Práctica: Diseño

para el proyecto

CAMBIOS GAMER

Preparado por:

Caballero Flores Gabriela Anaid

González Higuera Fernando

Jiménez González Melissa

Toral Maldonado José Ignacio

Salazar López Brandon Martín

Tabla de contenido

Historial de revisiones

Nombre	Fecha	Cambios hechos	Versión
Fernando González-FERGO19	25/01/2022	Se agregó el punto 2, Diagrama de Robustez, del documento Práctica-Diseño.	1
Melissa Jiménez-MelisBot	03/02/22	Agregué las descripciones de los puntos de la práctica de diseño.	2
Brandon Salazar-MartInBSL1	04/02/22	Agregué el punto 3. Diagramas de secuencia.	3
José Toral- VandalJI	05/02/22	Se completó el punto “Diagramas de clase” con la inserción de: Diagramas MVC, Diagramas DAO y el Diagrama de clases.	4
Melissa Jiménez-MelisBot	06/02/22	Se agregó la introducción y el diagrama de paquetes.	5
Gabriela Caballero-	06/02/22	Agregué el punto 6. Diseño de la bases de datos.	6

1. Introducción

El presente informe de diseño de software para la tienda online “Cambio Gamer” contiene los diseños preliminares para comprender el funcionamiento que se espera dentro de la tienda online tomando como referencia los casos de usos más importantes y llevándolos a su diseño para observar detalladamente si es claro y completo de los requisitos esperados por el cliente.

Para este informe de diseño tomamos en cuenta los siguientes diseños como son: diagramas de robustez, diagramas de secuencias, diagramas de clases, diagramas de paquetes, diagramas de bases de datos; estos nos darán la oportunidad de comprender muy a fondo sus usos y requisitos, con cada diagrama comprenderemos los distintos puntos principales para el diseño de la tienda online.

El informe de diseño se realizó con fines prácticos y de análisis para la comprensión clara y completa de los requisitos esperados por el cliente, para el desarrollo de la tienda online y su implementación para el funcionamiento correcto del mismo.

Por otro lado, la importancia de no dejar huecos y corregir errores que fueron implementados con anterioridad en los casos de usos, esto con el fin de no dejar nada fuera de lo requerido en una función del sistema y tener un proyecto completo y ordenado. Asimismo, la comprensión de cómo debe de funcionar un sistema en una tienda virtual de productos gamer.

Este informe se realizó mediante el análisis de los casos de usos, estos son: inscribir clientes, iniciar sesión, realizar compra, mostrar carrito y cerrar sesión, los cuales realizamos la implementación de sus flujos principales y alternativos de cada uno de ellos.

Implementamos en los diagramas de robustez la interacción del usuario en la tienda online esto nos permite una comprensión visual, para los diagramas de secuencias cómo reacciona el sistema a estas interacciones que hace el usuario y las diversas situaciones en las que se podría encontrar, para los diagramas de clases comprendimos los datos requeridos más importantes del usuario, productos, administradores y órdenes, para los diagramas de paquetes implementamos la interacción de cada paquete dentro de la tienda y finalmente para el diagrama de bases de datos el funcionamiento de los datos a almacenar tanto para usuarios, productos, administradores y llevar un orden de todos estos datos.

El presente informe contiene los diagramas diseñados en StarUML correspondientes a cada caso de uso establecidos, cada punto contiene una descripción de la importancia de cada diagrama iniciando con los diagramas de robustez, diagramas de secuencias, diagramas de clases, diagramas de paquetes, diagramas de bases de datos al igual de las tablas y relaciones de las mismas.

2. Diagramas de robustez

Los diagramas de robustez nos darán una visión más clara para nuestro funcionamiento de la tienda digital “Cambios Gamer” de la cuales realizamos sus casos de usos respectivos, y ahora comprobaremos si el caso de uso es apto y claro para la función de la tienda online y sino realizamos cambios en los casos de usos.

Para este proyecto tomamos 5 de nuestros casos de usos realizados en el “documento de requerimientos”, los cuales consideramos los más prioritarios para el sistema, usando la herramienta de StarUML planteamos las acciones del actor dentro de la tienda online.

A continuación, se muestran los casos de usos con sus flujos principales y flujos alternativos debajo de estos, los diagramas de robustez corresponden a cada uno de los casos de usos.

Inscribir clientes

A.1 Caso de uso “Inscribir clientes”

Flujos Principales:

1. El actor presiona el botón “Inscripción”.
2. El sistema le muestra al actor el formulario para crear una nueva cuenta.
3. El actor (A.13 <<include>>) introduce su nombre, apellido paterno, dirección, correo, y contraseña.
4. El actor presiona el botón “Aceptar”.
5. El sistema muestra un mensaje de “Los datos se guardaron correctamente, ahora puede iniciar sesión.” en la pantalla.
6. El actor presiona el botón “Aceptar” en el mensaje.
7. El sistema le muestra al usuario la pantalla principal para que inicie sesión.

Flujos Alternativos:

En 4 (Si el actor no ingresa ningún dato).

4.1. El actor presiona el botón “Aceptar”.
4.2. El sistema le muestra un mensaje al actor diciendo que complete los campos solicitados.
4.3 El actor permanece en el caso 3.

En 5 (Si el actor quiere cancelar la inscripción).

5.1. El actor presiona el botón “Cancelar”.
5.2. El sistema quita el formulario y se muestra la página principal.

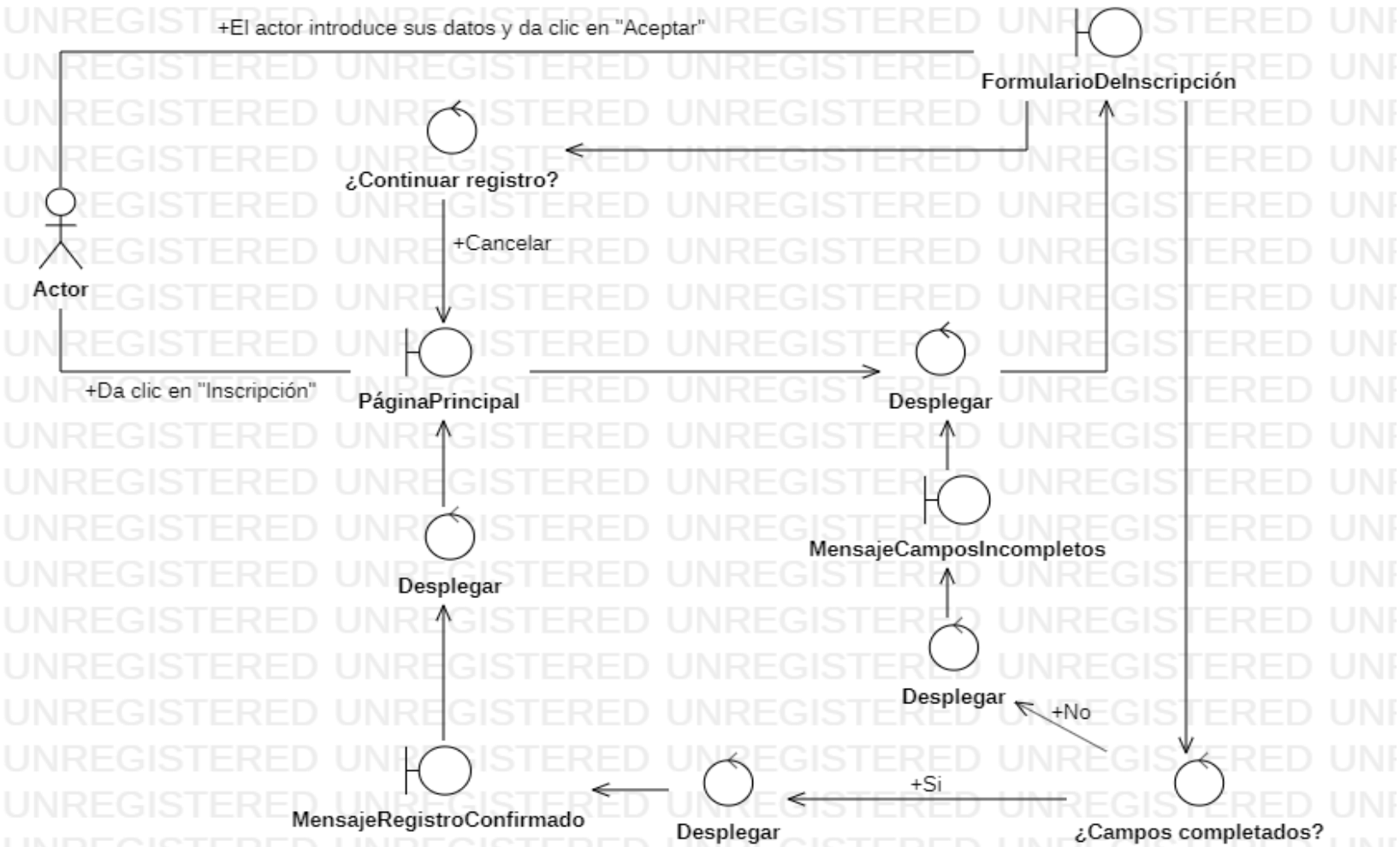


Diagrama de robustez de “Inscribir clientes”

En el diagrama de robustez de “Inscribir clientes” nos explica cuando el visitante se va a inscribir por primera vez dentro de la página online “Cambios Gamer” para que sea un cliente de la tienda virtual.

Iniciar sesión

A.2 Caso de uso “Iniciar sesión”

Flujos Principales:

1.El usuario presiona el botón “Iniciar sesión” desde la página principal.
2.El sistema le despliega al usuario el formulario para iniciar sesión.
3.El usuario (A.13 <<include>>) introduce su correo electrónico y contraseña.
4.El usuario presiona el botón “Aceptar”.
5.El sistema verifica que la cuenta realmente exista.
6.Si la cuenta existe, verifica que la contraseña dada es correcta.
7.El sistema devuelve información de la cuenta.
8.El sistema inicia la sesión una vez haya verificado que los datos son (A.1 <<extend>>) correctos y muestra un mensaje como este “Has iniciado sesión correctamente”.
9.El sistema despliega una nueva página con la sesión iniciada del usuario correspondiente.

Flujos Alternativos:

En 3 (Si el usuario no ingresa ningún dato).
3.1. El usuario presiona el botón “Aceptar”.
3.2. El sistema solicita al usuario que complete los campos solicitados.
3.3. El usuario ingresa los datos solicitados.
3.4. El usuario da clic en “Aceptar”.
3.5. El sistema verifica que la cuenta exista.

En 5 (Si el usuario puso datos incorrectos).
5.1. El sistema solicita al usuario que introduzca la información nuevamente.

En 2 (Si el actor no quiere iniciar sesión).
2.1. El actor presiona el botón “Cancelar”.
2.2. El sistema despliega la página principal.

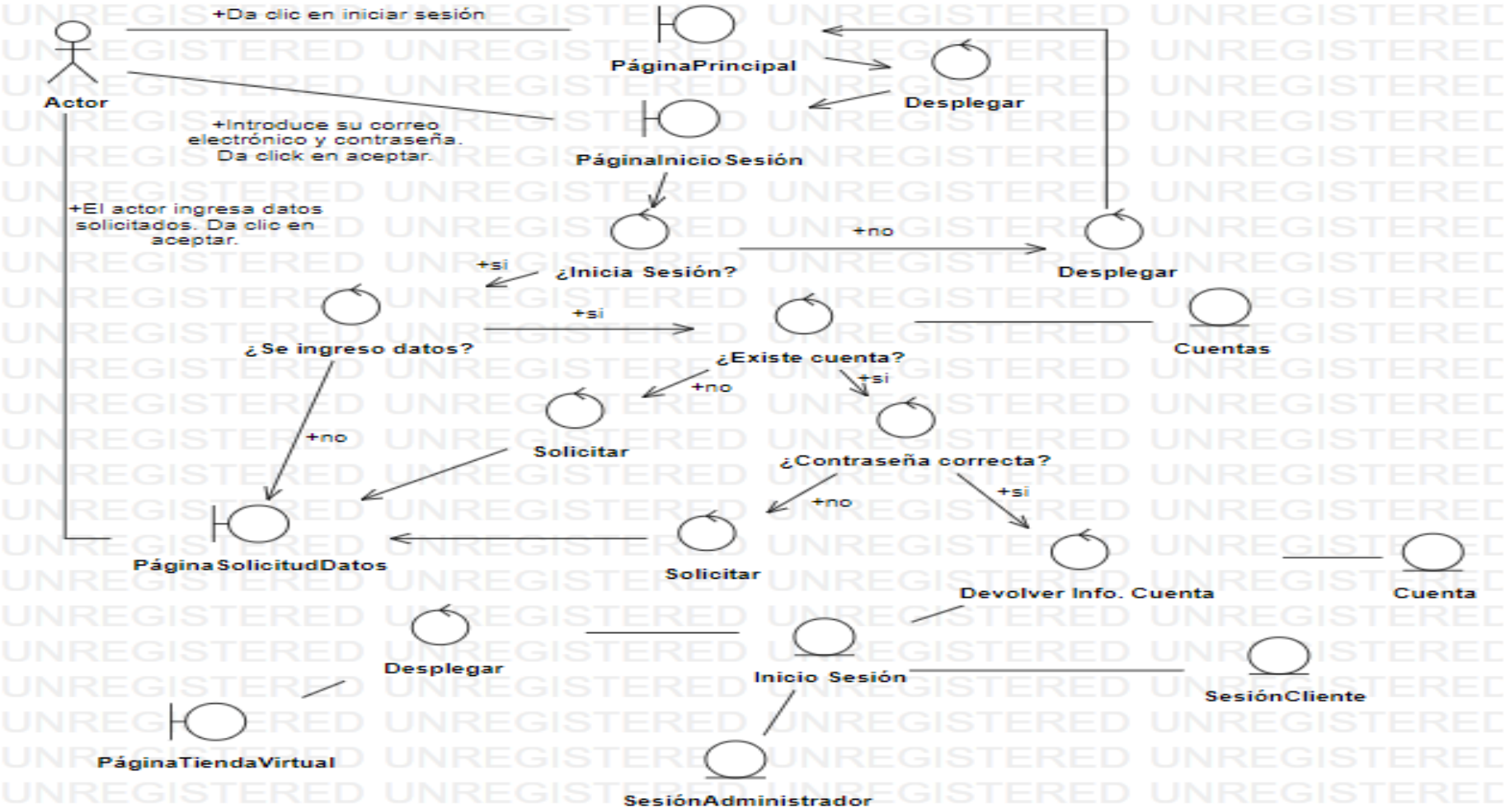


Diagrama de robustez de “Iniciar sesión”

En el diagrama de robustez “Iniciar sesión” nos muestra que cuando el cliente va a iniciar sesión dentro de la página online “Cambios Gamer” para que pueda revisar las ofertas que haya, ver productos si hay nuevos, para modificar los datos de su cuenta de perfil, entre otras cosas.

