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	?	? (Agregué el punto 2. Diagramas de robustez)	1
Melissa Jiménez-MelisBot	03/02/22	Descripciones	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

1. Introducción

2. Diagramas de robustez

El diagrama de robustez fue creado por Ivar Jacobson en 1991 para responder a la pregunta "qué objetos se necesitan para cada caso de uso".

El diagrama de robustez tiene como función verificar si la especificación del caso de uso es correcta y completa. El gráfico robusto es el método utilizado en el proceso de diseño de requisitos (análisis de robustez) el cual permite a los diseñadores tener una comprensión más clara y completa de los requisitos.

Los diagramas de robustez nos dará una visión más clara para nuestro funcionamiento de la tienda digital “Cambios Gamer” de la cuales realizamos sus casos de uso respectivos y ahora comprobaremos si el caso de uso es apto y claro para la función de la tienda online y si no, realizamos cambios en los casos de uso.

Para este proyecto tomamos 5 de nuestros casos de uso 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 uso con sus flujos alternativos y flujos principales de bajo de estos los diagramas de robustez correspondiente a cada uno de los casos de uso.

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.

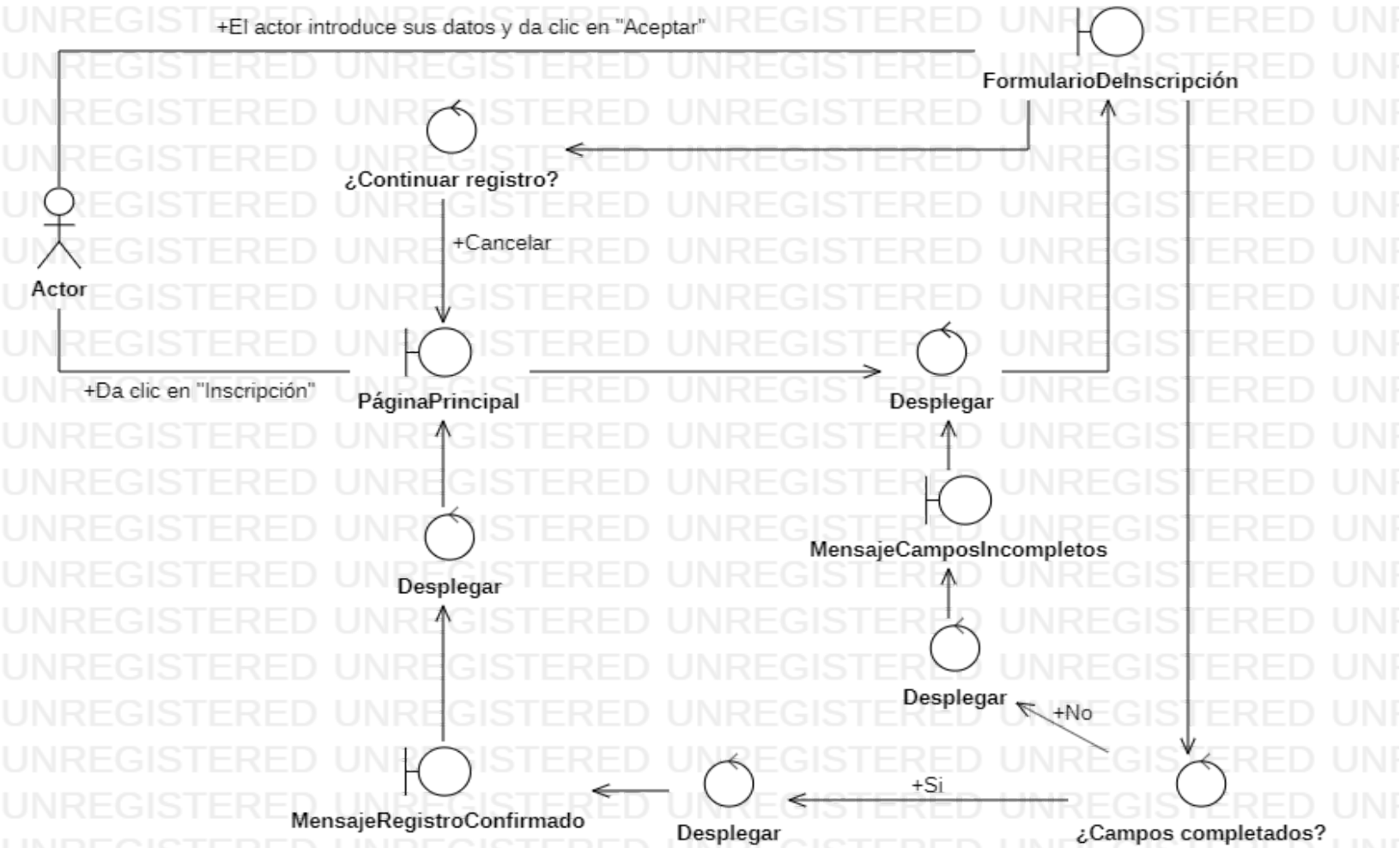


Diagrama de robustez de “Inscribir clientes”

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.

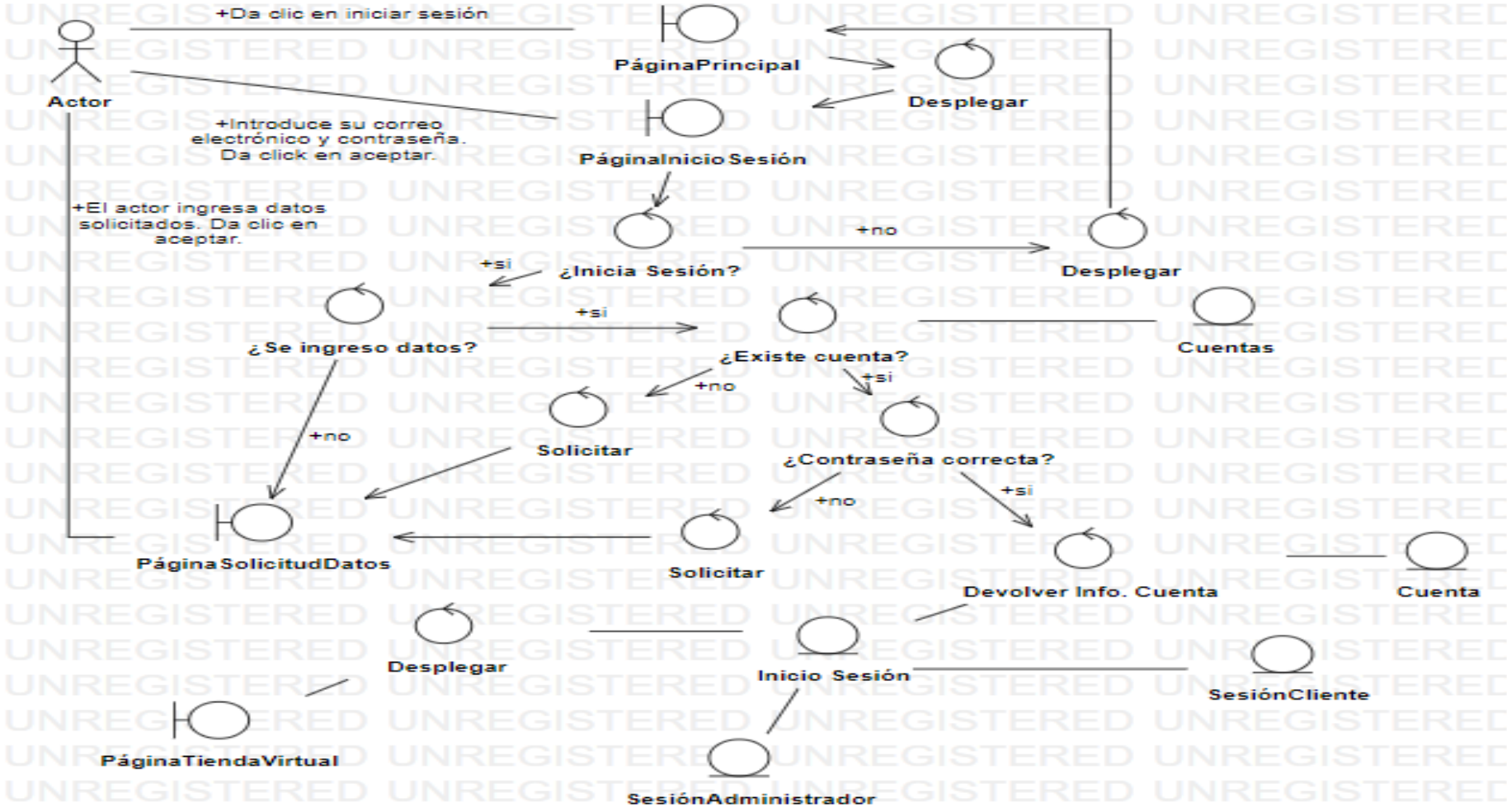


Diagrama de robustez de “Iniciar sesión”

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”.

