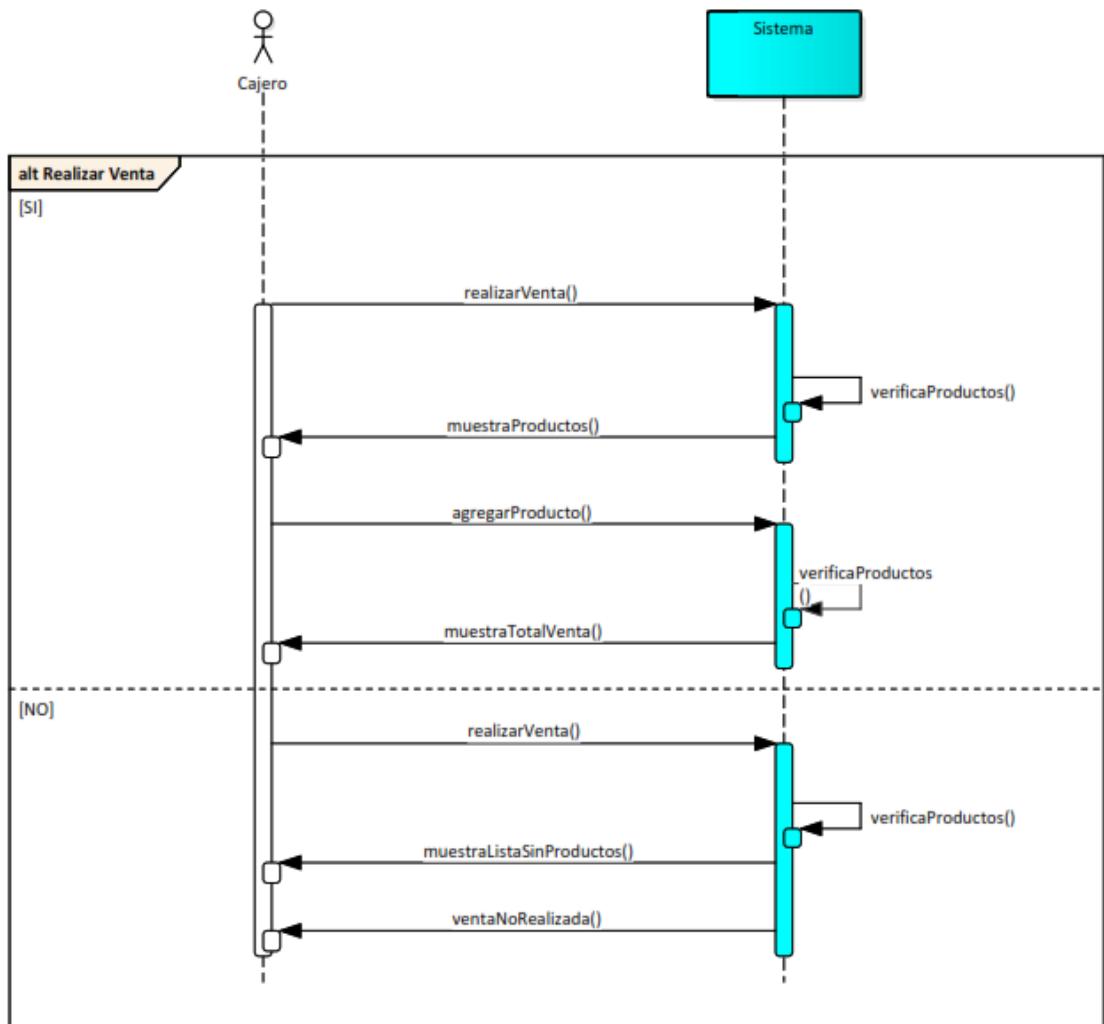


DIAGRAMA DE SECUENCIA DE DISEÑO #HU4

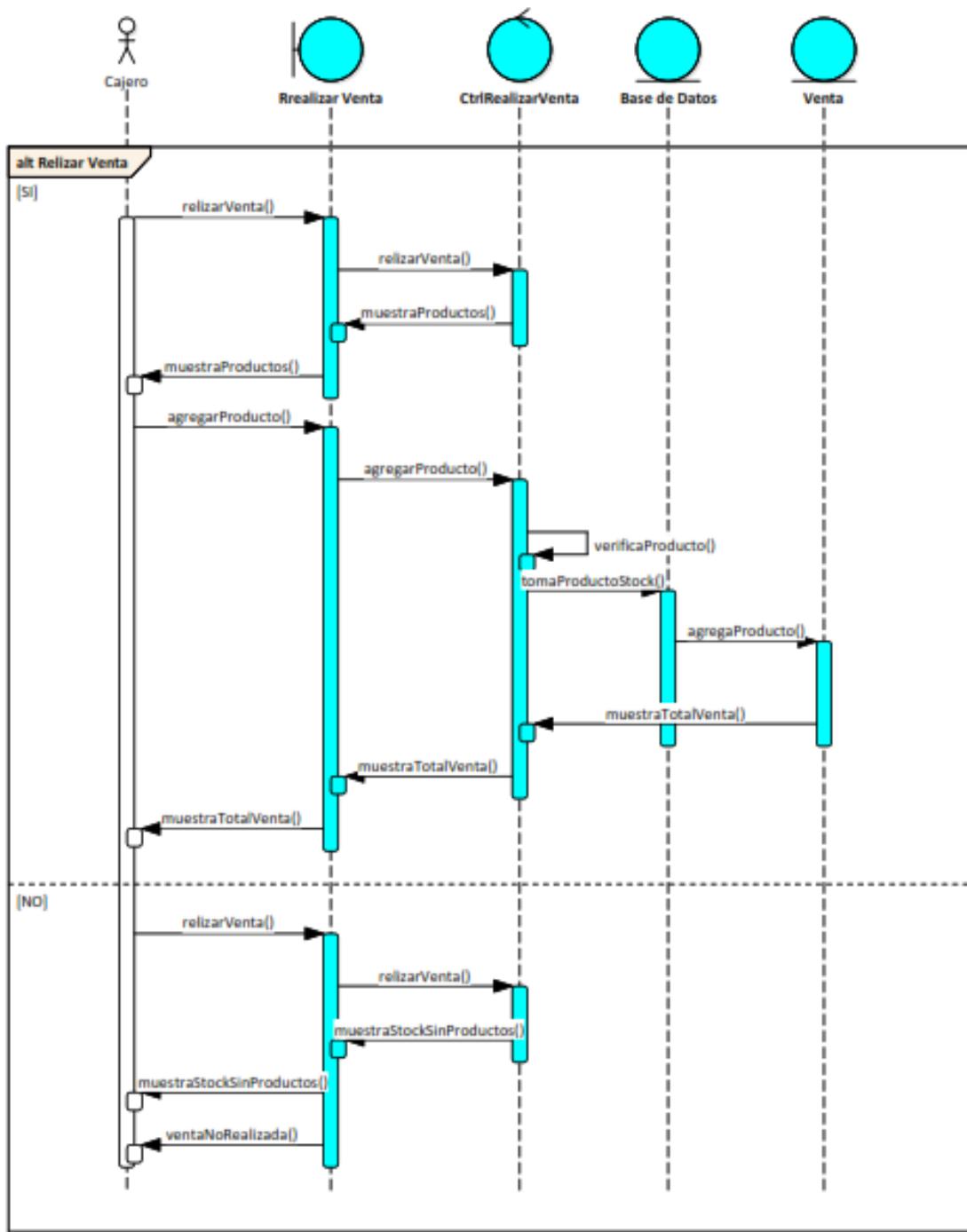


DIAGRAMA DE CLASES PARTICIPANTES #HU5

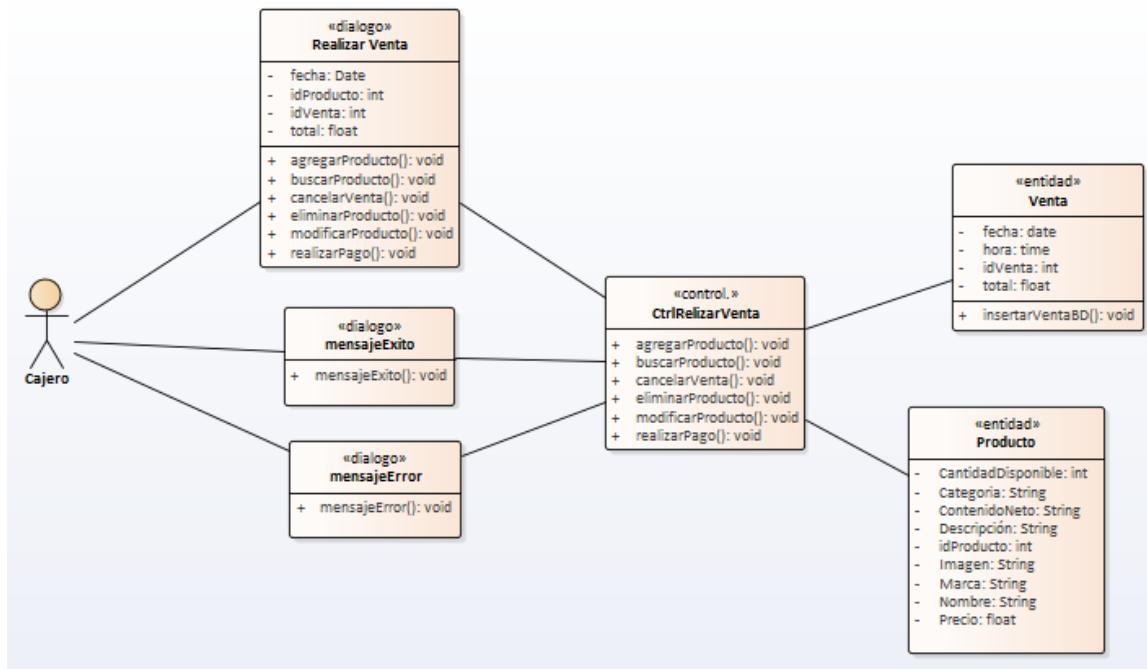


DIAGRAMA DE SECUENCIA DE ANALISIS #HU5

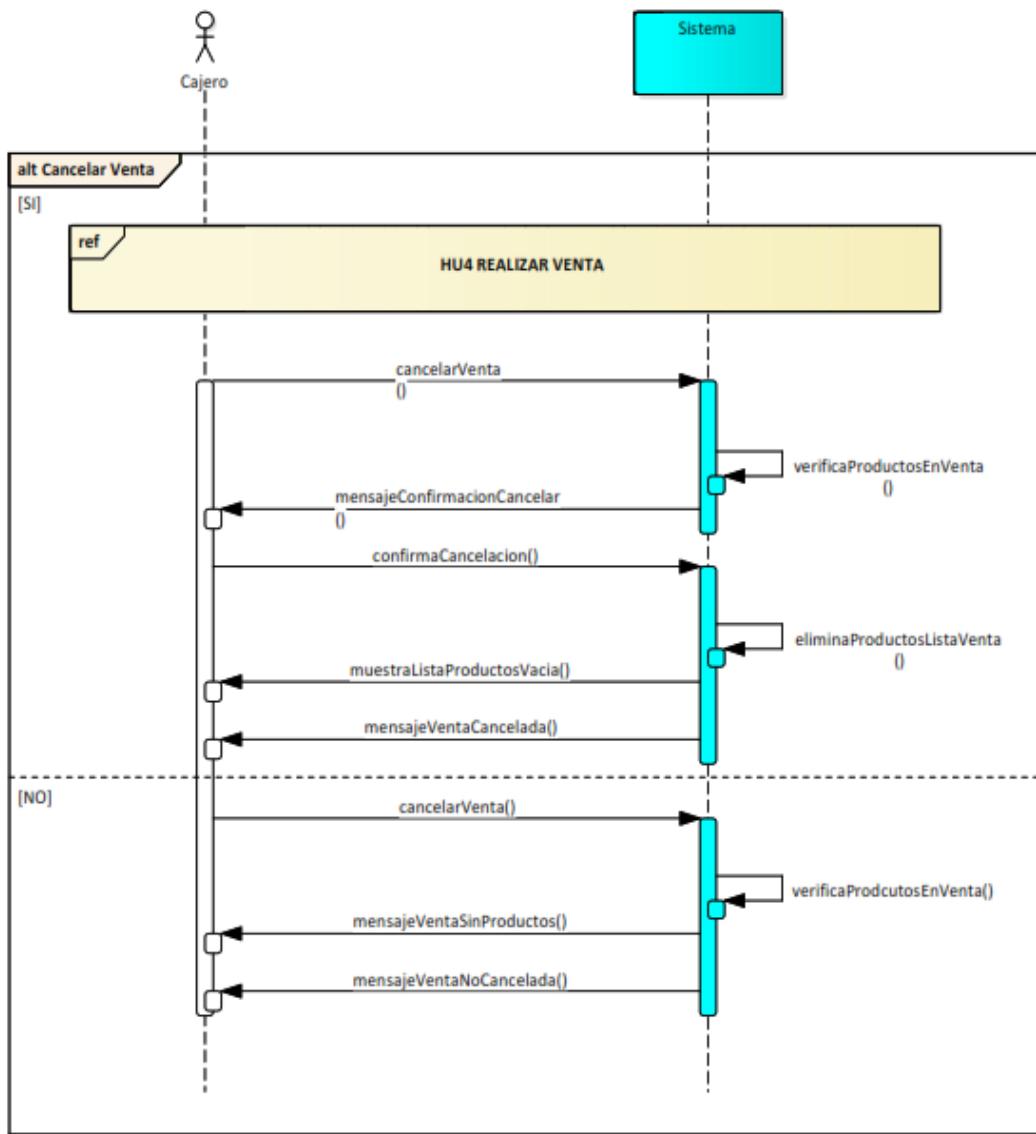


DIAGRAMA DE SECUENCIA DE DISEÑO #HU5

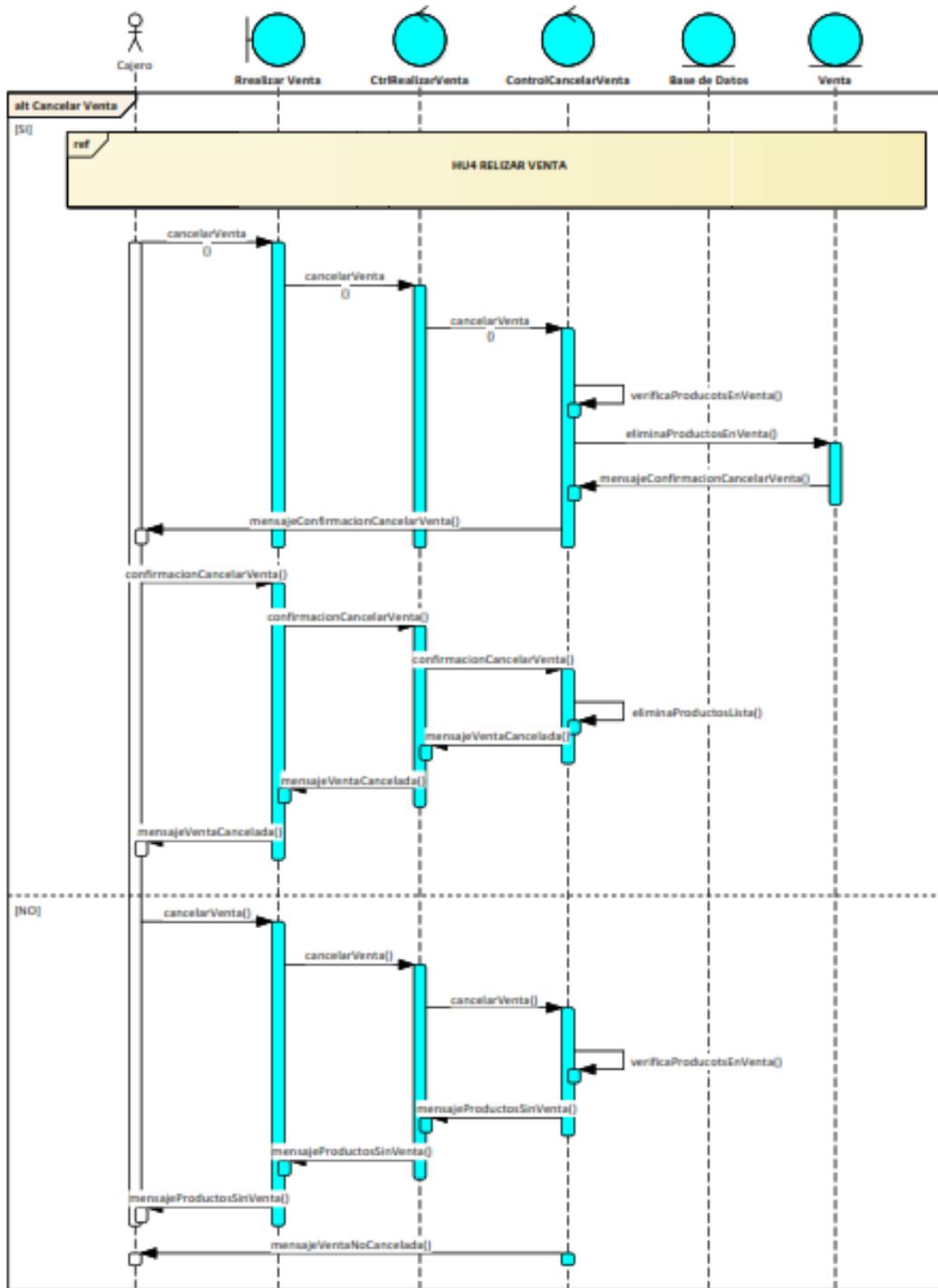


DIAGRAMA DE CLASES PARTICIPANTES #HU6

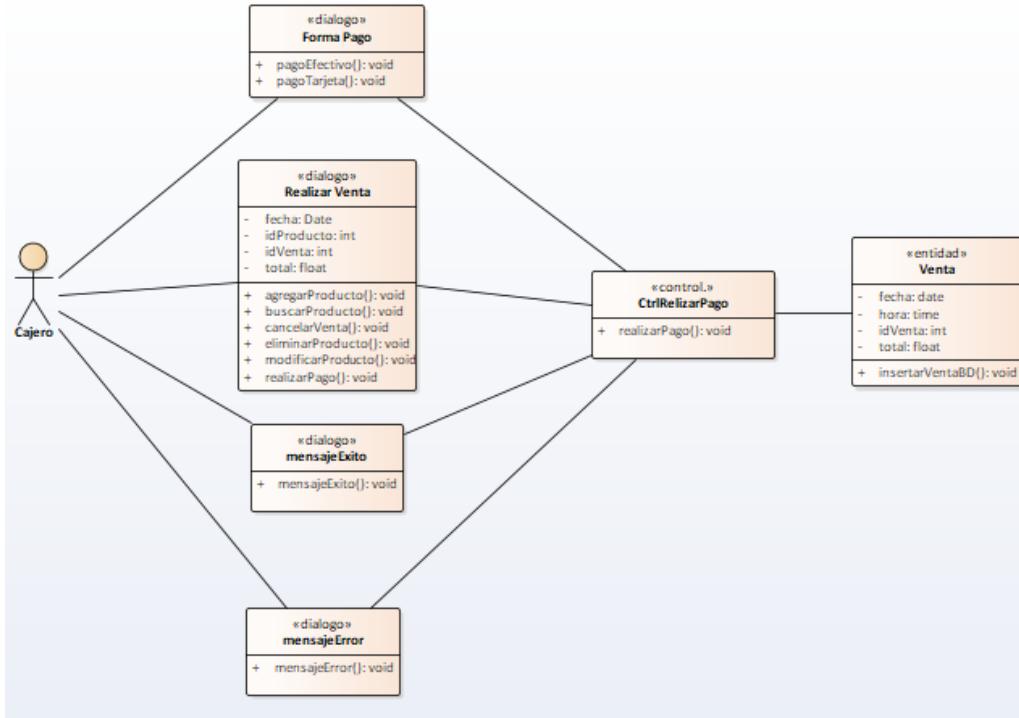


