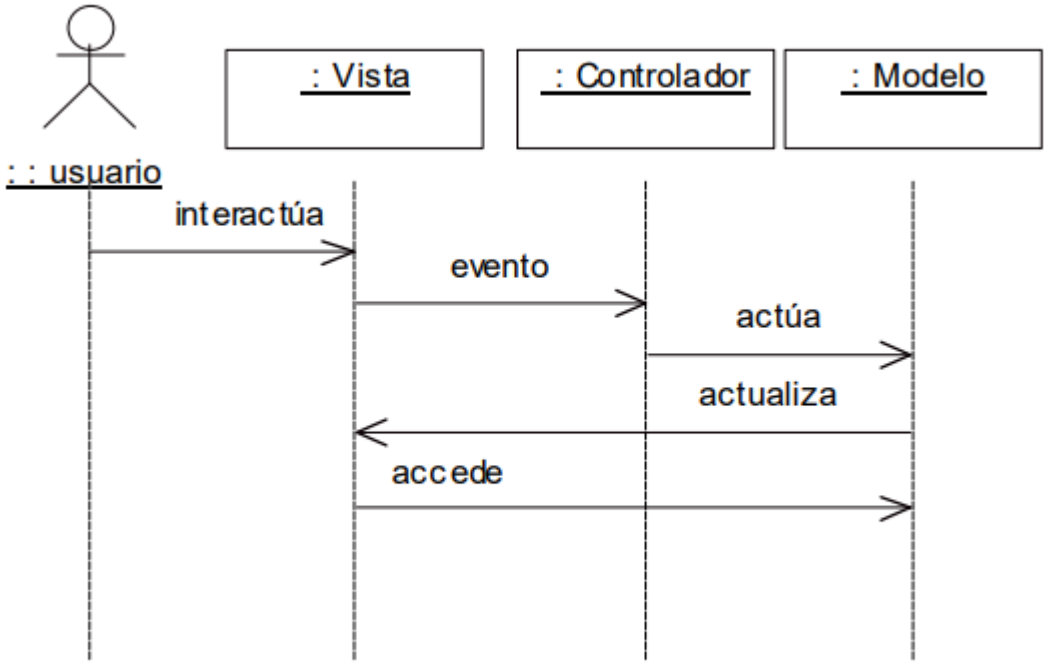
2. Documento de Diseño UML

2.1 Arquitectura

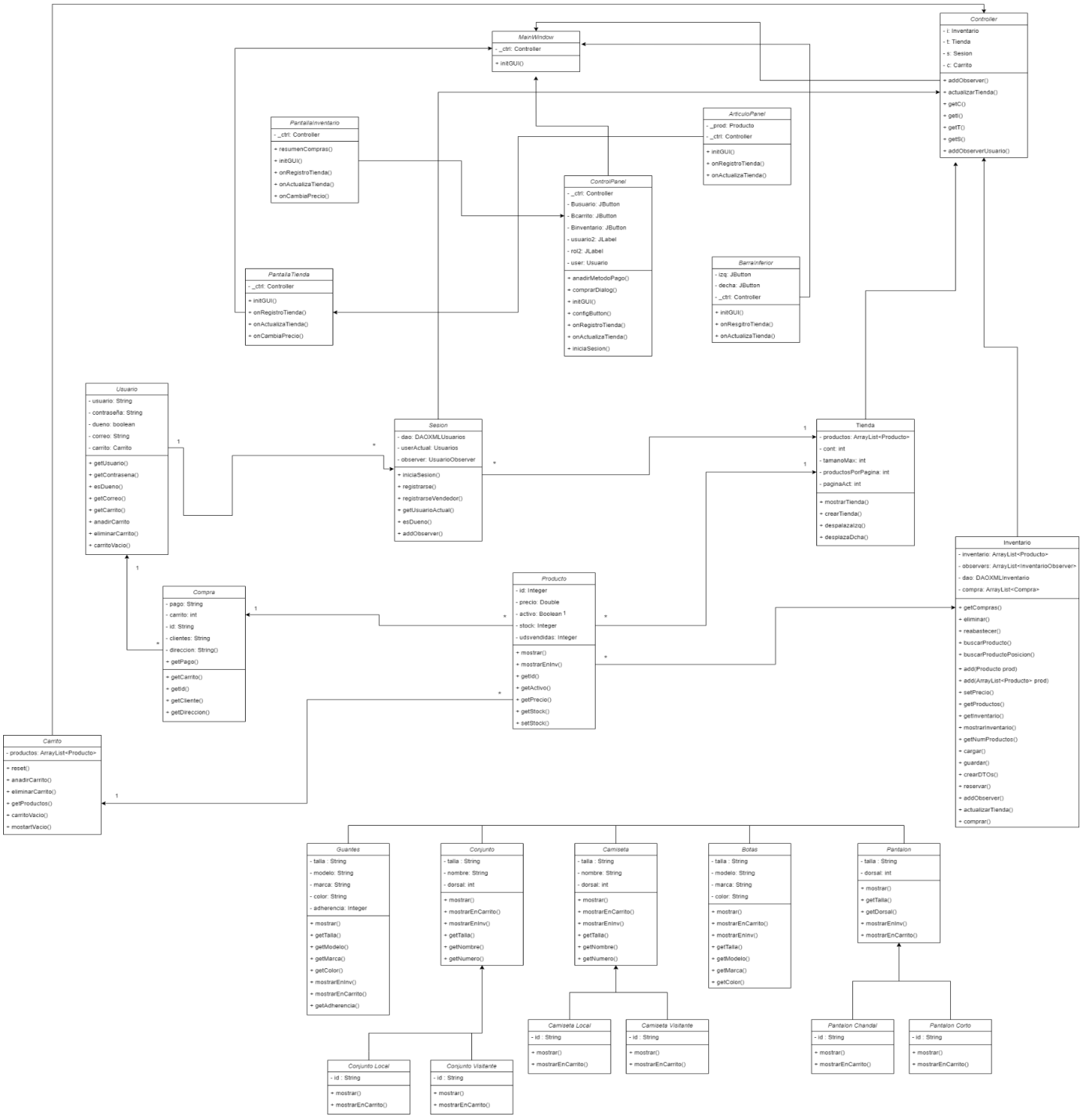
En nuestro proyecto hemos usado el modelo activo, ya que es el modelo el que actualiza la vista-controlador. El modelo notificará a las vistas los cambios que en los datos pueda producir un agente externo.