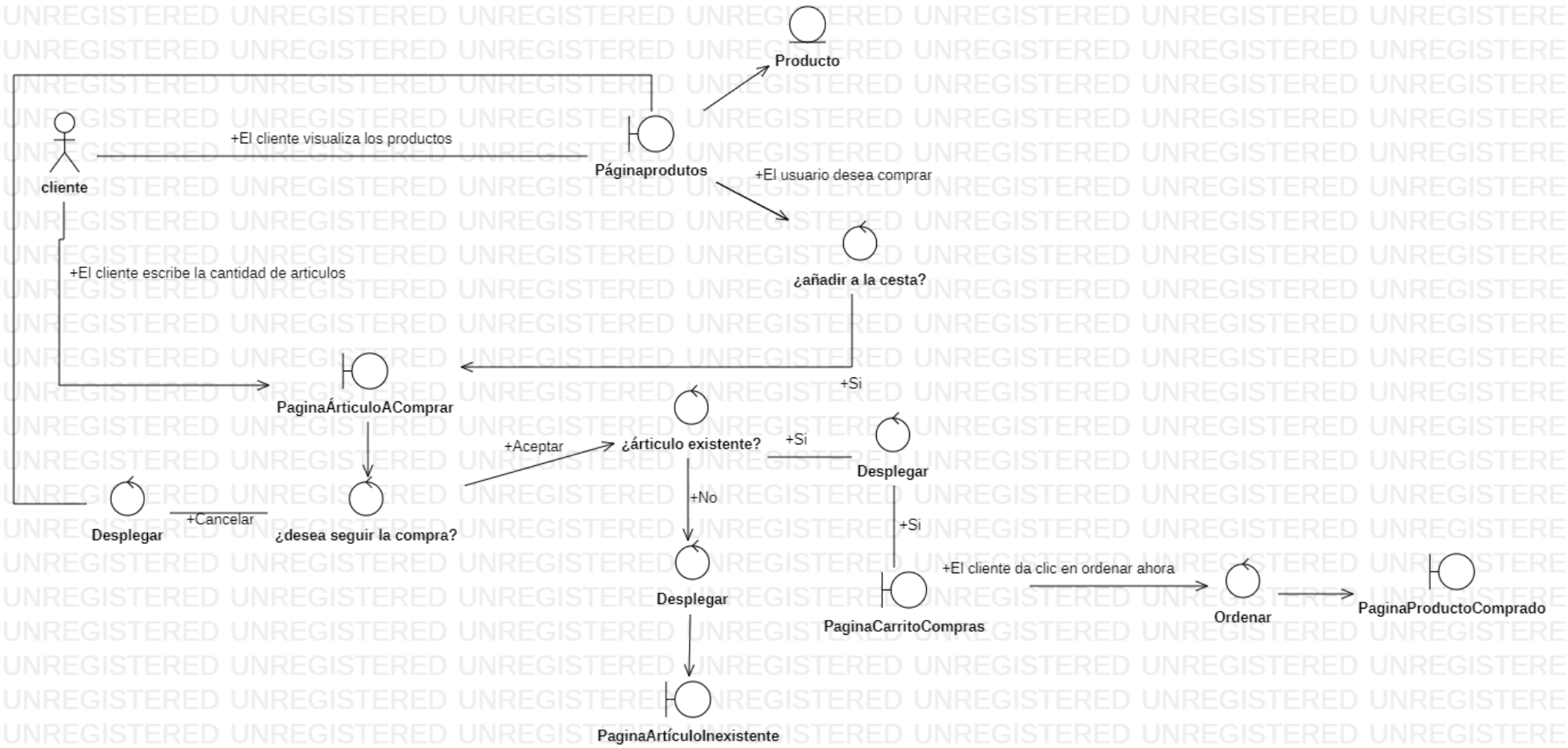


Diagrama de robustez de “Realizar compras”

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.

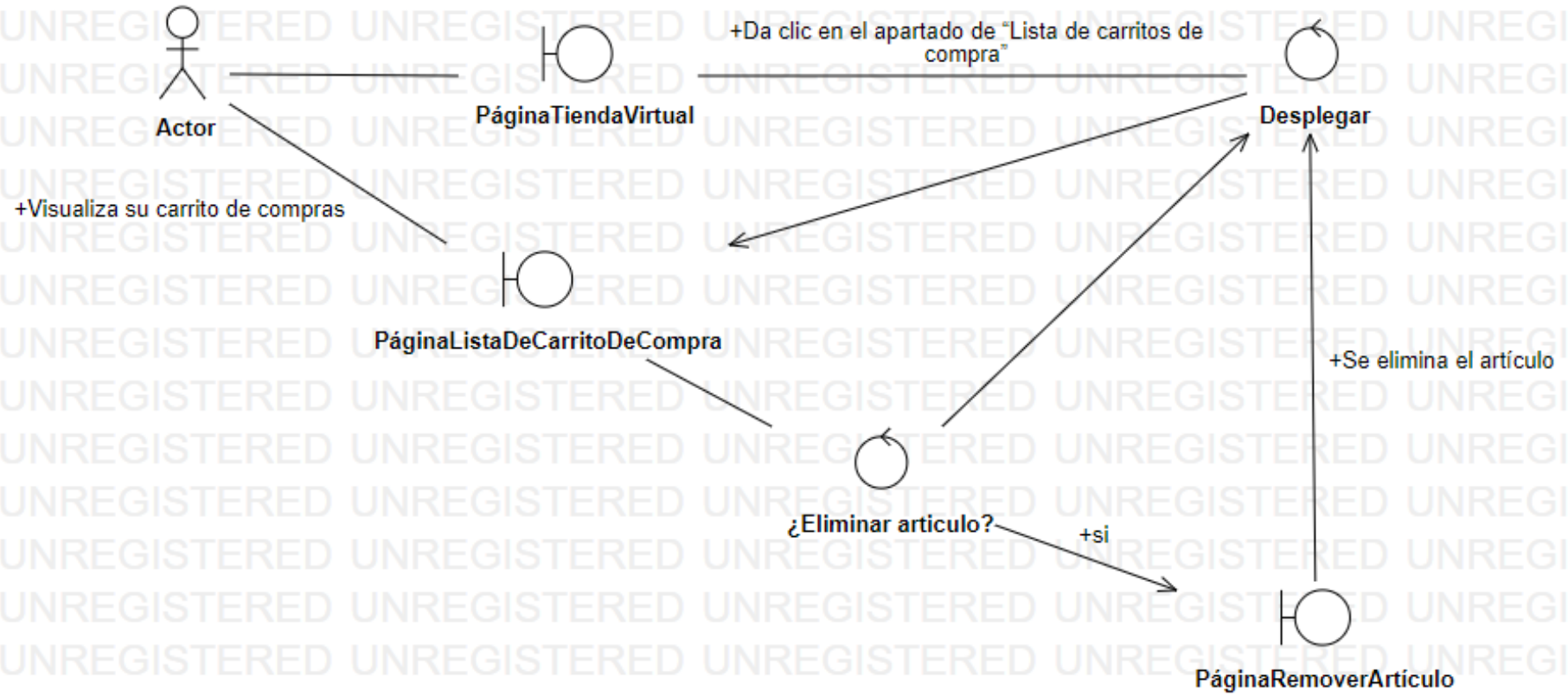


Diagrama de robustez de "Mostrar carrito"

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.

