

Actividad | 2 | Diagramas de Paradigma

Orientado a Objetos.

Análisis y Diseño de Sistemas

**Ingeniería en Desarrollo de
Software**



TUTOR: Eduardo Israel Castillo García.

ALUMNO: Sarahi Jaqueline Gómez Juárez, sara_2mil@outlook.com

FECHA: miercoles, 02 de noviembre de 2023.

Índice

Introducción	3
Descripción	7
<i>Elementos y símbolos en un Diagrama de Clases UML.....</i>	<i>11</i>
<i>Elementos y símbolos en un Diagrama de Casos de Uso UML.....</i>	<i>14</i>
Desarrollo.....	15
Contextualización:	15
<i>Diagrama de Clases</i>	<i>16</i>
Descripción del Diagrama del Diagrama de Clases:.....	17
<i>Diagrama de Casos de Uso: Sistema SISTEMA DE COMPRA-VENTA DE</i>	
<i>ROPA ONLINE (VENTAS A CLIENTES).....</i>	<i>29</i>
Descripción del Diagrama de Casos de Uso: Sistema SISTEMA DE COMPRA-	
VENTA DE ROPA ONLINE (VENTAS A CLIENTES)	30
<i>Diagrama de Casos de Uso: SISTEMA DE COMPRA-VENTA DE ROPA</i>	
<i>ONLINE (COMPRAS A PROVEEDOR)</i>	<i>32</i>
Descripción del Diagrama de Diagrama de Casos de Uso: SISTEMA DE COMPRA-	
VENTA DE ROPA ONLINE (COMPRAS A PROVEEDOR).....	33
Referencias:	35

Introducción

En el presente proyecto podrás visualizar dos tipos de Diagramas de Paradigmas Orientados a Objetos, recordemos que los Diagramas Orientados a Objetos: son herramientas gráficas utilizadas en el desarrollo de software y la ingeniería de software para representar visualmente la estructura y las interacciones de los objetos en un sistema orientado a objetos, estos diagramas son parte del Lenguaje de Modelado Unificado sus siglas en inglés son UML , qué es un estándar ampliamente aceptado en la industria del software para la representación de sistemas y aplicaciones.

Un diagrama de objetos ayuda a tener un panorama general con un alto nivel del sistema del que se esté tratando, ya que este se enfoca en los atributos, clases, operaciones o métodos de un conjunto de objetos y como estos objetos se relacionan entre sí, permitiéndonos representar una instantánea de un sistema en un momento específico.

UML Se ha convertido en el estándar de facto de la industria, este ha sido concebido por diferentes autores, entre ellos resaltan los siguientes:

Grady Booch, Ivar Jacobson, Jim Rumbaugh, Yordon- Coad, Martín, Shaer&Mellor, Rumbaugh, debido a qué son los métodos más usados de orientación a objetos.

El objetivo principal cuando se empezó a gestar UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado, por ello era necesario definir una anotación y semántica común, cabe mencionar que UML no define un proceso de desarrollo específico tan sólo se trata de una anotación.

Un proyecto de software orientado a objetos se compone de las siguientes etapas:

Análisis Orientado a Objetos (AOO):

Es la primera etapa en el proceso de desarrollo de software, durante esta etapa los analistas de

sistemas y desarrolladores de software se centrarán en comprender el dominio del problema y los requisitos del sistema, para ello utilizan conceptos de la Programación Orientada a Objetos para modelar las entidades, las propiedades y relaciones entre ellas, creando así Diagramas de Clases. Diagramas de Casos De Usos y otro tipo de modelos, que ayudan a representar el sistema de una manera más abstracta, con el objetivo de definir lo que se necesita y cómo se relacionan los objetos antes de llegar al diseño y la implementación.

Diseño Orientado a Objetos (DOO):

Es la segunda etapa donde los diseñadores de software toman los resultados del análisis y los utilizan para crear un diseño detallado del sistema, identificando las clases, métodos y atributos necesarios, estableciendo las relaciones entre las clases.

De igual manera se define la estructura general del sistema, la arquitectura y como se realizará la implementación, el Diseño Orientado a Objetos es fundamental para planificar cómo se considerará el software, asegurando así que cumpla con los requisitos identificados durante el análisis.

Programación Orientada a Objetos (POO):

Es la fase final en el proceso del desarrollo de dicho software, durante esta etapa los Desarrolladores escriben el código real del software, utilizando lenguajes de programación orientada a objetos, como pueden ser: Java, C ++, Python, entre otros. Los conceptos y modelos definidos durante el AOO y el DOO se traducen en clases y objetos reales en el código, ya que la Programación Orientada a Objetos se centra en la implementación de las funcionalidades del software, utilizándolos principios de Programación Orientada a Objetos como son la encapsulación, la herencia y el polimorfismo, para organizar y estructurar el código de manera eficiente y mantenible.

En resumen, el AOO se enfoca en comprender los requisitos del sistema, el DOO se enfoca en diseñar la estructura y la arquitectura del sistema, y la POO se centra en la implementación del software utilizando los conceptos de la programación

Aclarado este punto los diagramas de paradigma orientado a objetos son representaciones gráficas que se utilizan en el campo de la programación, y el diseño de software para visualizar y comunicar la estructura y el comportamiento de un sistema de un software desde una perspectiva orientada a objetos, estos diagramas son parte integral de una metodología de desarrollo de software orientada a objetos y se utilizan para modelar y diseñar sistemas de software, utilizando conceptos de programación orientada a objetos.

Algunos de los diagramas más utilizados en el paradigma orientado a objetos son:

Diagrama de Clases:

Este tipo de diagrama muestra las clases en el sistema, sus atributos, métodos y las relaciones entre ellas, es una representación estática de la estructura del sistema.

Diagrama de Objetos:

Como ya previamente se había descrito: es de representa instancias específicas de clases en un momento dado y muestra cómo interactúan entre sí en un escenario particular.

Diagrama de Secuencia:

Ilustra la interacción entre ambos objetos a lo largo del tiempo, mostrando la secuencia de mensajes intercambiados entre ellos.

Diagrama de Colaboración:

Este es similar al diagrama de secuencia, pero la diferencia se centra en la colaboración entre objetos y cómo se comunican entre sí.

Diagrama de Estados:

Describe el comportamiento de un objeto a lo largo de sus diferentes estados y como cambia de uno a otro en respuesta a eventos.

Diagrama de Actividad:

Representa el flujo de control en un proceso o algoritmo, mostrando actividades, decisiones y bifurcaciones.

Diagrama de Componentes:

Muestra los componentes de software y sus interacciones, lo que es útil en el diseño de sistemas de software a nivel de componentes.

Diagrama de Despliegue:

Ilustra la configuración de hardware y software en un entorno de despliegue, mostrando cómo los componentes de software se ejecutan en servidores y se conectan entre sí.

Cabe mencionar que solamente interactuaremos con 2 tipos de diagramas dentro de este proyecto, los cuáles son: el diagrama de clases y el diagrama de casos de uso.