Realizar compras

A.3 Caso de uso “Realizar compras”

Flujos Principales:

1. El actor busca entre las páginas el producto que desea comprar.
2. El actor selecciona el artículo que desea comprar con un clic sobre (A.12 <<extend>>) “Añadir la cesta”.
3. El sistema redirige al actor a una nueva página.
4. El sistema le muestra al actor una página donde se muestra el artículo a comprar.
5. El actor escribe la cantidad que desea comprar del artículo.
6. El actor hace clic sobre el botón “Aceptar”.
7. El sistema manda un mensaje al manejador de base de datos por disponibilidad de artículos para guardado del mismo.
8. El sistema envía confirmación de que el artículo (A.14 <<extend>>) existe en la tienda.
9. El sistema muestra el mensaje “Artículo agregado exitosamente al carrito”.
10. El actor hace clic sobre el botón “Aceptar”.
11. El actor se dirige a “Lista de carrito de compras” para (A.4 <<include>>) mostrar el carrito de compras y proceder con la orden.
12. El actor selecciona “Ordenar ahora” para concluir con la compra.

Flujos Alternativos:

En 6 (Si el actor desea dejar de ver el artículo seleccionado).

6.1. El usuario hace clic en “Cancelar” o en la barra superior, en regresar.

6.2. Regresa a 1.

En 9 (Si no hay artículos en almacén).

9.1. El sistema le muestra al actor un mensaje de “Insuficientes artículos en almacén”.

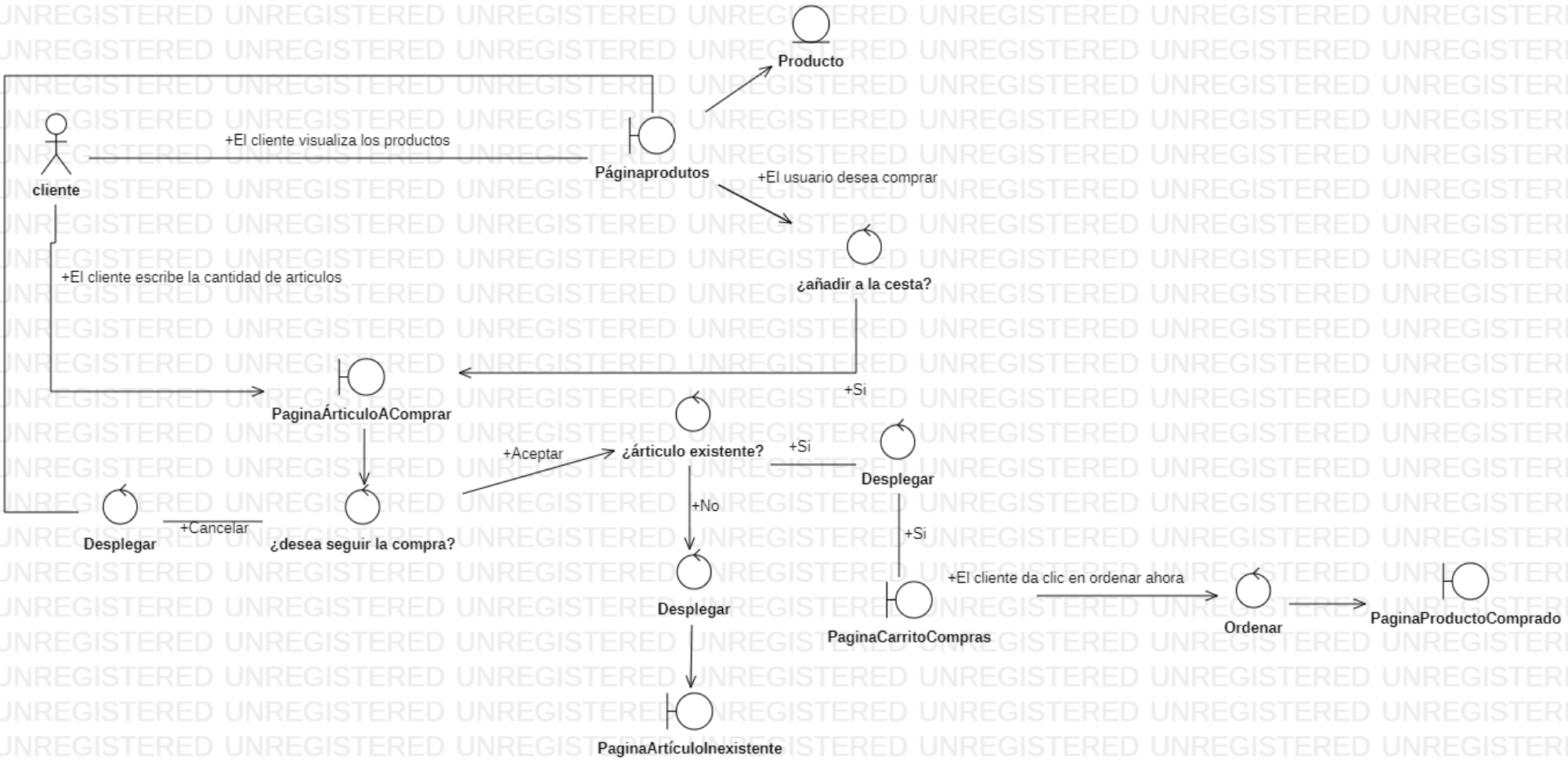


Diagrama de robustez de “Realizar compras”

En el diagrama de robustez “Realizar compras” nos explica que cuando el cliente (una vez iniciada su sesión) desea realizar la compra de un artículo o producto, puede seleccionar el artículo entre muchos en la página en línea, agregarlo a sus artículos de carrito, quitar artículos de carrito, dar descripción y características de su pedido, quitar características de su pedido, realizar compra o cancelar compra.

Mostrar carrito.

A.4 Caso de uso “Mostrar carrito”

Flujos Principales:

1. El sistema le muestra al actor la página de la tienda virtual.
2. El actor da un clic en el apartado de “Lista de carritos de compra” de la barra de menú lateral de la página.
3. El sistema redirecciona al actor al apartado de “Lista de carrito de compras”.
4. El actor visualiza su carrito de compras.

Flujos Alternativos:

En 4 (Si el actor desea eliminar el artículos).

4.1. El actor selecciona “Remover el artículo”.
4.2. El sistema lanza una ventana emergente para que decida eliminar el artículo.
4.3. El actor hace clic en “Aceptar”.
4.4. El sistema muestra la página lista del carrito de compras con el artículo eliminado.
4.5 Regresa al punto 3.

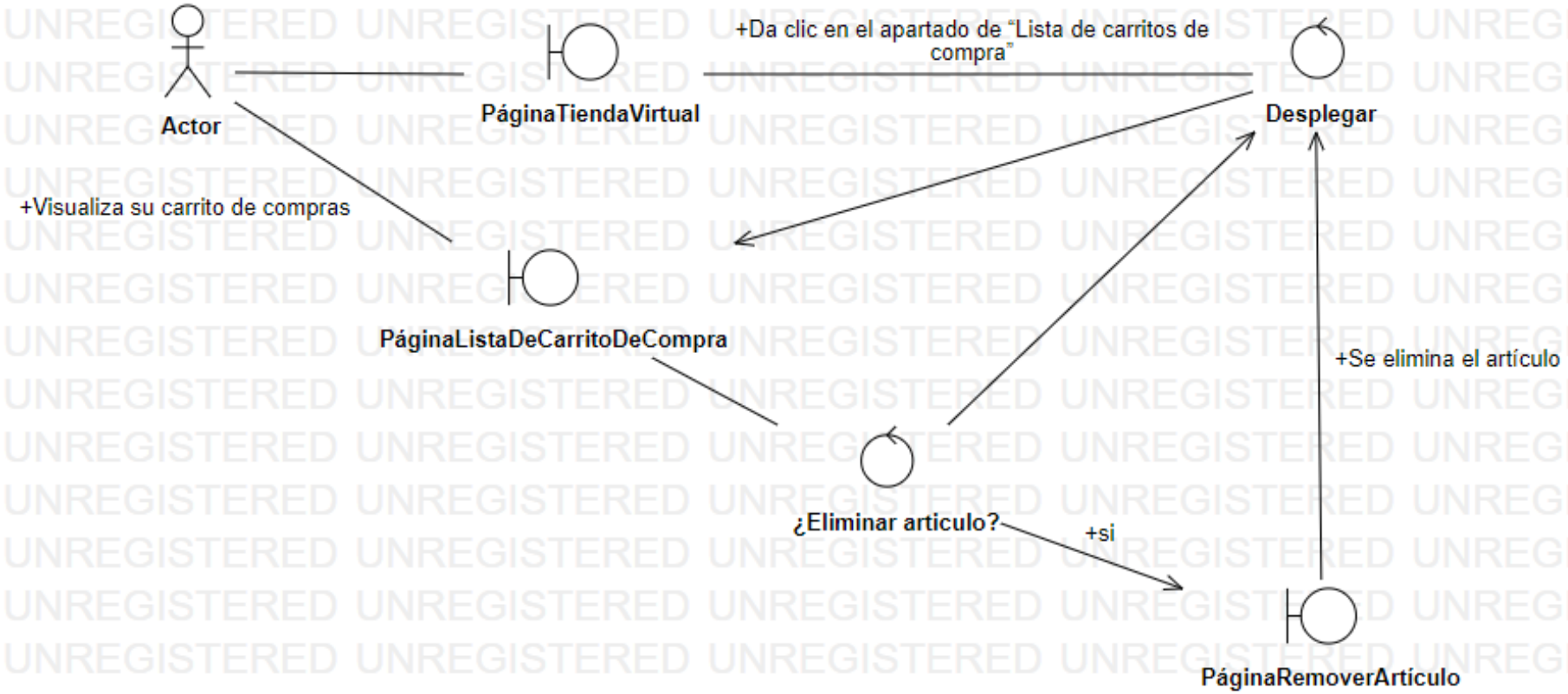


Diagrama de robustez de "Mostrar carrito"

En el diagrama de robustez “Mostrar carrito” nos señala que cuando el cliente una vez realizada la compra, este va a la “Lista de carritos de compras” para que pueda verificar que su pedido esté en orden y así pueda hacer la compra de manera segura.

Cerrar sesión

A.6 Caso de uso “Cerrar sesión”

Flujos Principales:

1. El sistema muestra la ventana correspondiente al apartado de la tienda que esté en ese momento el actor.
2. El actor visualiza en la parte inferior de la barra lateral, la opción de “Cerrar sesión”.
3. El actor da clic sobre “Cerrar sesión”.
4. El sistema devuelve el cierre de sesión.
5. El sistema le muestra al actor la página principal.

Flujos Alternativos:

En 2 (Otro camino para cerrar sesión).

2.1. El actor visualiza la barra en la parte superior derecha.
2.2. El actor presiona al correo de la parte superior derecha.
2.3. El actor presiona donde dice “Cerrar sesión”.
2.4 Regresa al paso 4.

En 3 (Si el actor no da el clic en cerrar sesión).