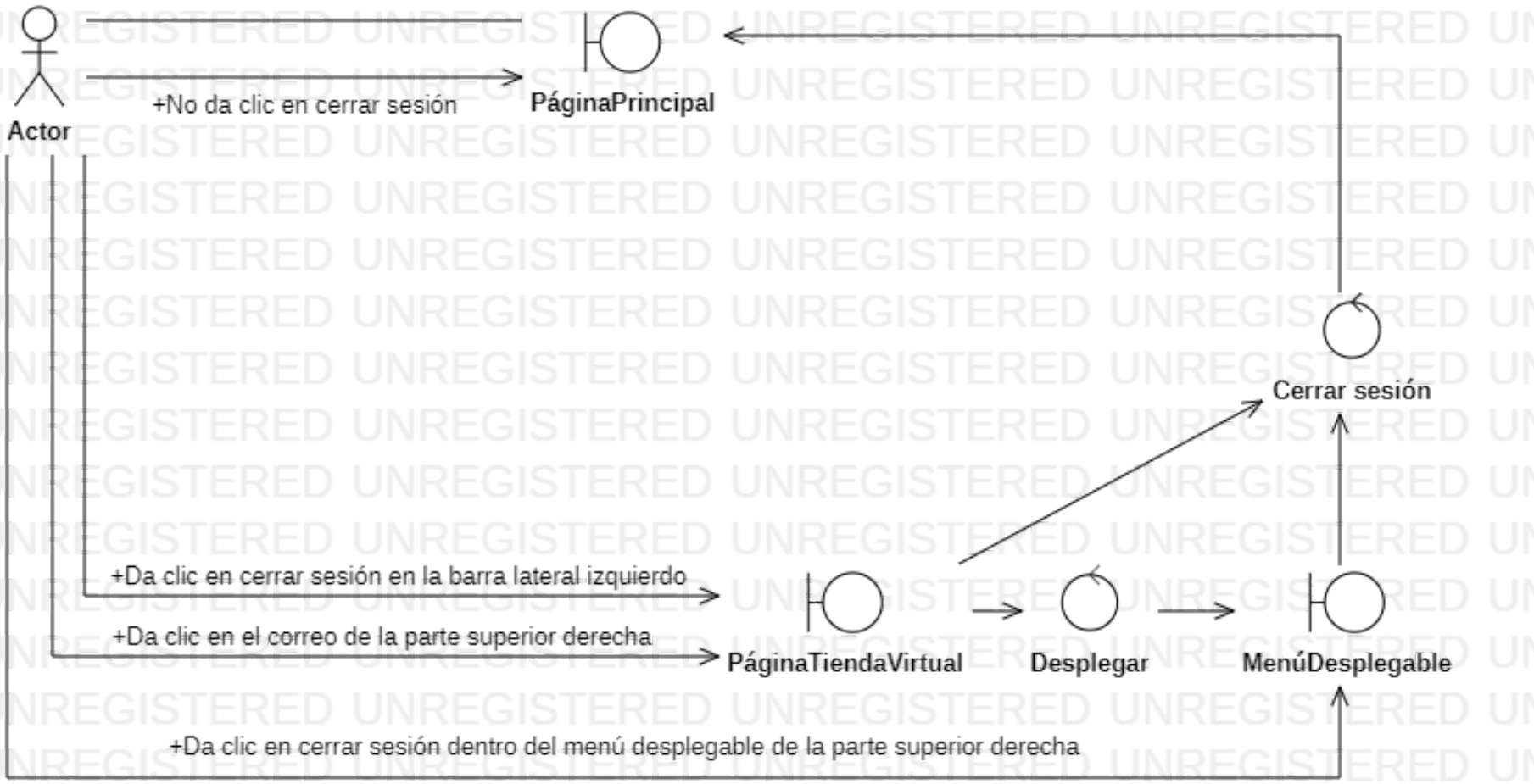


Diagrama de robustez de “Cerrar sesión”

3. Diagramas de secuencia

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

- Representa los detalles de un caso de uso.
- Modelar la lógica de una operación, una función o un procedimiento sofisticados.
- Ver como 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 uso principales y alternativos y su diagrama de robustez correspondiente al caso de uso.

En nuestro proyecto tomamos en cuenta 5 casos de uso:

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

Estos casos de uso están dados con sus flujos alternativos y principales junto con los diagramas de robustez en el paso (2-Diagrama de Robustez).

A continuación mostraremos los diagramas de secuencia de cada uno de los casos de uso descritos con anterioridad.

Diagrama de secuencia “Inscribir clientes”

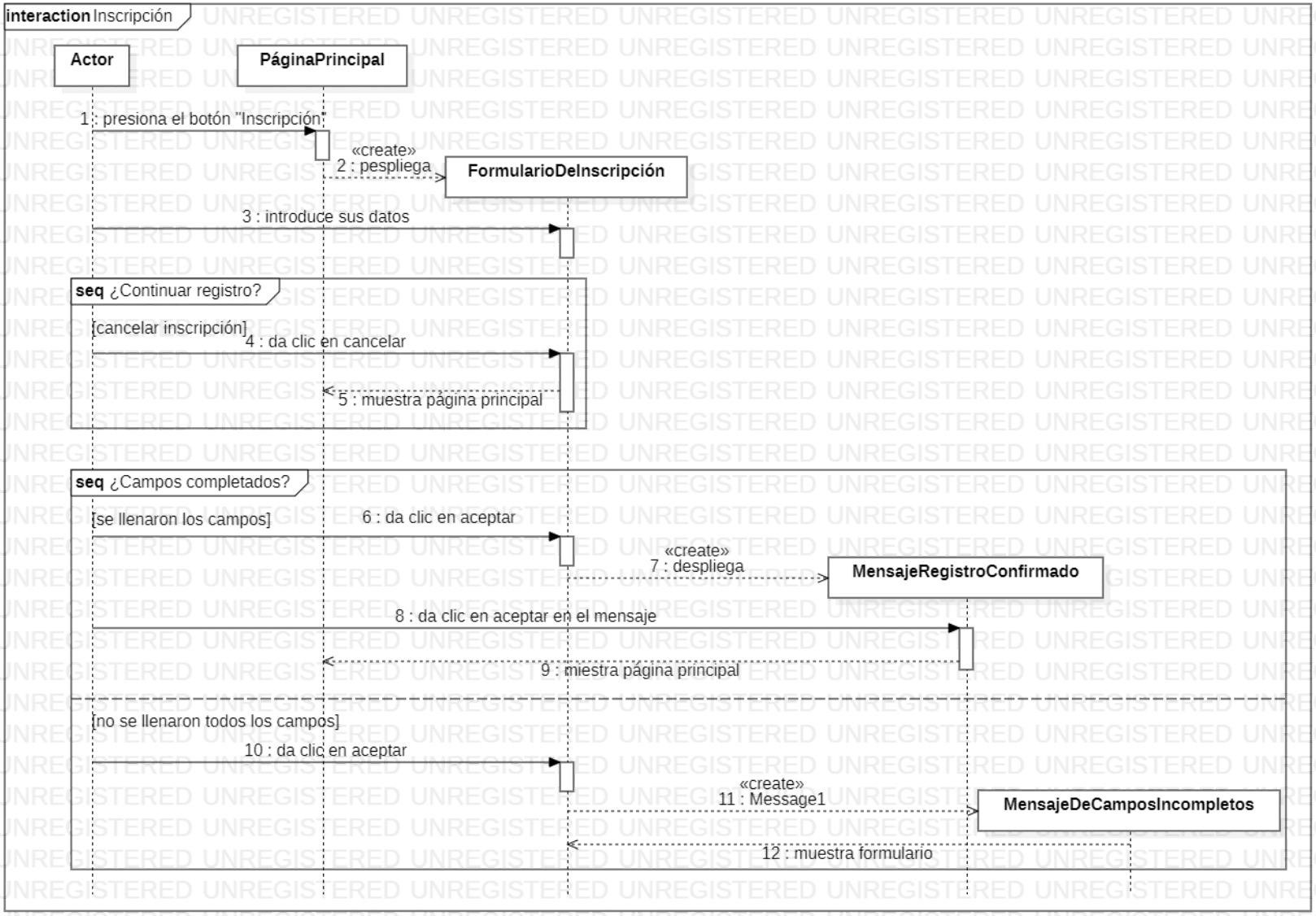


Diagrama de secuencia “Iniciar sesión” (FALTA)