Independientemente del tipo de diagrama que se utilice, el propósito de estos: es que los desarrolladores y diseñadores comprendan mejor la estructura y el comportamiento de un sistema de software, facilitando así la comunicación en la colaboración en los equipos de desarrollo, siendo así herramientas útiles para documentar sistemas de software, lo que hace que sea más fácil mantener y mejorar el software a largo tiempo.

Descripción

En el presente documento podrás visualizar la ejecución de un Diagrama de Clases y un Diagrama De Casos De Uso, estos son dos tipos de Diagramas de Paradigma Orientado a Objetos, estos estarán enfocados a la problemática de la contextualización; en la cual nos describe que una empresa de ropa desea tener un sistema que les permita una mejor organización, para poder diferenciarse de la competencia y de la misma manera les consienta ser más competitiva, por ello se identificaran los componentes externos e internos que lo rodean, se aplicará los diagramas para involucrar situaciones que presentan los actores convirtiendo la información en diagrama de clases y de grama de caso de uso, en donde los actores serán: los empleados, los clientes, los proveedores y las sucursales, para qué él cliente pueda acceder la página ya debió haberse registrado previamente dentro del Sistema de Compra-Venta de Ropa Online, en el cual va a poder interactuar con las diferentes clases y procesos que existen dentro del mismo sistema de esta manera podrá realizar adquirir productos en línea y ver sus compras, mediante su nombre de usuario y contraseña propiamente registrado.

Por ello es importante ya a ver identificado la asociación que conlleva estos procesos para brindarle un servicio de calidad a nuestro cliente y de esta manera tener un control sobre las ventas, los productos y el abastecimiento del inventario, junto con la parte proporcional que le corresponde a cada actor dentro de este.

Diagrama de Clases: Es una herramienta de modelado en UML, que se utiliza para representar la estructura estática de un sistema o aplicación, en éste se muestran las clases sus

atributos métodos y las relaciones que existen entre ellas,

Elementos y Símbolos en el Diagrama De Clases UML:

Clase:

Esto se representa por medio de un rectángulo dividido en 3 compartimientos:

El compartimiento superior es para el nombre de la clase.

El compartimiento intermedio es para los atributos de la clase

El compartimiento inferior es para las operaciones o métodos de la clase

Atributo:

Representa una propiedad o característica de una clase.

Se muestran en el compartimiento de atributos de la clase y se presenta con el siguiente formato:

[visibilidad] nombre: tipo de dato= valor por defecto

Cada tributo tiene un nombre un tipo de dato y una visibilidad.

La visibilidad:

Se utiliza para indicar el nivel de acceso a los atributos y métodos de una clase.

Los símbolos de visibilidad incluyen:

+ Público: Los atributos o métodos son accesibles desde cualquier parte del sistema.

- Privado: Los atributos o métodos son accesibles dentro de la propia clase.

Protegido: Los atributos o métodos son accesibles por las subclases.

/ Derivado: Indica que un atributo es calculado a partir de otros atributos.

~ Paquete: Los atributos son métodos son accesibles sólo dentro del paquete.

Las operaciones o métodos:

Representan las operaciones o funciones que una clase puede realizar y se muestra en el

compartimiento de métodos. Cada método tiene un nombre una lista de parámetros un tipo de retorno una visibilidad.

Se representa con el siguiente formato:

[visibilidad] nombre (parámetro): tipo_retorno

La visibilidad se representa de la misma manera que en los atributos.

Interacciones:

Representan los diferentes tipos de relaciones entre las clases los cuales son:

Agregación:

Representa una relación parte-todo entre una clase(compuesta) y una clase que contiene(compuesta), las partes pueden existir independientemente de la clase compuesta. Se representa con un rombo blanco en el extremo de la línea de la relación.

Composición:

Representa una relación de todo-parte, en donde una clase(compuesta) está formado por otras clases(partes).

Se representa con un rombo negro en un extremo de la línea de relación.

Herencia:

Representa una relación de herencia entre clases, donde una clase hereda atributos y operaciones de otra clase, se dibuja con una línea sólida y una flecha desde la clase derivada hacia una clase base.

Dependencia:

Representa una relación en la que una clase depende de otra clase, es decir, indica que una clase depende de otra clase en algún aspecto cómo la llamada a métodos.

Se representa con una línea punteada y una flecha desde la clase que depende hacia la

clase que la que depende.

Asociación:

Representa una relación entre 2 clases que no implica una jerarquía, se dibuja con una línea sólida que conecta las clases y puede tener etiquetas que indican la multiplicidad de la relación.

Multiplicidad:

Indica el número de elementos relacionados entre dos clases en una asociación, representando un rango numérico en la línea de asociación, es decir la multiplicidad se utiliza para indicar el número de elementos relacionados entre las clases, puede representar un conjunto de relaciones o asociaciones con notaciones, que describen cuántos elementos de una clase están relacionadas con objetos de otra clase.

Algunas de las anotaciones más comunes son:

1 no más de uno: Indica una relación uno a uno.

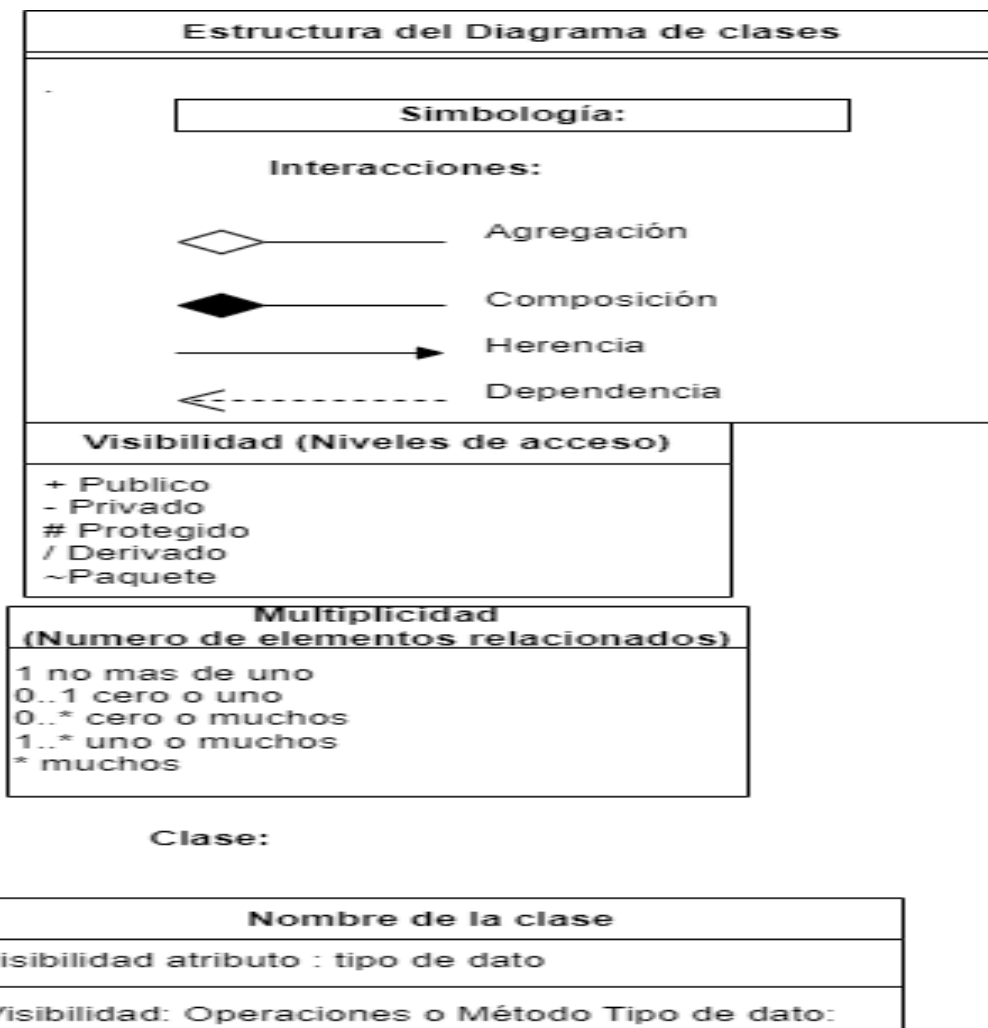
0..1 cero o uno: Indica que puede haber cero o un elemento relacionado.