Respecto a la estructura del programa, hemos creado paquetes para el modelo, la vista y el controlador. En la vista encontramos en la pantalla principal la interfaz de la tienda y botones para acceder al carrito, a los usuarios y al inventario dentro de la clase controlPanel. En el modelo tenemos todas las clases que hemos creado, los productos, los tipos de productos, los usuarios y las clases necesarias para el funcionamiento de estos. Por último en el paquete del controller encontramos el controller, que nos ayuda a realizar todos los cambios necesarios para la actualización.

2.2 Diseño

Diagrama de clases general del proyecto

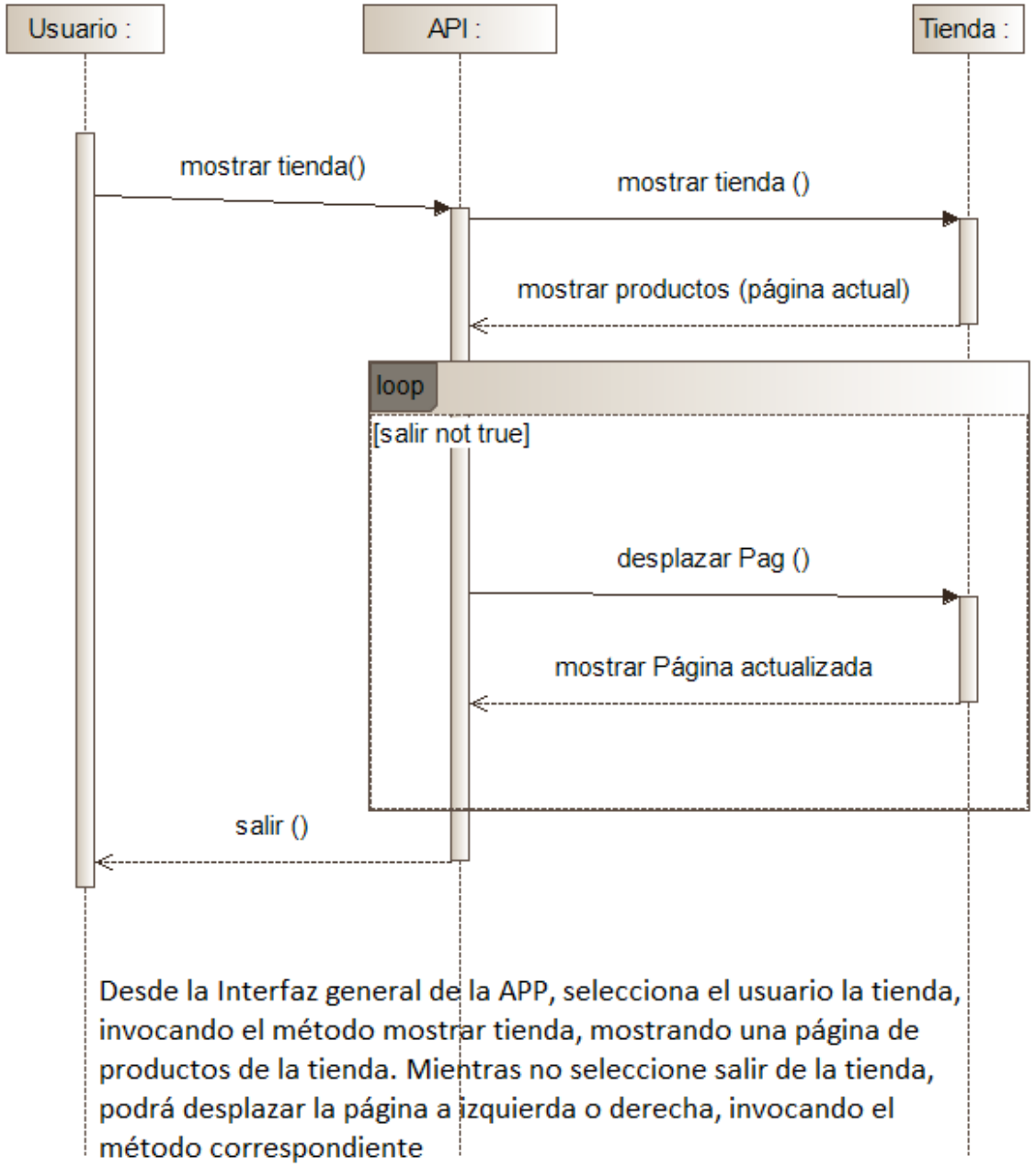


Historia de usuario Mostrar Tienda

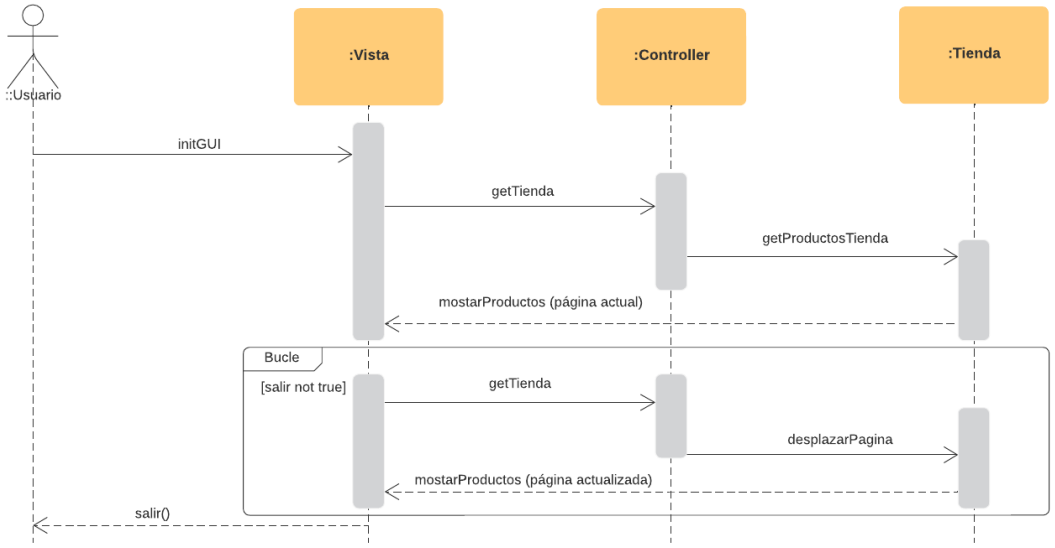
En el primer sprint, antes de tener la interfaz gráfica creada, simplemente creamos un método para mostrar los productos de nuestra tienda por pantalla, en formato textual. Dividimos los productos en páginas, que en el futuro serán páginas reales de la interfaz, y mostramos ese número de productos. Permitimos al usuario desplazar páginas a izquierda y derecha.

En el segundo sprint, modificamos la clase, para que tienda solo se pueda instanciar una vez, mediante el patrón de diseño Singleton.