DIAGRAMA DE SECUENCIA DE ANALISIS #HU6

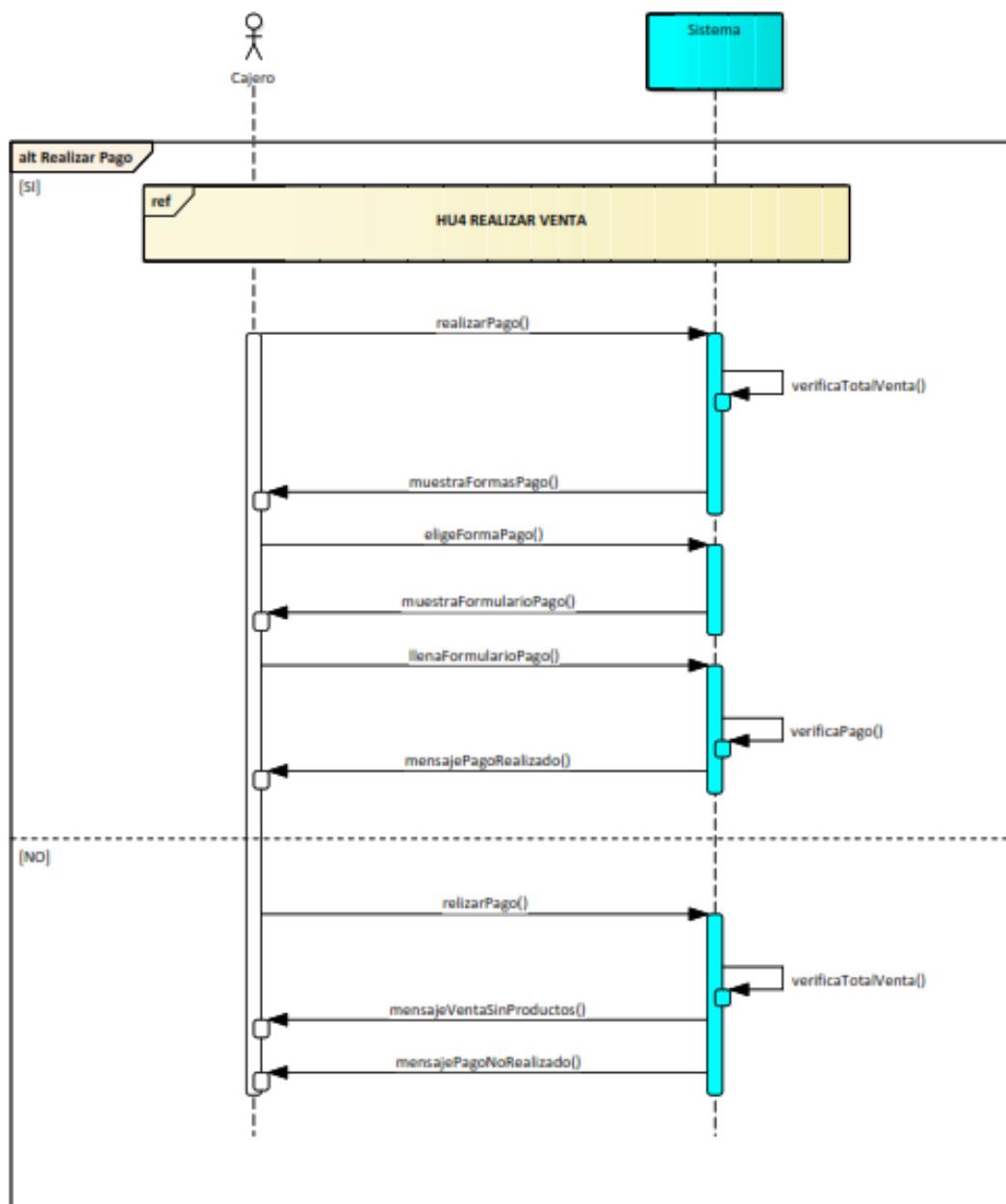


DIAGRAMA DE SECUENCIA DE DISEÑO #HU6

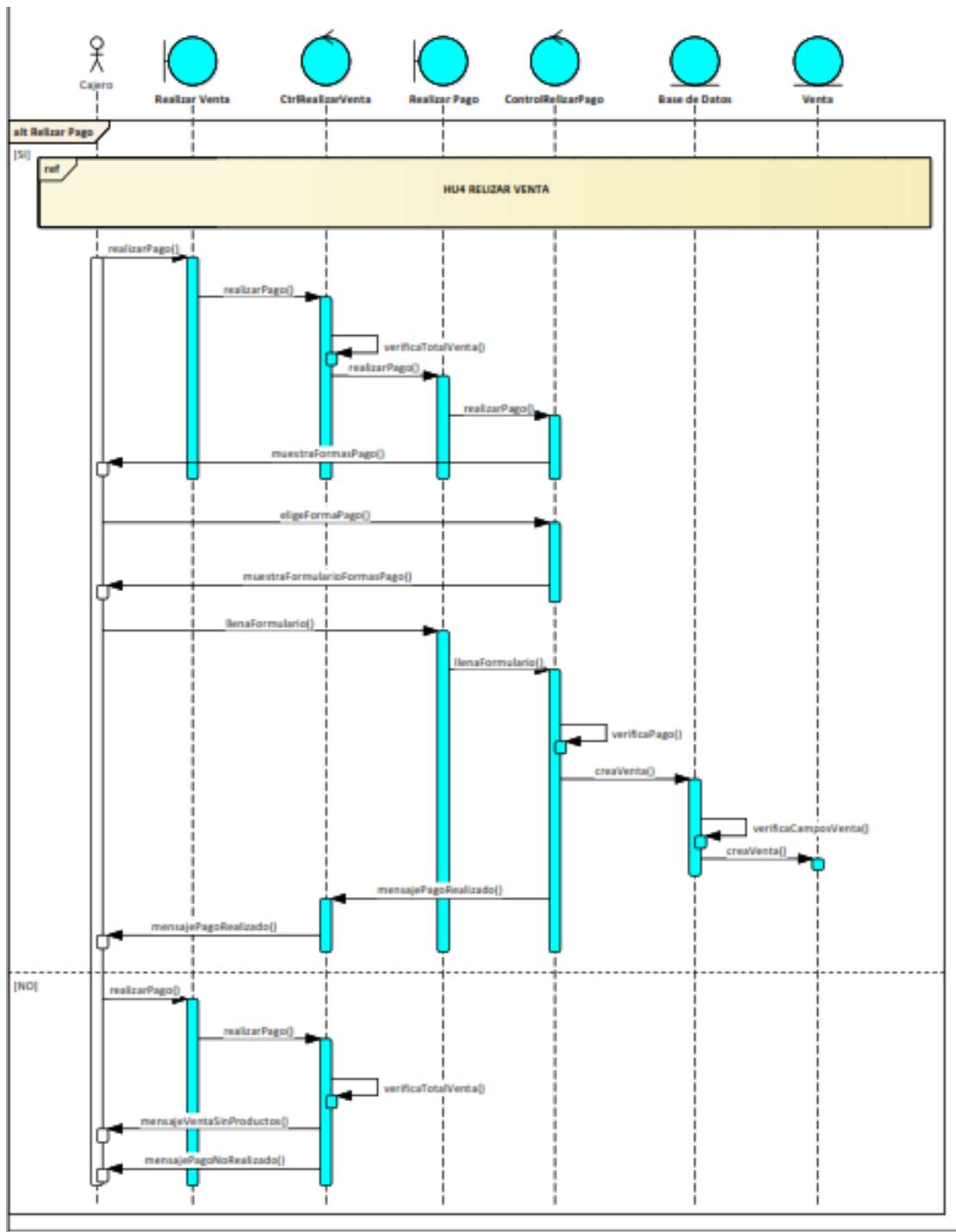


DIAGRAMA DE CLASES PARTICIPANTES #HU7

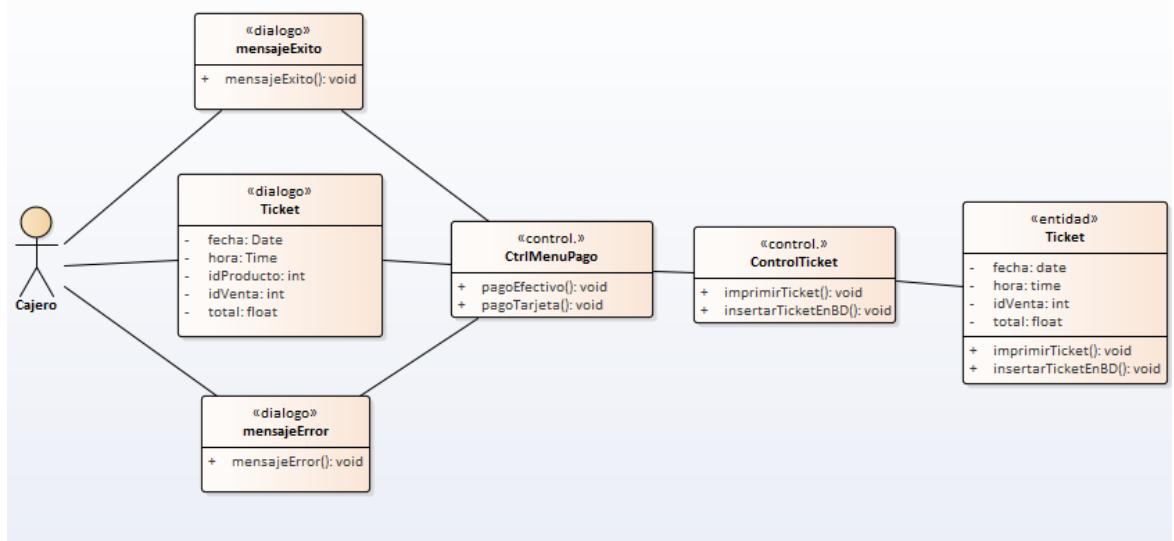


DIAGRAMA DE SECUENCIA DE ANALISIS #HU7

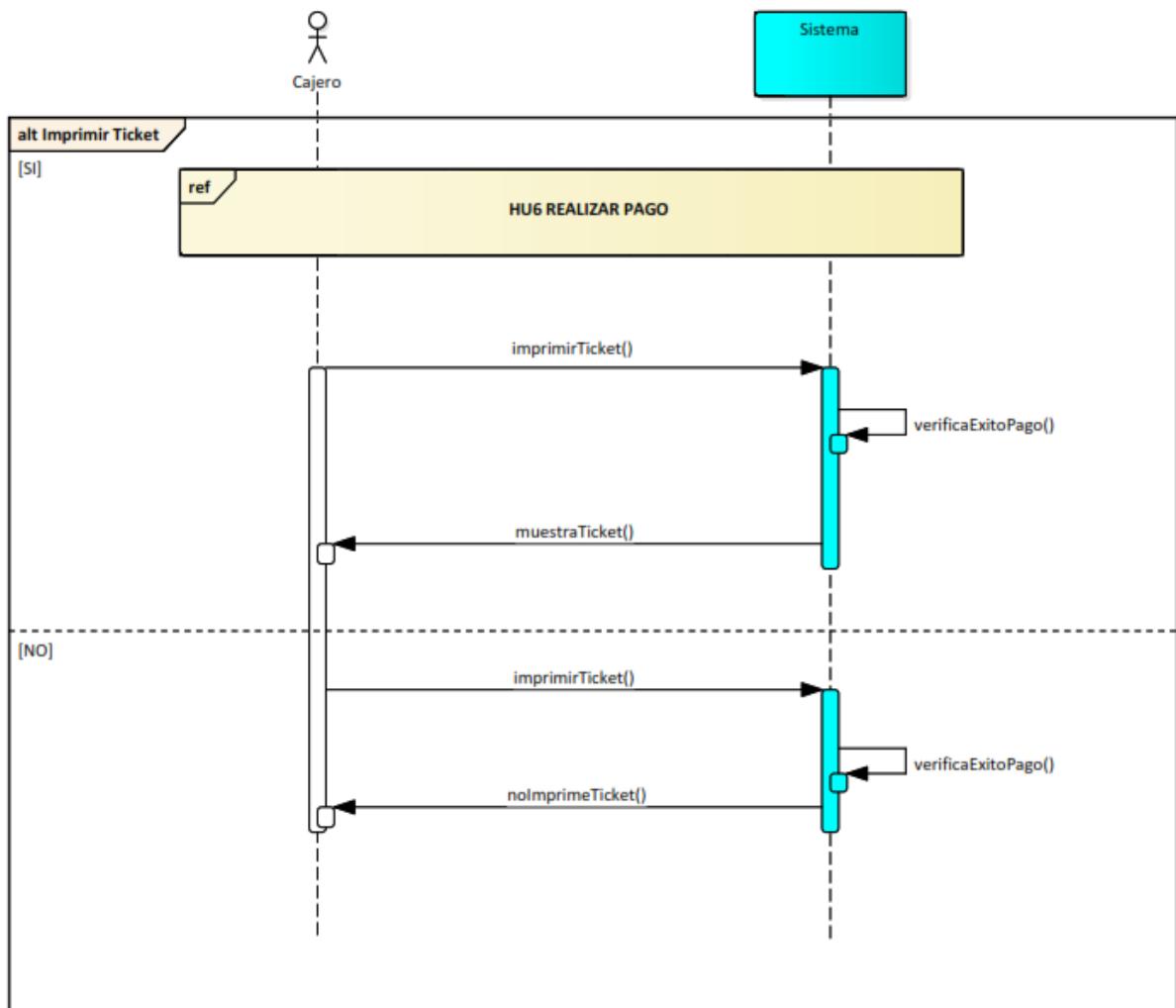


DIAGRAMA DE SECUENCIA DE DISEÑO #HU7

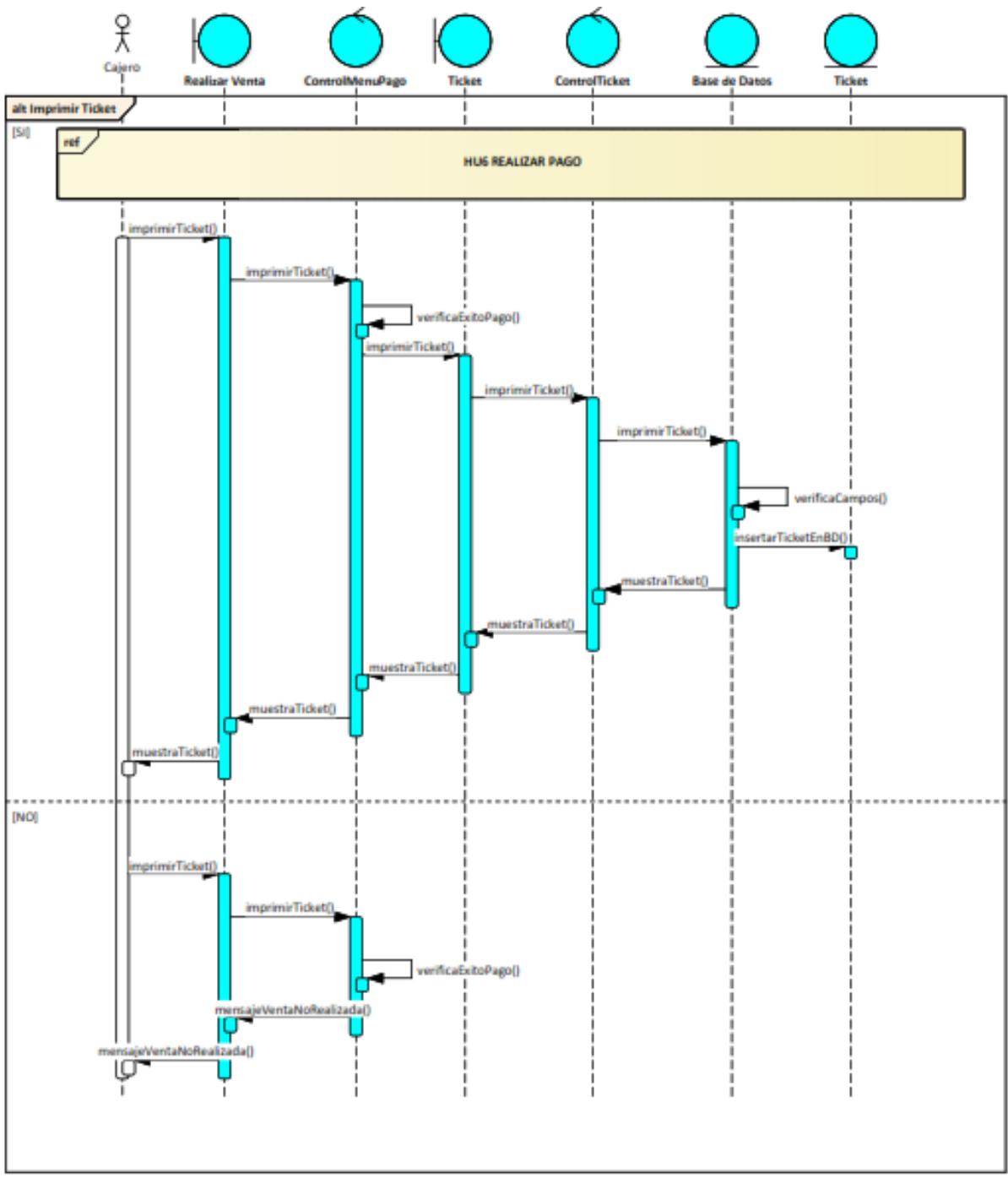


DIAGRAMA DE CLASES PARTICIPANTES #HUG8

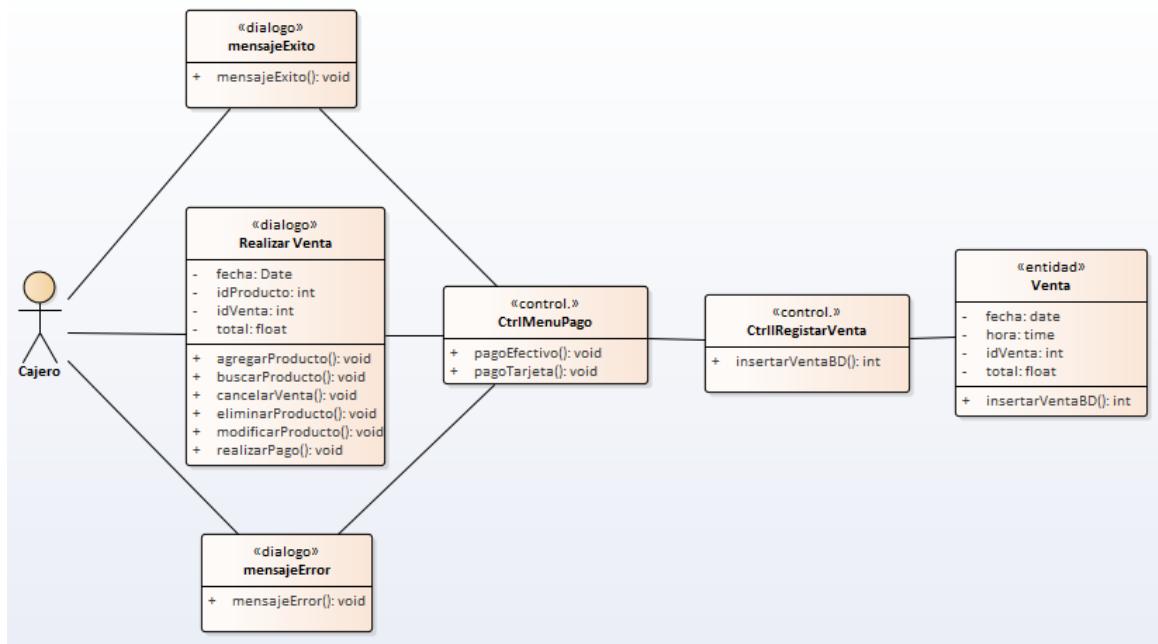