0.. * cero o muchos: Indica que puede haber cero o más elementos relacionados.

1.. * uno o muchos: Indica que debe a ver al menos un elemento relacionado, pero puede haber más.

* muchos: Indica que puede haber muchos elementos relacionados

Todos estos elementos se muestran en la figura 1 de manera gráfica:

Figura 1***Elementos y símbolos en un Diagrama de Clases UML***

Nota: En la siguiente imagen se presentan los diferentes elementos y símbolos de un Diagrama de Clases de una forma gráfica, con ellos se puede modelar la precisión de la arquitectura del sistema, estos serán utilizados más adelante con El Sistema Compra-Venta de Ropa, el enlace para poder visualizar más a detalle la presente imagen es el siguiente:

<https://app.diagrams.net/#G1VnNfOGDdrkezcghJWmUKuq0CWEaIFJNH> .realizado en diagrams.com. Creación propia

Un Diagrama de Casos de Uso: Es una herramienta de modelado, en El Lenguaje de Modelado Unificado UML, este se utiliza para representar la interacción entre un sistema y sus actores.

Elementos principales de un Diagrama de Casos:

Actor:

Representa un Usuario o un Sistema externo que interactúa con el Sistema. Gráficamente se representa como una figura de una persona un bloque rectangular, estos están fuera del sistema y se conectan por medio de líneas al caso de uso con las que interactúan.

Caso de uso:

Un caso de uso representa una funcionalidad o escenario específico que el sistema proporciona a sus actores. Se representa gráficamente como una elipse y se nombra de manera descriptiva, por ejemplo “Registrar Usuario” o “Realizar Compra”, los casos de uso describen lo que el sistema hace, pero no como lo hace.

Relaciones entre actores y Casos de Uso:

Las relaciones entre actores y casos de uso se representan con líneas de comunicación o asociación, dichas líneas indican la interacción entre los actores y los casos de uso.

Las relaciones más comunes son:

Relación de Inclusión:

Indica que un caso de uso incluye (o es influido por) otro caso de uso: es representada gráficamente con una flecha contra la que apunta al caso de su incluido con la etiqueta <<Include>>, esto se utiliza para representar la reutilización de funcionalidades comunes en varios casos de uso y a su vez indica que un caso de uso incluye a otro como uno de su

funcionalidad.

Relación de Extensión:

La relación extensión indica que un caso puede extender el comportamiento de otro caso de uso con ciertas condiciones, se representa gráficamente con una flecha punteada con una etiqueta que conecta a un caso de uso de otro, indicando que un caso de uso se extiende a otro caso de uso en situaciones específicas, esto se realiza para representar la posibilidad de agregar funcionalidades adicionales con ciertos puntos de un caso de uso.

La etiqueta es: <<Extend>>

Generalización:

Representa una relación de herencia entre casos de uso, un caso de uso hijo hereda comportamientos del caso de uso padre, se representa gráficamente con una línea sólida y una flecha blanca, las relaciones de generalización se utilizan para mostrar la jerarquía entre los casos de uso, un caso de uso puede ser generalizado por otro, lo que significa que el caso de uso generalizado hereda su funcionalidad.

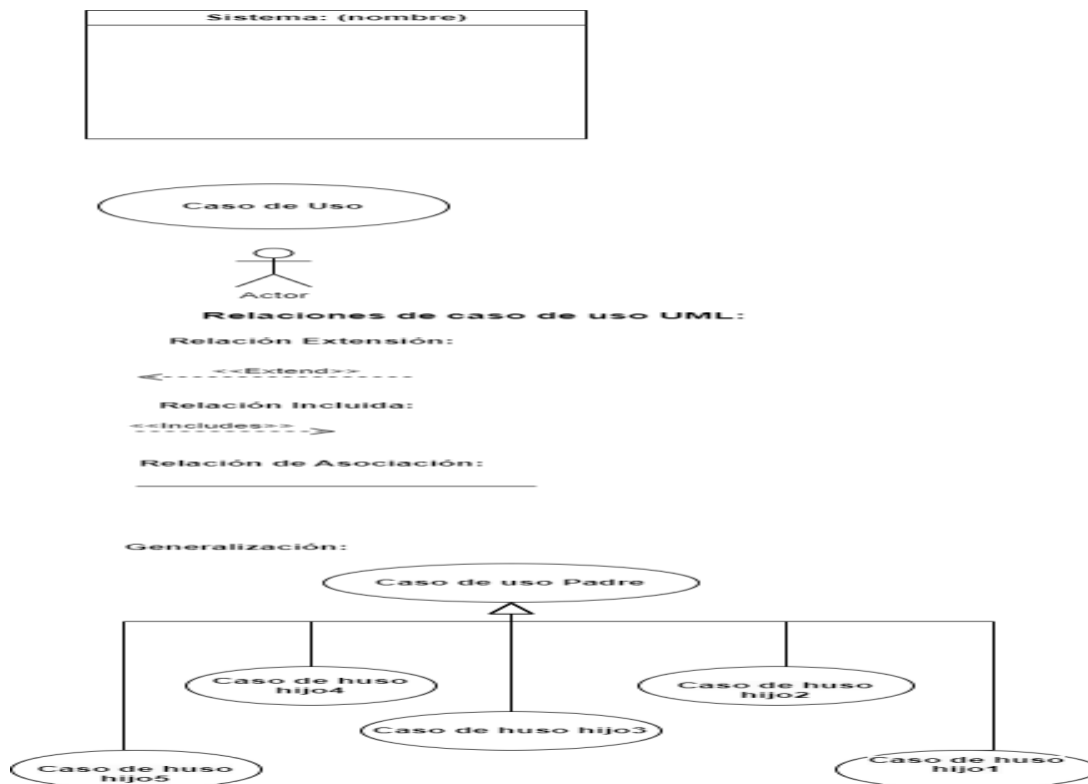
Asociación:

Se representa con una línea sólida que conecta un actor a un caso de uso, indica que el actor está involucrado en el caso de uso de alguna manera.