Diagrama de secuencia:

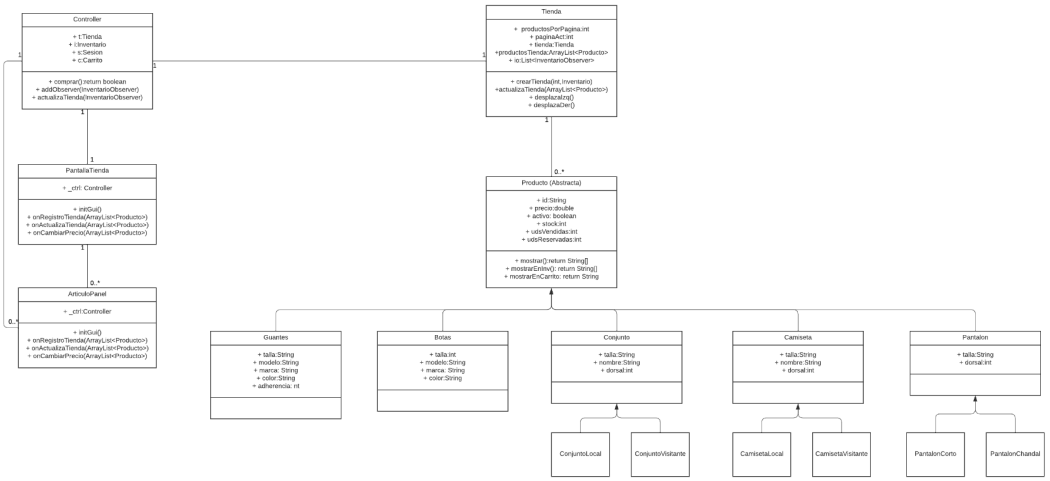


En el cuarto sprint, se ha modificado este diagrama de secuencias al introducir el modelo-vista-controlador:



A diferencia del anterior diagrama de secuencia, en este tenemos que para llegar a mostrar los productos en la interfaz hay que hacer uso del controller el cual tiene una instancia con el estado de la Tienda y por lo tanto a través de este podemos llamar a la función getProductos() de Tienda que nos devuelve la lista de productos que se mostrará en la interfaz gráfica. Al igual que anteriormente, te puedes ir desplazando por las diferentes páginas.

Diagrama de clases:

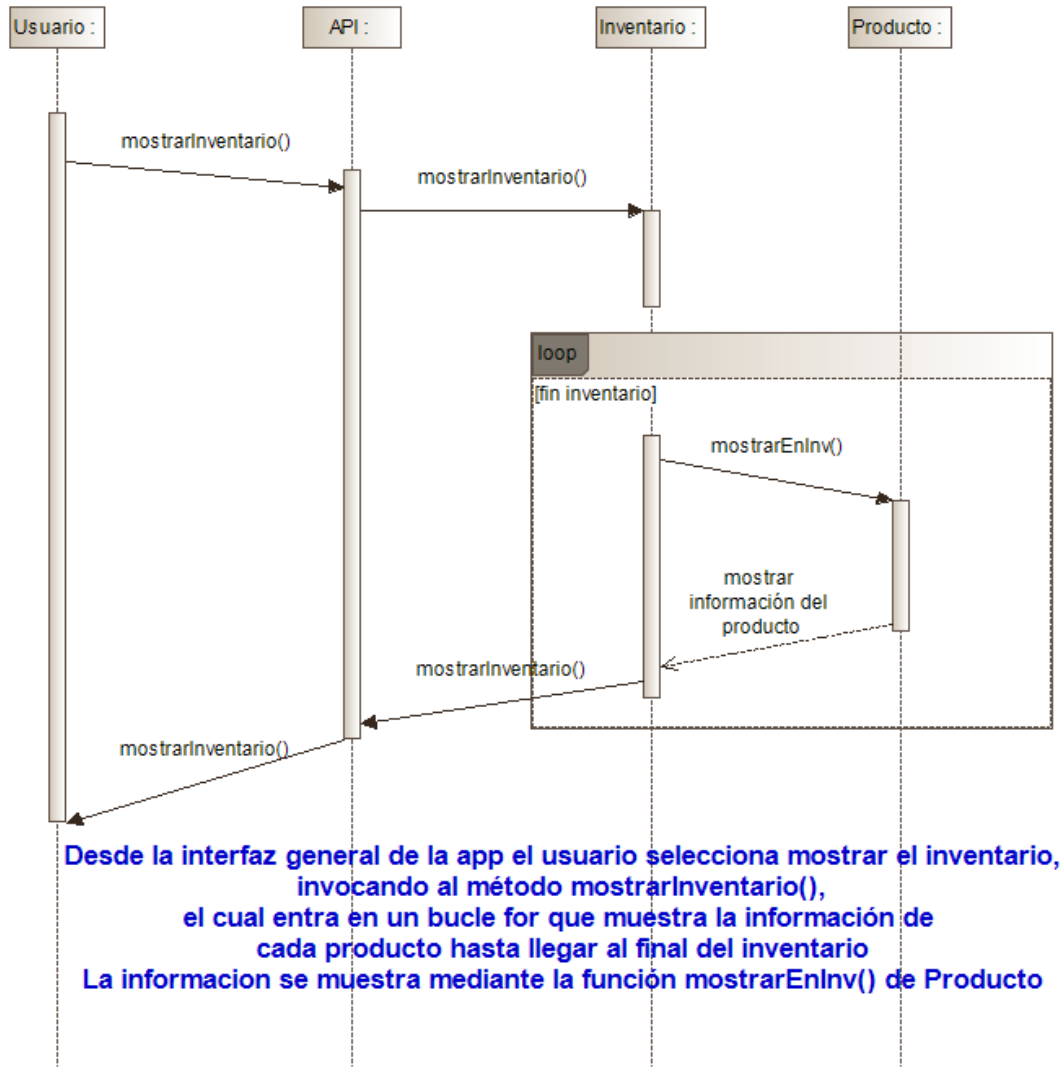


Historia de usuario Mostrar Inventario

En el primer sprint, creamos un método para mostrar los productos del inventario por pantalla, en formato textual. Se muestran en lista todos los productos con la cantidad

disponible de cada uno y su información. Esta lista en un