DIAGRAMA DE SECUENCIA DE ANALISIS #HU8

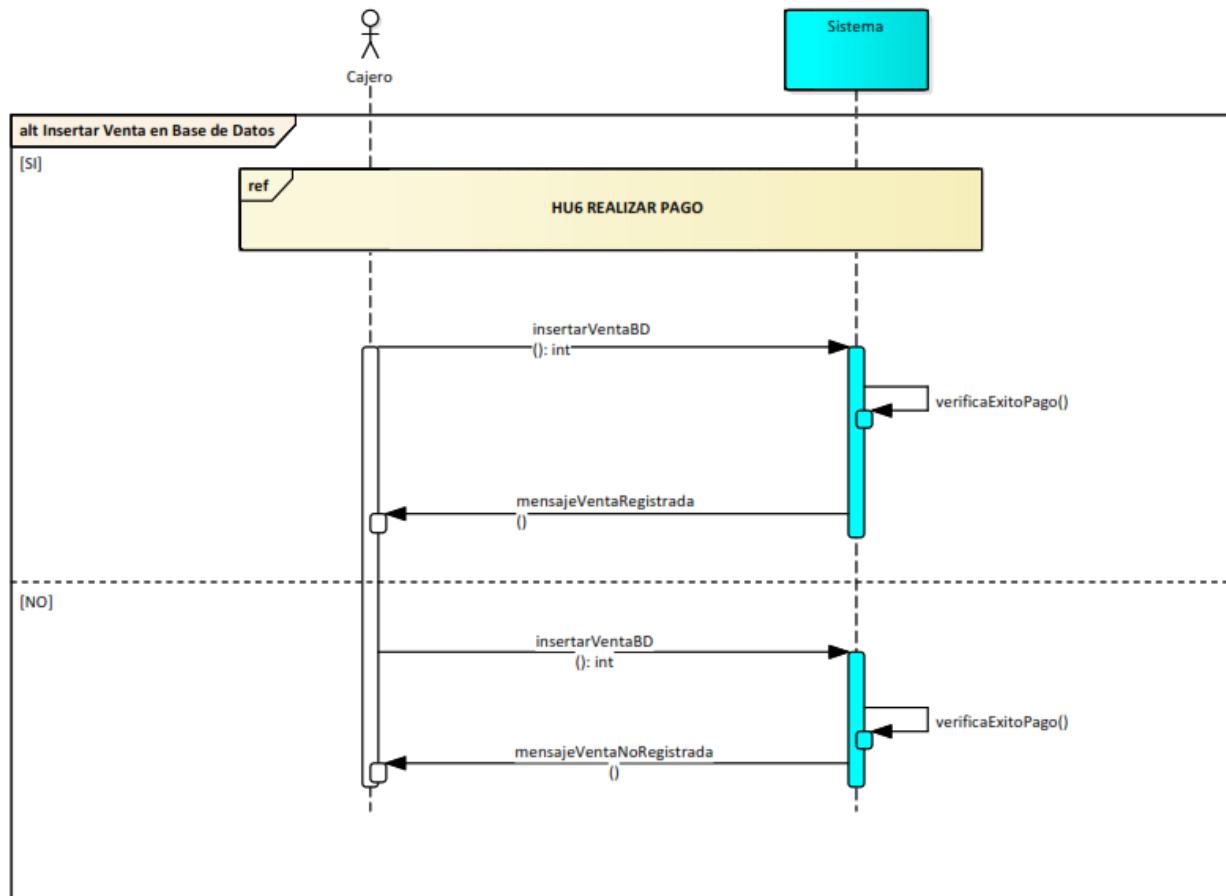


DIAGRAMA DE SECUENCIA DE DISEÑO #HU8

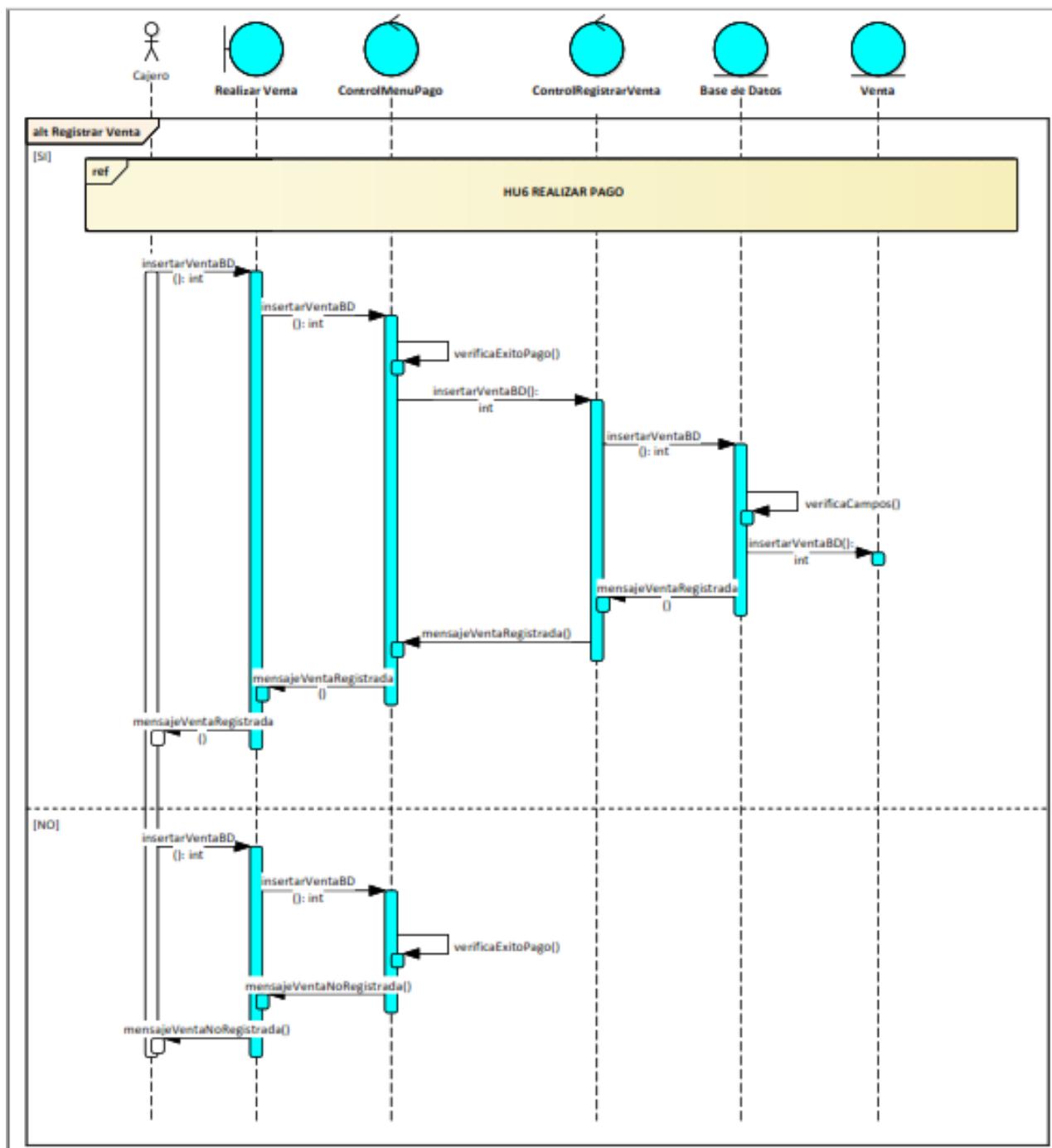


DIAGRAMA DE CLASES PARTICIPANTES #HU9

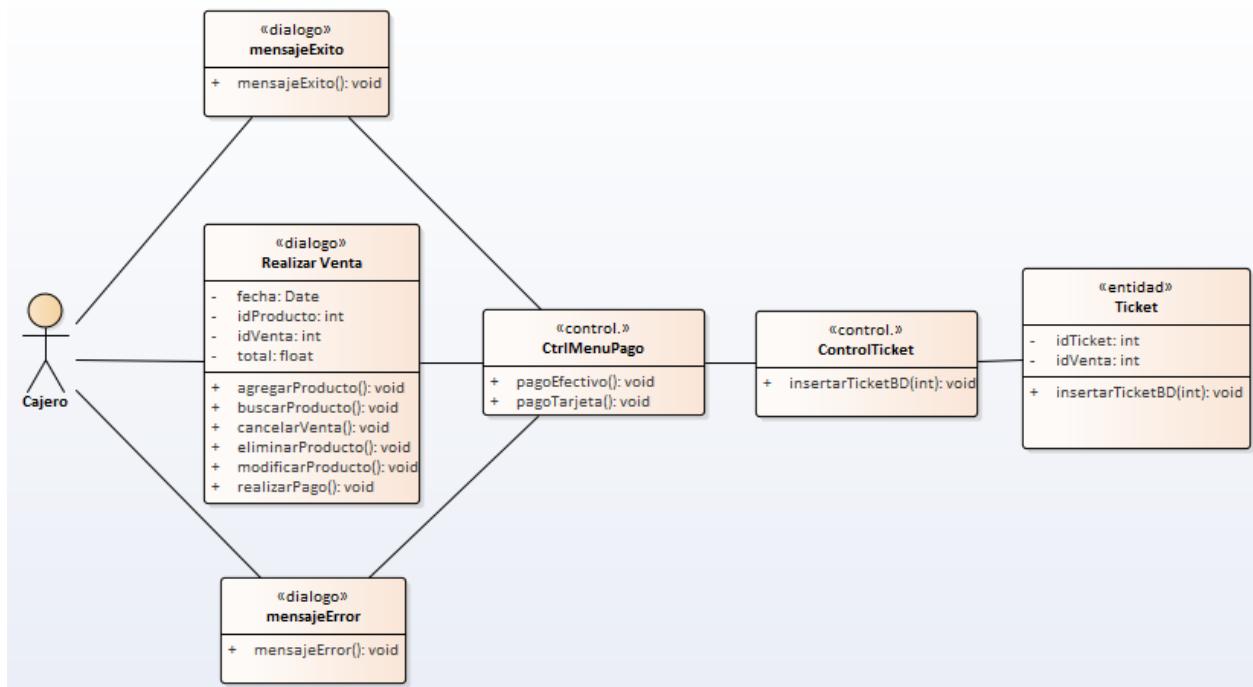


DIAGRAMA DE SECUENCIA DE ANALISIS #HU9

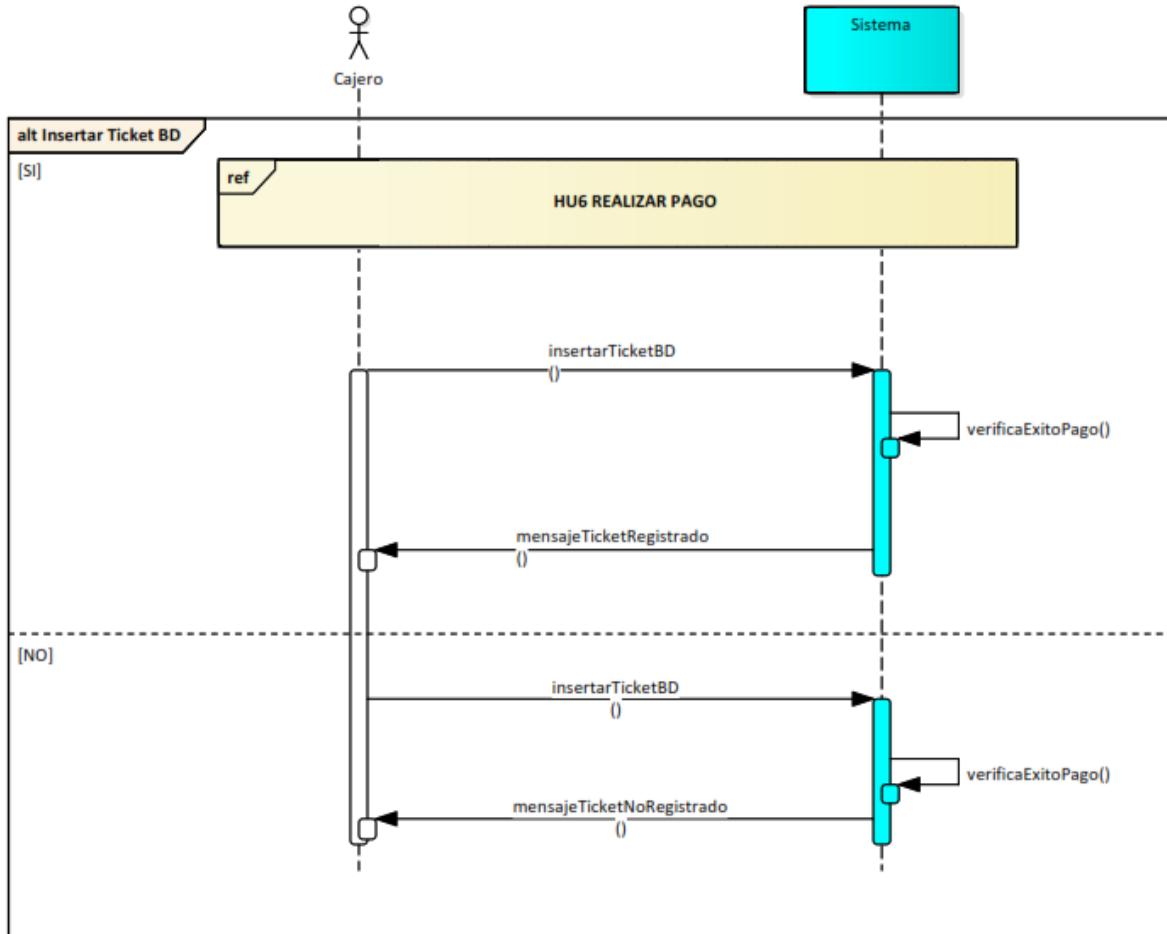


DIAGRAMA DE SECUENCIA DE DISEÑO #HU9

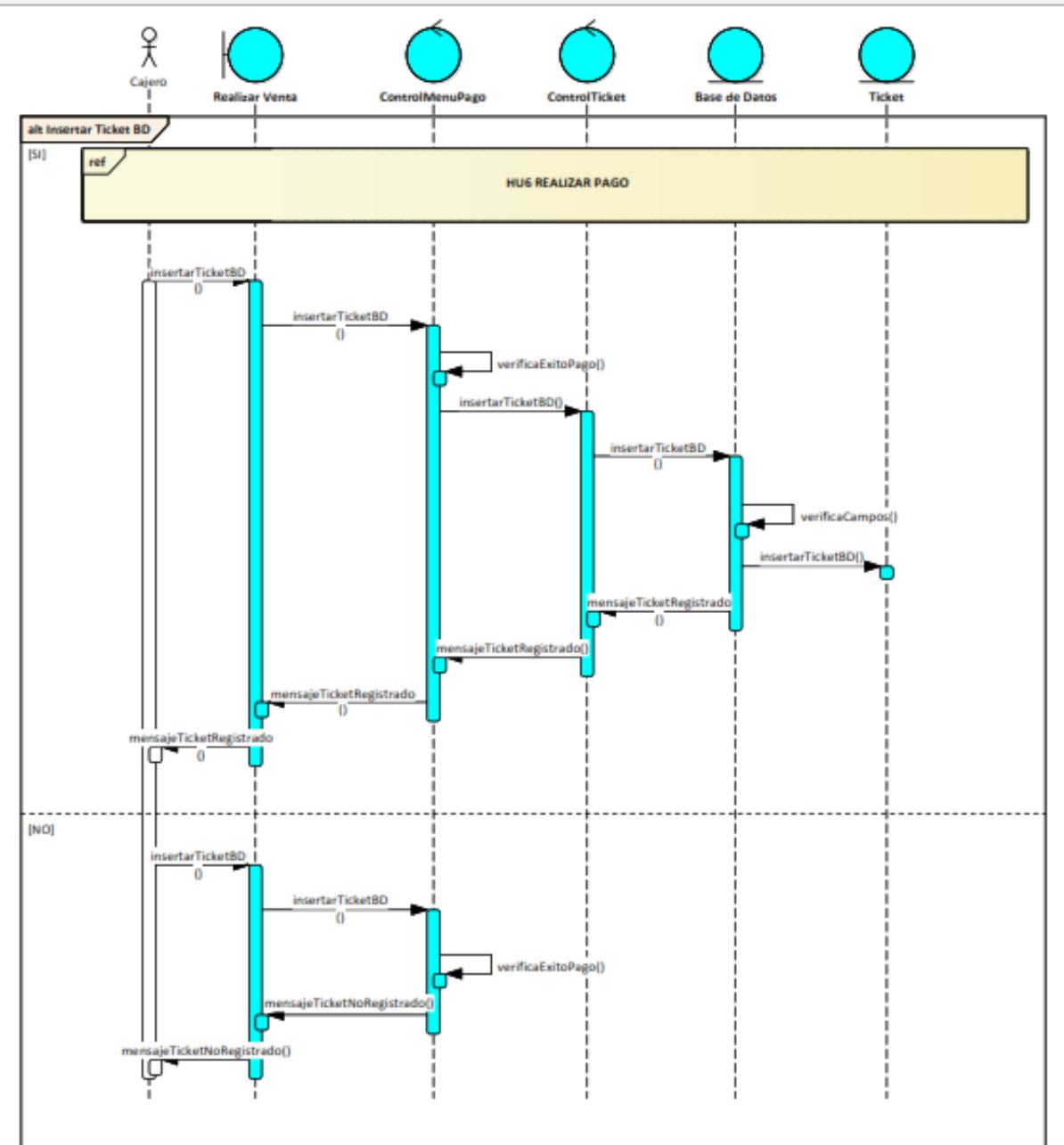


DIAGRAMA DE CLASES PARTICIPANTES #HU10