3.1. La cuenta permanece activa dentro de la página correspondiente a su objetivo cumplido.

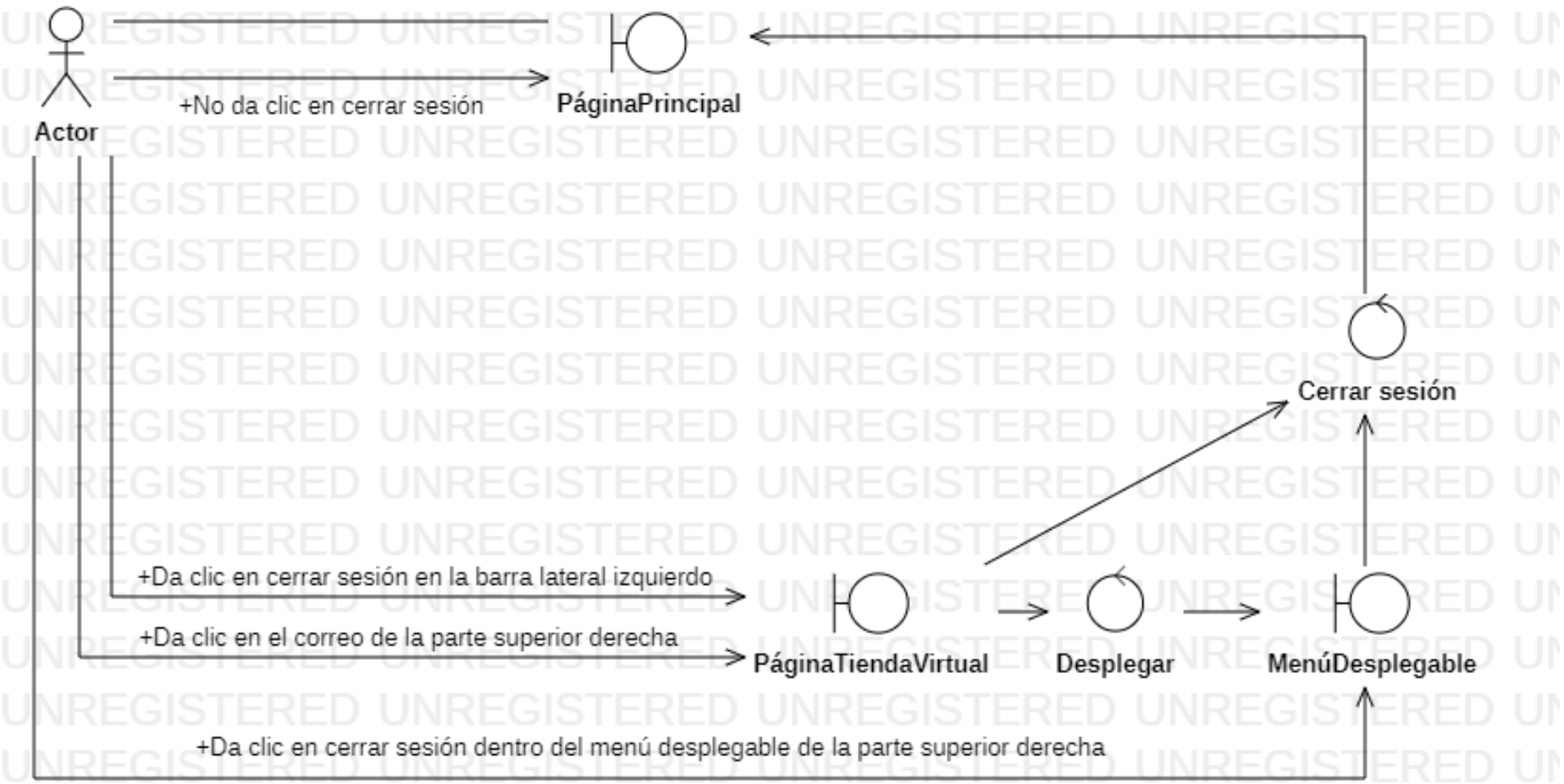


Diagrama de robustez de “Cerrar sesión”

En el diagrama de robustez “Cerrar sesión” nos muestra que cuando el cliente desea cerrar su sesión dentro de la página online “Cambios Gamer”, este lo puede realizar de dos maneras, una es estando en cualquier página se dirige a la barra lateral inferior y darle clic en el botón y la otra es en la barra superior derecha en la parte de su correo, esto es que donde en cualquiera existe la opción de cerrar sesión para que el cliente seleccione esta y lo saque de su sesión en la página.

3. Diagramas de secuencia

Los diagramas de secuencias permiten capturar el orden de las interacciones entre las diferentes partes del sistema. Usando un diagrama de secuencia podemos describir cuáles interacciones se activarán cuando se ejecute un caso de uso específico y en qué orden ocurrirán esas interacciones.

Los diagramas de secuencias pueden ser referencias útiles para las empresas y otras organizaciones, sus beneficios son:

- Representar los detalles de un caso de uso.
- Modelar la lógica de una operación, una función o un procedimiento sofisticados.
- Ver cómo los objetos y los componentes interactúan entre sí para completar un proceso.
- Planificar y comprender la funcionalidad detallada de un escenario actual o futuro.

Para la realización de los diagramas de secuencia ocuparemos la herramienta de StarUML, al igual utilizaremos los casos de usos con sus flujos principales y alternativos, y sus diagramas de robustez correspondiente al caso de uso.

En nuestro proyecto tomamos en cuenta los 5 casos de usos:

1. “Inscribir clientes”
2. “Iniciar sesión”
3. “Realizar compras”
4. “Mostrar carrito”
5. “Cerrar sesión”

Estos casos de usos están dados con sus flujos principales y alternativos junto con los diagramas de robustez en el paso (2. Diagramas de robustez).

A continuación, mostraremos los diagramas de secuencias de cada uno de los casos de usos descritos con anterioridad.

Diagrama de secuencia “Inscribir clientes”

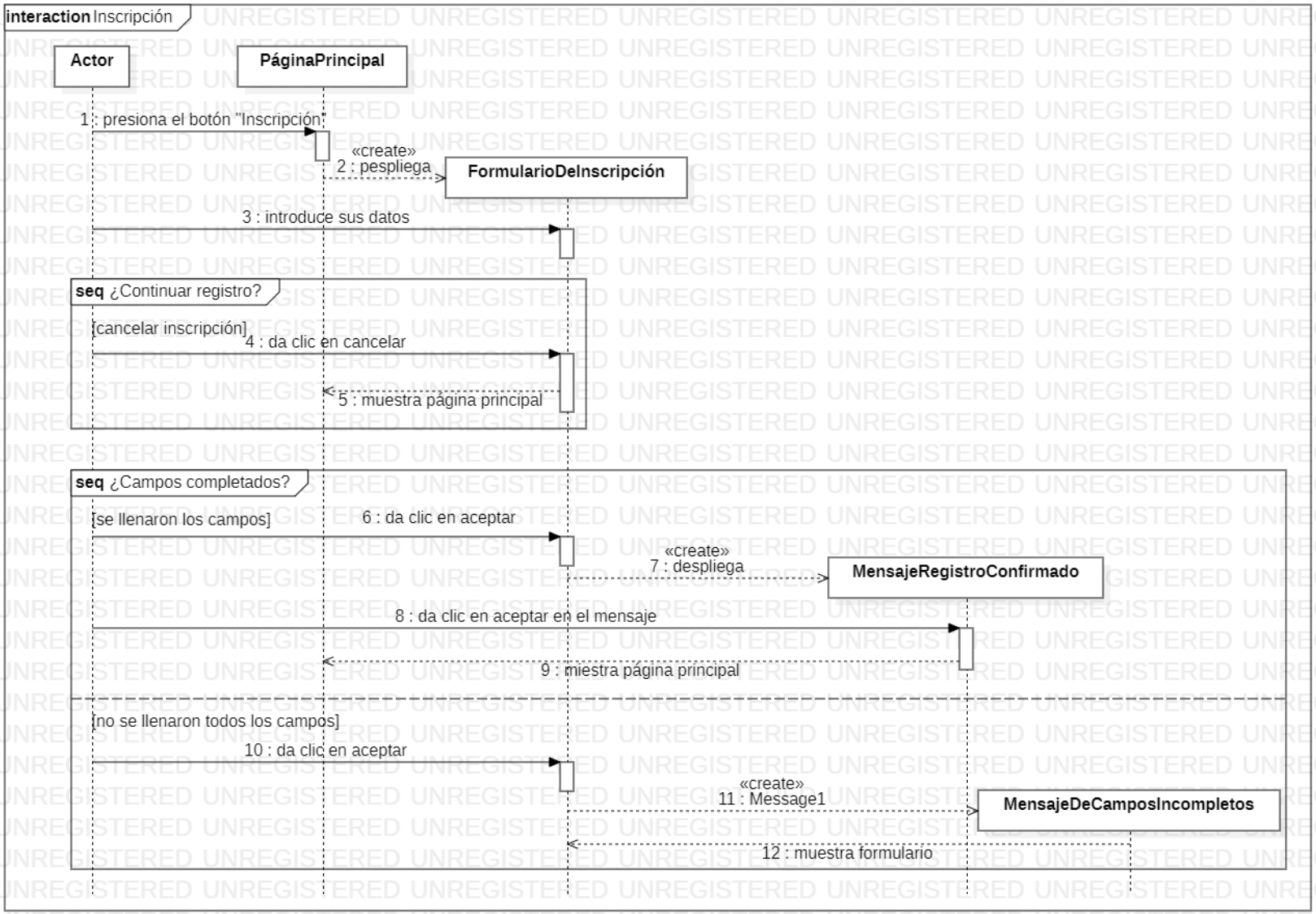


Diagrama de secuencia “Iniciar sesión”

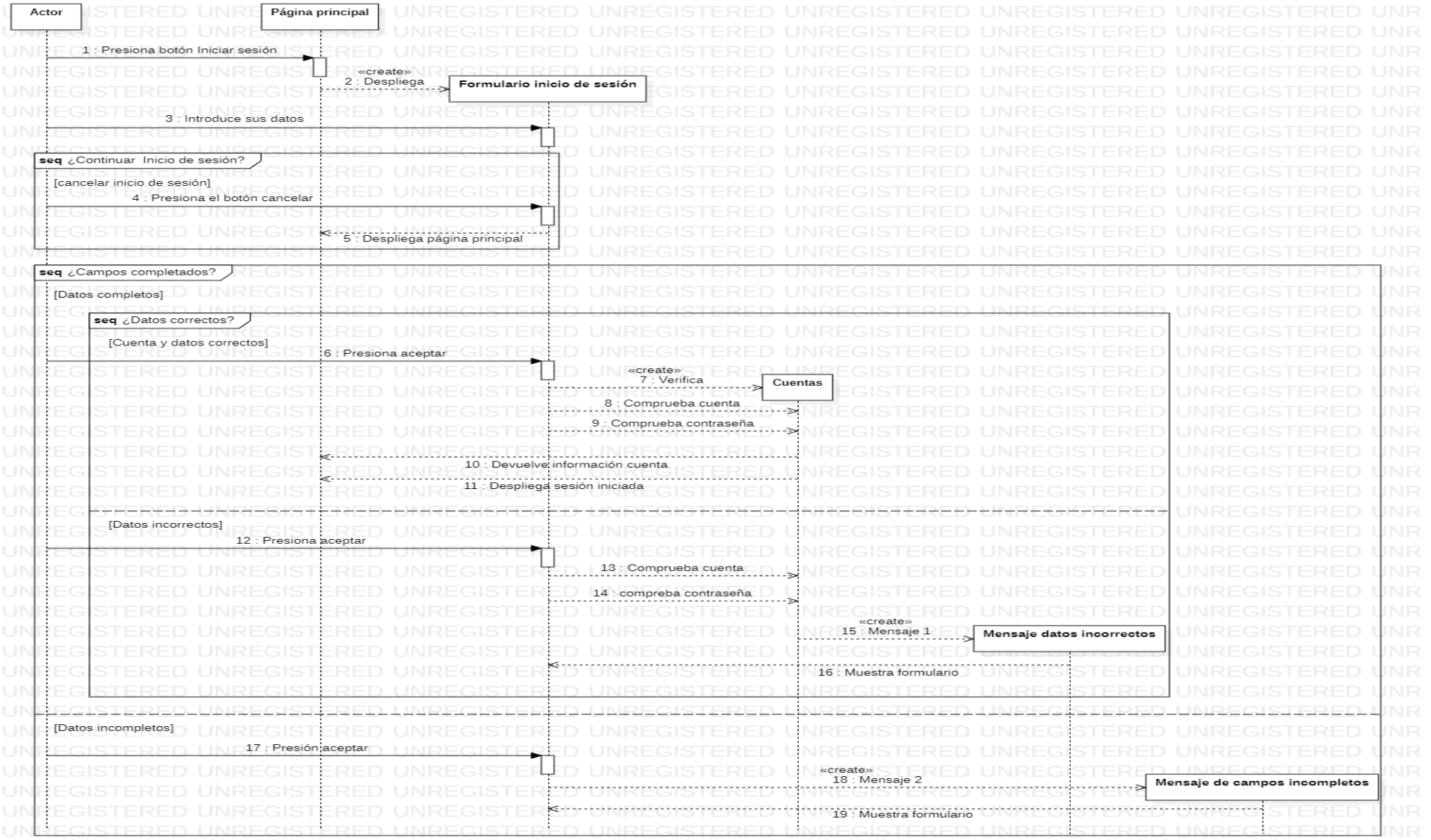


Diagrama de secuencia “Realizar compras”