futuro, cuando tengamos la interfaz gráfica estará más compacta y categorizada.

Diagrama de secuencias:



En el **cuarto Sprint**, modificamos este diagrama de secuencias debido a la implementación del Modelo-Vista-Controlador:

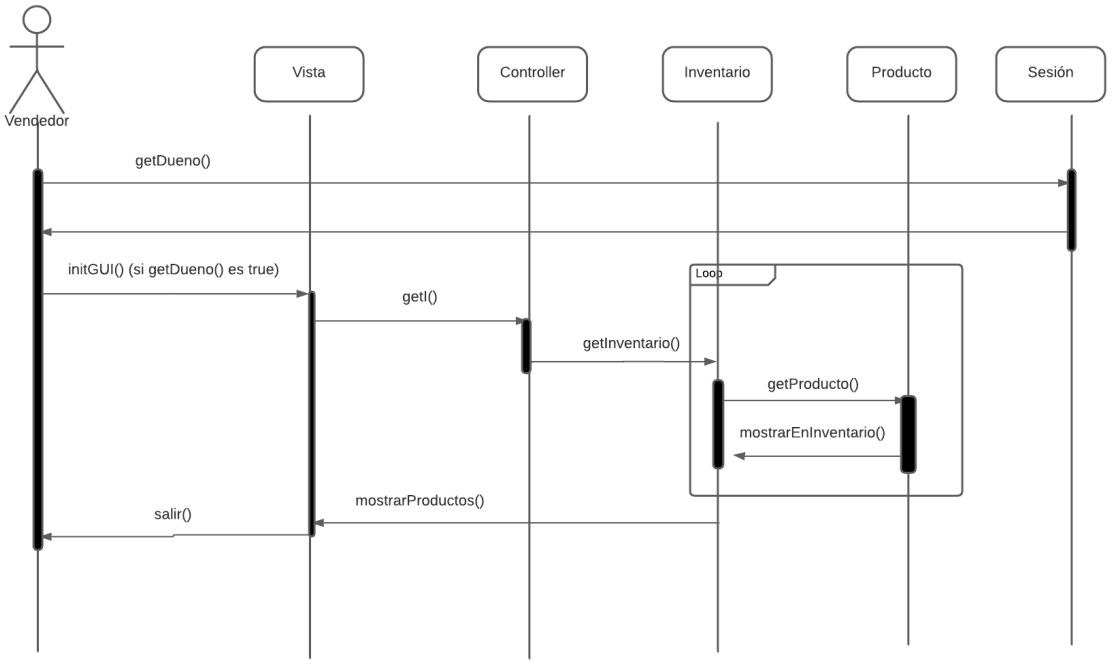
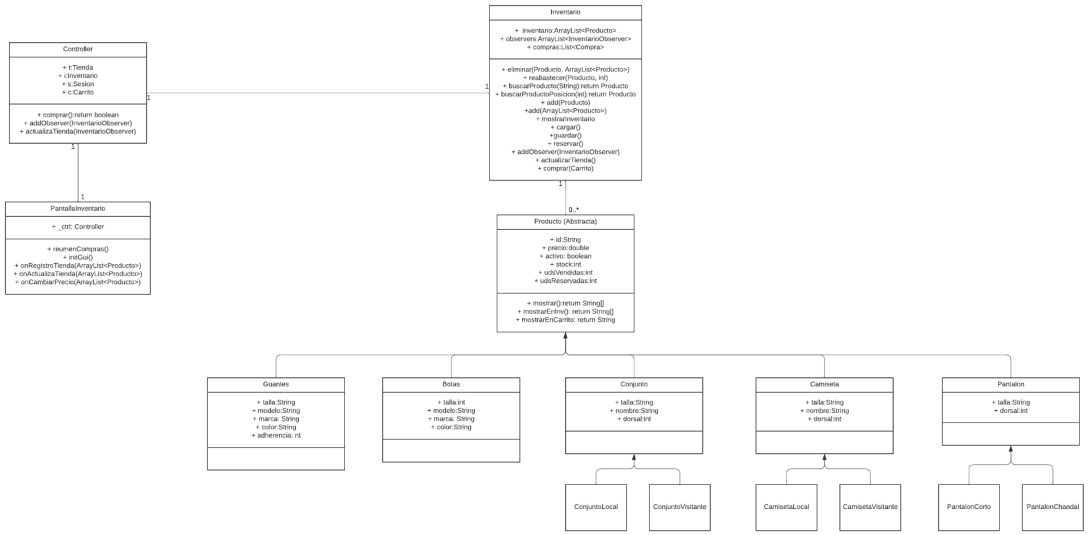