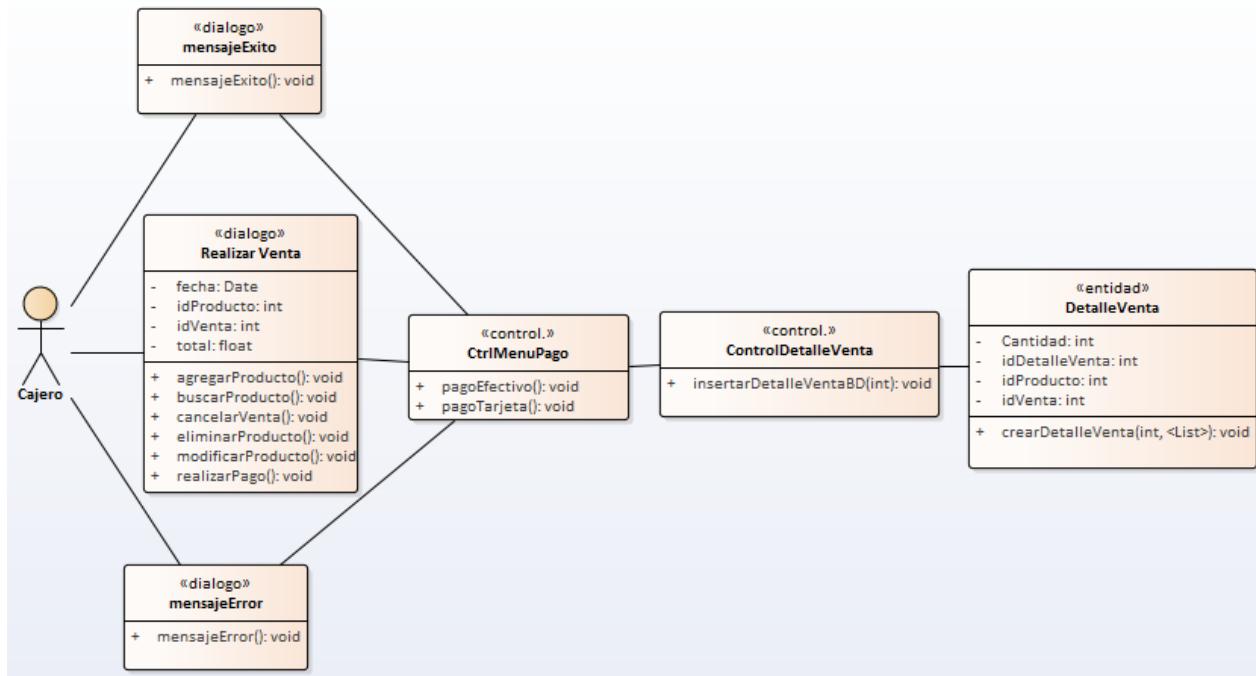


DIAGRAMA DE SECUENCIA DE ANALISIS #HU10

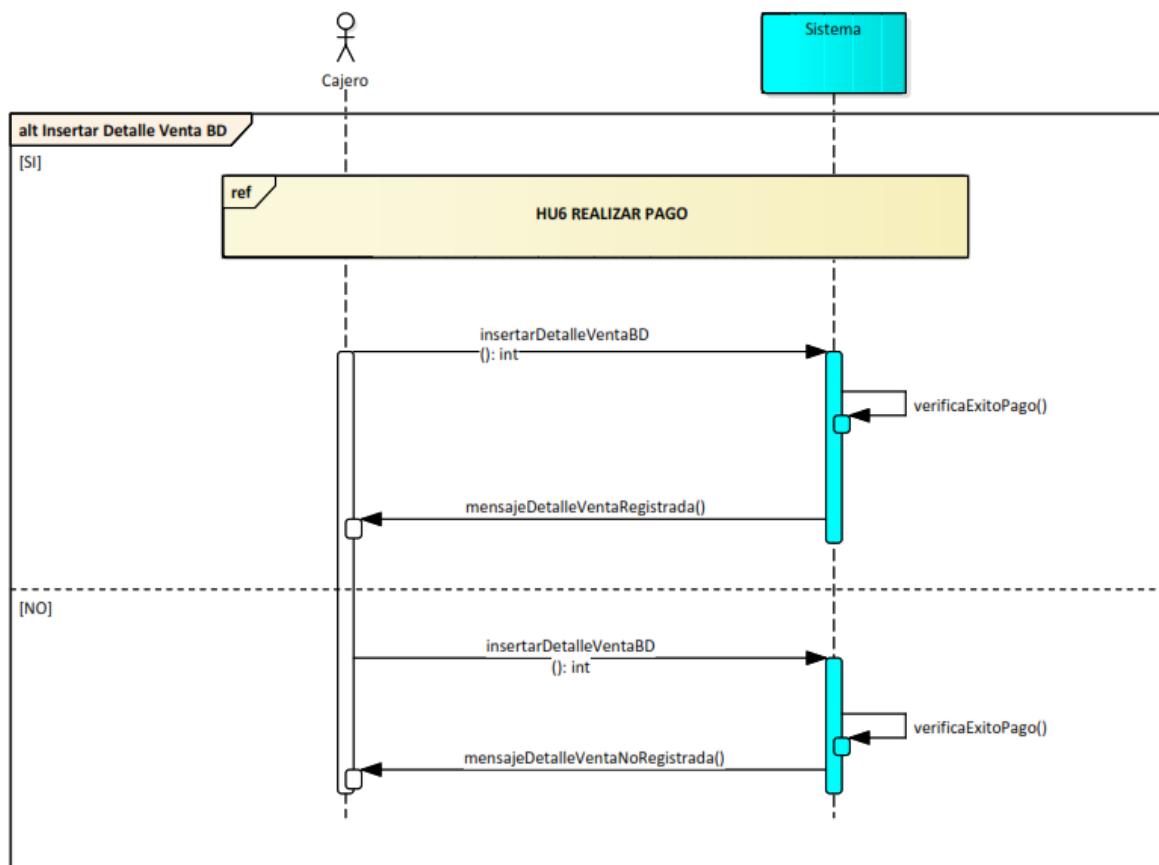


DIAGRAMA DE SECUENCIA DE DISEÑO #HU10

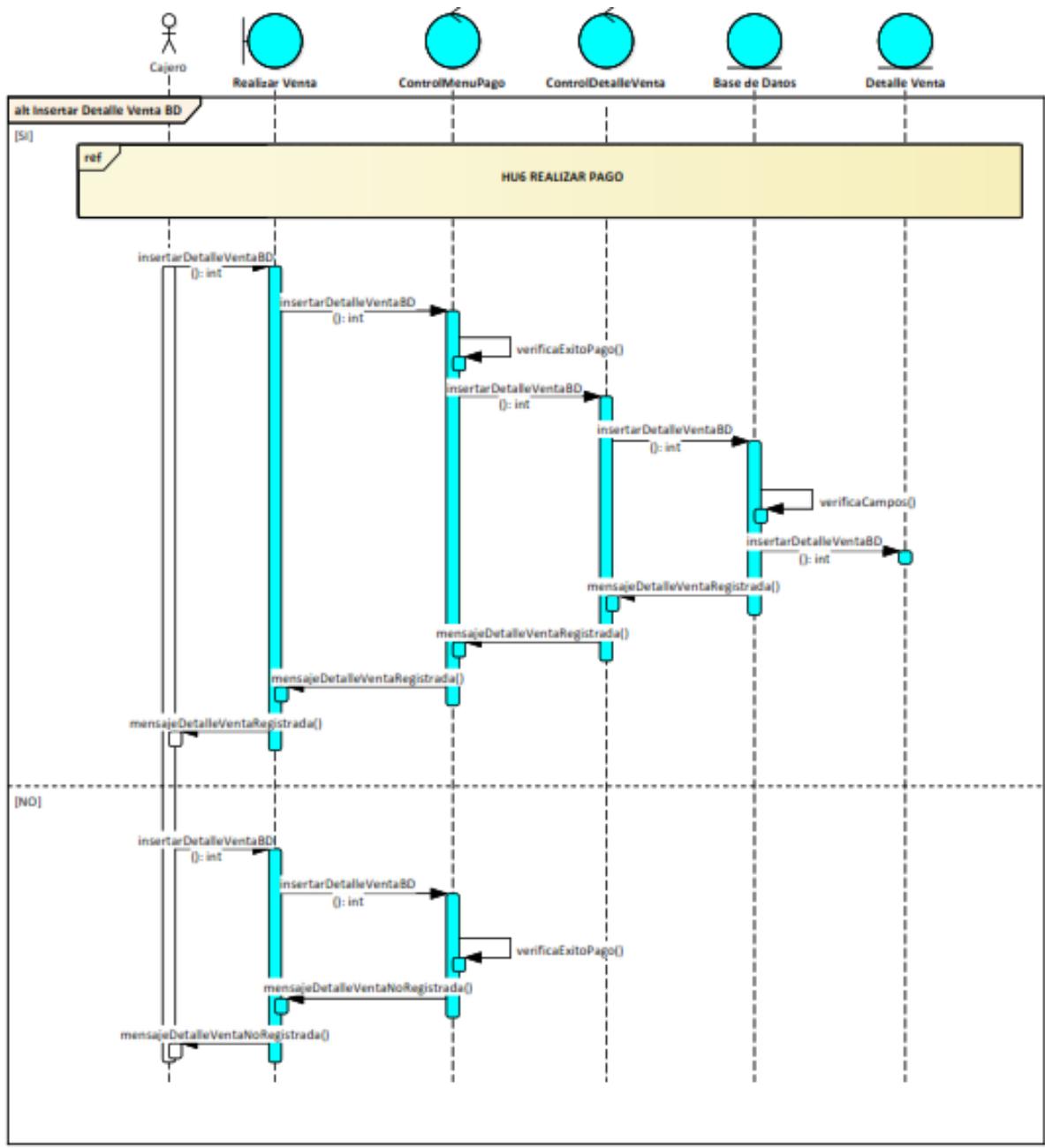


DIAGRAMA DE CLASES PARTICIPANTES #HU11

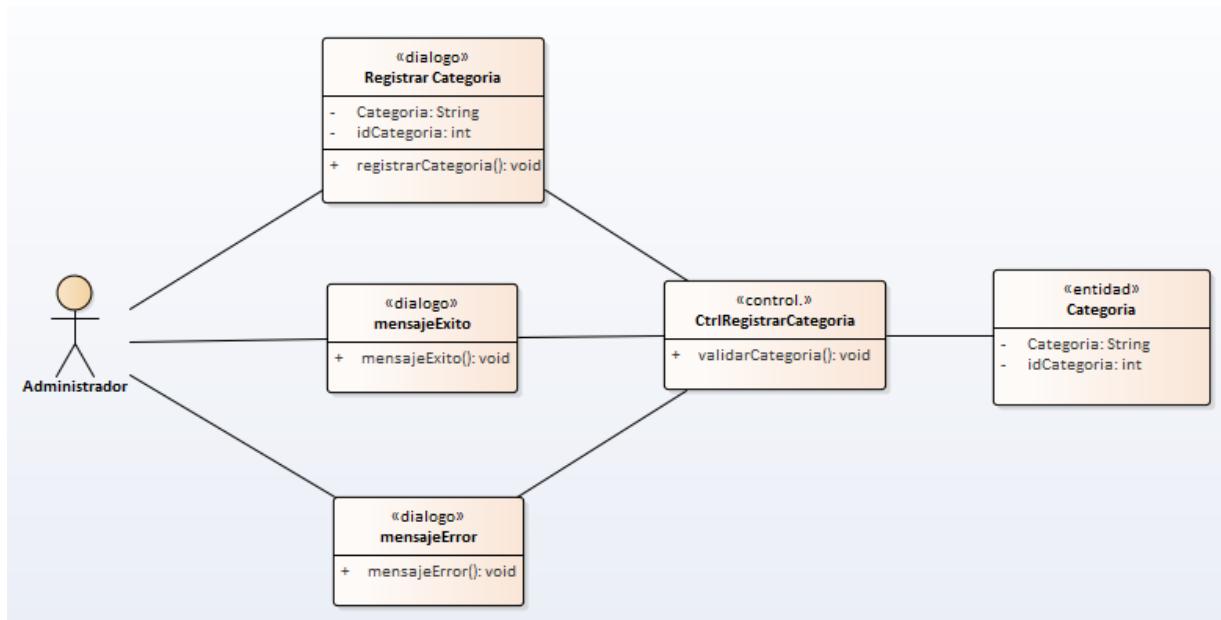


DIAGRAMA DE SECUENCIA DE ANALISIS #HU11

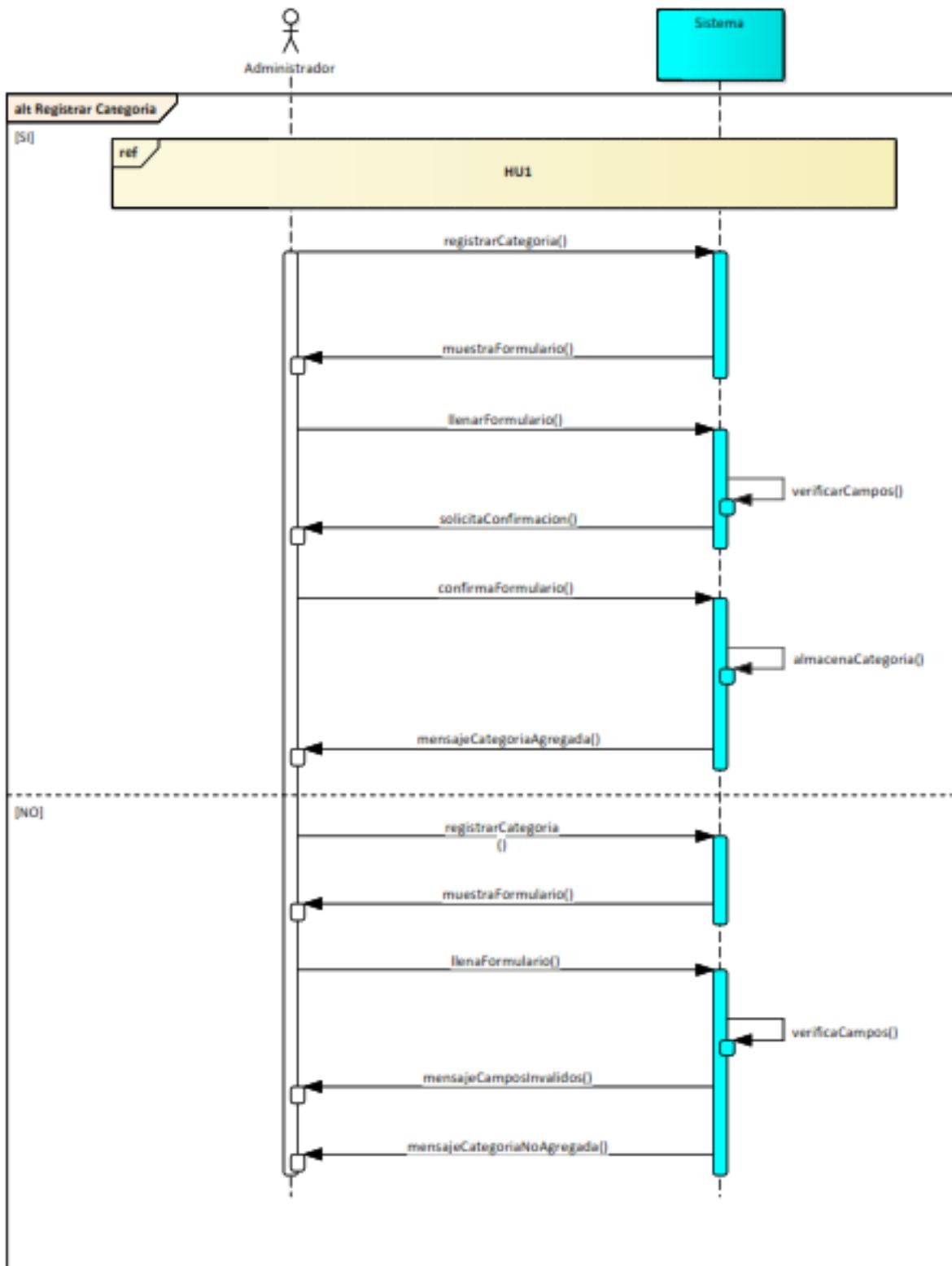
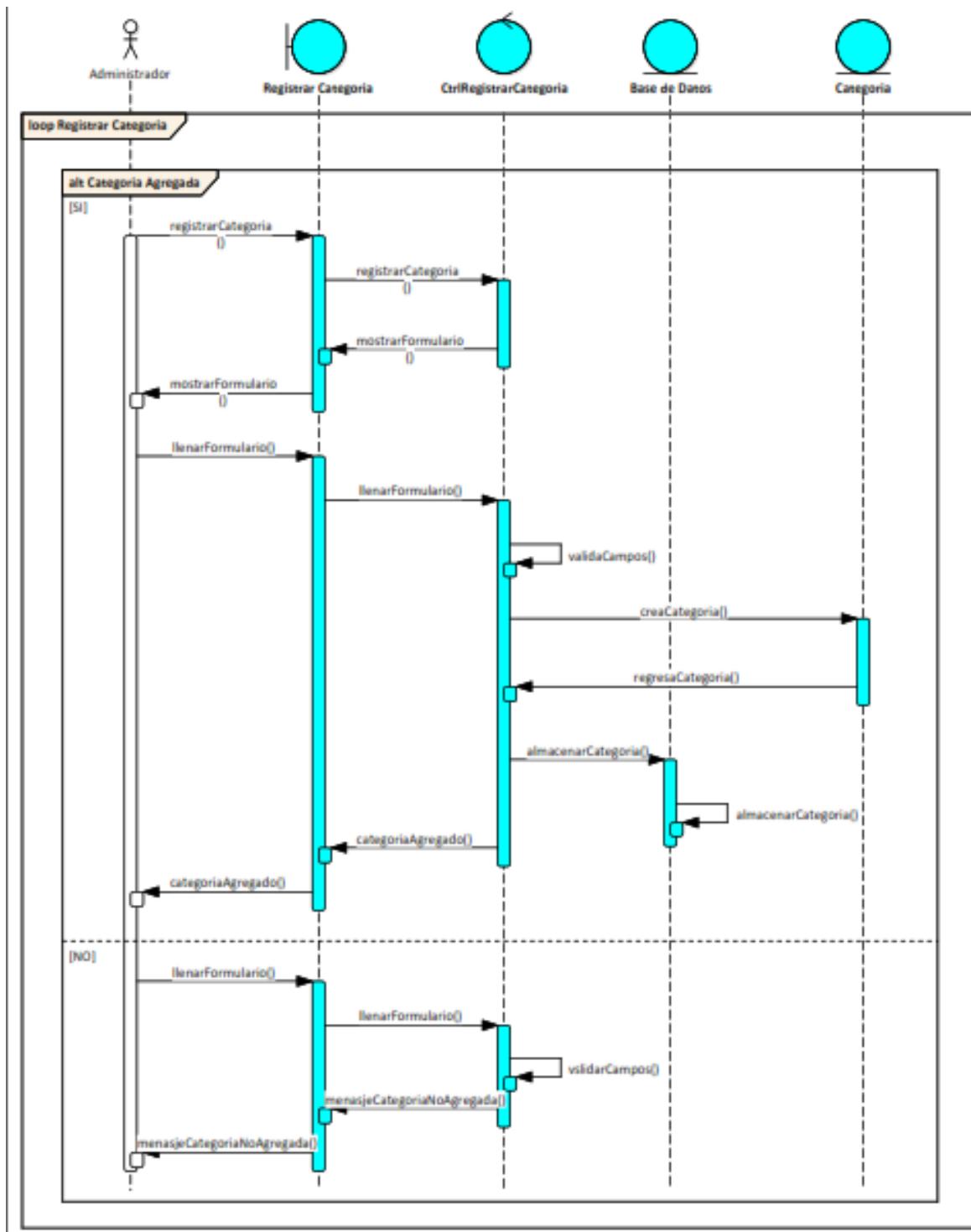
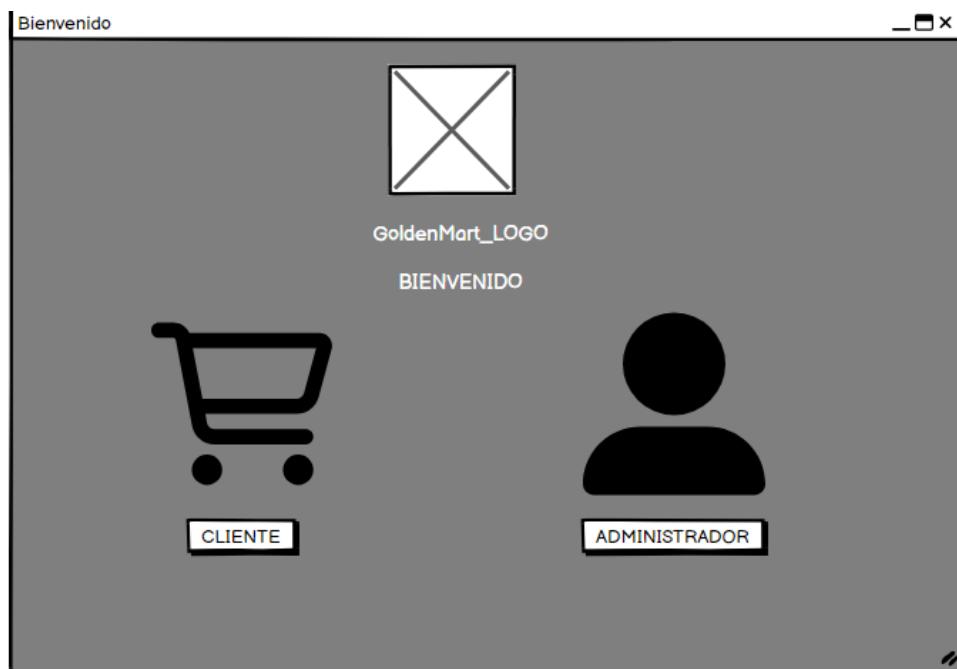