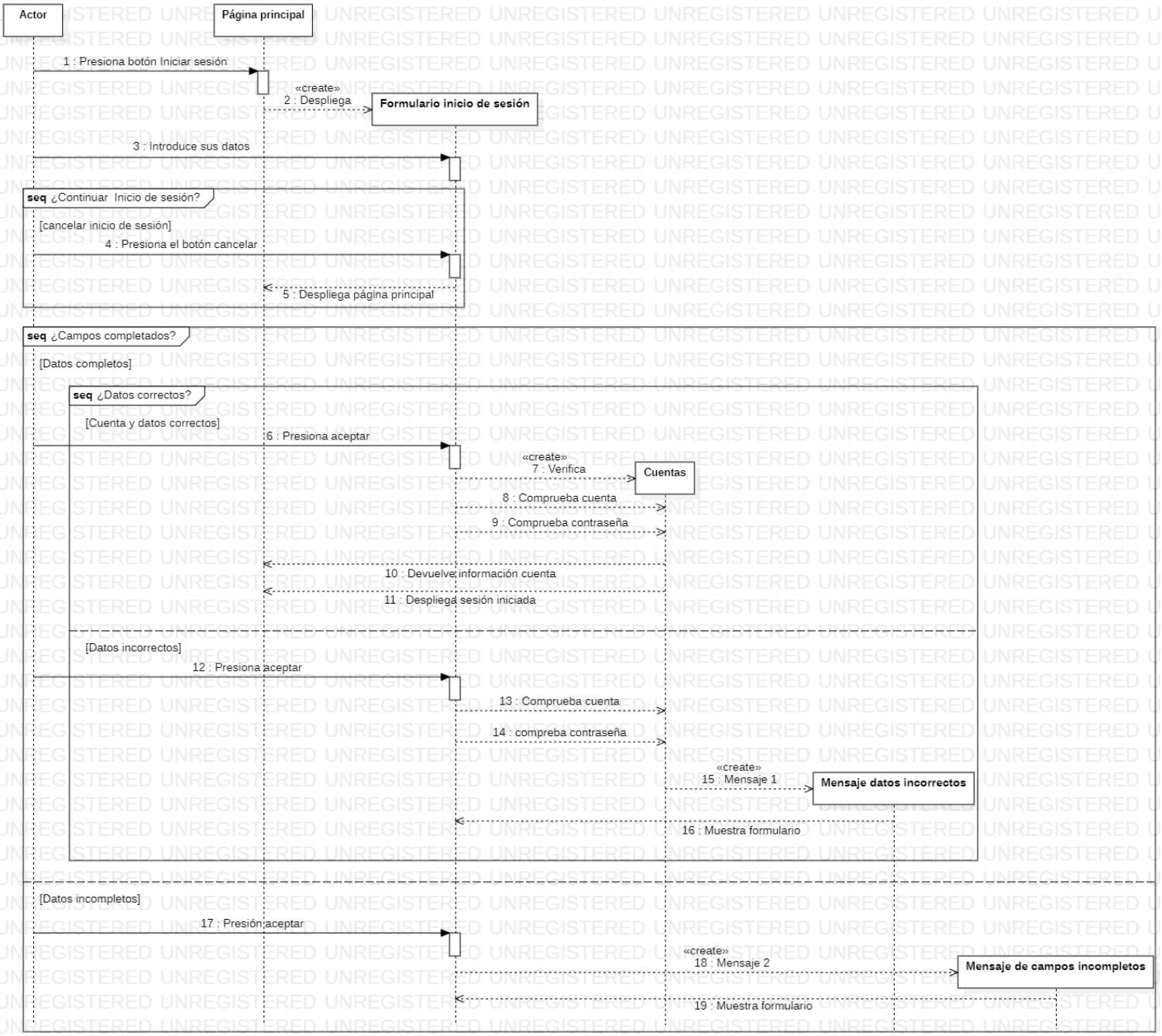


Diagrama de secuencia “Realizar compras”

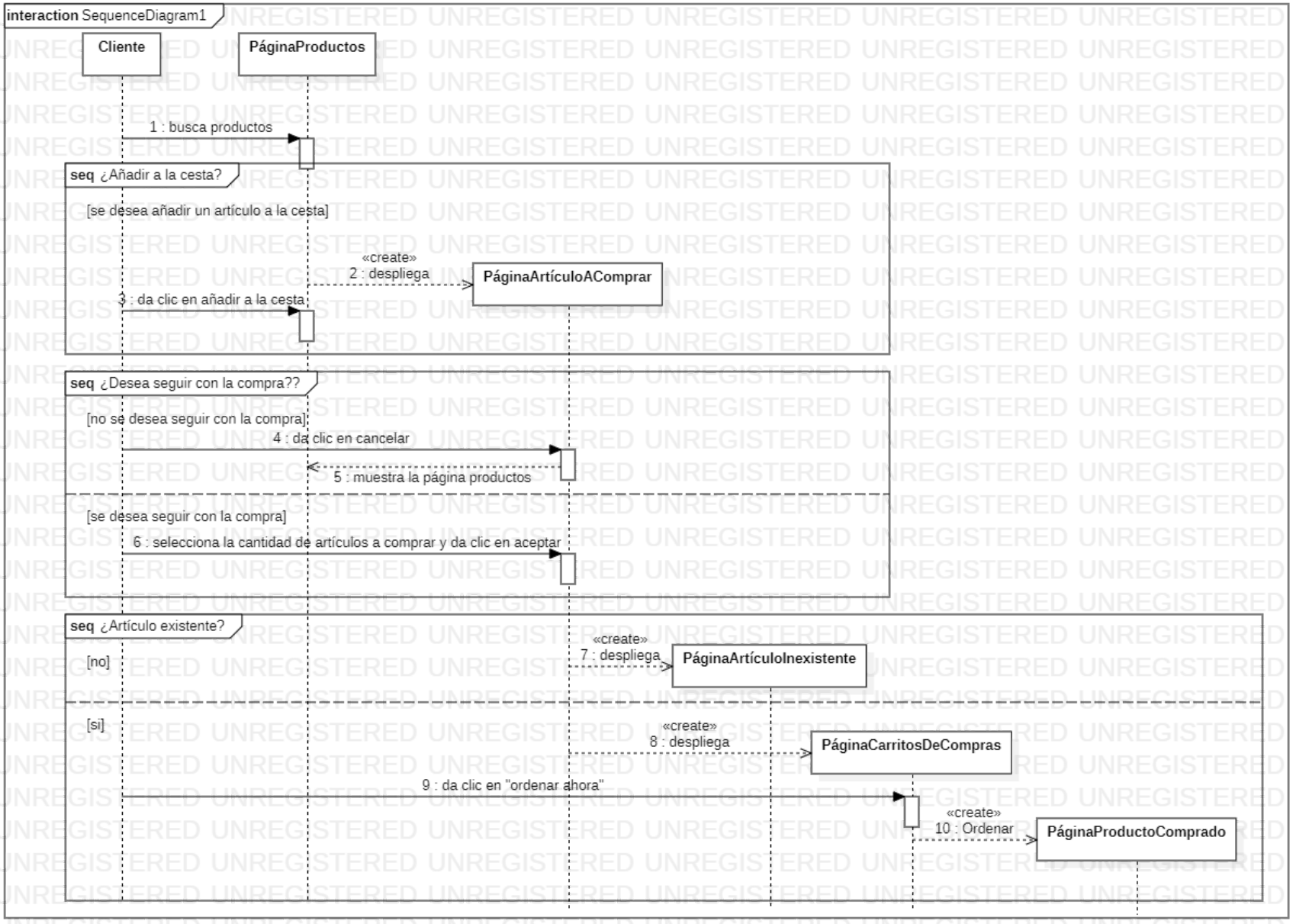


Diagrama de secuencia “Mostrar carrito”