Diagrama de clases:



(Diagrama de clases Inventario)

Historia de usuario Añadir Producto

En el primer sprint creamos las clases del producto y un inventario. Se implementa la función add de la libreria, donde pasamos a la lista de productos el producto nuevo que deseamos añadir.

En el cuarto sprint siguiendo el modelo vista controlador se crea la interfaz del inventario y añadimos el botón de añadir producto, que llama a la función add implementada anteriormente.

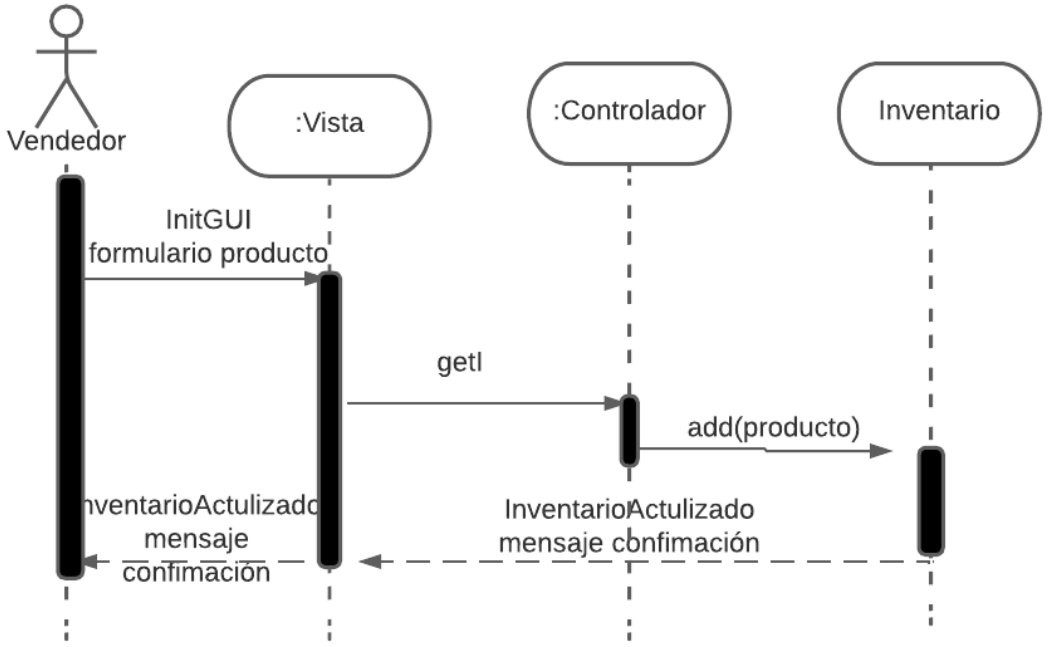


Diagrama de clases igual que en mostrar inventario

Historia de usuario Cambiar Precio Producto

En el primer sprint creamos las clases del producto y un inventario. Se implementa la función setPrice() que cambia el precio
En el cuarto sprint siguiendo el modelo vista controlador se crea la interfaz del inventario y añadimos el botón de cambiar precio, que llama a la función add implementada anteriormente.