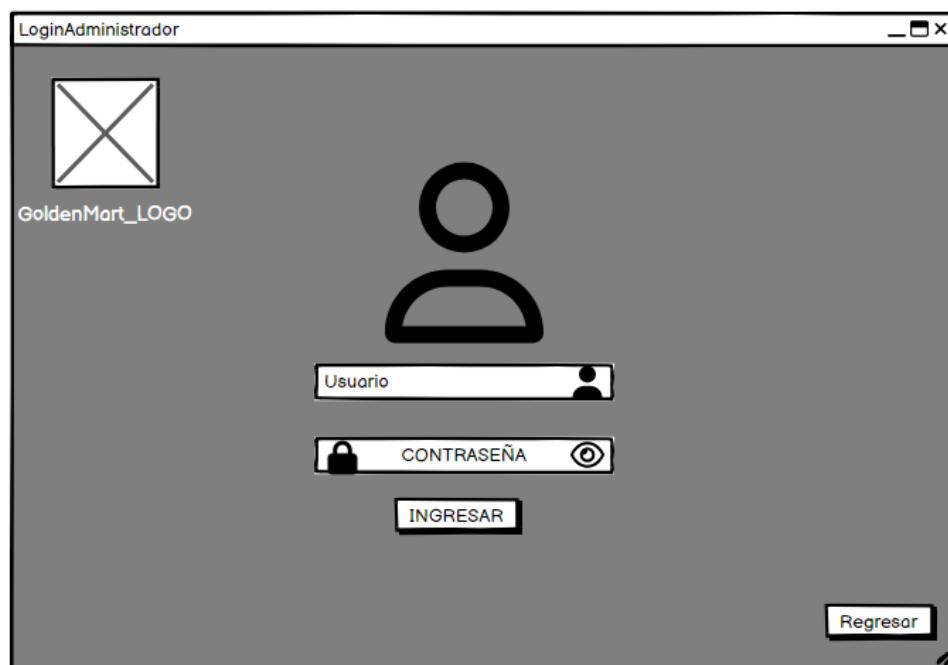
DIAGRAMA DE SECUENCIA DE DISEÑO #HU11



INTERFACES PREELIMINARES EN BALSAMIQ



INTERFAZ BIENVENIDO



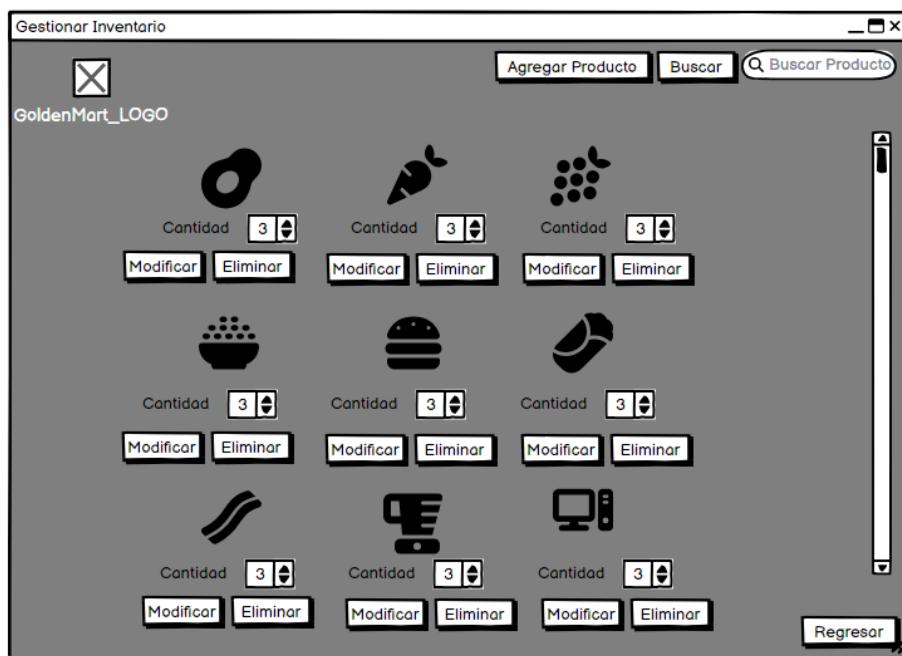
INTERFAZ LOGIN ADMINISTRADOR



INTERFAZ SESION ADMINISTRADOR

A screenshot of a Windows-style application window titled "Agregar Producto". It features a "GoldenMart_LOGO" header. The form includes fields for: Nombre del Producto (text input), Marca (text input), Contenido Neto (text input), Categoría (dropdown menu showing "3"), Precio (text input), Imagen (file input), Cantidad Disponible (text input), and Descripción (text input). At the bottom are "Agregar Producto", "Cancelar", and "Regresar" buttons. A blue callout bubble on the right says "Golden Mart" and "Producto agregado con éxito".

INTERFAZ REGISTRAR PRODUCTO(ADMINISTRADOR)



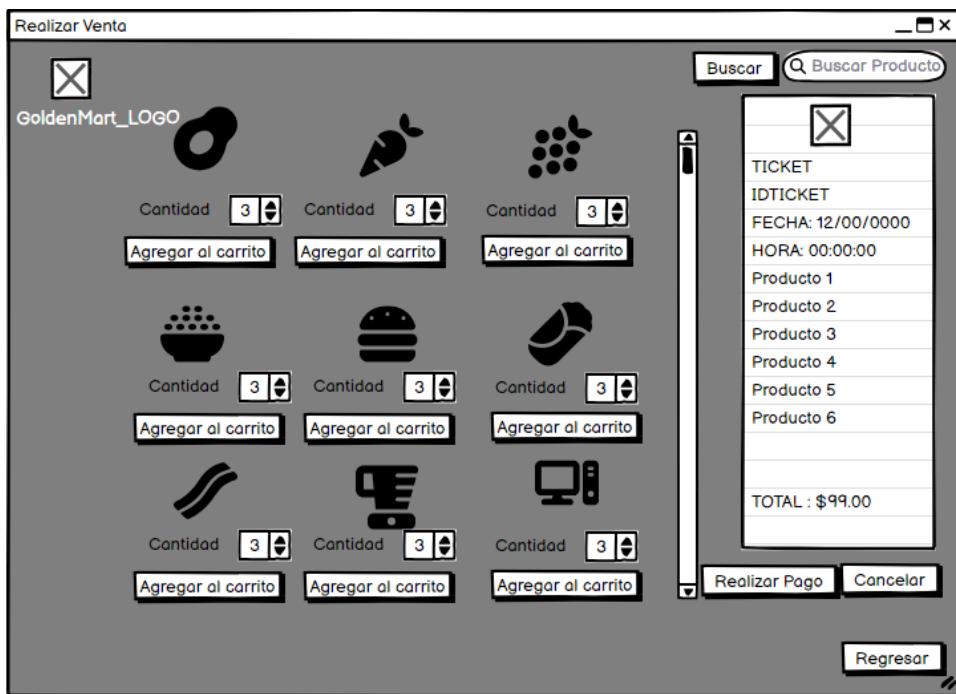
INTERFAZ GESTIONAR INVENTARIO(ADMINISTRADOR)

The screenshot shows a window titled 'Modificar Producto'. At the top right are buttons for 'Regresar' (Return) and 'Cancelar' (Cancel). The main area contains fields for product details:

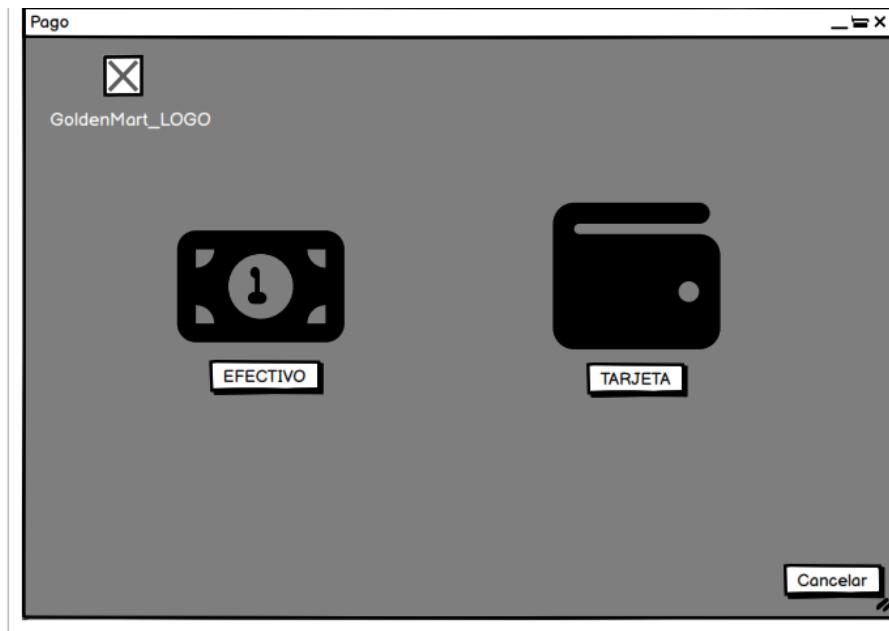
- Nombre del Producto: [Text input]
- Marca: [Text input]
- Contenido Neto: [Text input]
- Categoría: [Spinner input set to 3]
- Precio: [Text input]
- Imagen: [Text input]
- Cantidad Disponible: [Text input]
- Descripción: [Text input]

At the bottom are buttons for 'Modificar Producto' (Modify Product) and 'Cancelar' (Cancel).

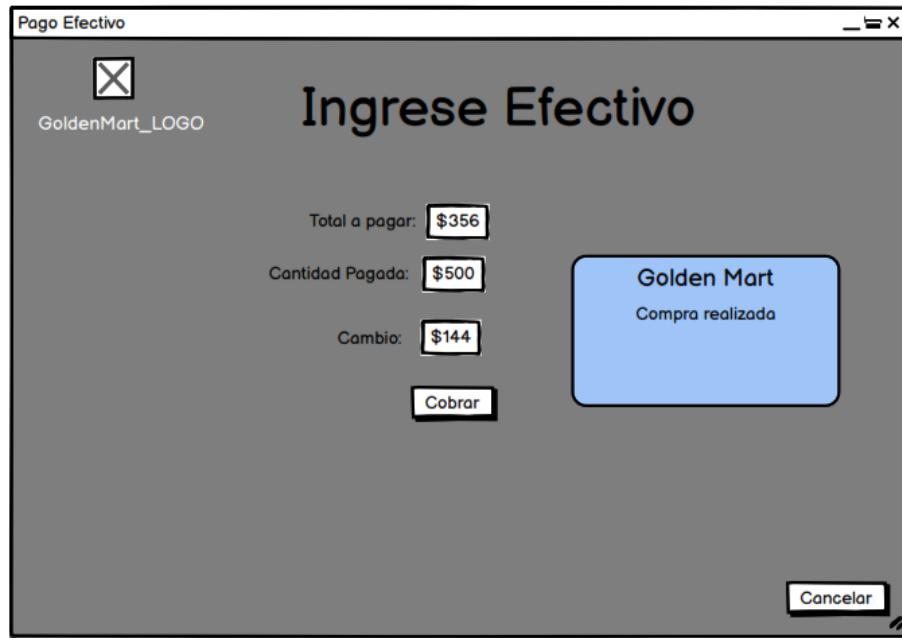
INTERFAZ MODIFICAR PRUDCTO(ADMINISTRADOR)



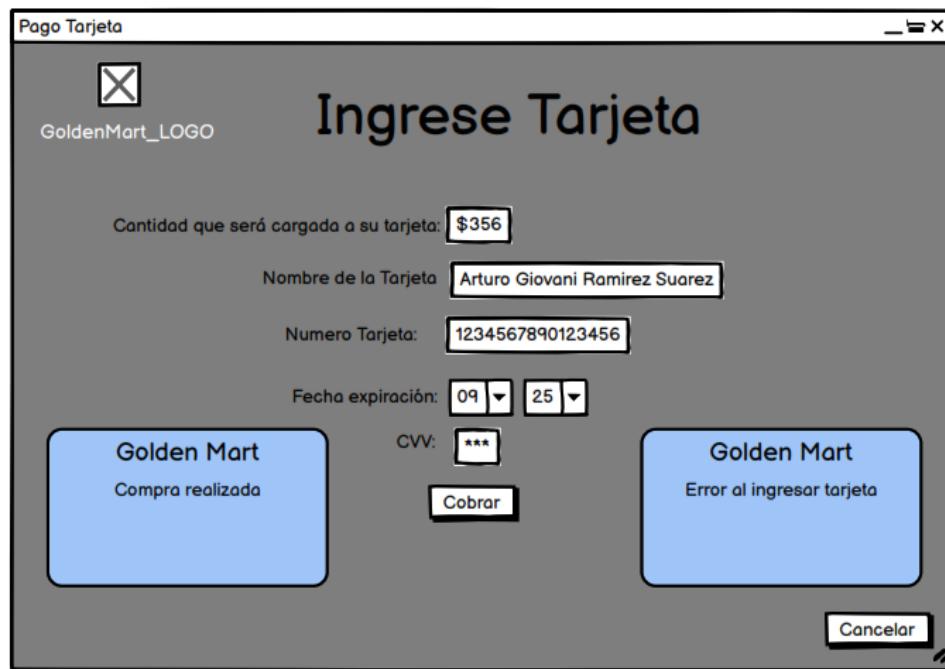
INTERFAZ REALIZAR VENTA(CAJERO)



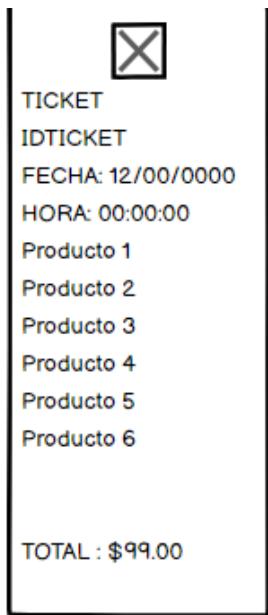
INTERFAZ GENERAR PAGO-(CAJERO)