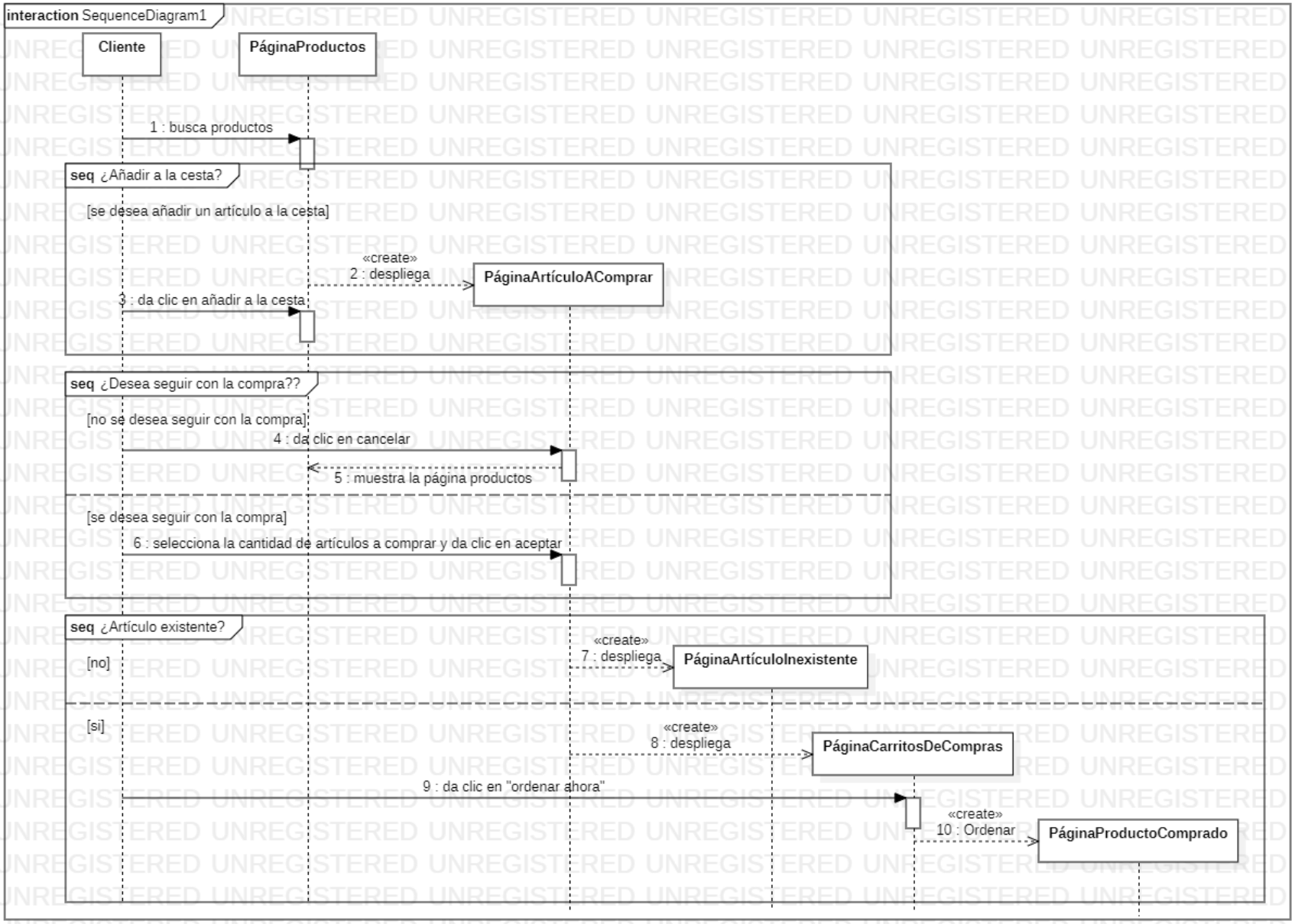


Diagrama de secuencia “Mostrar carrito”

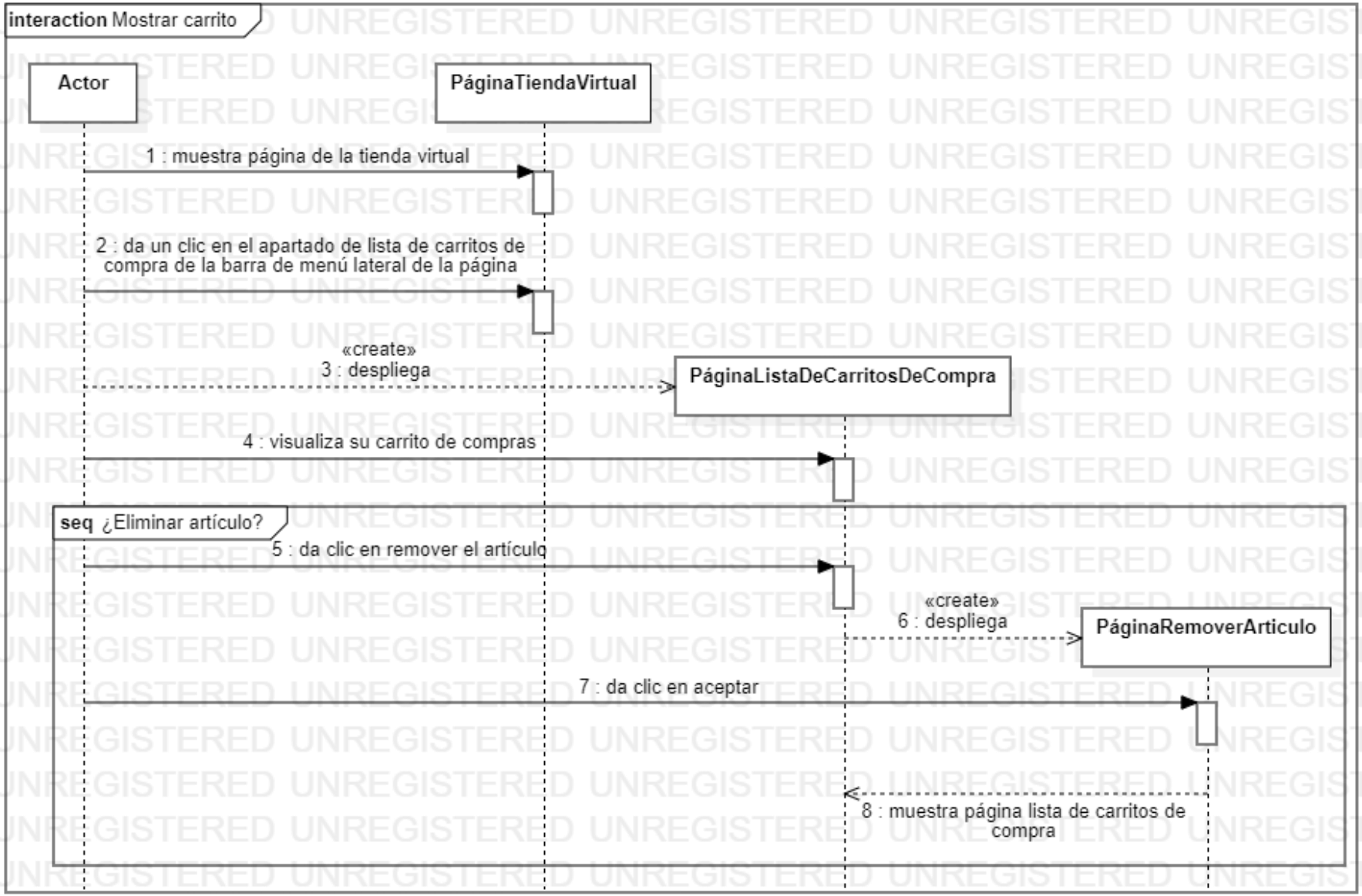
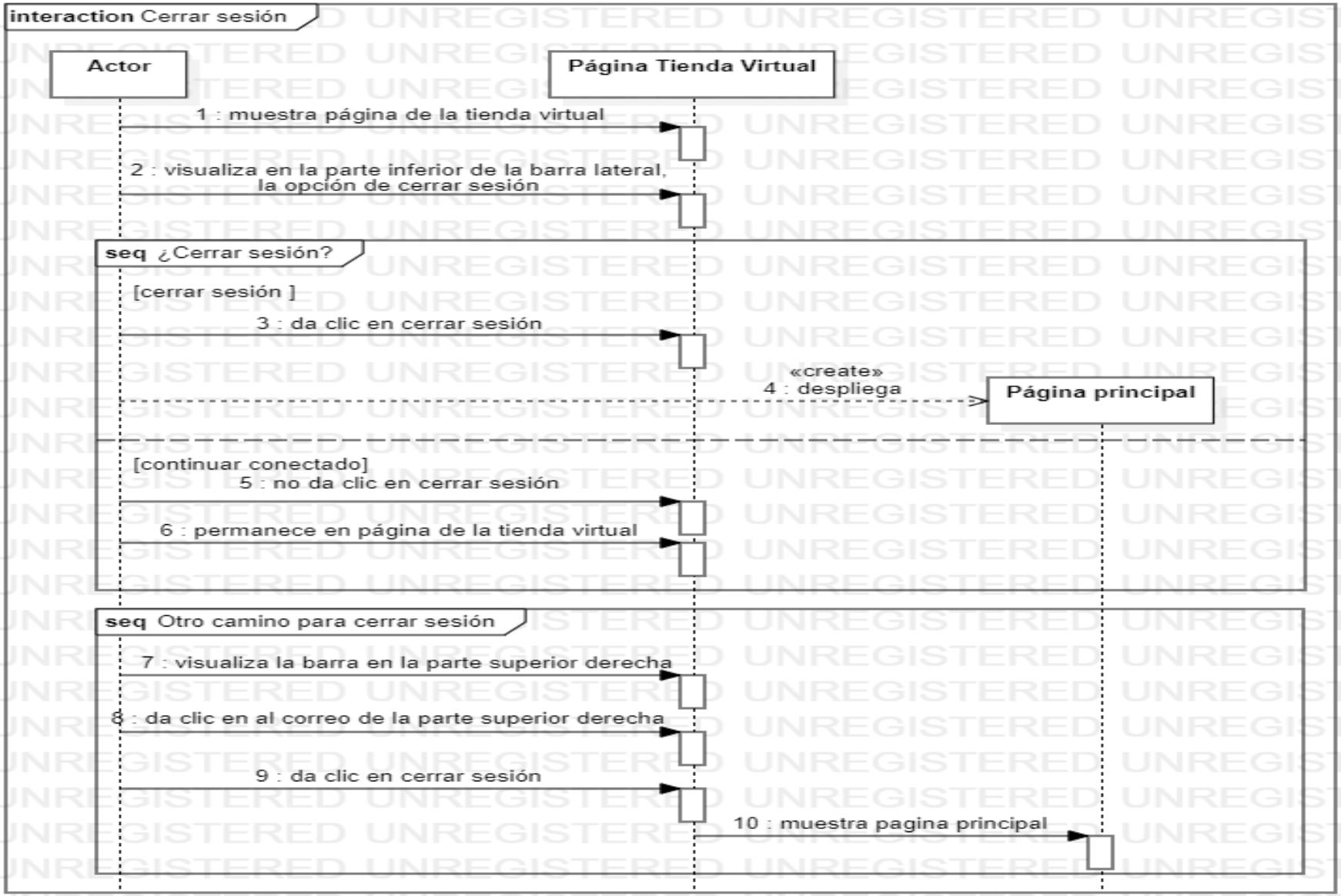


Diagrama de secuencia “Cerrar sesión”



4. Diagrama o diagramas de clases

Los diagramas de clases son una técnica central y ampliamente difundida en los distintos métodos orientados a objetos, cada método incluye sus propias variantes a esta técnica.

Los diagramas de clases describen los tipos de objetos de un sistema, así como los distintos tipos de relaciones que pueden existir entre ellos.

Este tipo de diagrama nos da el modelado conceptual de un sistema de software, la cual suele recoger los conceptos necesarios para las variables requeridas y su funcionamiento.

Los elementos principales de un diagrama de clases son las relaciones entre clases dadas entre sí, de forma que las clases y sus principales relaciones son elementos esenciales de este diagrama, las clases contienen (atributos y operaciones), las asociaciones pueden ser de 1 a muchos (1:n), 1 a 1 (1:1), de 0 a 1 (n:1), o de muchos a muchos (n:m), las agregaciones es la relación de parte-de, que presenta a una entidad como un agregado de partes y por último la herencia es la relación de generalización entre clases.

Para nuestro proyecto de la tienda online “Cambios Gamer” ocupamos el modelo vista-controlador, diseño del MVC y Implementations DAO.

Contiene las entidades de: Cliente, Administrador, Productos y Orden; también, realizamos una implementación para los mismos, finalizando con el diagrama de clases que contiene lo requerido para el sistema.

Modelo Vista-Controlador (MVC)

Modelo Vista-Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, estos tres componentes son conocidos como modelo, vista, controlador. Gracias al MVC se pueden definir componentes para la representación de la información, y por otro lado para la interacción del usuario.

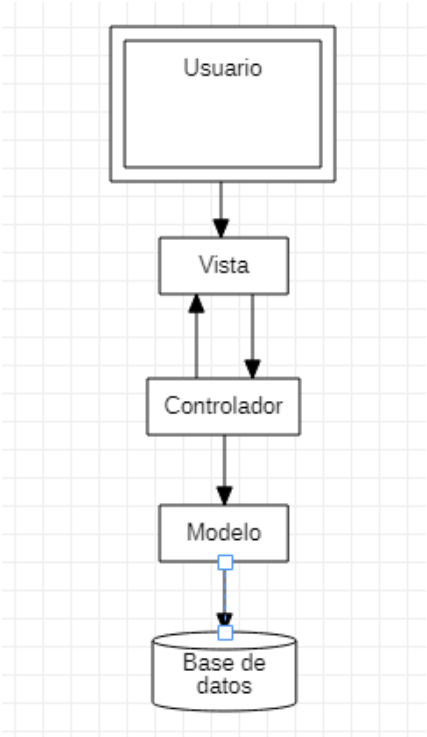
En el apartado de **modelo** se trabaja con datos, eso quiere decir que contendrá mecanismos para acceder a la información y también para actualizar su estado. (Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas).

En la **vista** o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con éste en un formato adecuado para interactuar, por lo tanto, requiere de la información que debe representar como salida.

El **controlador** actúa como intermediario entre el modelo y la vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Diseño del MVC

El modelo funciona en base a la interacción del usuario con la vista, la primera interacción o movimiento lo vamos a tener con el usuario y la vista. Esta instrucción que el usuario realiza en la vista (En la interfaz de usuario), y el controlador se va a encargar de mediar la interacción entre el modelo y la vista para ser guardada en la base de datos.