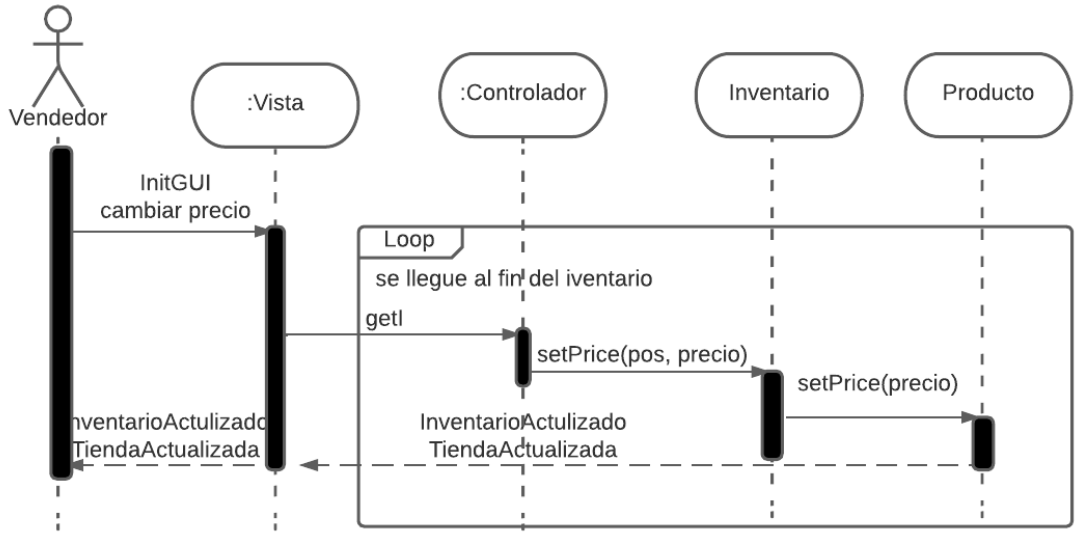


Diagrama de clases igual que en mostrar inventario

Historia de usuario Actualizar Tienda

En el cuatro sprint se añade la función Actualizar tienda, que actualiza la tienda cuando se añade un producto al inventario.

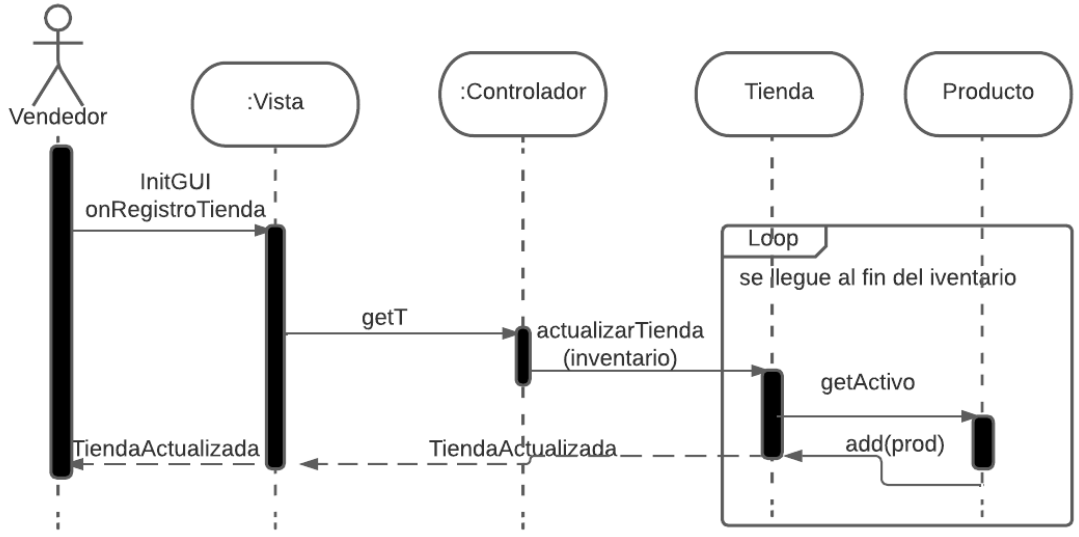


Diagrama de clases igual que en mostrar tienda

Historia de usuario Eliminar Producto

En el primer sprint hemos creado la clase Inventario donde metemos todos los productos existentes, tanto si se muestran en la tienda o no.
Añadimos diversos métodos, entre los cuales se encuentra eliminar(), donde pasándole la lista de productos de la tienda y el elemento a eliminar, eliminamos el producto tanto del inventario como de la tienda.