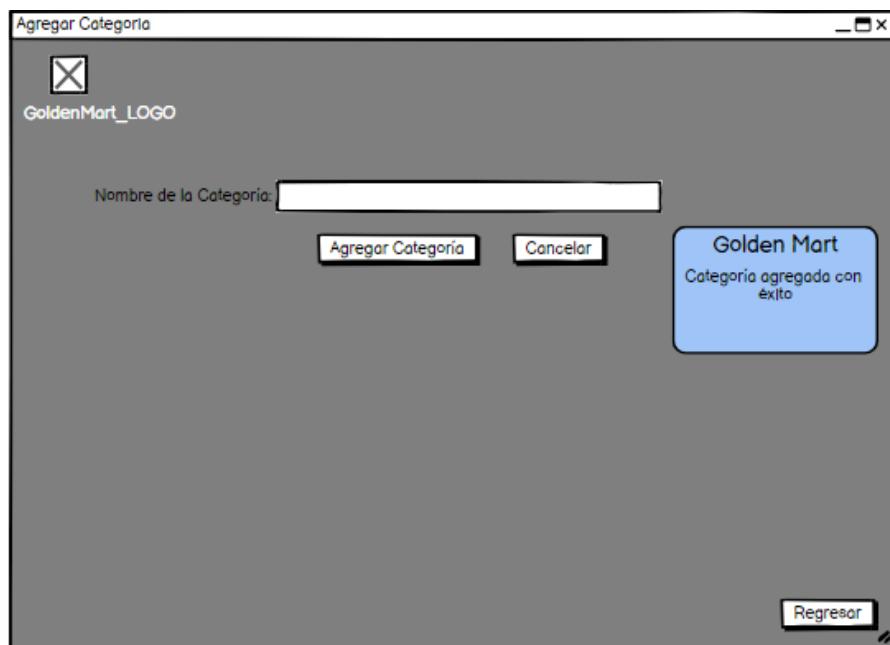
INTERFAZ PAGO-EFECTIVO(CAJERO)



INTERFAZ PAGO-TARJETA(CAJERO)

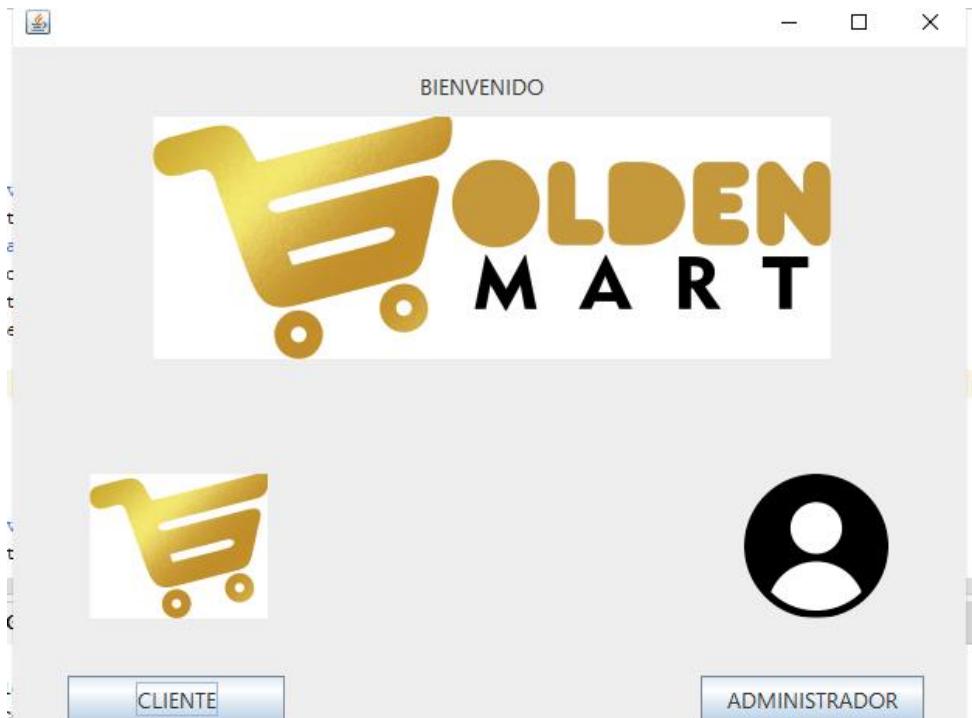


INTERFAZ TICKET(CAJERO)

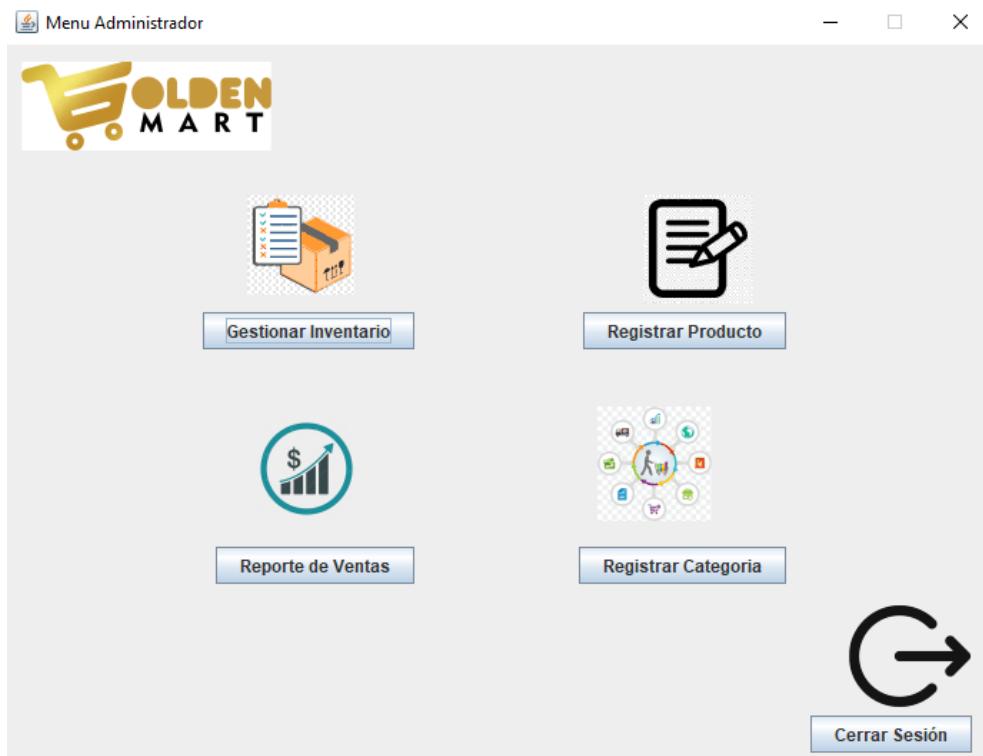


INTERFAZ REGISTRAR CATEGORIA(ADMINISTRADOR)

INTERFACES FINALES EN JAVA SWING



INTERFAZ BIENVENIDO



INTERFAZ SESION ADMINISTRADOR

Modificar Producto



Nombre del Producto: Carda desenredante

Marca: Dry Pet

Contenido Neto: 1 unidad

Categoría: Mascotas

Precio: 130.0

Imagen: /imagenes/card.png

Cantidad Disponible: 15

Descripción: illo para el cuidado del pelaje

Modificar Producto **Cancelar**

Regresar

INTERFAZ MODIFICAR PRODUCTO

Gestionar Inventario



Agregar Producto Agregar Categoría Buscar

Nombre	Marca	Contenido Neto	Categoría	Precio	Cantida...	Imagen	Modificar Producto	Eliminar
Leche Entera	Alpura	1L.	Lacteos	30.0	31	 CLASICA	Modificar Producto	Eliminar
Aceite	Nutriol	946 ml.	Abarrotes	48.5	12	 946 ml	Modificar Producto	Eliminar
Pan Blanco Mediano	Bimbo	460 g.	Panaderia	45.5	8	 BIMBO	Modificar Producto	Eliminar

Escribe aquí para buscar

Regresar domingo, 19 de mayo de 2024 10:05 p.m. 19/05/2024

INTERFAZ GESTIONAR INVENTARIO(ADMINISTRADOR)

Realizar Venta



Buscar

Nombre	Marca	Contenido Neto	Precio	Cantidad Disponible	Imagen	Agregar	Eliminar
Leche Entera	Alpura	1L.	30.0	100		Agregar al Carrito	Eliminar
Aceite	Nutrioli	946 ml.	48.5	50		Agregar al Carrito	Eliminar
Pan Blanco Mediano	Bimbo	460 g.	45.5	10		Agregar al Carrito	Eliminar

GOLDEN MART
 Fecha: 2024-04-23
 Hora: 17:08:54
 Productos:
 Leche Entera - Precio: \$30.0
 Aceite - Precio: \$48.5
 Pan Blanco Mediano - Precio: \$45.5
 Total: \$124.0

Realizar Pago Cancelar

Regresar

INTERFAZ REALIZAR VENTA(CAJERO)

Realizar Venta



Buscar

Nombre	Marca	Contenido Neto	Precio	Cantidad Disponible	Imagen	Agregar	Eliminar
Leche Entera	Alpura	1L.	30.0	100		Agregar al Carrito	Eliminar
ACEITE	Nutrioli	946 ml.	48.5	30		Venta cancelada.	Eliminar
Pan Blanco Mediano	Bimbo	460 g.	45.5	10		Agregar al Carrito	Eliminar

GOLDEN MART
 Fecha: 2024-04-29
 Hora: 02:39:21
 Productos:
 Leche Entera - Precio: \$30.0
 Total: \$30.0

Realizar Pago Cancelar

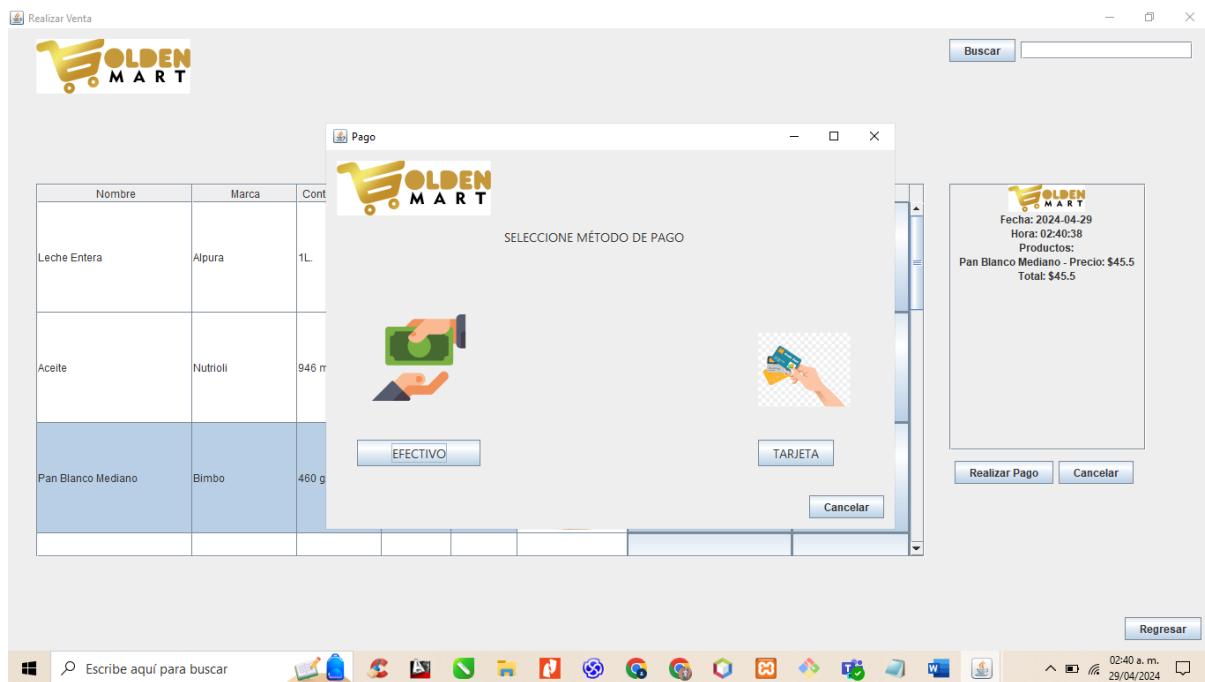
Regresar

Escribe aquí para buscar

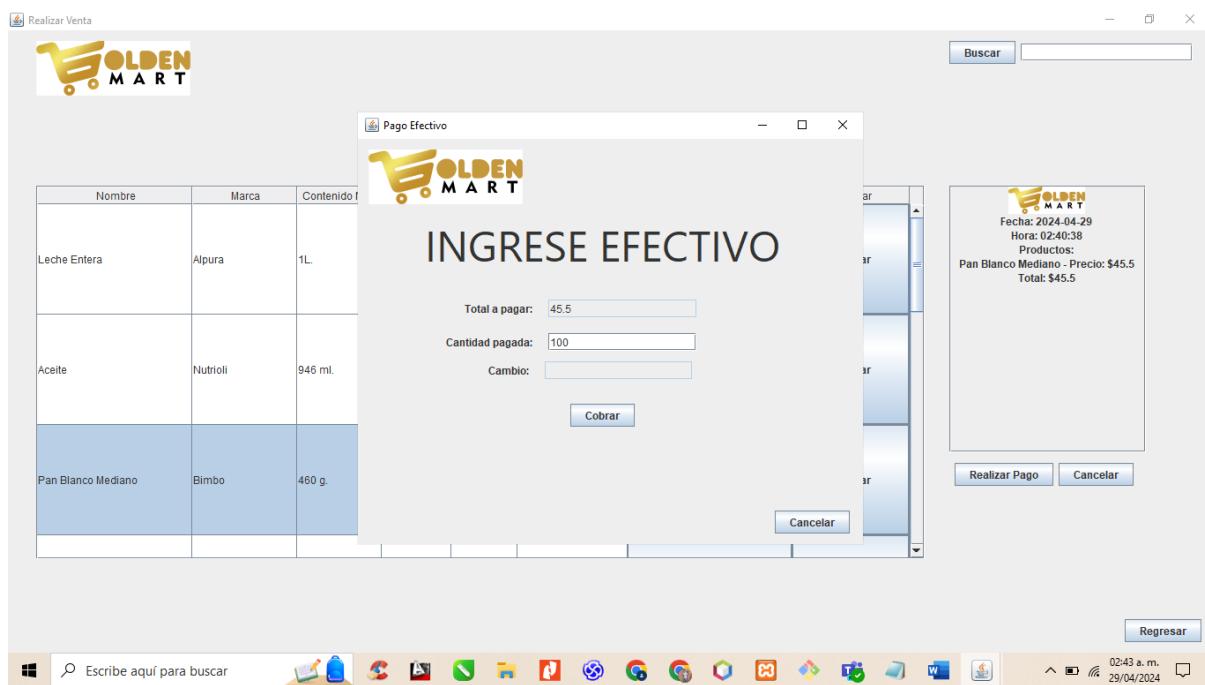


02:39 a. m.
29/04/2024

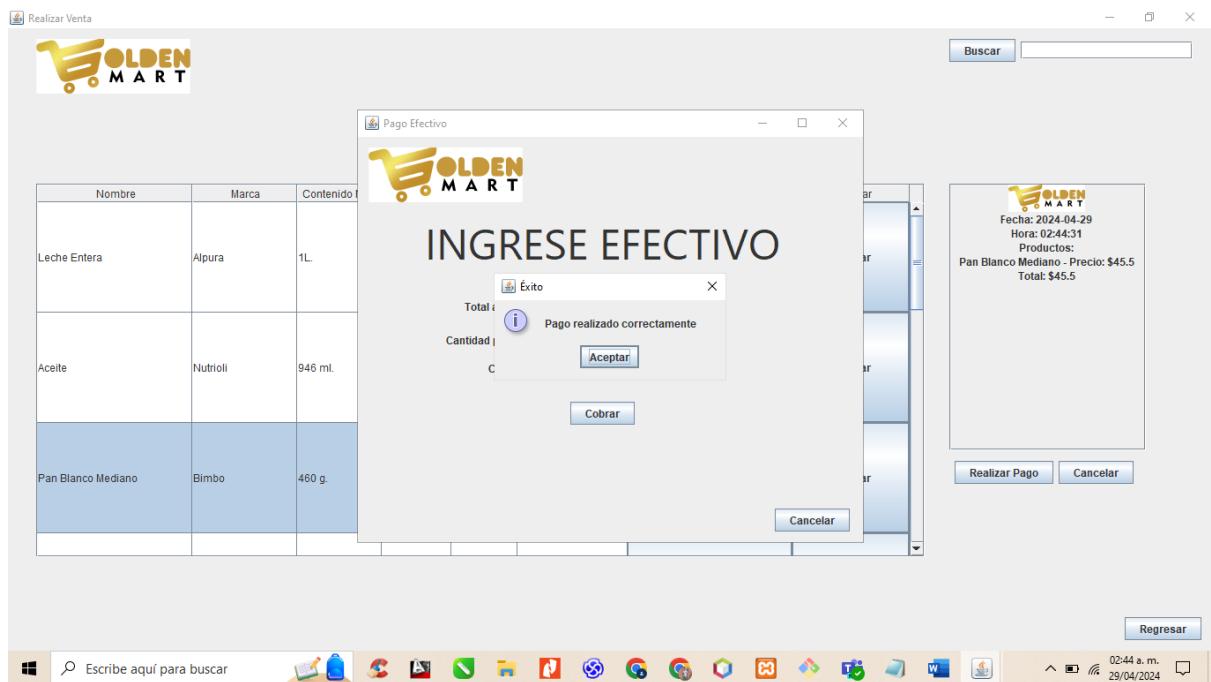
INTERFAZ REALIZAR VENTA-(CAJERO) VENTA CANCELADA



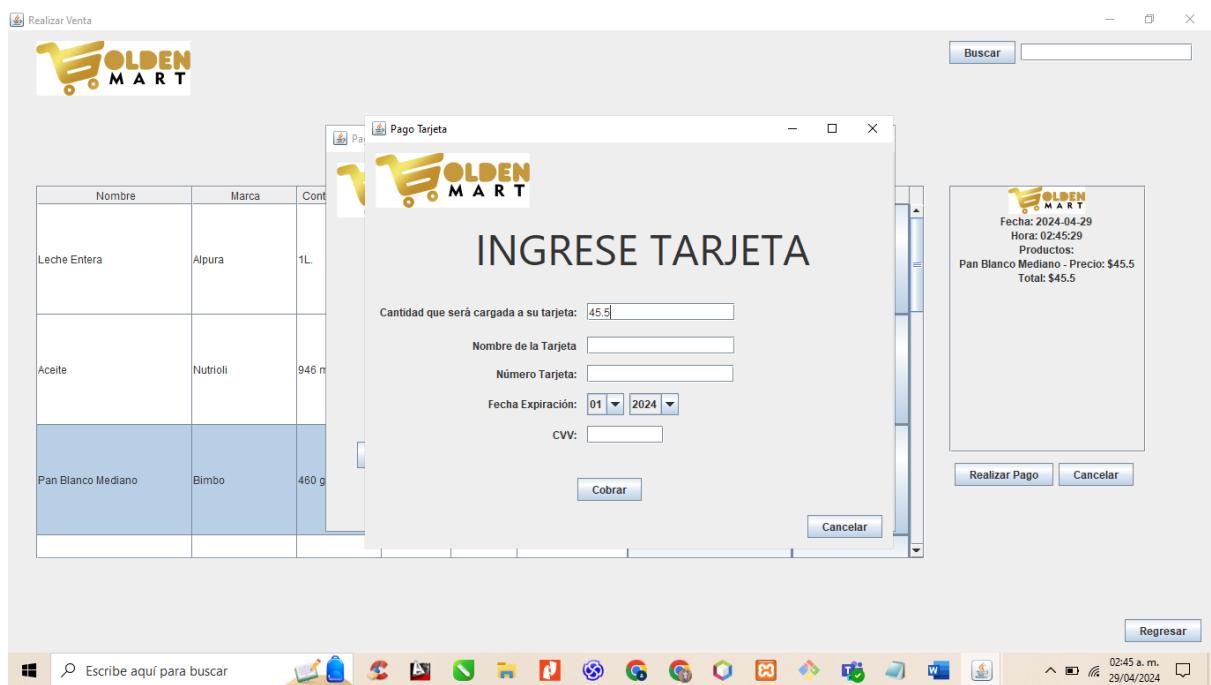
INTERFAZ GENERAR PAGO-(CAJERO)