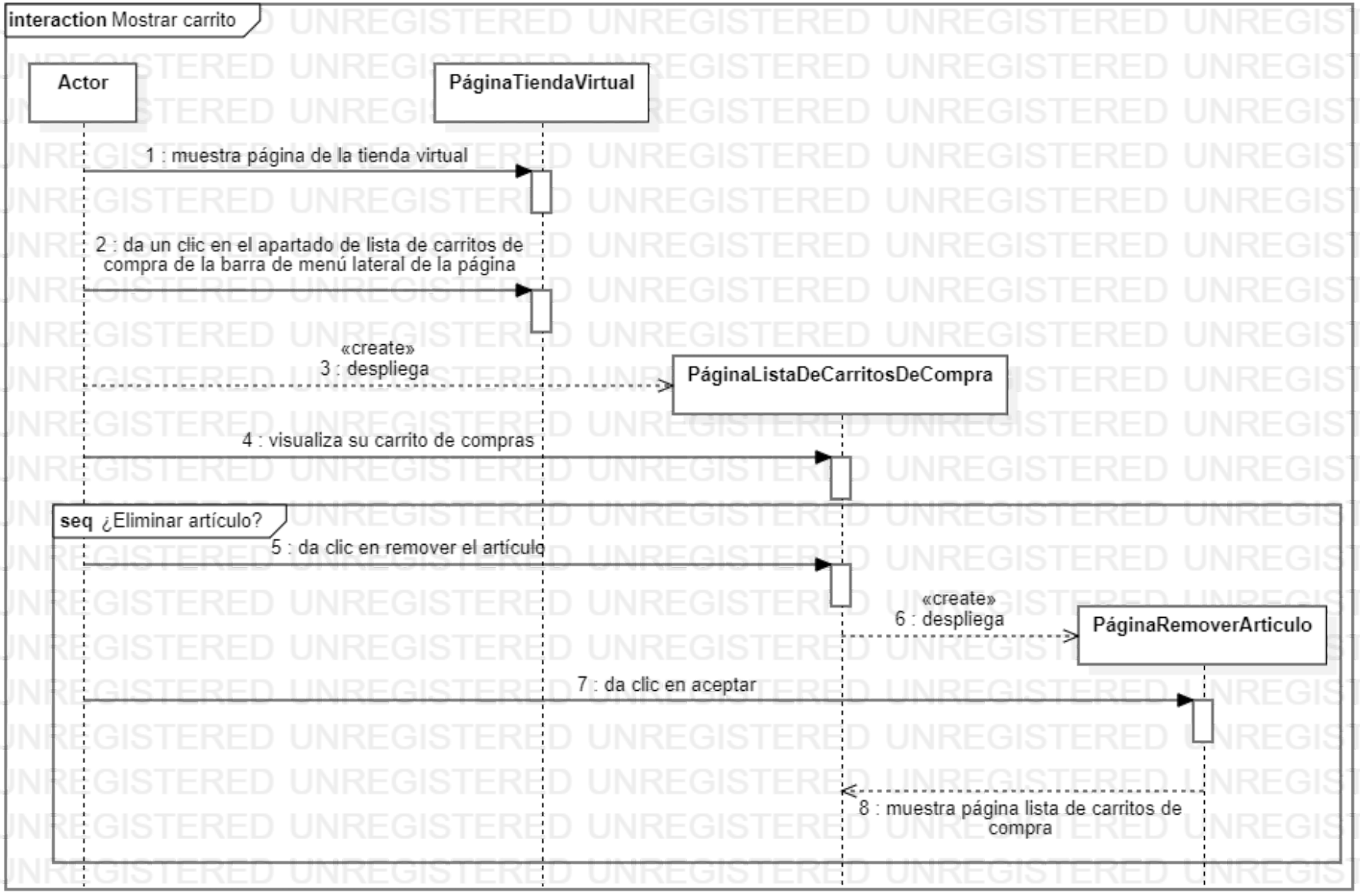
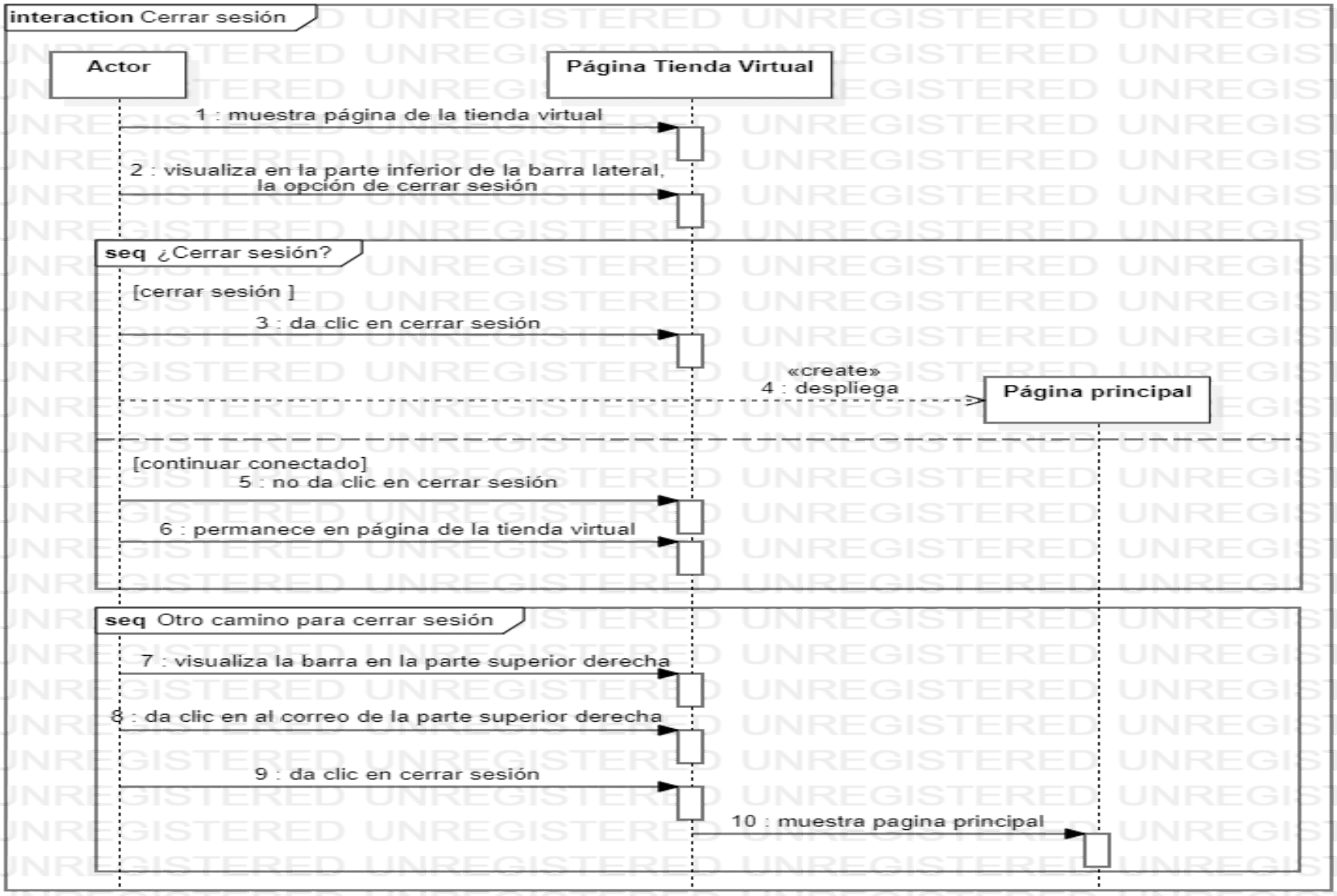


Diagrama de secuencia “Cerrar sesión”



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 se de 1 a muchos, 1 a 1 o de 0 a 1, 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.

(Información de lo que realizamos)

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.

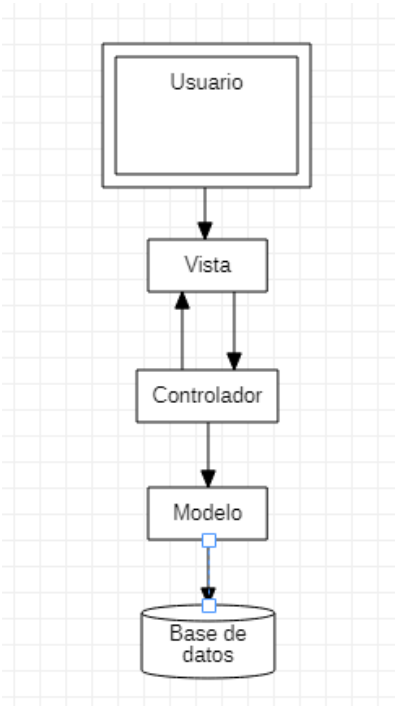
En el apartado de **Modelo** se trabaja con datos, eso quiere decir que contendrá mecanismos para acceder a la información y también para actualizar su estado. (Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas)

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 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