Sistema:

En el Diagrama de Casos de uso se representa como un rectángulo que rodea todos los casos de uso, el sistema es el contexto en el que se desarrolla todas las interacciones y funcionalidades representadas en el diagrama, es decir el sistema en sí, se representa como un gran rectángulo que encierra todos los casos de uso, esto ayuda a delimitar el alcance del sistema.

Todos estos elementos se representan de manera grafica en la figura 2.

Figura 2***Elementos y símbolos en un Diagrama de Casos de Uso UML***

Nota: En la siguiente imagen se presentan los diferentes elementos y símbolos de un Diagrama de Casos de Uso de una forma gráfica, un diagrama de casos de uso es una herramienta efectiva para comprender y comunicar las interacciones entre los actores y las funcionalidades del sistema en un alto nivel de abstracción ayudando a definir los requisitos del sistema y proporciona una vista general como los usuarios interactúan con él, , estos serán utilizados más adelante con El Sistema Compra-Venta de Ropa, el enlace para poder visualizar más a detalle la presente imagen es el siguiente:

<https://app.diagrams.net/#G1P3g1lgzYa5vyflfVuDoVvcHg2i7rw0pt> .Creación propia

Desarrollo

Contextualización:

En una empresa de venta de ropa online, se requiere un sistema que conlleve los requerimientos para una gestión adecuada de los involucrados, así como del proceso de compra-venta de sus productos.

El contexto general es el siguiente:

- La empresa cuenta con: empleados, clientes, proveedores y sucursales.
- Sus procesos son: venta minorista y venta de mayoreo.

Cuando el cliente ingresa al sistema, este elige los productos de su preferencia.

Posteriormente, al realizar la compra, tiene la posibilidad de adquirir en mayoreo a partir de 5 piezas. Todos los clientes deben tener un registro dentro del sistema. En este sentido, a cada cliente se le solicitan sus datos generales (nombre, apellido, dirección, correo electrónico, usuario y contraseña).

En la página del comercio el cliente podrá acceder para ver sus compras y podrá adquirir sus productos en línea. Esto mediante su nombre de usuario y contraseña previamente registrados.

Identificar a los actores que se involucran en este proceso, así como cada una de sus actividades. De la misma manera, encontrar la asociación que conlleva hacia estos procesos. Por su parte, para la presentación del análisis de esta información, es necesario generar los diagramas correspondientes. Es importante considerar que el contexto es general, y dentro de este mismo se presentan otras actividades que involucran a los actores.

Actividad:

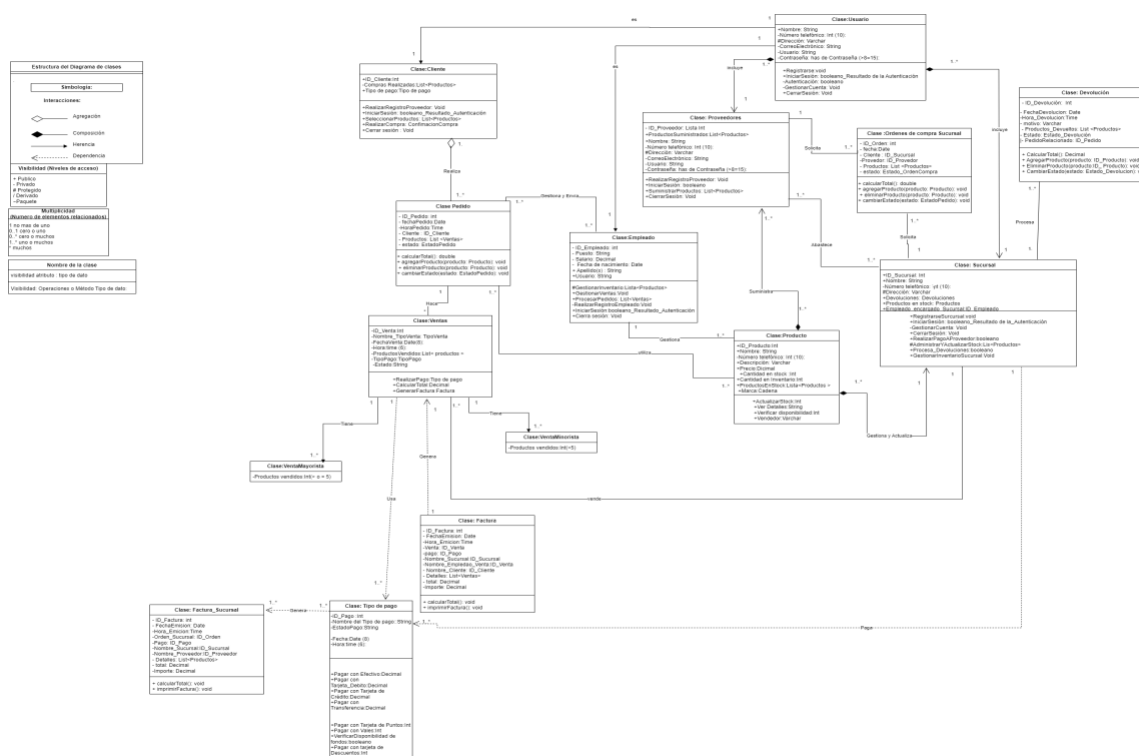
Considerando el contexto anterior respecto a la empresa de venta de ropa online aparte de

diseñar los diagramas de contexto es necesario involucrar las situaciones que presentan los actores convirtiendo la información en diagramas de clases y diagramas de casos de uso.

Generar los diagramas para el Paradigma Orientado a Objetos (diagramas de clases y diagrama de casos de uso).

Figura 3

Diagrama de Clases



Nota: En la siguiente imagen se presenta un diagrama de, de igual manera en la parte de

“Descripción del Diagrama de Clase”, representar la estructura del sistema de información

desde un punto de vista estático, el enlace para poder visualizar más a detalle la presente imagen

es el siguiente:

<https://app.diagrams.net/#G1mHuWOjApIu551yneqhr4-QPqNPogwKQo> realizado en diagrams.com. Creación propia

Descripción del Diagrama del Diagrama de Clases:

Clase: Usuario

Atributos:

+Nombre: String

-Número telefónico: Int (10):

#Dirección: Varchar

-CorreoElectrónico: String

-Usuario: String

-Contraseña: has de Contraseña (>8=15):

Métodos:

+Registrarse:void

+IniciarSesión: booleano_Resultado de la Autenticación

-Autenticación: booleano

-GestionarCuenta: Void

+CerrarSesión: Void

La Clase:Usuario tiene una Relación de herencia con la Clase:Cliente

La Multiplicidad (cardinalidad) es: 1 a 1 (un Usuario puede ser solo un cliente), es decir que ningún ID_Cliente será igual al de otro Usuario, esto nos ayudará a realizar la autenticación de los datos y verificar si existe el registro de dicho Cliente en el sistema.