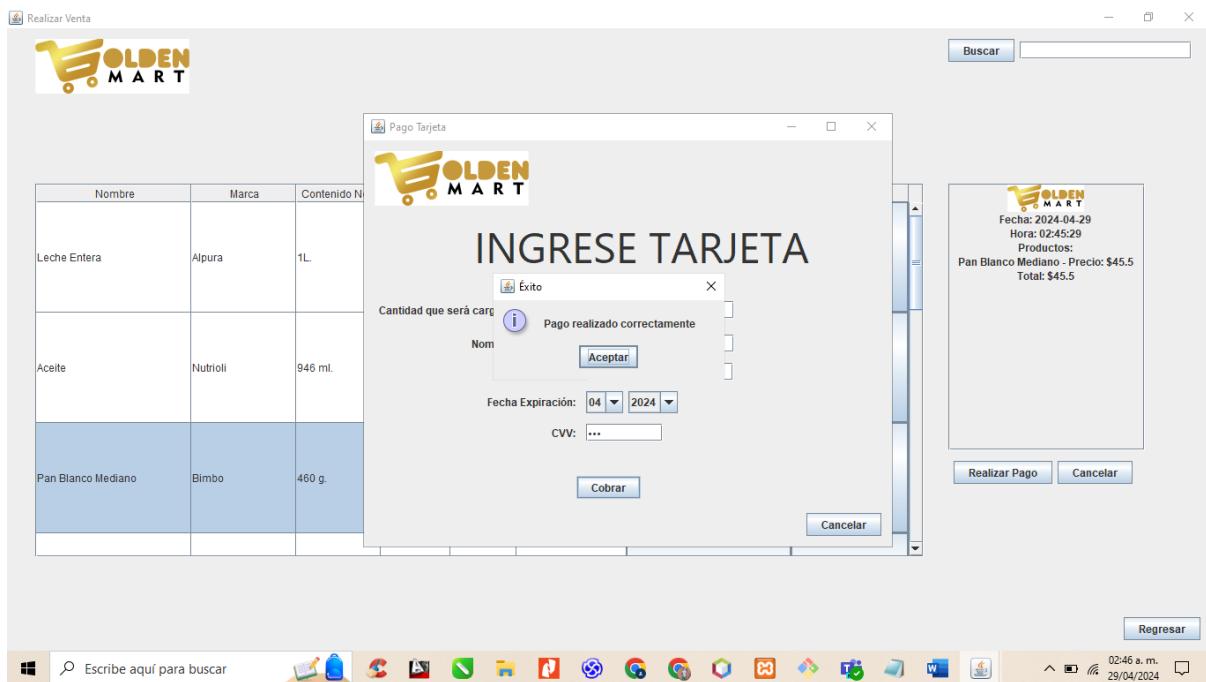
INTERFAZ PAGO-EFECTIVO(CAJERO)



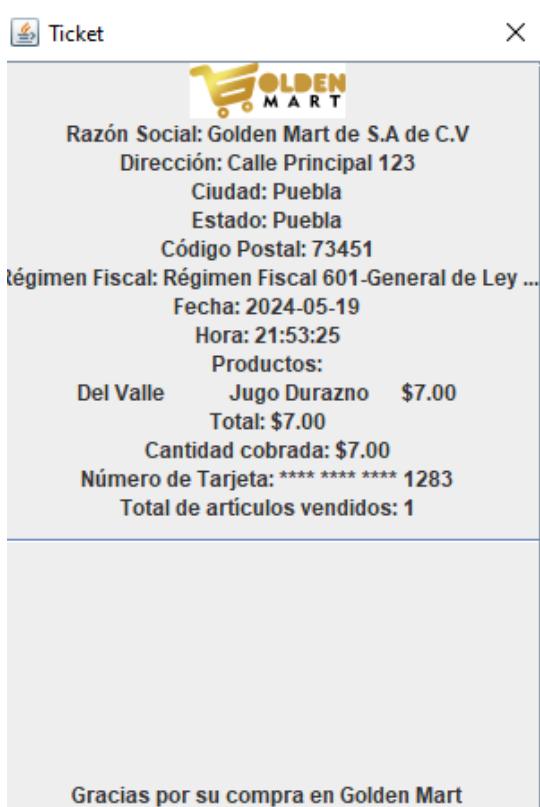
INTERFAZ PAGO-EFECTIVO(CAJERO) – MENSAJE ÉXITO



INTERFAZ PAGO-TARJETA(CAJERO)



INTERFAZ PAGO-TARJETA (CAJERO) – MENSAJE ÉXITO



INTERFAZ TICKET CLIENTE

Registrar Categoría

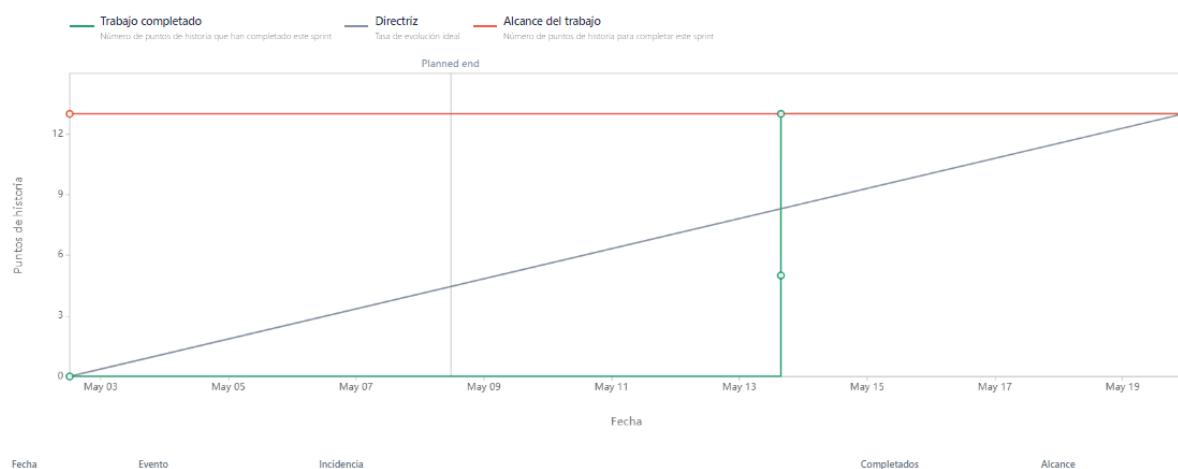


Nombre de la Categoría:

Agregar Categoría **Cancelar**

Regresar

INFORMES JIRA



INFORME DE TRABAJO COMPLETADO



GRAFICO DE TRABAJO PENDIENTE

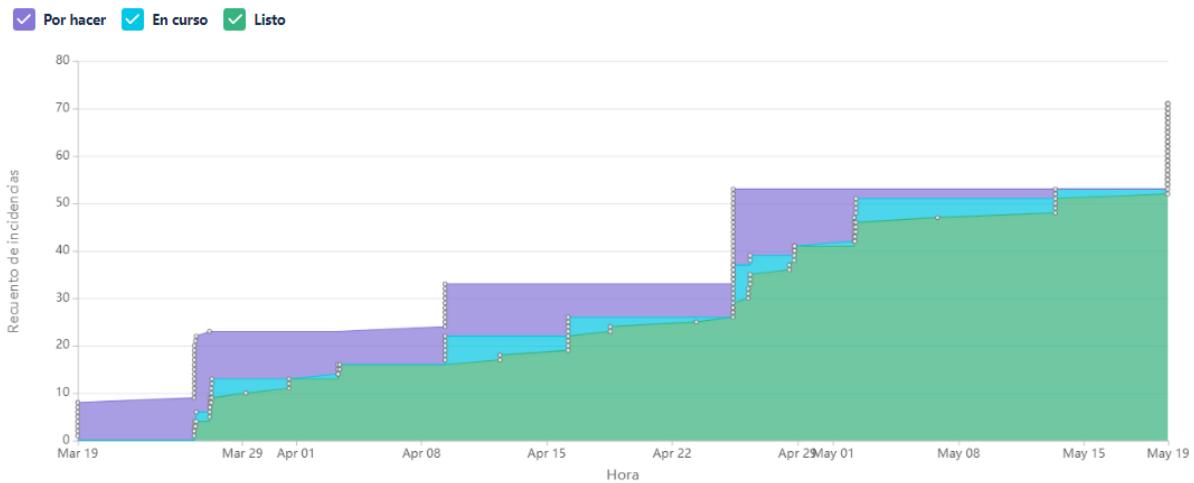
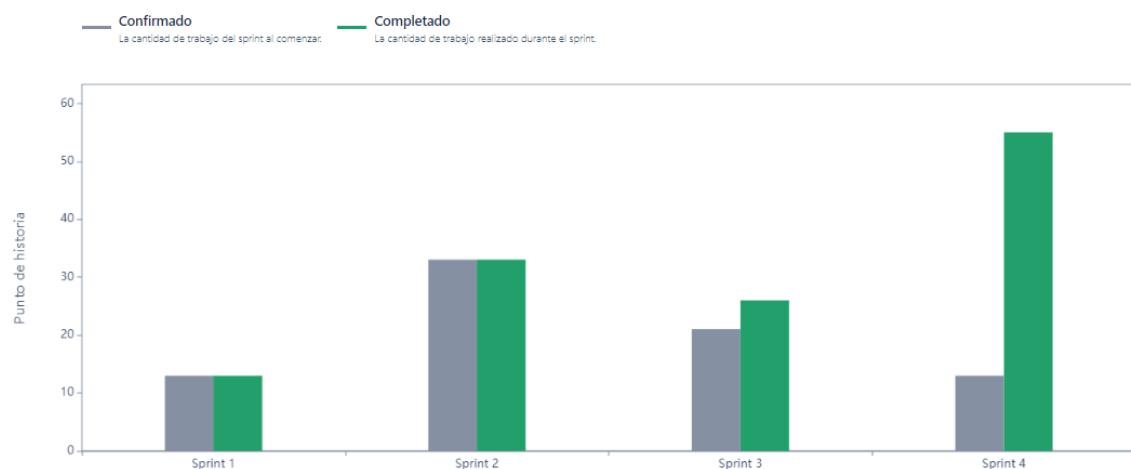


DIAGRAMA DE FLUJO ACUMULADO



INFORME DE VELOCIDAD

PRUEBAS

1er Sprint

GoldenMart - Apache NetBeans IDE 20

```

13 import java.sql.SQLException;
14 import java.sql.Statement;
15 import javax.swing.JOptionPane;
16
17
18 public class Producto {
19     private Connection con;
20     Conexion conexion = new Conexion();
21
22     private String nombre;
23     private float precio;
24     private String contenidoNeto;
25     private String categoria;
26     private String descripcion;
27     private String marca;
28     private int cantidadDisponible;
29     private String imagen;
30
31
32     public Producto(String nombre, String marca, String contenidoNeto, String categoria, float precio, String imagen, int cantidadDisponible) {
33         this.nombre = nombre;
34     }
35
36
37     public void setNombre(String nombre) {
38         this.nombre = nombre;
39     }
40
41     public void setPrecio(float precio) {
42         this.precio = precio;
43     }
44
45     public void setCantidadDisponible(int cantidadDisponible) {
46         this.cantidadDisponible = cantidadDisponible;
47     }
48
49     public void setCategoria(String categoria) {
50         this.categoria = categoria;
51     }
52
53     public void setDescripcion(String descripcion) {
54         this.descripcion = descripcion;
55     }
56
57     public void setMarca(String marca) {
58         this.marca = marca;
59     }
60
61     public void setImagen(String imagen) {
62         this.imagen = imagen;
63     }
64
65     public String getNombre() {
66         return nombre;
67     }
68
69     public float getPrecio() {
70         return precio;
71     }
72
73     public String getContenidoNeto() {
74         return contenidoNeto;
75     }
76
77     public String getCategoría() {
78         return categoria;
79     }
80
81     public String getDescripción() {
82         return descripción;
83     }
84
85     public String getMarca() {
86         return marca;
87     }
88
89     public int getCantidadDisponible() {
90         return cantidadDisponible;
91     }
92
93     public String getImagen() {
94         return imagen;
95     }
96
97 }

```

Test Results

Model Admin Test	Model Producto Test
Tests passed: 100.00 %	Tests passed: 100.00 %
model.AdminTest passed	model.AdminTest passed
testSetPassword passed (0.027 s)	testSetPassword passed (0.027 s)
testGetPassword passed (0.002 s)	testGetPassword passed (0.002 s)
testVerificarCredenciales passed (1.000 s)	testVerificarCredenciales passed (1.000 s)
testsetUsername passed (0.003 s)	testsetUsername passed (0.003 s)

PRUEBAS DEL ADMINISTRADOR

GoldenMart - Apache NetBeans IDE 20

```

13 import java.sql.SQLException;
14 import java.sql.Statement;
15 import javax.swing.JOptionPane;
16
17
18 public class Producto {
19     private Connection con;
20     Conexion conexion = new Conexion();
21
22     private String nombre;
23     private float precio;
24     private String contenidoNeto;
25     private String categoria;
26     private String descripcion;
27     private String marca;
28     private int cantidadDisponible;
29     private String imagen;
30
31
32     public Producto(String nombre, String marca, String contenidoNeto, String categoria, float precio, String imagen, int cantidadDisponible) {
33         this.nombre = nombre;
34     }
35
36
37     public void setNombre(String nombre) {
38         this.nombre = nombre;
39     }
40
41     public void setPrecio(float precio) {
42         this.precio = precio;
43     }
44
45     public void setCantidadDisponible(int cantidadDisponible) {
46         this.cantidadDisponible = cantidadDisponible;
47     }
48
49     public void setCategoria(String categoria) {
50         this.categoria = categoria;
51     }
52
53     public void setDescripcion(String descripcion) {
54         this.descripcion = descripcion;
55     }
56
57     public void setMarca(String marca) {
58         this.marca = marca;
59     }
60
61     public void setImagen(String imagen) {
62         this.imagen = imagen;
63     }
64
65     public String getNombre() {
66         return nombre;
67     }
68
69     public float getPrecio() {
70         return precio;
71     }
72
73     public String getContenidoNeto() {
74         return contenidoNeto;
75     }
76
77     public String getCategoría() {
78         return categoria;
79     }
80
81     public String getDescripción() {
82         return descripción;
83     }
84
85     public String getMarca() {
86         return marca;
87     }
88
89     public int getCantidadDisponible() {
90         return cantidadDisponible;
91     }
92
93     public String getImagen() {
94         return imagen;
95     }
96
97 }

```

Test Results

Model Admin Test	Model Producto Test
Tests passed: 100.00 %	Tests passed: 100.00 %
All 7 tests passed. (0.952 s)	model.ProductoTest passed running...
model.ProductoTest passed (0.066 s)	testInsertarProducto passed (0.066 s)
testSetCantidadDisponible passed (0.004 s)	testSetCantidadDisponible passed (0.004 s)
testSetNombre passed (0.004 s)	testSetNombre passed (0.004 s)

PRUEBAS DE LA CLASE PRODUCTO

2do Sprint

The screenshot shows the Apache NetBeans IDE 20 interface. The top menu includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and GoldenMart - Apache NetBeans IDE 20. The toolbar has icons for file operations like New, Open, Save, and Print. The Projects tab shows a hierarchy of Java files: Categoría.java, Conexión.java, DetalleVenta.java, Producto.java [-/M], Ticket.java, and Venta.java. Below these are view, Test Packages, controller, and model packages. The model package contains AdminTest.java, ConexiónTest.java, and ProductoTest.java [-/M]. The Files tab shows additional files like Test Packages, Libraries, and Test Libraries. The central workspace displays the source code for ProductoTest.java. The code is testing product modification, comparing original and new product details. The right side shows the Test Results window with the message "Tests passed: 100.00%" and a detailed breakdown of passed tests for modificarProducto.

```

String marcaOriginal = "La Lechera";
String contenidoNetoOriginal = "1 litro";
String categoriaOriginal = "Lácteos";
float precioOriginal = 1.5f;
String imagenOriginal = "Imagen.png";
int cantidadDisponibleOriginal = 10;
String descripcionOriginal = "Leche condensada";

// Nuevos datos para modificar el producto
String nombreNuevo = "Leche condensada";
String marcaNuevo = "Nestlé";
String contenidoNetoNuevo = "375 g.";
String categoriaNuevo = "Lácteos";
float precioNuevo = 1.5f;
String imagenNuevo = "/imagenes/lechera.png";
int cantidadDisponibleNuevo = 10;
String descripcionNuevo = "Por más de 85 años la leche condensada La Lechera ha sido el complemento perfecto p

```

PRUEBAS DEL CRUD-MODIFICAR PRODUCTO

This screenshot is similar to the previous one but focuses on the test for deleting a product. The central workspace shows the source code for ProductoTest.java, specifically the testEliminarProducto() method. It creates a new product, adds it to the database, and then attempts to delete it by ID. The test checks if the deleted product is null. The right side shows the Test Results window with the message "Tests passed: 100.00%" and a breakdown of passed tests for testEliminarProducto.

```

    assertEquals(imagenNuevo, productoModificado.getImagen());
    assertEquals(cantidadDisponibleNuevo, productoModificado.getCantidadDisponible());
    assertEquals(descripcionNuevo, productoModificado.getDescripcion());
}

@Test
public void testEliminarProducto() {
    System.out.println("testEliminarProducto");
    // Eliminamos el producto con ID 11
    try {
        Producto producto = new Producto();
        producto.eliminarProducto(11);
        // Verificamos que el producto se haya eliminado correctamente
        Producto productoEliminado = producto.obtenerProductoPorId(11);
        assertNull(productoEliminado);
    } catch (Exception e) {
        fail("Se lanzó una excepción: " + e.getMessage());
    }
}