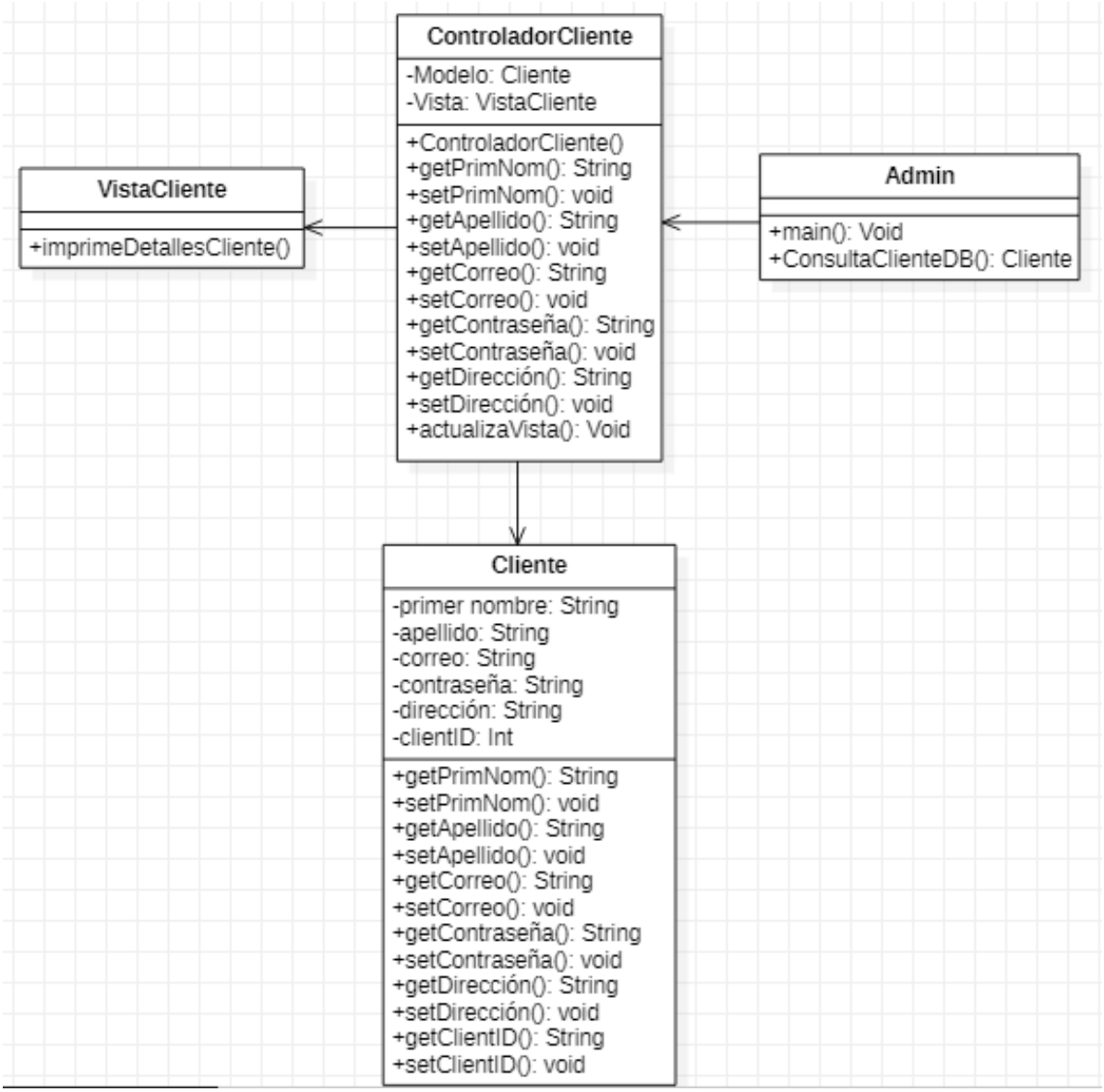


Implementación del MVC en nuestro sistema de la tienda online “Cambios Gamer”

Entidad: Cliente (Modelo)

En este modelo de cliente observamos que hay una relación de ControladorCliente con las 3 tablas (VistaCliente, Cliente, Admin) las cuales comparten datos entre ellas.

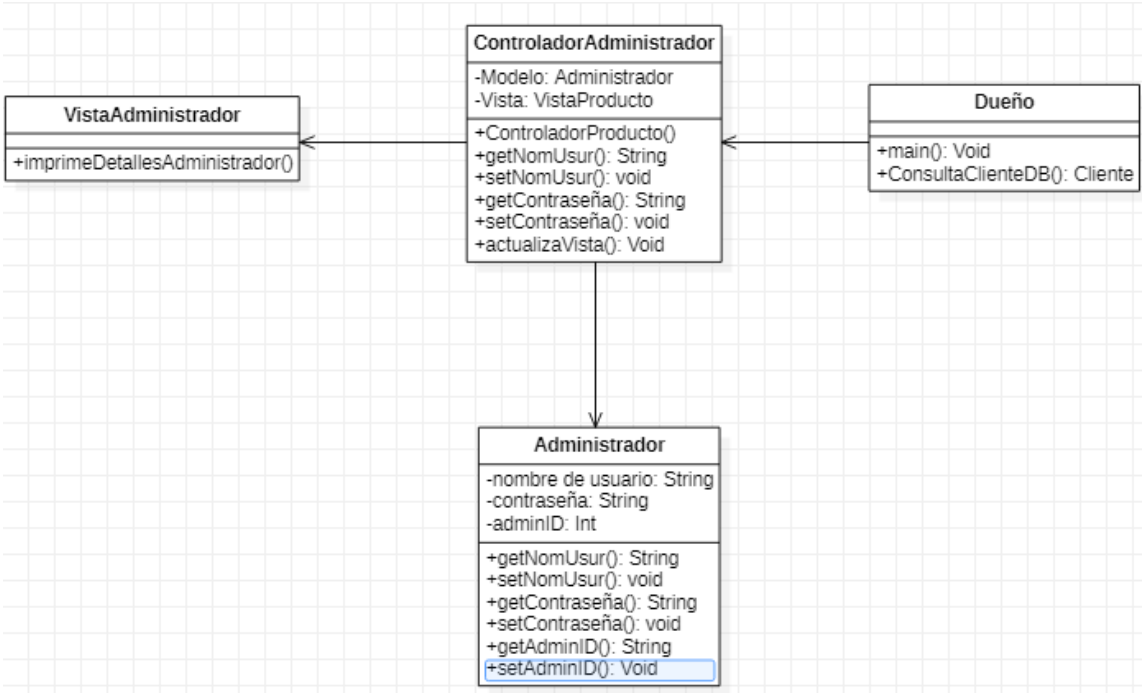
Este modelo nos permite visualizar los datos a requerir de un cliente y administrador, y los datos que interactúan con la tabla principal ControladorCliente.



Entidad: Administrador (Modelo)

En este modelo de administrador podemos observar que esta nuestra tabla ControladorAdmistrador que contiene datos a relacionar con las 3 tablas (VistaAdministrador, Dueño, Administrador).

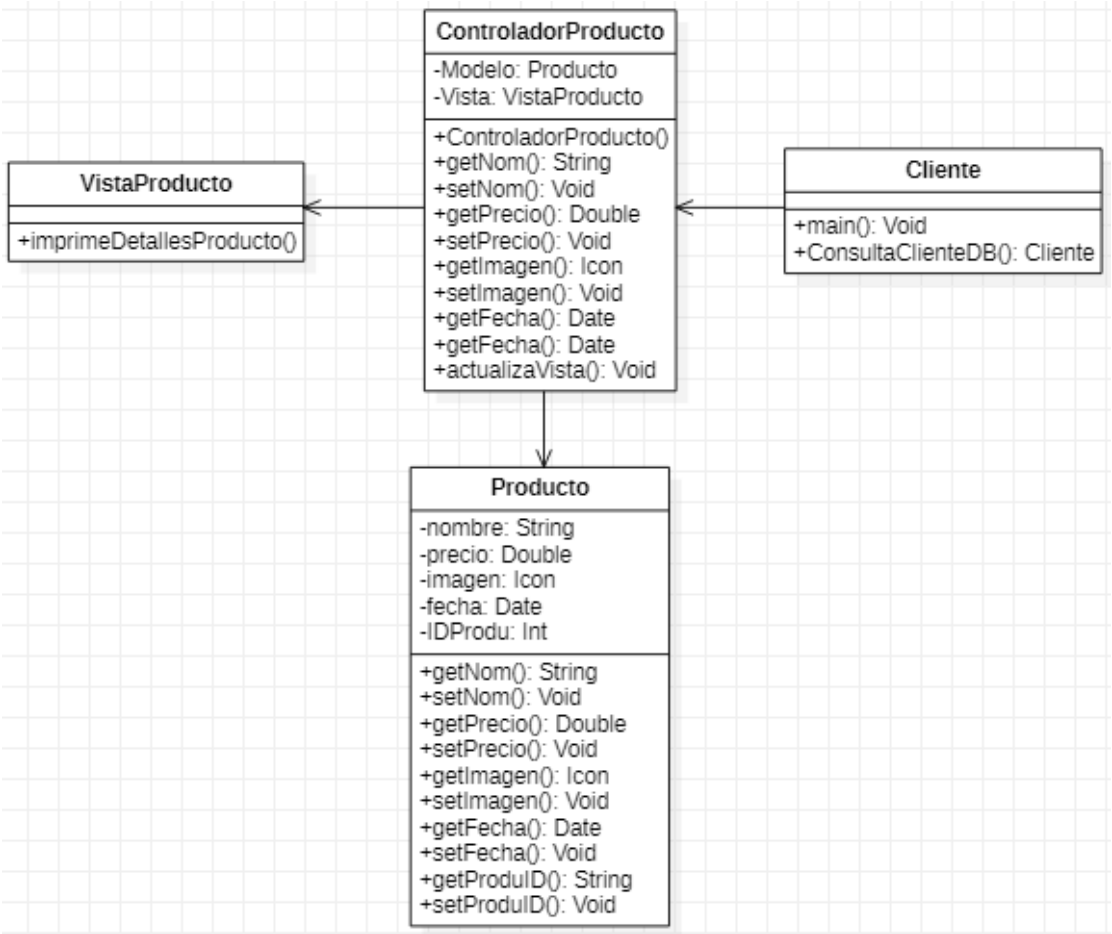
Este modelo es principalmente para los datos de un administrador podemos ver que está su nombre de usuario, contraseña y ID; también, las interacciones de los datos para las tablas dueño y VistaAdministrador.



Entidad: Producto (Modelo)

En este modelo de producto podemos observar la interacción de datos dentro de la tabla ControladorProducto con las 3 tablas (Cliente, Producto, VistaProducto).

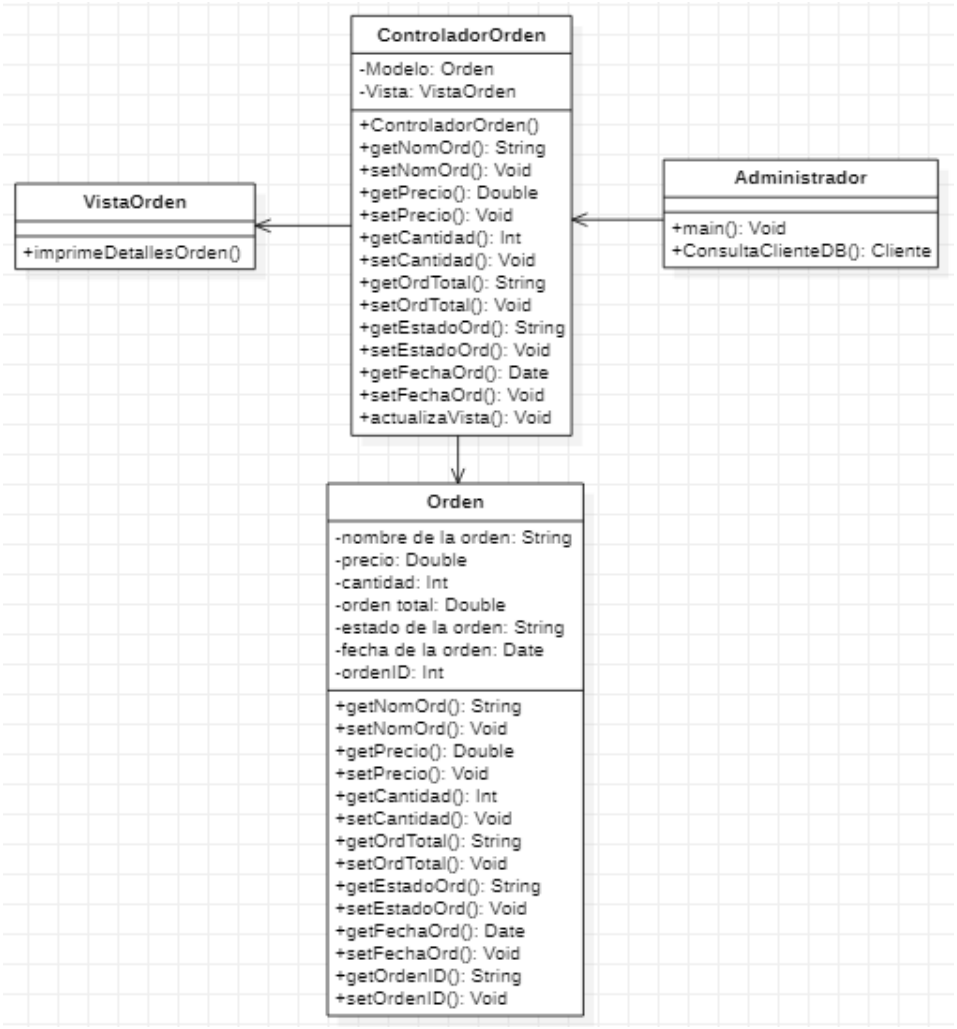
Este modelo contiene datos de los productos a vender dentro de la tienda online, contiene los nombres, imagen del producto, precio e imagen.