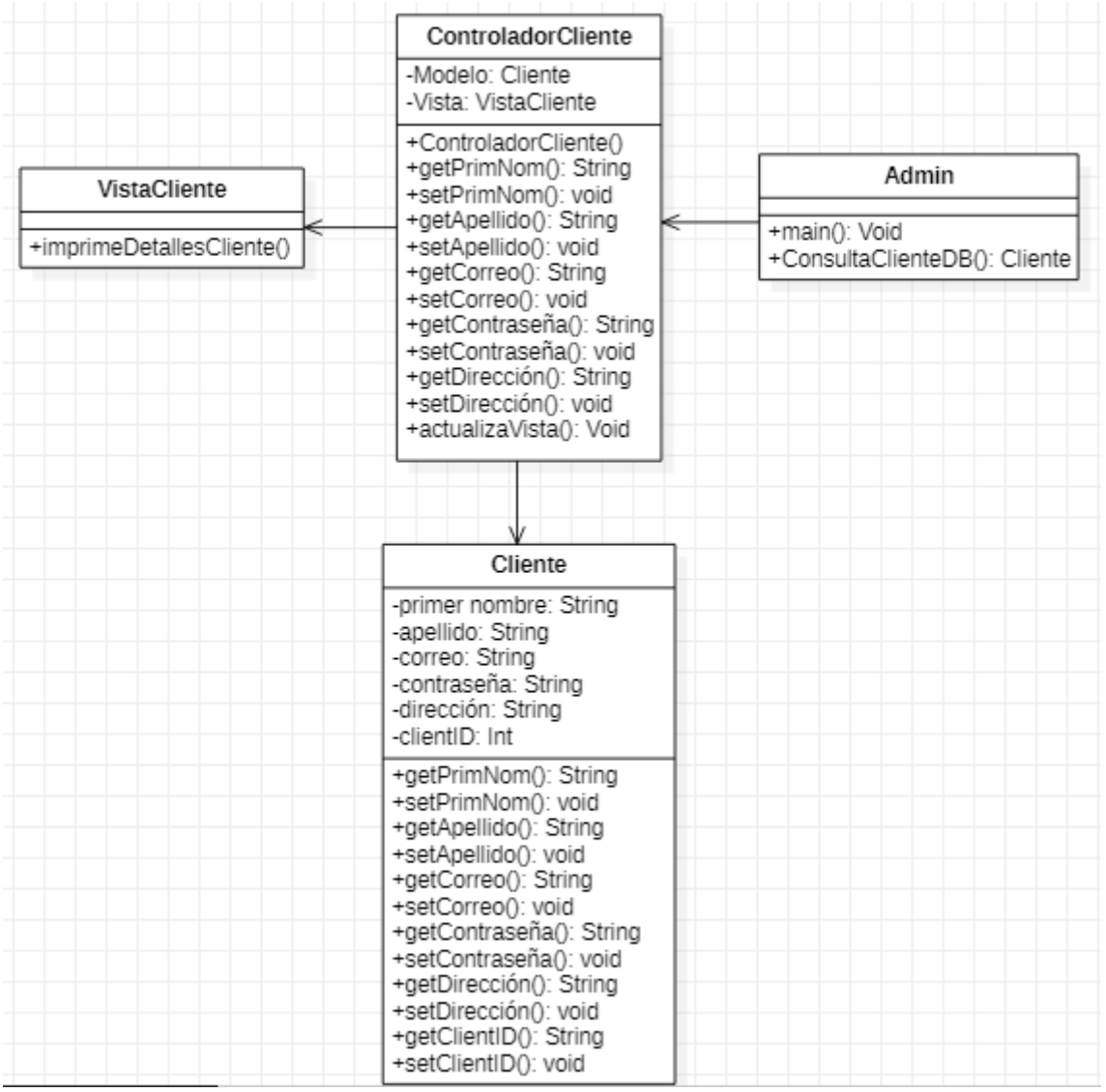
El modelo funciona en base a la interacción del usuario con la vista, la primera interacción o movimiento lo vamos a tener con el usuario y la vista. Esta instrucción que el usuario realiza en la vista (En la interfaz de usuario), y el controlador se va a encargar de mediar la interacción entre el modelo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



Implementación del MVC en nuestro sistema

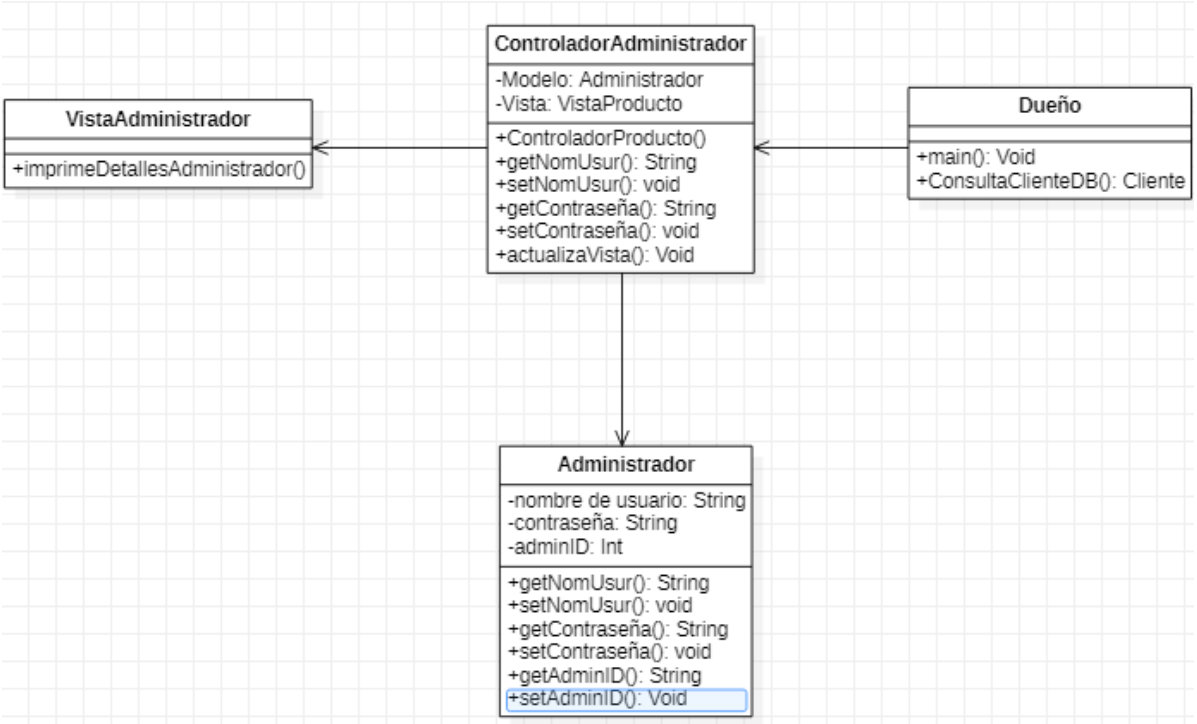
Entidad: Cliente (Modelo)

(desc)



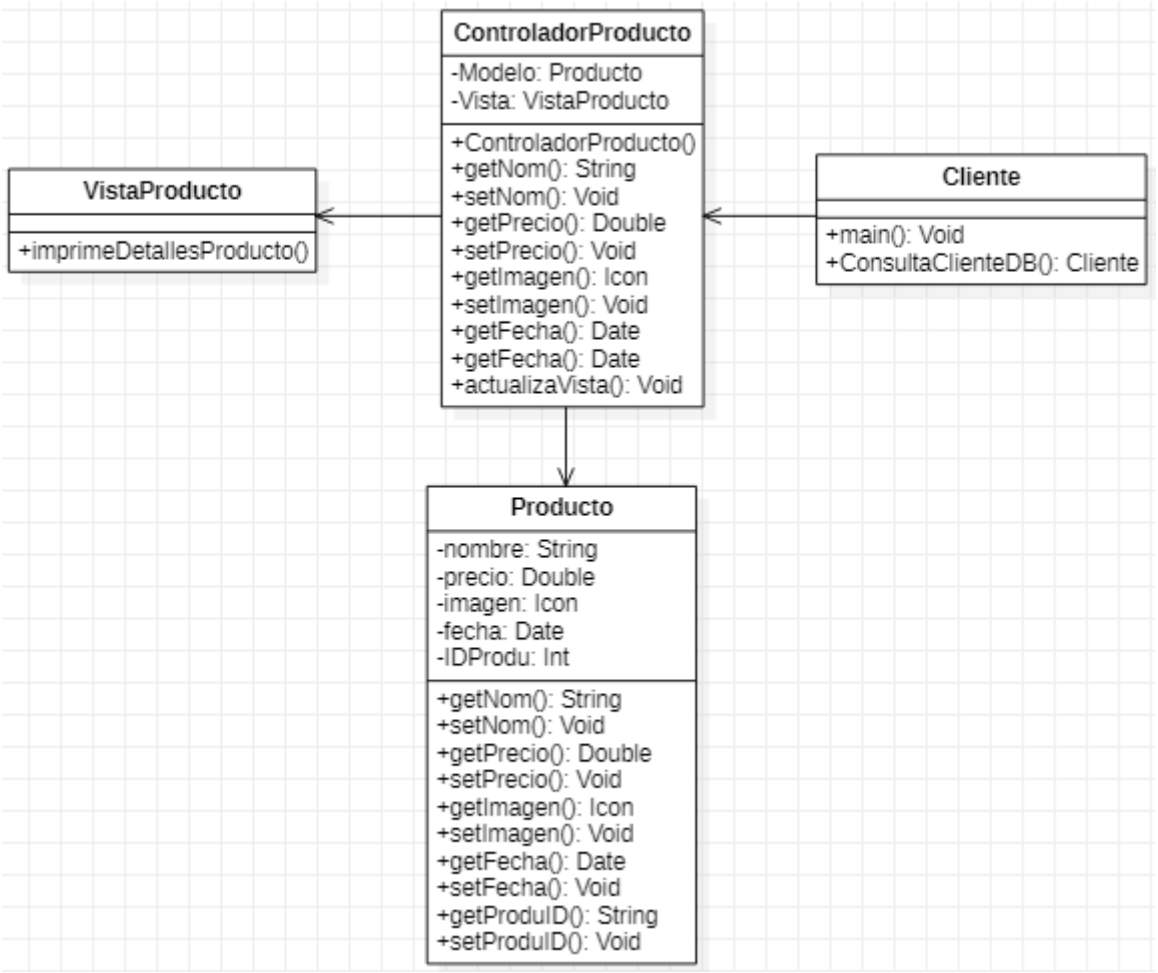
Entidad: Administrador (Modelo)