```

PRUEBAS DEL CRUD-ELIMINAR PRODUCTO

The screenshot shows the Apache NetBeans IDE 20 interface. The left sidebar displays the project structure under 'Projects' and 'Services'. The main area shows the source code for `ProductoTest.java`. The code implements a test for searching products by name ('Leche condensada'). It creates a new `Producto` object, calls the `buscarProductos` method with the search term, prints the size of the resulting list, prints each product's name, and asserts that at least one product was found. The `Test Results` window at the bottom shows a single test named `testBuscarProductos` has passed.

```

180     @Test
181     public void testBuscarProductos() {
182         System.out.println("testBuscarProductos");
183         // Realizamos una búsqueda de productos con texto "Leche condensada"
184         try {
185             Producto producto = new Producto();
186             List<Producto> productosEncontrados = producto.buscarProductos("Leche condensada");
187
188             // Imprimimos el tamaño de la lista de productos encontrados
189             System.out.println("Número de productos encontrados: " + productosEncontrados.size());
190
191             // Imprimimos los nombres de los productos encontrados
192             for (Producto p : productosEncontrados) {
193                 System.out.println("Nombre del producto encontrado: " + p.getNombre());
194             }
195
196             // Verificamos que se haya encontrado al menos un producto
197             assertFalse(productosEncontrados.isEmpty());
198         } catch (Exception e) {
199             fail("Se lanzó una excepción: " + e.getMessage());
200         }
201     }
202 }
203
204     @Test
205     public void testAgregarVenta_ProductoDisponible() {
206         System.out.println("testAgregarVenta_ProductoDisponible");
207         Producto producto = new Producto();
208
209         // Verificamos la cantidad disponible inicial del producto
210         int cantidadInicial = producto.obtenerCantidadDisponibleInicial(1);

```

PRUEBAS DEL CRUD-BUSCAR PRODUCTO

The screenshot shows the Apache NetBeans IDE 20 interface. The left sidebar displays the project structure under 'Projects' and 'Services'. The main area shows the source code for `ProductoTest.java`. This code includes a test for adding a product to a sale ('venta'). It creates a new `Producto` object, adds it to a `Venta` object, and then checks if the initial quantity is still available. The `Test Results` window at the bottom shows two tests have passed: `testAgregarVenta_ProductoDisponible` and `testBuscarProductos`.

```

193     // Imprimimos los nombres de los productos encontrados
194     for (Producto p : productosEncontrados) {
195         System.out.println("Nombre del producto encontrado: " + p.getNombre());
196     }
197
198     // Verificamos que se haya encontrado al menos un producto
199     assertFalse(productosEncontrados.isEmpty());
200 } catch (Exception e) {
201     fail("Se lanzó una excepción: " + e.getMessage());
202 }
203
204     @Test
205     public void testAgregarVenta_ProductoDisponible() {
206         System.out.println("testAgregarVenta_ProductoDisponible");
207         Producto producto = new Producto();
208
209         // Verificamos la cantidad disponible inicial del producto
210         int cantidadInicial = producto.obtenerCantidadDisponibleInicial(1);

```

PRUEBAS DE LA CRUD-REALIZAR VENTA-AGREGAR PRODUCTO DISPONIBLE

The screenshot shows the Apache NetBeans IDE 20 interface. The code editor displays a Java test class named `model.ProductoTest`. The specific method shown is `testAgregarVenta_ProductoDisponible()`. The code performs several assertions to verify the state of the product in the database after a sale is added.

```

201     fail("Se lanzó una excepción: " + e.getMessage());
202 }
203 }
204 @Test
205 public void testAgregarVenta_ProductoDisponible() {
206     System.out.println("testAgregarVenta_ProductoDisponible");
207     Producto producto = new Producto();
208
209     // Verificamos la cantidad disponible inicial del producto
210     int cantidadInicial = producto.obtenerCantidadDisponibleInicial(11);
211
212     // Realizamos la venta del producto
213     JTextArea jTicket = new JTextArea();
214     List<Producto> productosVendidos = new ArrayList<>();
215     producto.agregarVenta(11, jTicket, productosVendidos);
216
217     // Verificamos que se haya reducido la cantidad disponible en la base de datos
218     int cantidadFinal = producto.obtenerCantidadDisponibleInicial(11);

```

The test results window shows all tests passed at 100.00%.

PRUEBAS DE LA CRUD-REALIZAR VENTA-AGREGAR PRODUCTO AGOTADO

The screenshot shows the Apache NetBeans IDE 20 interface. The code editor displays a Java test class named `model.ProductoTest`. The specific method shown is `testAgregarVenta_ProductoAgotado()`. The code checks if attempting to add a sold-out product to the sales list fails correctly.

```

217     // Verificamos que se haya reducido la cantidad disponible en la base de datos
218     int cantidadFinal = producto.obtenerCantidadDisponibleInicial(11);
219     assertEquals(cantidadInicial - 1, cantidadFinal);
220
221     // Verificamos que el producto se haya agregado a la lista de productos vendidos
222     assertEquals(1, productosVendidos.size());
223 }
224
225 @Test
226 public void testAgregarVenta_ProductoAgotado() {
227     System.out.println("testAgregarVenta_ProductoAgotado");
228     Producto producto = new Producto();
229
230     // Agotamos el producto
231     producto.actualizarCantidadDisponibleEnBD(11, 0);
232
233     // Intentamos agregar un producto agotado
234     JTextArea jTicket = new JTextArea();

```

The test results window shows all tests passed at 100.00%.

PRUEBAS DE LA CRUD-REALIZAR VENTA-AGREGAR PRODUCTO NO DISPONIBLE

GoldenMart - Apache NetBeans IDE 20

```

    // Verificamos que no se haya reducido la cantidad disponible en la base de datos
    assertEquals(0, cantidadFinal);

    // Verificamos que la lista de productos vendidos esté vacía
    assertTrue(productosVendidos.isEmpty());
}

@Test
public void testAgregarVenta_ProductoNoDisponible() {
    System.out.println("testAgregarVenta_ProductoNoDisponible");
    Producto producto = new Producto();

    // Eliminamos el producto
    producto.eliminarProducto(11);

    // Intentamos agregar un producto no disponible
    JTextArea jTicket = new JTextArea();
    List<Producto> productosVendidos = new ArrayList<>();
}

```

Test Results x Output - GoldenMart (test)

model.ProductoTest.testAgregarVenta_ProductoDisponible x model.ProductoTest.testAgregarVenta_ProductoAgotado x model.ProductoTest.testAgregarVenta_ProductoNoDisponible x

Tests passed: 100.00 %

The test passed. (11.835 s)

model.ProductoTest passed

testAgregarVenta_ProductoNoDisponible passed (11.254 s)

PRUEBAS DE LA CRUD-REALIZAR VENTA-ELIMINAR PRODUCTO DE LA VENTA

GoldenMart - Apache NetBeans IDE 20

```

    public void testAgregarVenta_ProductoAgotado() {...19 lines...}

    @Test
    public void testAgregarVenta_ProductoNoDisponible() {...19 lines...}

    @Test
    public void testEliminarVenta_ProductoEnLista() {
        System.out.println("testEliminarVenta_ProductoEnLista");

        // Creamos una instancia de Producto para simular la venta
        Producto producto = new Producto();
        producto.setIdProducto(11); // ID del producto a eliminar
        producto.setNombre("Leche condensada");
        producto.setPrecio(26);
        producto.setCantidadDisponible(40); // Cantidad inicial del producto

        // Creamos una lista de productos vendidos y agregamos el producto
        List<Producto> productosVendidos = new ArrayList<>();
    }

```

Test Results x Output - GoldenMart (test)

model.ProductoTest.testAgregarVenta_ProductoAgotado x model.ProductoTest.testAgregarVenta_ProductoNoDisponible x model.ProductoTest.testEliminarVenta_ProductoEnLista x

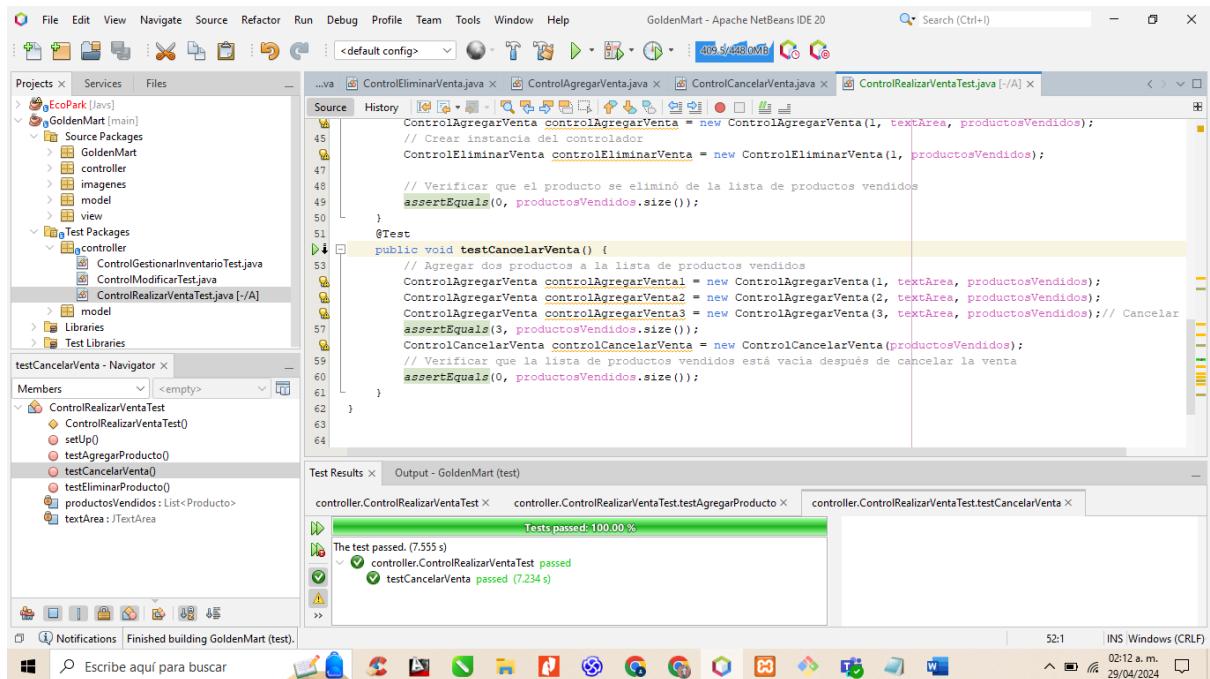
Tests passed: 100.00 %

The test passed. (1.458 s)

model.ProductoTest passed

testEliminarVenta_ProductoEnLista passed (1.082 s)

3er Sprint

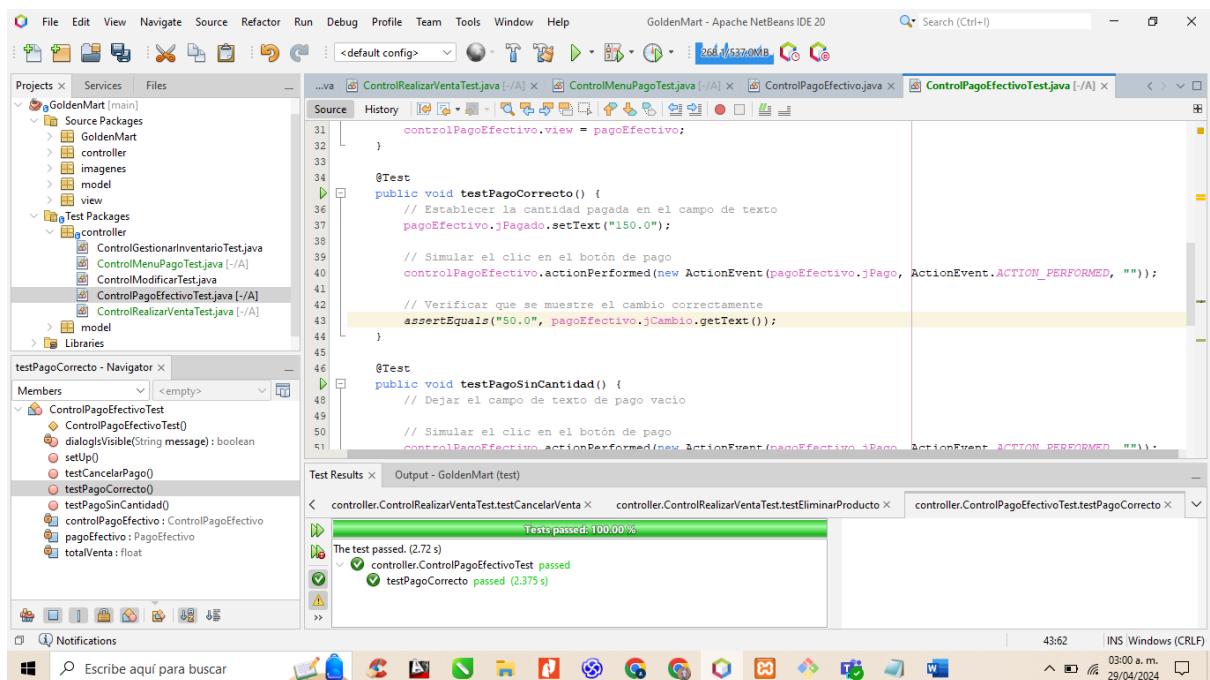


The screenshot shows the Apache NetBeans IDE 20 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and GoldenMart - Apache NetBeans IDE 20. The left sidebar displays the Projects view with 'EcoPark [Java]' and 'GoldenMart [main]' selected. Under 'GoldenMart [main]', there are Source Packages (GoldenMart, controller, imagenes, model, view), Test Packages (controller, model, Libraries, Test Libraries), and a test class 'ControlRealizarVentaTest.java'. The main workspace shows the source code for 'ControlRealizarVentaTest.java'. The code contains several test methods, including @Test annotations and assertions like assertEquals. The bottom status bar shows the date and time as 29/04/2024, 02:12 a.m.

```
public void testRealizarVenta() {
    // Crear instancia del controlador
    ControlAgregarVenta controlAgregarVenta = new ControlAgregarVenta(l, textArea, productosVendidos);
    // Agregar dos productos a la lista de productos vendidos
    ControlAgregarVenta controlAgregarVenta1 = new ControlAgregarVenta(l, textArea, productosVendidos);
    ControlAgregarVenta controlAgregarVenta2 = new ControlAgregarVenta(l, textArea, productosVendidos);
    ControlAgregarVenta controlAgregarVenta3 = new ControlAgregarVenta(l, textArea, productosVendidos);
    // Verificar que el producto se eliminó de la lista de productos vendidos
    assertEquals(0, productosVendidos.size());
}

@Test
public void testCancelarVenta() {
    // Agregar dos productos a la lista de productos vendidos
    ControlAgregarVenta controlAgregarVenta1 = new ControlAgregarVenta(l, textArea, productosVendidos);
    ControlAgregarVenta controlAgregarVenta2 = new ControlAgregarVenta(l, textArea, productosVendidos);
    ControlAgregarVenta controlAgregarVenta3 = new ControlAgregarVenta(l, textArea, productosVendidos);
    // Cancelar la venta
    ControlCancelarVenta controlCancelarVenta = new ControlCancelarVenta(productosVendidos);
    // Verificar que la lista de productos vendidos está vacía después de cancelar la venta
    assertEquals(0, productosVendidos.size());
}
```

PRUEBAS DE LA CRUD-REALIZAR VENTA-CANCELAR VENTA



The screenshot shows the Apache NetBeans IDE 20 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and GoldenMart - Apache NetBeans IDE 20. The left sidebar displays the Projects view with 'GoldenMart [main]' selected. Under 'GoldenMart [main]', there are Source Packages (GoldenMart, controller, imagenes, model, view), Test Packages (controller, model, Libraries, Test Libraries), and a test class 'ControlPagoEfectivoTest.java'. The main workspace shows the source code for 'ControlPagoEfectivoTest.java'. The code contains several test methods, including @Test annotations and assertions like assertEquals. The bottom status bar shows the date and time as 29/04/2024, 03:00 a.m.

```
public void testPagoCorrecto() {
    // Establecer la cantidad pagada en el campo de texto
    pagoEfectivo.jPagado.setText("150.0");

    // Simular el clic en el botón de pago
    controlPagoEfectivo.actionPerformed(new ActionEvent(pagoEfectivo.jPago, ActionEvent.ACTION_PERFORMED, ""));

    // Verificar que se muestre el cambio correctamente
    assertEquals("50.0", pagoEfectivo.jCambio.getText());
}

@Test
public void testPagoSinCantidad() {
    // Dejar el campo de texto de pago vacío
    pagoEfectivo.jPagado.setText("");

    // Simular el clic en el botón de pago
    controlPagoEfectivo.actionPerformed(new ActionEvent(pagoEfectivo.jPago, ActionEvent.ACTION_PERFORMED, ""));
}
```

PRUEBAS VERIFICAR PAGO EFECTIVO

```

public void testInsertarVenta() {
    // Simular una venta con una fecha, hora y total específicos
    LocalDate fechaVenta = LocalDate.now();
    LocalTime horaVenta = LocalTime.now();
    float total = 100.0f; // Total de la venta

    // Insertar la venta en la base de datos y obtener el ID de la venta
    int idVenta = controlRegistrarVenta.insertarVenta(fechaVenta, horaVenta, total);

    // Verificar que el ID de la venta devuelto sea válido
    assertTrue(idVenta > 0); // El ID de la venta debe ser un número positivo
}

```

Test Results

controller.ControlRegistrarVentaTest.testInsertarVenta

- The test passed. (1.44 s)
- >> controller.ControlRegistrarVentaTest passed
 - testInsertarVenta passed (1.061 s)

PRUEBAS INSERTAR VENTA EN BD

```

public void testInsertarVentaYCrearDetalleVenta() {
    // Simular una venta con una fecha, hora y total específicos
    LocalDate fechaVenta = LocalDate.now();
    LocalTime horaVenta = LocalTime.now();
    float total = 230.30f; // Total de la venta

    // Crear instancia del controlador para agregar productos
    ControlAregarVenta controlAregarVenta = new ControlAregarVenta(1, textArea, productosVendidos);
    // Agregar algunos productos a la lista de productos vendidos
    // Por ejemplo, podríamos agregar los productos con IDs 1 y 2
    productosVendidos.add(new Producto(1));
    productosVendidos.add(new Producto(2));

    // Insertar la venta en la base de datos y obtener el ID de la venta
    // Luego, crear instancia del controlador para registrar la venta
    ControlRegistrarVenta controlRegistrarVenta = new ControlRegistrarVenta(null); // Puedes pasar null para controlRealizarVenta
    int idVenta = controlRegistrarVenta.insertarVenta(fechaVenta, horaVenta, total);

    // Verificar que el ID de la venta devuelto sea válido
    assertTrue(idVenta > 0); // El ID de la venta debe ser un número positivo
}

```

Test Results

controller.ControlRegistrarVentaTest.testInsertarVenta

- The test passed. (4.701 s)
- >> controller.ControlDetalleVentaTest passed
 - testInsertarVentaYCrearDetalleVenta passed (4.351 s)

PRUEBAS INSERTAR DETALLES EN LA VENTA EN BD