Entidad: Orden (Modelo)

En este modelo de orden podemos observar la interacción de datos dentro de la tabla ControladorOrden con las 3 tablas (Administrador, Orden, VistaOrden).

Este modelo contiene datos de las compras realizadas en la tienda online para el envío y distribución del mismo, por ello es uno de los modelos más importantes, este modelo contiene datos como: fecha, precio, cantidad, estado, nombre de orden.



DAO

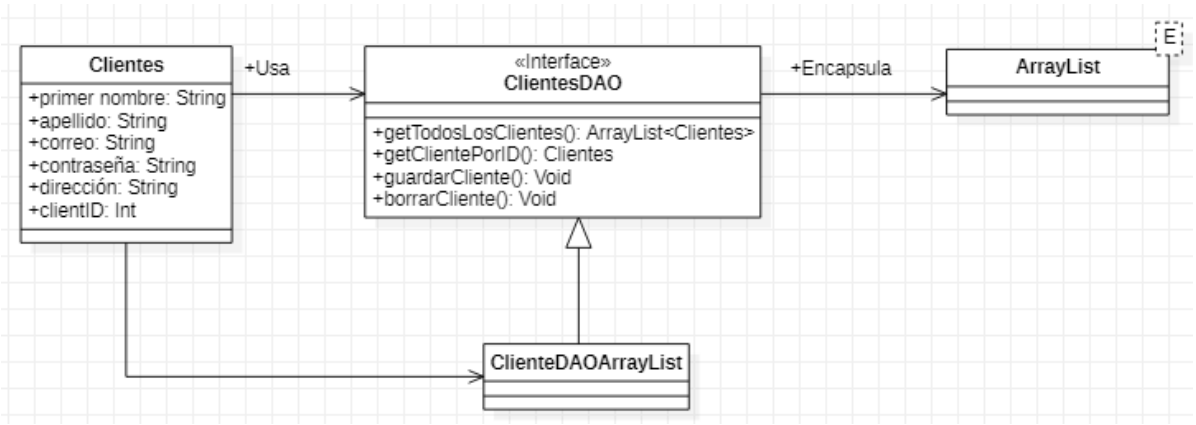
El patrón **Data Access Object** (DAO) propone separar por completo la lógica de negocio de la lógica para acceder a los datos, de esta forma el **DAO** proporcionará los métodos necesarios para insertar, actualizar, borrar y consultar la información; por otra parte, la capa de negocio sólo se preocupa por lógica de negocio y utiliza el **DAO** para interactuar con la fuente de datos.

El patrón **Data Access Object** (DAO) es fácil de implementar y proporciona claros beneficios, incluso, si sólo tenemos una fuente de datos y esta no cambia, pues permite separar por completo la lógica de acceso a datos en una capa separada, y así, sólo nos preocupamos por la lógica de negocio sin preocuparnos de dónde viene los datos o los detalles técnicos para consultarlos o actualizarlos.

Implementación de DAO

Para Cliente

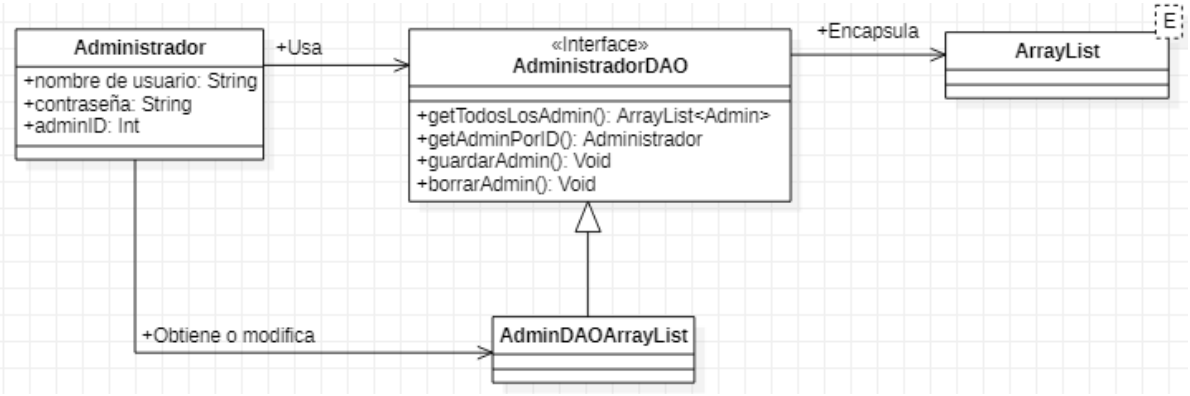
Tenemos nuestra tabla clientesDAO la cual tiene la interacción con 3 tipos de tablas (Clientes, ClienteDAOArrayList y ArrayList), contiene los datos principales del cliente y un ID con el que lo identificamos, esto se encapsula en la ArrayList y esto se modifica en ClienteDAOArrayList.



Dentro de nuestros diagramas con el patrón DAO seguimos trabajando con los clientes ya que estos son parte de nuestra colección de datos, datos en el sistema interactuarán con arraylists para ser almacenados en el sistema.

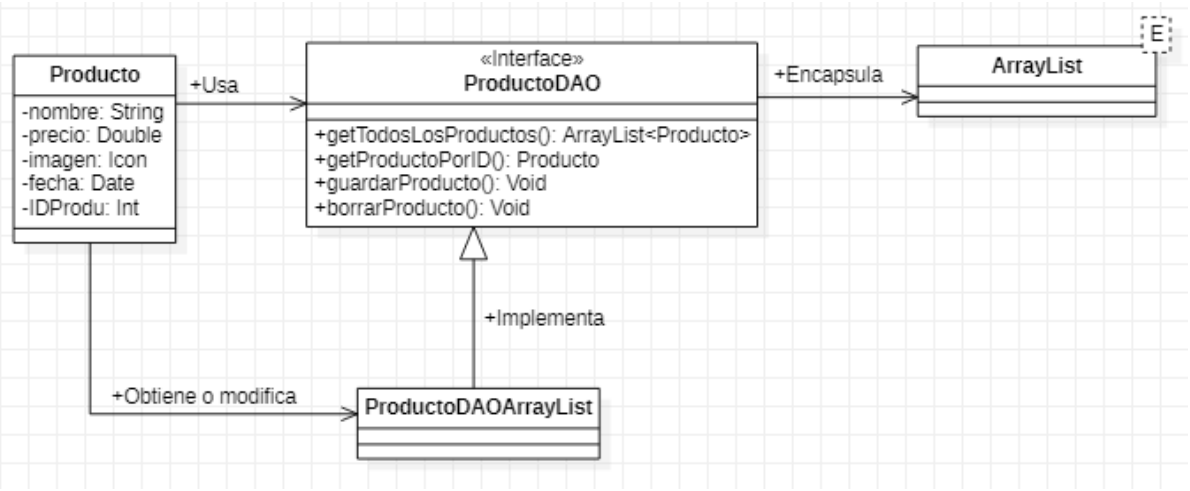
Para Administrador

Tenemos nuestra tabla AdmistradorDAO la cual tiene la interacción con 3 tipos de tablas (Administrador, AdministradorDAOArrayList y ArrayList), contiene los datos principales del Administrador y un ID con el que lo identificamos, esto se encapsula en la ArrayList y esto se modifica en AdministradorDAOArrayList.



Para Productos

Tenemos nuestra tabla AdmistradorDAO la cual tiene la interacción con 3 tipos de tablas (Producto, ProductoDAOArrayList y ArrayList), contiene los datos principales del Administrador y un ID con el que lo identificamos, esto se encapsula en la ArrayList y esto se modifica en ProductoDAOArrayList.



Para Órdenes

Tenemos nuestra tabla AdmistradorDAO la cual tiene la interacción con 3 tipos de tablas (Orden, OrdenDAOArrayList y ArrayList), contiene los datos principales del Administrador y un ID con el que lo identificamos, esto se encapsula en la ArrayList y esto se modifica en OrdenDAOArrayList.

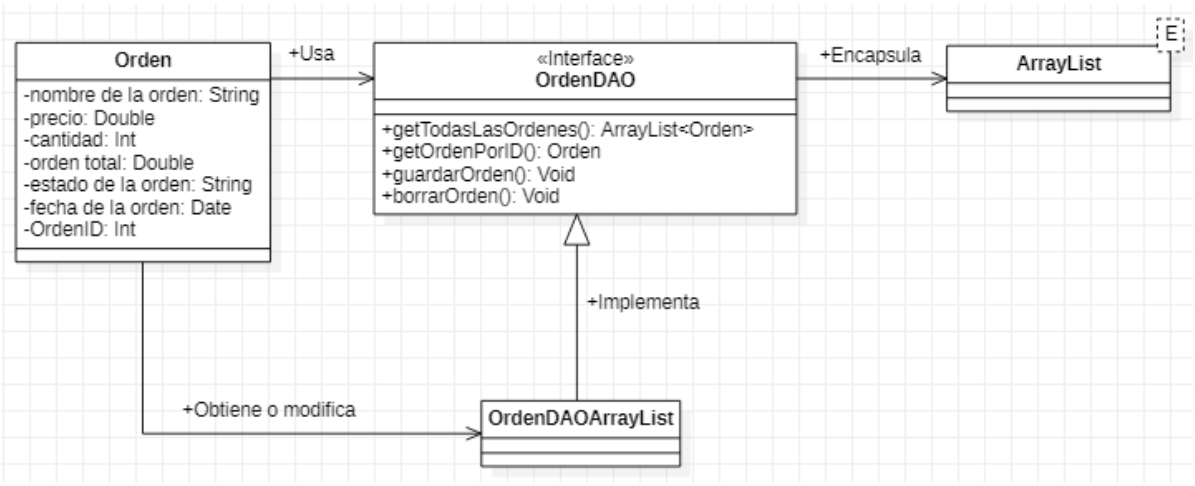
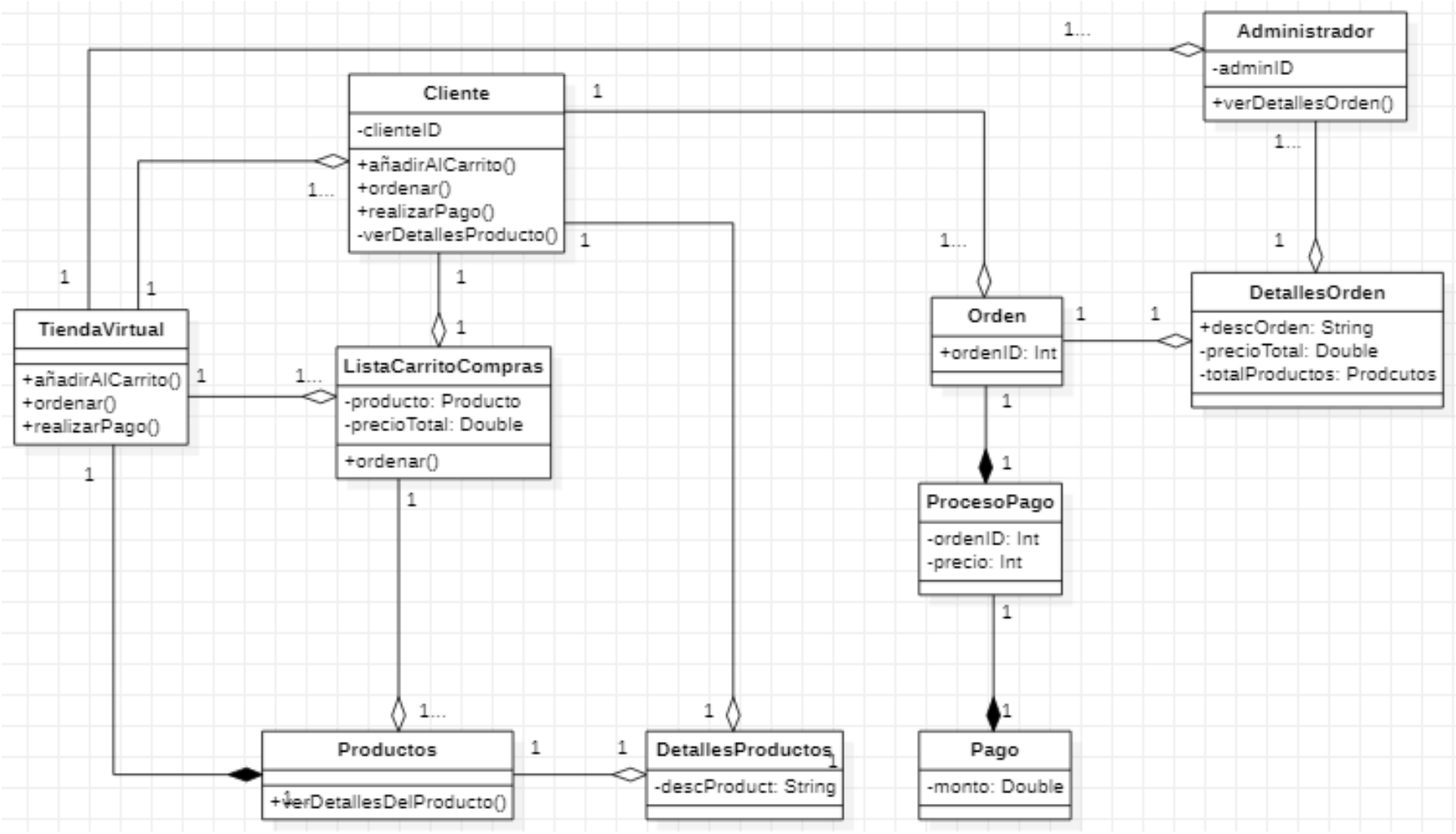