(Desc)



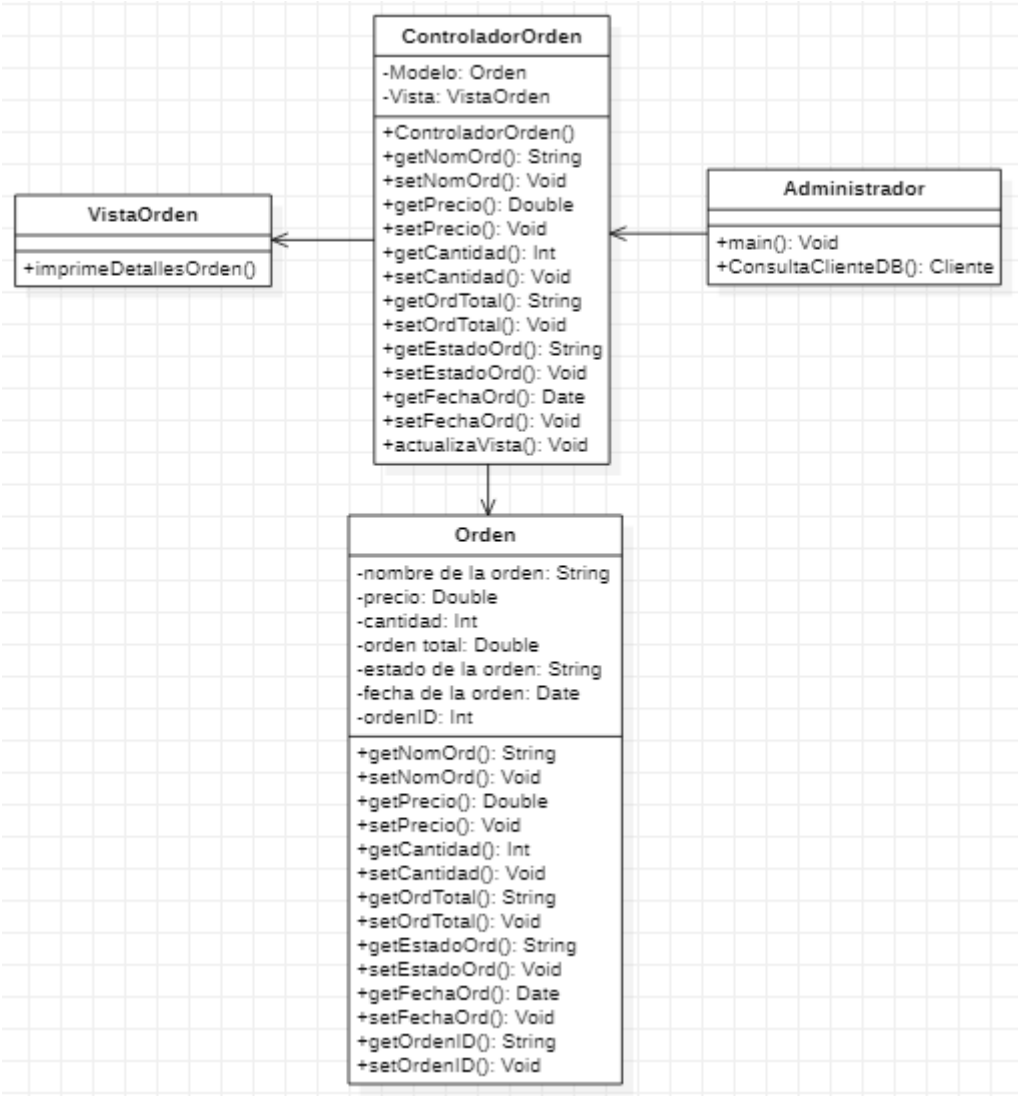
Entidad: Producto (Modelo)

(Desc)



Entidad: Orden (Modelo)

(Desc)



DAO

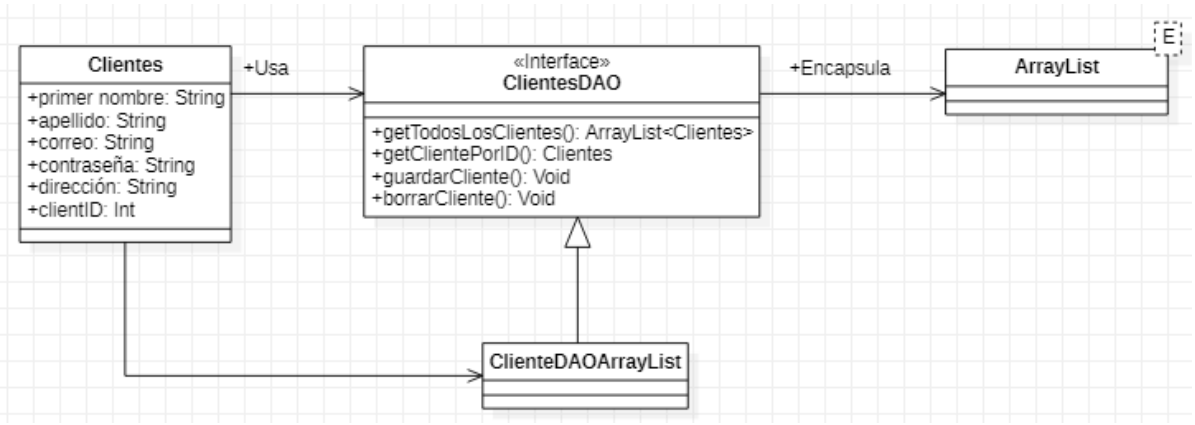
El patrón **Data Access Object** (DAO) propone separar por completo la lógica de negocio de la lógica para acceder a los datos, de esta forma, el **DAO** proporcionará los métodos necesarios para insertar, actualizar, borrar y consultar la información; por otra parte, la capa de negocio solo se preocupa 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 solo 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, solo nos preocupamos por la lógica de negocio sin preocuparnos de donde viene los datos o los detalles técnicos para consultarlos o actualizarlos.