Clase:Cliente

Atributos:

+ID_Cliente:Int

-Compras Realizadas:List<Productos>

+Tipo de pago:Tipo de pago

Métodos:

+RealizarRegistroProveedor: Void

+IniciarSesión: booleano_Resultado_Autenticación

+SeleccionarProductos: List<Productos>

+RealizarCompra: ConfimacionCompra

+Cerrar sesión: Void

La Clase: Usuario tiene una Relación de herencia con la Clase: Empleado

La Multiplicidad (cardinalidad) es: 1 a 1 (un Usuario puede ser solo un Empleado), es decir que ningún ID_Empleado será igual al de otro Usuario, esto nos ayudará a realizar la autenticación de los datos y verificar si existe el registro de dicho Empleado en el sistema.

La Multiplicidad (cardinalidad) es: 1 a 1

Clase:Empleado

Atributos:

- ID_Empleado: int

- Puesto: String

- Salario: Decimal

- Fecha de nacimiento: Date

+ Apellido(s): String

+Usuario: String

Métodos:

#GestionarInventario:Lista<Productos>

+GestionarVentas:Void

+ProcesarPedidos: List<Ventas>

-RealizarRegistroEmpleado:Void

+IniciarSesión:booleano_Resultado_Autenticación

+Cierra sesión: Void

La Clase:Usuario tiene una Relación Compuesta con la Clase:proveedores

La Multiplicidad (cardinalidad) es: 1..* a 1 (un Usuario puede ser solo un proveedor, pero uno o muchos proveedores pueden ser un Usuario) , es decir que ningún ID_Proveedor será igual al de otro Usuario, esto nos ayudara a realizar la autenticación de los datos y verificar si existe el registro de dicho Proveedor en el sistema.

Clase: Proveedores

Atributos:

- ID_Proveedor: Lista Int

+ProductosSuministrados:List<Productos>

+Nombre: String

-Número telefónico: Int (10):

#Dirección: Varchar

-CorreoElectrónico: String

-Usuario: String

-Contraseña: has de Contraseña (>8=15):

Métodos:

+RealizarRegistroProveedor: Void

+IniciarSesión: booleano

+SuministrarProductos: List<Productos>

+CerrarSesión: Void

La Clase:Usuario tiene una Relación Compuesta con la Clase:Sucursal

La Multiplicidad (cardinalidad) es: 1..* a 1 (un Usuario puede ser solo una Sucursal, pero una o muchas Sucursal pueden ser un Usuario) , es decir que ningún ID_ Sucursal será igual al de otro Usuario, esto nos ayudara a realizar la autenticación de los datos y verificar si existe el registro de dicha Sucursal en el sistema.

Clase: Sucursal

Atributos:

+ID_Sucursal: Int

+Nombre: String

-Número telefónico: Int (10):

#Dirección: Varchar

+Devoluciones: Devoluciones

+Productos en stock: Productos

+Empleado_encargado_Sucursal:ID_Empleado

Métodos:

+RegistrarseSurcursal:void

+IniciarSesión: booleano_Resultado de la_Autenticación

-GestionarCuenta: Void

+CerrarSesión: Void

+RealizarPagoAProveedor:booleano

#AdministrarYActualizarStock:Lis<Productos>

+Procesa_Devoluciones:booleano

+GestionarInventarioSucursal:Void

Cada Clase: tiene sus propios atributos que la diferencian del resto y las vuelve especiales, sin perder de vista el tipo de relación que cada una tiene con el Usuario.

Clase: Pedido

Atributos:

- ID_Pedido: int
- fechaPedido:Date
- HoraPedido:Time
- Cliente : ID_Cliente
- Productos: List <Ventas>
- estado: EstadoPedido

Método:

- + calcularTotal(): double
- + agregarProducto(Producto: Producto): void
- + eliminarProducto(Producto: Producto): void
- + cambiarEstado(estado: EstadoPedido): void

La Clase: Cliente tiene una Relación de Agregación con la Clase: Proveedor

La Multiplicidad (cardinalidad) es: 1 a 1..* (un cliente puede realizar uno o muchos pedidos pero uno o muchos Pedidos pueden ser de un cliente), es decir que ningún ID_Pedido será igual al de otro Cliente, esto nos ayudara a realizar la venta, la factura y él envió(estos 3 se

explicaran más adelante).

Clase: Producto

Atributo:

+ID_Producto: Int

+Nombre: String

-Número telefónico: Int (10):

+Descripción: Varchar

+Precio: Ddecimal

+Cantidad en stock :Int

+Cantidad en Inventario: Int

+ProductosEnStock: Lista<Productos >

+Marca: Cadena

Método:

+ActualizarStock: Int

+Ver Detalles: String

+Verificar disponibilidad: Int

+Vendedor: Varchar

La Clase: Pedido tiene una Relación de Asociación con la Clase: Producto

La Multiplicidad (cardinalidad) es: 1..* a 1..* (un pedido o muchos utilizan uno o muchos Productos, pero uno o muchos Productos utilizar uno o muchos pedidos), es decir que ningún ID será igual al de otro, esto nos ayudara a realizar la venta, la factura y el envío (estos 3 se explicaran más adelante), de esta manera se podrá llevar una mejor gestión de pedidos y productos.

Clase: Ventas**Atributo:**

-ID_Venta: Int

-Nombre_TipoVenta: TipoVenta

-FechaVenta: Date(8):

-Hora: time (6):

-ProductosVendidos: List< Productos >

-TipoPago: TipoPago

-Estado: String

Método:

+RealizarPago: Tipo de pago

+CalcularTotal: Decimal

+GenerarFactura: Factura

La Clase: Pedido tiene una Relación de Asociación con la Clase: Venta

La Multiplicidad (cardinalidad) es: 1 a * (un pedido hace muchas ventas, pero muchas ventas hacen un pedido), es decir que ningún ID_Venta será igual al de otro, esto nos ayudara a realizar la venta, la factura y el envío (estos 3 se explicaran más adelante) de esta manera se podrá llevar una mejor gestión de ventas y pedidos

Clase: VentaMinorista**Atributo**

-Productos vendidos: Int(<5)

La Clase: VentaMinorista tiene una Relación de herencia con la Clase: Venta

La Multiplicidad (cardinalidad) es: 1 a 1..* (una venta tiene una o muchas ventas)

minoristas , pero una o muchas ventas minoristas tiene una venta), es decir que ningún ID_VentaMinorista será igual a otro, esto nos ayudara a tener mejor gestión de ventas con diferentes roles y tipos de ventas.