Diagrama de clases



5. Diagrama de paquetes

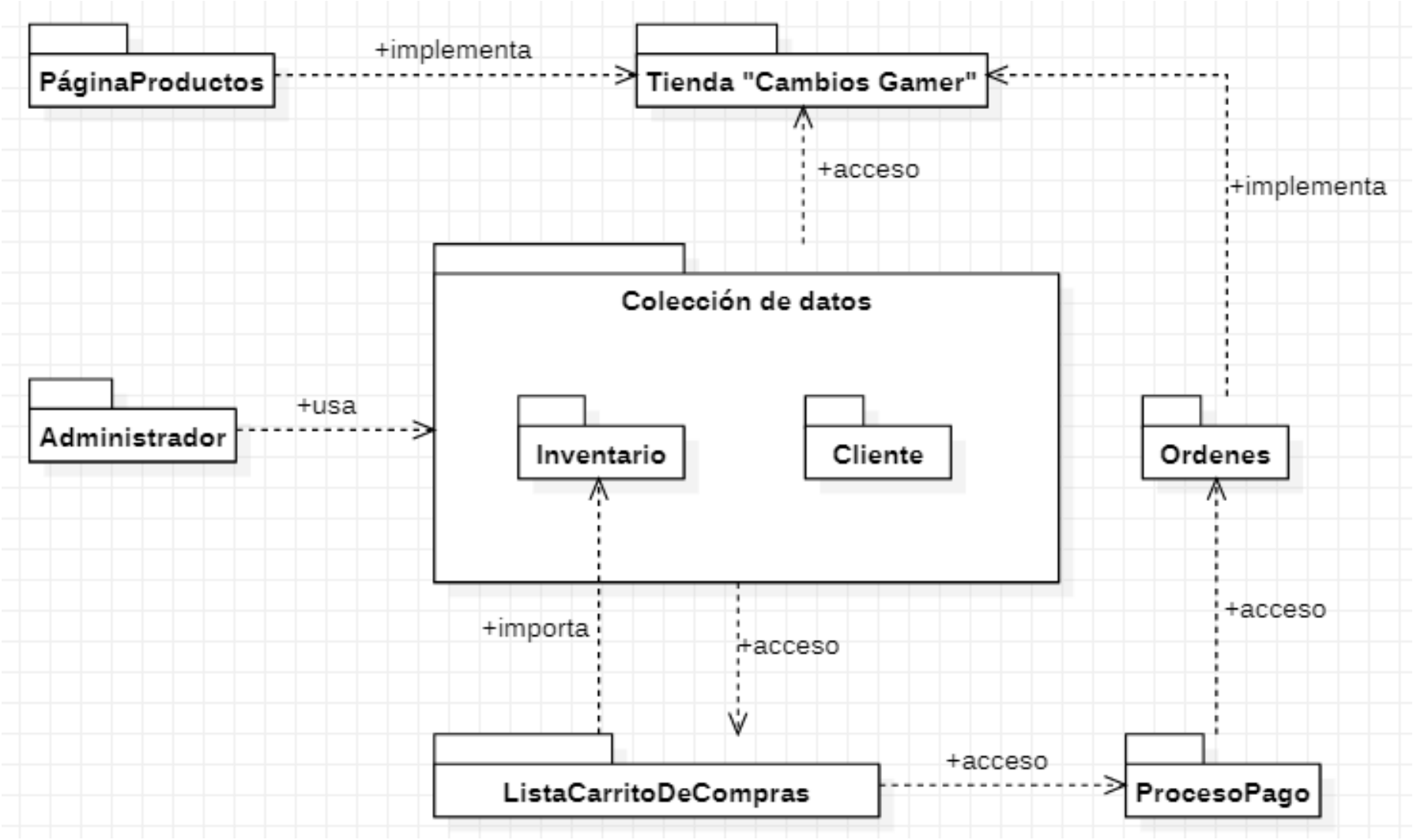
El diagrama de paquetes es uno de los diagramas estructurales comprendidos en UML 2.5, por lo que, como tal, representa de forma estática los componentes del sistema de información que está siendo modelado. Es utilizado para definir los distintos paquetes a nivel lógico que forman parte de la aplicación y la dependencia entre ellos. Es principalmente utilizado por desarrolladores y analistas.

Hablando estrictamente, los paquetes y las dependencias son elementos de un diagrama de clases, por lo cual un diagrama de paquetes es sólo una forma de un diagrama de clases.

Los paquetes son un elemento clave del diagrama, este da un nombre al mismo, podemos comprender que un paquete es un conjunto de elementos, estos paquetes pueden incluir más de un paquete.

Las dependencias de paquetes son aquellos que necesitan elementos de otro paquete para poder funcionar de manera óptima, este elemento se representa con flechas discontinuas que va desde el paquete que requiere la función hasta el paquete que ofrece esa función.

Para nuestra tienda online “Cambios Gamer” tomamos en cuenta los diagramas de clases y comprendimos la interacción de cada uno dentro de la tienda.


PáginaProductos
+implementa
Tienda "Cambios Gamer"
+acceso
+implementa
Colección de datos
Administrador
+usa
Inventario
Cliente
Ordenes
+importa
+acceso
+acceso
ListaCarritoDeCompras
+acceso
ProcesoPago

6. Diseño de la bases de datos

Una base de datos correctamente diseñada le permite obtener acceso a información actualizada y precisa.

El diseño de una base de datos es un proceso que se guía por varios principios bien definidos, partiendo de un dominio del cual se obtendrá un modelo conceptual, seguidamente un modelo lógico, al cual se le debe aplicar normalización y finalmente obtener un modelo físico y poder implementarlo.

El proceso de diseño de bases de datos consiste en definir la estructura lógica y física de una o más bases de datos para responder a las necesidades de los usuarios con respecto a la información y para un conjunto concreto de aplicaciones.

Mediante un proceso de diseño de bases de datos, se pueden decidir las tablas y relaciones que debe tener una base de datos determinada, los atributos de las diferentes tablas, las claves primarias y las claves foráneas que se deben declarar en cada tabla, etc. Todas estas tareas forman parte del proceso de diseño de bases de datos. Para poder tomar estas decisiones de la manera más correcta posible, hay que tener en cuenta las necesidades de información de los usuarios en relación con un conjunto concreto de aplicaciones.

En nuestro proyecto de tienda online “Cambios Gamer” modelamos un diseño de base de datos con sus respectivos atributos, los cuales son importantes para la comprensión de cómo nuestra tienda online contendrá datos esenciales como: el registro de usuarios, datos de usuarios, datos de productos, cantidad de productos y demás.

A continuación, observaremos la base de datos para nuestra tienda online “Cambios Gamer”.

MODELO ER

Los pasos del modelo ER son los siguientes:

PASO 1.- IDENTIFICAR CONJUNTOS DE ENTIDADES

En una **tienda de videojuegos** se interesa recoger en lo que a cada uno de sus clientes se refiere, el nombre, apellido paterno, dirección, e-mail y contraseña.

Para los **clientes**, la tienda de videojuegos dispone de varios artículos de varias marcas que pueden comprar con sus respectivos gustos. Asimismo, los clientes pueden revisar más a detalle las especificaciones de cada uno de los artículos que van a comprar como, por ejemplo: nombre, precio, cantidad, total y estado.

En los artículos se pueden escoger si están disponibles, estos se van actualizando cuando se vayan acabando.

Los **administradores de la tienda** son los encargados de administrar los artículos y que pueden hacer pedidos de otros artículos cuando ya estén agotados. Además, ellos pueden revisar los **detalles del pedido** para que así puedan ver las compras de los clientes que hicieron y que también puedan ver si hay artículos disponibles o sino hacer pedidos.

Los **artículos** que se encuentran en la tienda pueden ser de las consolas de videojuegos antiguas hasta las más recientes (hoy en día) y con sus respectivos controles, además pueden encontrar stickers para ellas mismas, es decir, las consolas de videojuegos y sus controles, pero no sólo stickers sino también las fundas para que estos no se dañen y las gomitas para los joysticks para que así sea más fácil de mover las palancas.

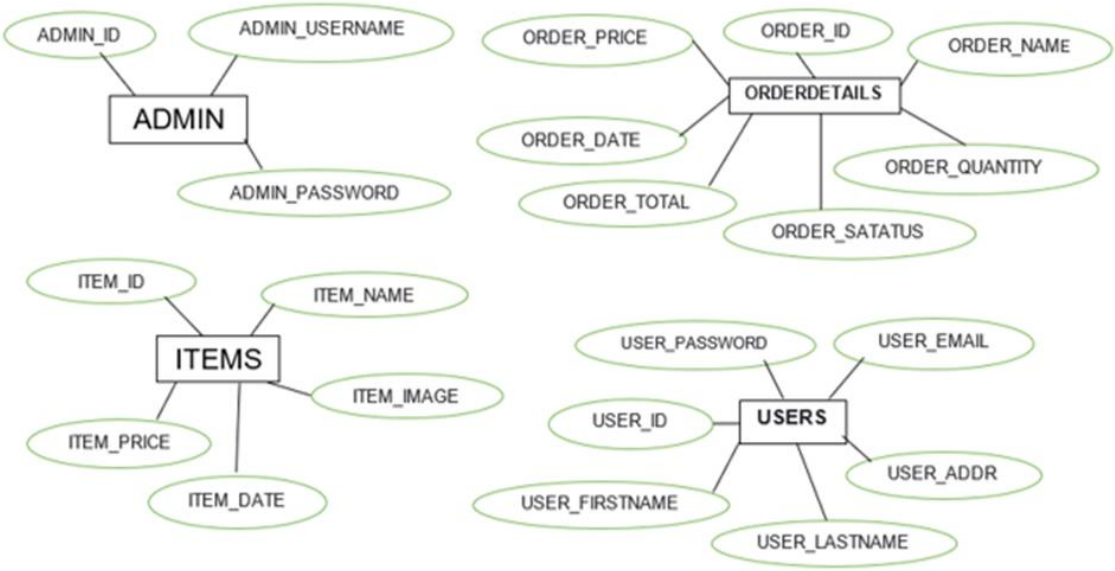
También pueden encontrar una gran variedad de audífonos como, por ejemplo: audífonos con bluetooth (diadema y sin diadema) para que así puedan escuchar música a su gusto y hay varios cables de todo tipo como los sencillos hasta los más complicados, esto es como una extensión eléctrica hasta cables HDMI, etc. En otras palabras, los clientes pueden ver las características de todos los artículos que estén disponibles en la tienda.

Si los clientes se equivocan al hacer sus pedidos se les puede reembolsar su dinero y los artículos para que así no haya problemas.

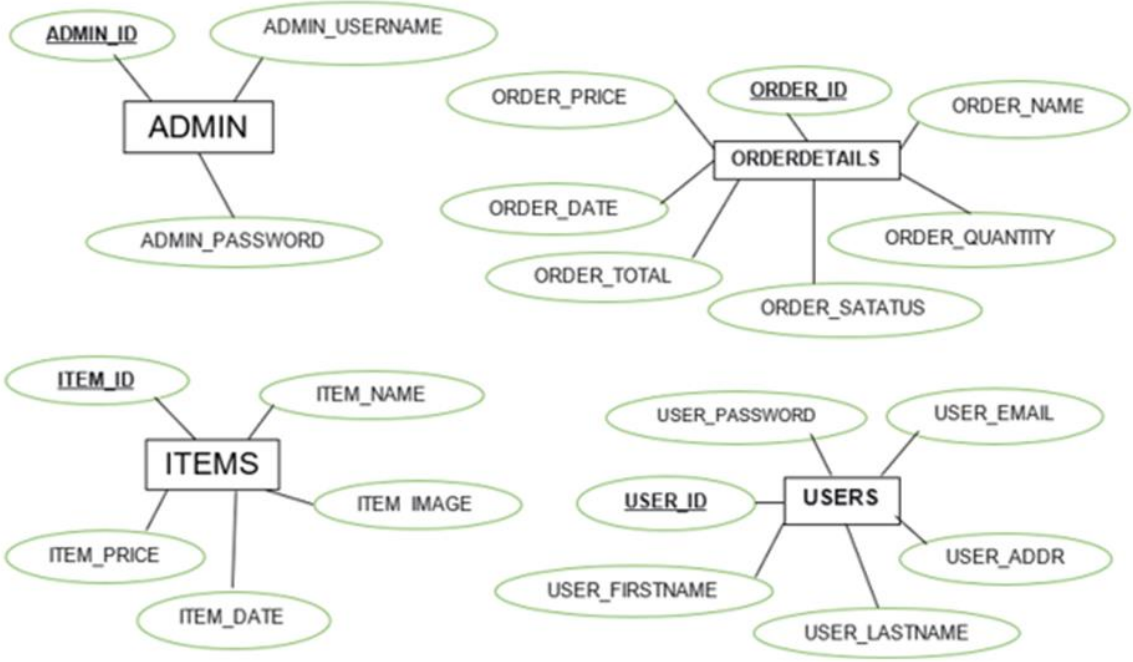
Los clientes deben de entender que los precios de los artículos están en pesos mexicanos y no en otro, ya que así pueden estar confundidos al momento de hacer la compra.