Implementación de DAO

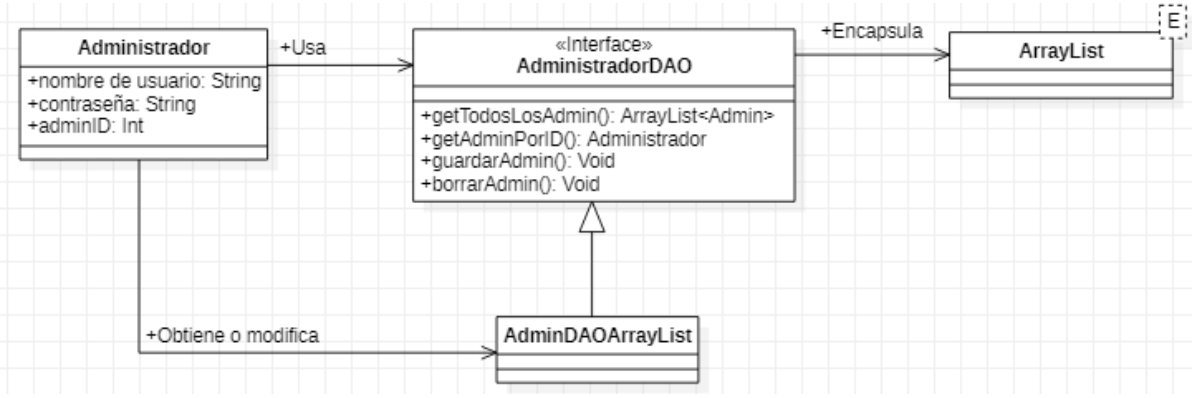
Para Clientes

(Desc)



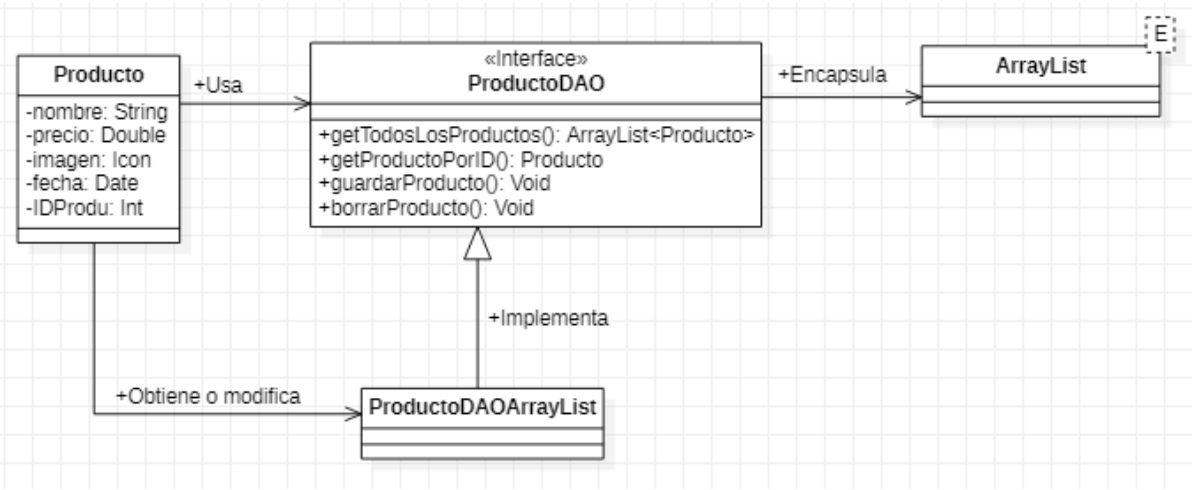
Para Administrador

(Desc)



Para Productos

(Desc)



Para Órdenes

(Desc)

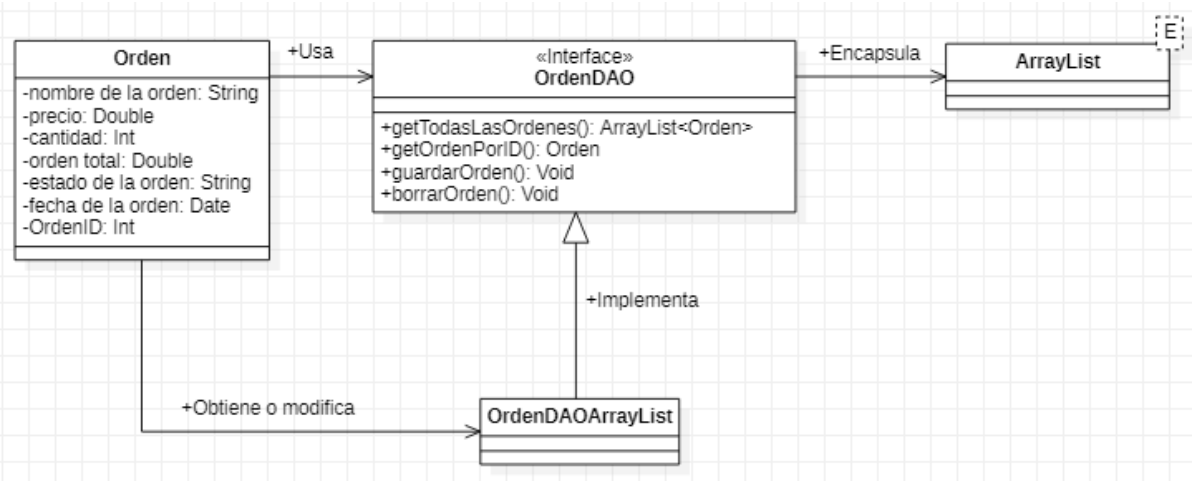
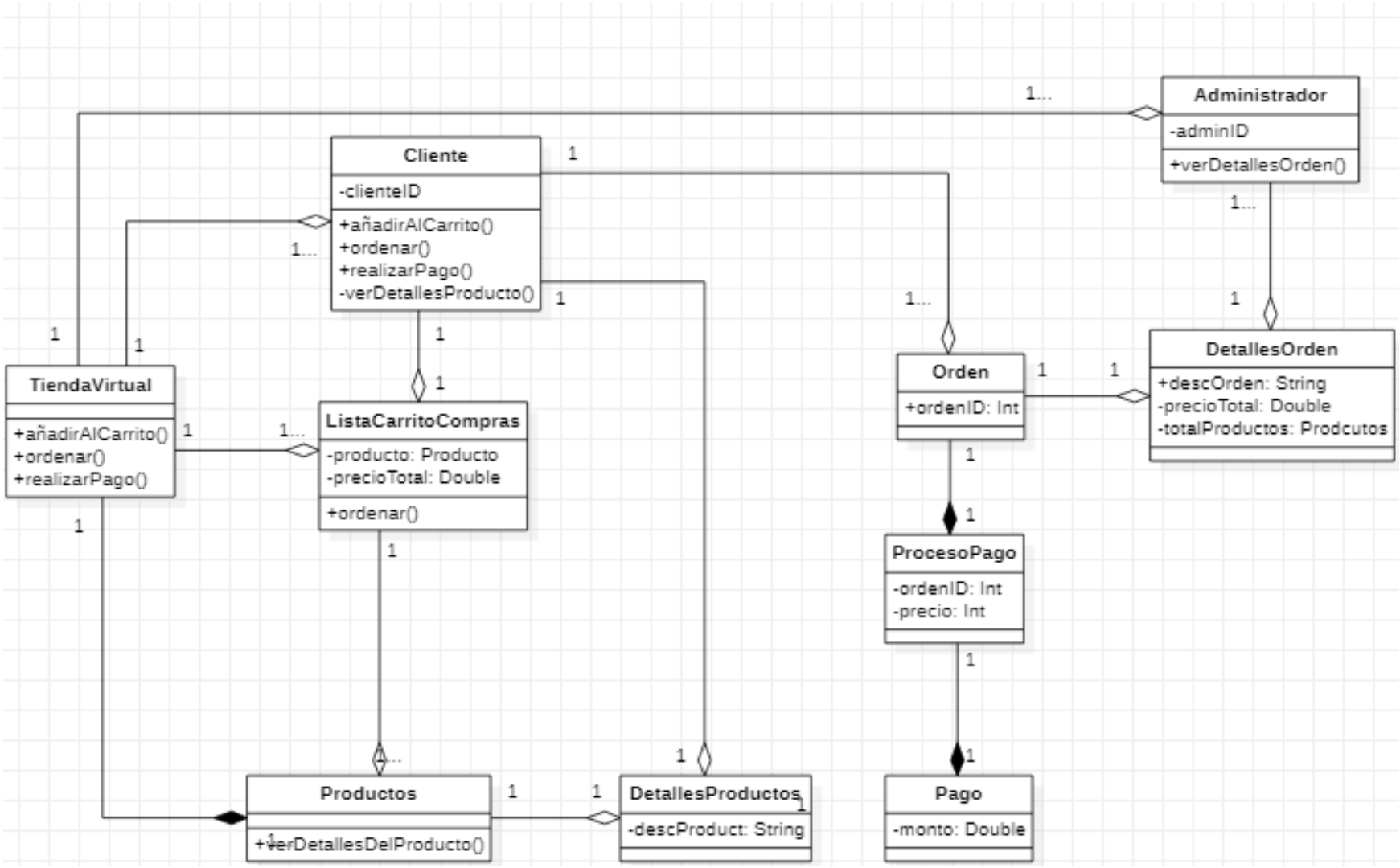


Diagrama de clases



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.

La dependencia 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”

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 de más.

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

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

Introducción a la arquitectura de software