Clase: VentaMayorista

Atributo:

-Productos vendidos: Int(> 0 = 5)

La Clase: VentaMayorista tiene una Relación de herencia con la Clase: Venta

La Multiplicidad (cardinalidad) es: 1 a 1..* (una venta tiene una o muchas ventas mayoristas , pero una o muchas ventas minoristas tiene una venta), es decir que ningún ID_VentaMayorista será igual a otro, esto nos ayudara a tener mejor gestión de ventas con diferentes roles y tipos de ventas.

Clase:: Tipo de pago

Atributo:

-ID_Pago :Int

-Nombre del Tipo de pago::String -EstadoPago:String

-Fecha:Date (8)

-Hora:time (6):

Método:

+Pagar con Efectivo:Decimal +Pagar con Tarjeta_Debito:Decimal

+Pagar con Tarjeta de Crédito:Decimal

+Pagar con Transferencia:Decimal

+Pagar con Tarjeta de Puntos:Int +Pagar con Vales:Int +VerificarDisponibilidad de fondos:booleano

+Pagar con tarjeta de Descuentos: Int

La Clase: Tipo de Pago tiene una Relación de dependencia con la Clase: Venta

La Multiplicidad (cardinalidad) es: 1 a 1..* (una venta tiene uno o muchos tipos de pago , pero una o muchas tipos de pago tiene una venta), es decir que ningún ID_ Tipo de Pago será igual a otro, esto nos ayudara a gestionar cómo se realizan los pagos en el sistema y cómo se relacionan con las ventas.

Clase: Factura

Atributo:

- ID_Factura: int
- FechaEmision: Date
- Hora_Emision: Time
- Venta: ID_Venta
- pago: ID_Pago
- Nombre_Sucursal: ID_Sucursal
- Nombre_Empleado_Venta: ID_Venta
- Nombre_Cliente: ID_Cliente
- Detalles: List<Ventas>
- total: Decimal
- Importe: Decimal

Método:

- + calcularTotal(): void
- + imprimirFactura(): void

La Clase: Factura tiene una Relación de dependencia con la Clase: Venta

La Multiplicidad (cardinalidad) es: 1 a 1 (una factura depende de una venta para poderse generar, y una venta solo genera una factura a la vez), es decir que ningún ID_Factura será igual a otro, esto nos dará la capacidad de gestionar y generar facturas para cada venta en el sistema.

La Clase:Empleado tiene una relación de asociación con la Clase:Producto.

La Multiplicidad es: 1 a 1..* (un Empleado gestiona uno o muchos Productos)

La Clase:Empleado tiene una relación de asociación con la Clase:Pedido

La Multiplicidad es: 1..* a 1..* (uno o muchos Empleados pueden gestionar y enviar uno o muchos pedidos y viceversa)

La Clase:proveedores tiene una relación compuesta con la Clase:Producto .

La Multiplicidad es: 1 a 1..* (uno proveedor suministra uno o muchos Productos)

Las Sucursal tiene una relación de asociación con la Clase: Órdenes de Compra de Sucursal:

La Multiplicidad es: 1 a 1..* (una Sucursal solicita una o muchas Órdenes de compra Sucursal, pero una o muchas Órdenes de compra Sucursal son de una Sucursal)

Clase: Órdenes de compra Sucursal

Atributos:

- ID_Orden: int
- fecha:Date
- Cliente : ID_Sucursal
- Proveedor: ID_Proveedor
- Productos: List <Productos>
- estado: Estado_OrdenCompra

Métodos:

```

+ calcularTotal(): double
+ agregarProducto(Producto: Producto): void
+ eliminarProducto(Producto: Producto): void
+ cambiarEstado(estado: EstadoPedido): void

```

La Clase Órdenes de compra Sucursal tiene una relación de asociación con la Clase: Proveedores.

La Multiplicidad La Multiplicidad es: 1..* a 1 (Una o muchas Órdenes de compra Sucursal son solicitadas a un proveedor)

La Clase: Proveedor tiene una relación de asociación con la Clase: Sucursal.

La Multiplicidad es: 1..* a 1..* (uno o muchos Proveedores pueden abastecer a una o muchas Sucursales)

La Clase: Sucursal tiene una relación de dependencia con la Clase: Tipo de Pago

La Multiplicidad (cardinalidad) es: 1 a 1..* (una Sucursal depende uno o muchos tipos de pago para pagar , pero una o muchas tipos de pago es expedido por una Sucursal), es decir que ningún ID_ Tipo de Pago será igual a otro, esto nos ayudara a gestionar cómo se realizan los pagos en el sistema y cómo se relacionan con las órdenes de compra que suministra el proveedor.

La Clase: Tipo de Pago tiene una Relación de dependencia con la Clase: Factura_Sucursal

La Multiplicidad (cardinalidad) es: 1..* a 1..* (una Factura_Sucursal tiene uno o muchos tipos de pago , y viceversa un tipo de pago puede generar una o muchas facturas), es decir que ningún ID_ Factura será igual a otro, esto nos ayudara a gestionar cómo se realizan los pagos en el sistema y cómo se relacionan con las ventas.

Clase: Factura_Sucursal

Atributos:

- ID_Factura: int
- FechaEmision: Date
- Hora_Emision:Time
- Orden_Sucursal: ID_Orden
- Pago: ID_Pago
- Nombre_Sucursal:ID_Sucursal
- Nombre_Proveedor:ID_Proveedor
- Detalles: List<Productos>
- total: Decimal
- Importe: Decimal

Método:

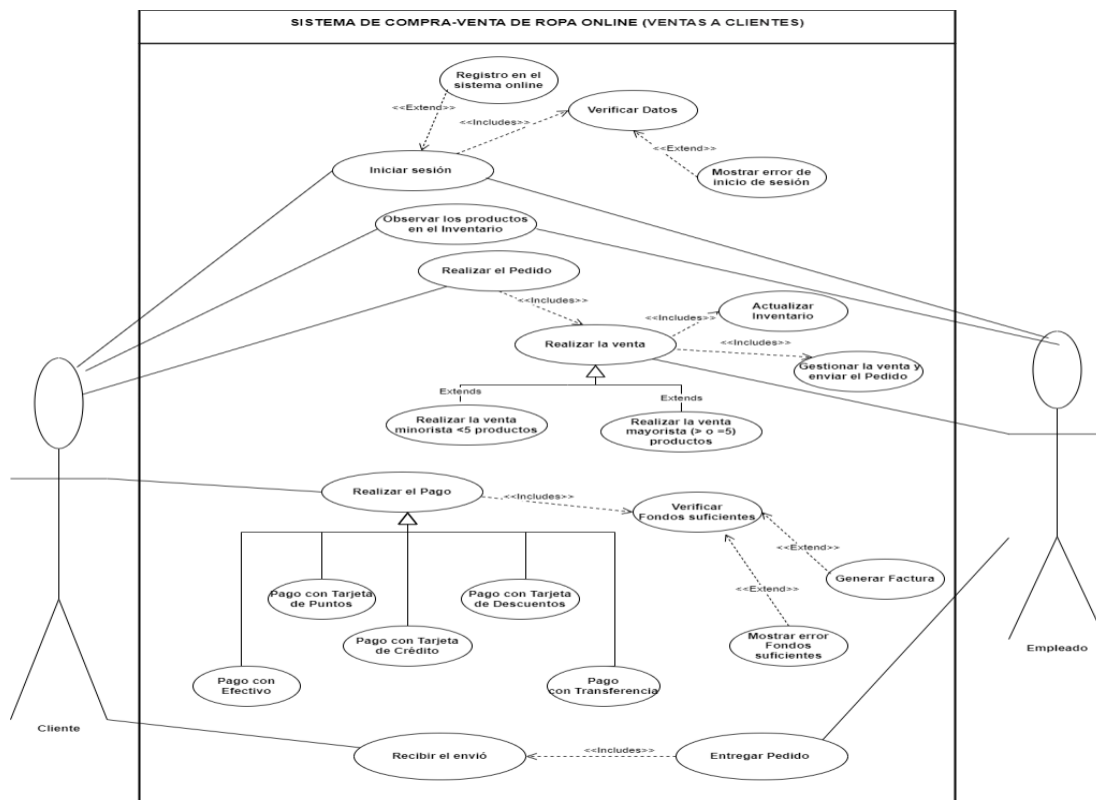
- + calcularTotal(): void
- + imprimirFactura(): void

La Sucursal tiene una relación de asociación con la Clase: Devolución.

La Multiplicidad es: 1 a 1..*(una Sucursal procesa una o muchas devoluciones, una o muchas devoluciones son procesadas por una Sucursal), esto nos ayudará a tener una mejor gestión de las devoluciones que entran al sistema.

Figura 4

**Diagrama de Casos de Uso: Sistema SISTEMA DE COMPRA-VENTA DE ROPA ONLINE
(VENTAS A CLIENTES)**



Nota: En la siguiente imagen se presenta un diagrama de tiene el sistema, asimismo en la parte de “**Descripción del Diagrama de Diagrama de Casos de Uso: SISTEMA DE COMPRA-VENTA DE ROPA ONLINE (VENTAS A CLIENTES)**” se describe a detalle la figura en donde se detalla la interacción y comportamiento de cada elemento durante el proceso de venta al cliente, el enlace para poder visualizar más a detalle la presente imagen es el siguiente: https://app.diagrams.net/#G1WKVKfyUmNVDZvX9rw4_K9CJ98Qeq65r9 .

Realizado en diagrams.com. Creación propia

Descripción del Diagrama de Casos de Uso: SISTEMA DE COMPRA-VENTA DE ROPA ONLINE (VENTAS A CLIENTES)

El nombre del sistema es: SISTEMA DE COMPRA-VENTA DE ROPA ONLINE (VENTAS A CLIENTES)

Está conformado por 2 actores los cuales son:

Cliente

Empleado

Estos estarán fuera del sistema, ellos interactúan con todo lo que sea describirá a continuación según sea el caso.

El Cliente y Empleado, están asociado con el caso de uso: Iniciar Sesión, este caso de uso se extiende al caso de uso: Registro en el Sistema Online, además el caso de usos: Iniciar Sesión incluye el caso de uso: Verificar Datos, el cual se extiende al caso de uso mostrar error de inicio de sesión.

El Cliente y Empleado, están asociado con el caso de uso: Observar los productos en el Inventario.

El Cliente está asociado al caso de uso: Realizar El Pedido, el cual se incluye al caso de uso: Realizar Venta, en este caso de uso también está asociado el empleado.

El caso de uso: Realizar Venta, tiene 2 tipos de ventas, es decir se generaliza y extiende al caso de uso: Realizar la venta minorista <5 productos, así mismo se generaliza y extiende al caso de uso: Realizar la venta mayorista (≤ 5) productos, cualquiera de los 3 casos de uso relacionados a la venta a la venta, se incluye al caso de uso: Actualizar Inventario y el caso: Gestionar la venta y enviar el Pedido.

El cliente está asociado al caso de uso: Realizar el Pago, este caso se generaliza a 5 casos de uso

que son tipos de pago, los cuales son:

Caso de uso: Pago con Efectivo.

Caso de uso: Pago con mi Tarjeta De Puntos.

Caso de uso: Pago con Tarjeta de Descuentos.

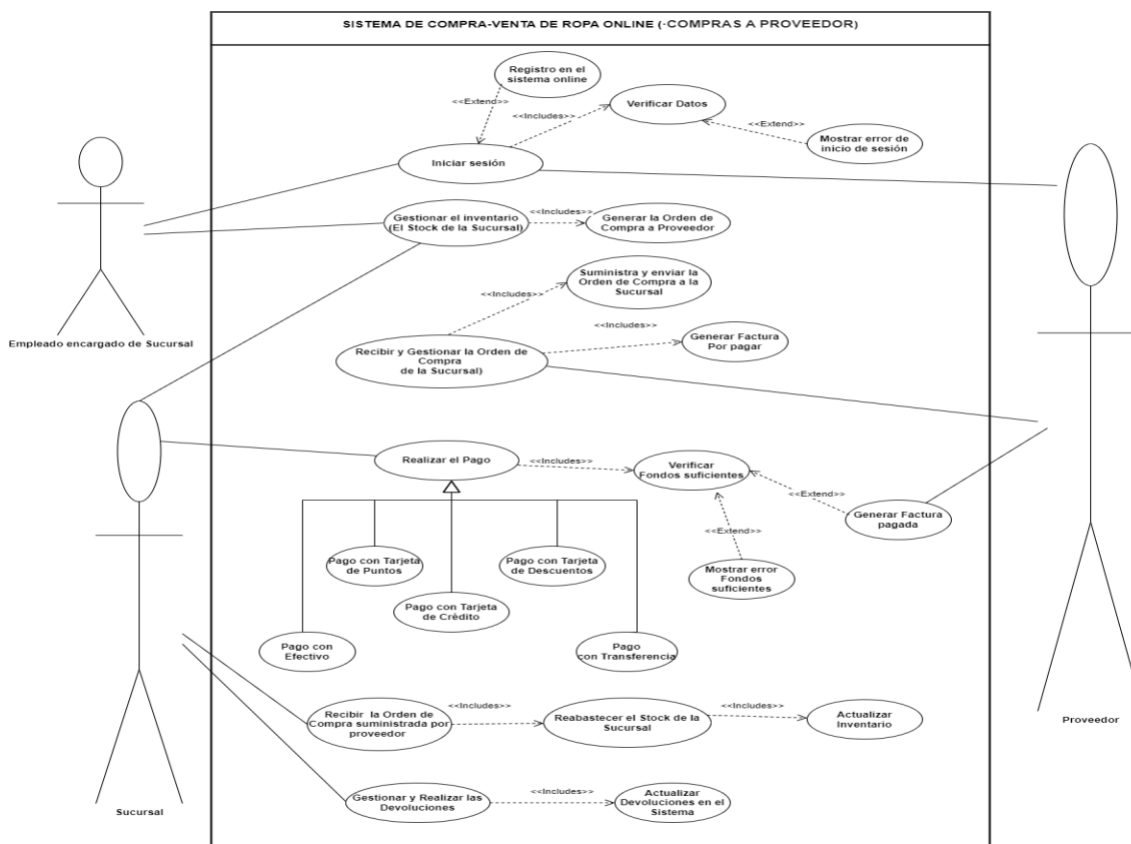
Caso de uso: Pago con Transferencia.