Candidato a conjunto de entidades	¿Existe más de una entidad?	¿Almacena información?	Acción
Tienda de videojuegos	✗	✓	_____
Clientes	✓	✓	Se toma como conjunto de entidades
Artículos	✓	✓	Se toma como conjunto de entidades
Administradores	✓	✓	Se toma como conjunto de entidades
Detalles del pedido	✓	✓	Se toma como conjunto de entidades

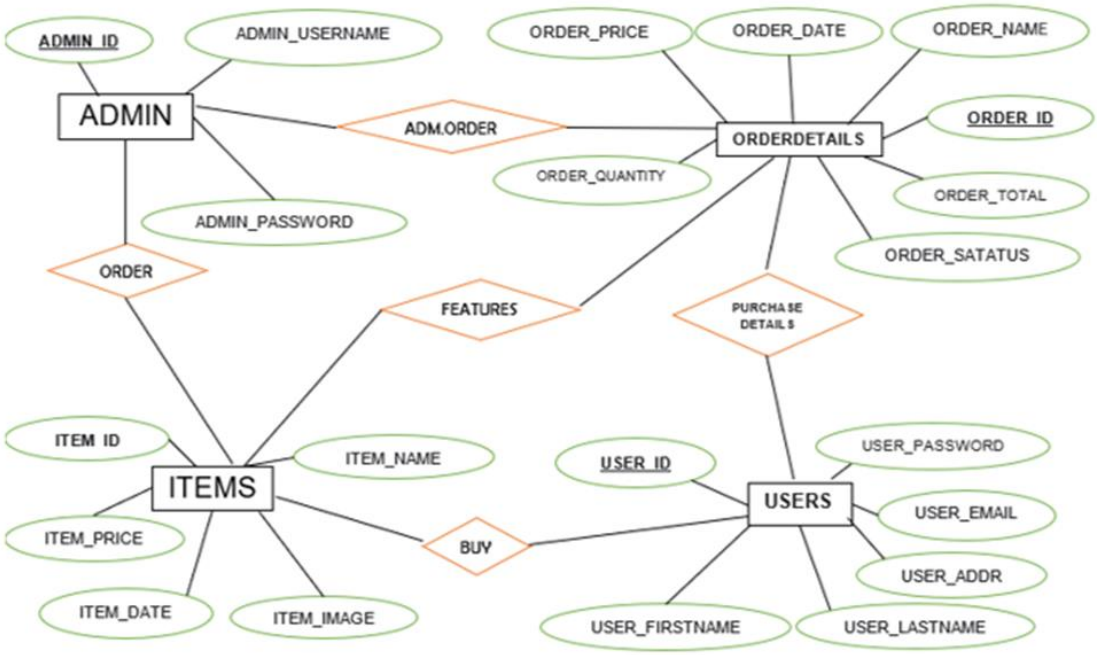
PASO 2.- IDENTIFICAR ATRIBUTOS BÁSICOS



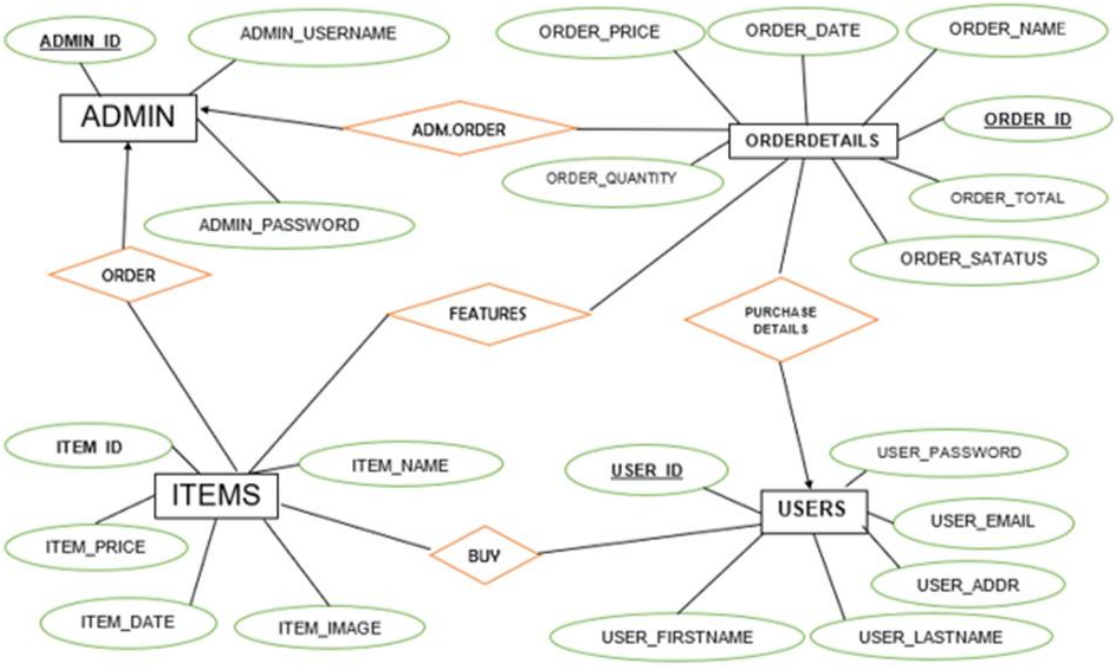
PASO 3.- IDENTIFICACIÓN DE LAS LLAVES RELACIONALES



PASO 4.- IDENTIFICAR CONJUNTOS DE RELACIONES



PASO 5.- IDENTIFICAR LA CARDINALIDAD DE LAS ASOCIACIONES BINARIAS



PASO 6.- REVISAR ASOCIACIONES

I. **Revisar asociaciones de muchos a muchos**
   - ITEMS-USERS, es una asociación que aparece con una tabla extra en la base de datos.
   - ITEMS-ORDERS, es una asociación que aparece con una tabla extra en la base de datos.

II. **Resolver asociaciones uno a uno**
   - No hay asociaciones de UNO-UNO.

ESQUEMA RELACIONAL

ADMINS={ ADMIN_ID , ADMIN_USERNAME, ADMIN_PASSWORD}

ORDERDETAILS={ ORDER_ID, ORDER_NAME, ORDER_PRICE, ORDER_QUANTITY, ORDER_DATE, ORDER_TOTAL**,** ORDER_SATATUS**,** ADMIN_ID, USER_ID**}**

USERS={ USER_ID, USER_EMAIL, USER_PASSWORD**,** USER_ADDRES**,** USER_LASTNAME**,** USER_FIRSTNAME}

ITEMS={ ITEM_ID, ITEM_NAME**,** ITEM_PRICE**,** ITEM_DATE**,** ITEM_IMAGE**,** ADMIN_ID}

FEATURES={ ITEM_ID, ADMIN_ID , ORDER_ID, USER_ID}

BUY={ ITEM_ID, ADMIN_ID **,** USER_ID}

CREATE TABLE admin (

admin_id INT(10) PRIMARY KEY AUTO_INCREMENT,

admin_username VARCHAR(500) NOT NULL DEFAULT '',

admin_password VARCHAR(500) NOT NULL DEFAULT '',

) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

CREATE TABLE items (

item_id INT(10) PRIMARY KEY AUTO_INCREMENT,

item_name VARCHAR(5000) NOT NULL DEFAULT '',

item_price DOUBLE DEFAULT NULL,

item_image VARCHAR(5000) NOT NULL DEFAULT '',

item_date DATE NOT NULL DEFAULT '0000-00-00',

) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=latin1;

CREATE TABLE users (

user_id INT(11) PRIMARY KEY AUTO_INCREMENT,

user_email VARCHAR(1000) NOT NULL,

user_password VARCHAR(1000) NOT NULL,

user_firstname VARCHAR(1000) NOT NULL,

user_lastname VARCHAR(1000) NOT NULL,

user_address VARCHAR(1000) NOT NULL,

) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

CREATE TABLE orderdetails (

order_id INT(10) PRIMARY KEY AUTO_INCREMENT,

user_id INT(11) NOT NULL DEFAULT '0',

order_name VARCHAR(1000) NOT NULL DEFAULT '',

order_price DOUBLE NOT NULL DEFAULT '0',

order_quantity INT(10) NOT NULL DEFAULT '0',

order_total DOUBLE NOT NULL DEFAULT '0',

order_status VARCHAR(45) NOT NULL DEFAULT '',

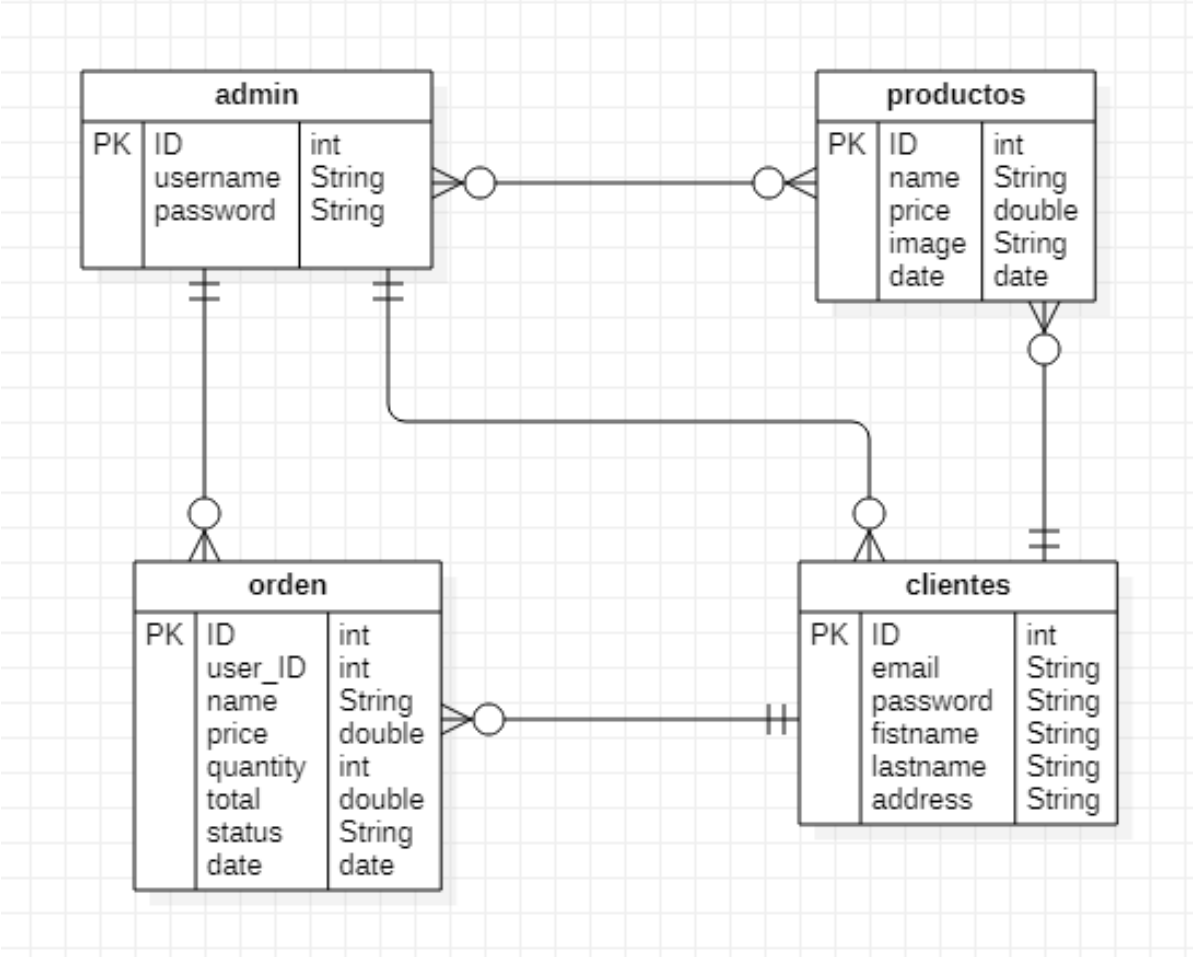
order_date DATE NOT NULL DEFAULT '0000-00-00',

CONSTRAINT FK_orderdetails_1 FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=33 DEFAULT CHARSET=latin1;

Mapeo de relaciones (asociación, composición y agregación)

El diagrama nos queda que:



Podemos ver que en la parte de admin - clientes tenemos que cada administrador individualmente dentro de su sesión tiene la autoridad de gestionar las cuentas de los clientes que él crea que necesiten de gestión, que puede ser una o tres o todas (1:n). Además, en la parte de admin - productos tenemos que todos los administradores son los encargados de la agregación, modificación y eliminación de los productos, por ejemplo en un día de reabastecimiento, todos los administradores pueden trabajar en conjunto para el reabastecimiento de los productos (n:m). Asimismo, en la parte de admin - orden tenemos que este caso cada administrador dentro de su sesión tiene la autoridad de la gestión de todas las órdenes de todos los clientes (1:n). También, en la parte de clientes - productos tenemos que el cliente tiene acceso a la visualización de todos los productos que él quiera siempre y cuando estén subidos en el sistema, cada cuenta de cliente es individual, es una cuenta por cada persona ya que al pedir datos sólo pedimos el nombre, apellido, dirección, e-mail personales (1:n). Por último, tenemos que las cuentas de los cliente son personales, cada cliente tiene la opción de hacer las compras que él quiera, no se limita a sólo una por un tiempo limitado, el cliente puede hacer una orden y si se olvidó de comprar algo lo puede comprar en otra orden (1:n).

Referencias

1- García Peñalvo, Francisco José y Pardo Aguilar, Carlos. “Introducción al Análisis y Diseño Orientado a Objetos”. RPP, Nº37. Febrero, 1998.

2-. Abundio Mendoza A., Rosa Lopez. “Base de Datos” [Consultada: 03-02-22] https://repositorio.uchile.cl/bitstream/handle/2250/151632/Bases-de-datos.pdf?sequence=1&isAllowed=y

3-. Blancarte Iturralde, O. J. (Enero 2020). Introducción a la arquitectura de software -Un enfoque práctico. Ciudad de México: Independently published.

4-. Larman C. (2002). Uml y patrones, inducción al análisis y diseño orientado a objetos: Pearson.

5.- Fowler Martin y Scott Kendall, "UML gota a gota”. Addison Wesley Longman de México. S.A. de C.V. México .1999