Cualquiera de los casos de tipo de pago, incluye el caso de uso: Verificar Fondos Suficientes, este mismo caso de uso se extiende al caso de uso Generar Factura y el caso de uso: Mostrar Error Fondos Suficientes.

El empleado está asociado al caso de uso: Entregar Pedido, esto se incluye el caso de uso: Recibir el envío en el cual se encuentra asociado al cliente.

Figura 5

**Diagrama de Casos de Uso: SISTEMA DE COMPRA-VENTA DE ROPA ONLINE
(COMPRAS A PROVEEDOR)**



Nota: En la siguiente imagen se presenta un diagrama de tiene el sistema, asimismo en la parte de “**Descripción del Diagrama de Diagrama de Casos de Uso: SISTEMA DE COMPRA-VENTA DE ROPA ONLINE (COMPRAS A PROVEEDOR)**” se describe a detalle la figura, en donde se describe la interacción y comportamiento de los actores, durante la gestión y el abastecimiento del inventario (Stock de sucursal), el enlace para poder visualizar más a detalle la presente imagen es el siguiente:

<https://app.diagrams.net/#G1oscuIEe4jjgqwgFWAQGO1AbH0-CMWcg3>, realizado en diagrams.com. Creación propia

Descripción del Diagrama de Diagrama de Casos de Uso: SISTEMA DE COMPRA-VENTA DE ROPA ONLINE (COMPRAS A PROVEEDOR)

El nombre del sistema es: SISTEMA DE COMPRA-VENTA DE ROPA ONLINE
(COMPRAS A PROVEEDOR)

Está conformado por 3 actores los cuales son:

Cliente

Empleado encargado de la sucursal.

Sucursal.

Estos estarán fuera del sistema, ellos interactúan con todo lo que sea describirá a continuación según sea el caso.

El Cliente y Empleado encargado de la sucursal, están asociado con el caso de uso: Iniciar Sesión, este caso de uso se extiende al caso de uso: Registro en el Sistema Online, además el caso de usos: Iniciar Sesión incluye el caso de uso: Verificar Datos, el cual se extiende al caso de uso mostrar error de inicio de sesión.

El Empleado y Sucursal, están asociado con el caso de uso: Gestionar el Inventario (El Stock De La Sucursal), este caso de uso incluye el caso de uso: Generar la Orden de Compra a Proveedor.

El Proveedor está asociado al caso de uso: Recibir y Gestionar la Orden De Compra de la Sucursal, este caso de uso incluye el caso de uso: Suministrar y enviar la Orden de Compra a la Sucursal y el caso de uso: Generar Factura por pagar.

Sucursal está asociado al caso de uso: Realizar el Pago, este caso se generaliza a 5 casos de uso que son tipos de pago, los cuales son:

Caso de uso: Pago con Efectivo.

Caso de uso: Pago con mi Tarjeta De Puntos.

Caso de uso: Pago con Tarjeta de Descuentos.

Caso de uso: Pago con Transferencia.


Cualquiera de los casos de tipo de pago, incluye el caso de uso: Verificar Fondos Suficientes, este mismo caso de uso se extiende al caso de uso: Generar Factura y el caso de uso: Mostrar Error Fondos Suficientes.

El caso de uso: Generar Factura Pagada está asociado al proveedor.

Sucursal está asociado al caso de uso: Recibir la Orden de Compra suministrada por el proveedor, que incluye el caso de uso: Reabastecer el Stock de la Sucursal, este mismo caso de uso incluye el caso de uso: Actualizar Inventario.

Sucursal está asociado al caso de uso: Gestionar y Realizar las Devoluciones, este mismo caso de uso incluye el caso de uso: Actualizar Devoluciones en el Sistema

Referencias:

- Charly Cimino. (2021a, julio 19). *UML <unk> Qué es 😊 y cuándo usarlo ?* [Vídeo]. YouTube. Recuperado el jueves, 02 de noviembre de 2023 de:
https://www.youtube.com/watch?v=d_y2RcEP5Is
- Charly Cimino. (2021b, agosto 2). *DIAGRAMA de CLASES UML <unk> ¡Empezá por acá! 😊* [Vídeo]. YouTube. Recuperado el jueves, 02 de noviembre de 2023 de:
<https://www.youtube.com/watch?v=txxU2x5e3HM>
- Equipo editorial de IONOS. (2020, 24 julio). *El diagrama de casos de uso en UML*. IONOS Digital Guide. Recuperado el jueves, 02 de noviembre de 2023 de:
<https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/>
- Lucid Software Español. (2019, 4 febrero). *Tutorial - Diagrama de clases UML* [Vídeo]. YouTube. Recuperado el jueves, 02 de noviembre de 2023 de:
<https://www.youtube.com/watch?v=Z0yLerU0g-Q>
- Marcasco. (2022, 2 diciembre).  ¿Qué es un caso de uso? (Con elementos clave y un ejemplo). *Historiadelapresa.com*. Recuperado el jueves, 02 de noviembre de 2023 de:
<https://historiadelapresa.com/caso-de-uso>
- Yahoo is part of the Yahoo family of brands*. (s. f.). Recuperado el jueves, 02 de noviembre de 2023 de:
<https://mx.video.search.yahoo.com/yhs/search?fr=yhs-fc-2461&ei=UTF-8&hsimp=yhs-2461&hspart=fc¶m1=7¶m2=eJwti0EOgjAQRa8yS0gMTFso07L1BG6NiwoVGwolgMF4ettoZvP%2B%2F28G11%2Fb2%2BXMEDmhvJ5uc8xKKYqYJmxQchFD9%2BsjuSVisjWXpCtSmtiDNGfUayZqoQXZWquGJ3mwIdrTO%2BLLJAof570p6wIhO9zch2ODeQeGBbY>

QC1m18JZVDmZZvD3sfXR7WYumEBKy8blP%2FgTejRYG240hh%2B65hsmWjFGB6WAZ
D7O6%2F8sXDt5A9Q%3D%3D&p=casos+de+uso&type=fc_AC934C13286_s58_g_e_d07062
3_n9998_c999#id=2&vid=f3cb9a8eca2e1a13e84546a33389a014&action=click