Diagrama de secuencia:

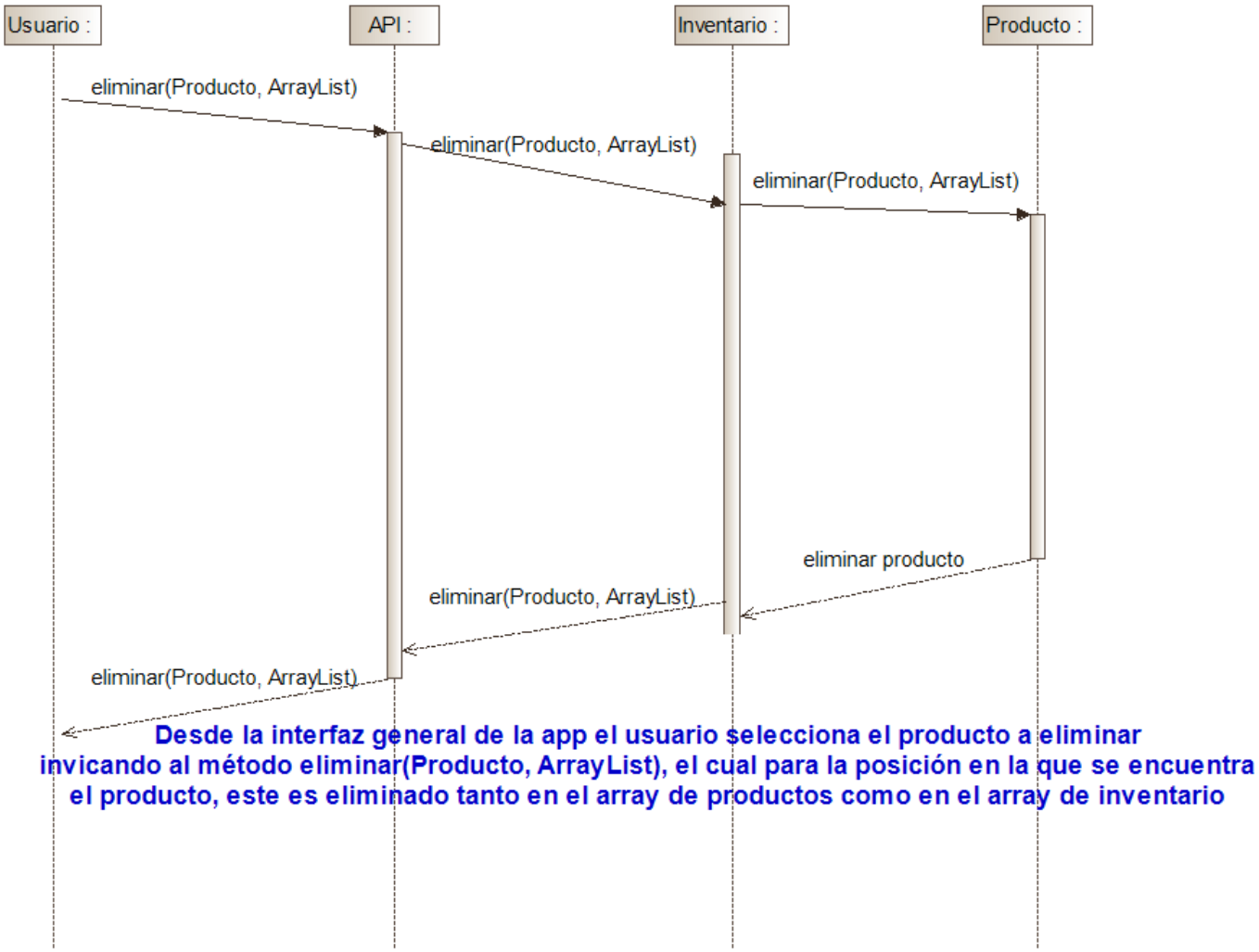


Diagrama de clases:
Mismo al de historia de usuario “Mostrar inventario”

Historia de usuario Reabastecer Producto

En el primer sprint, igual que pasaba con el método eliminar() de la clase Inventario, hemos creado un método llamado reabastecer(), el cual se encarga de añadir una cantidad “n” a un

elemento concreto del inventario. La modificación de dicho elemento queda reflejado tanto en el inventario como en la lista de productos de la tienda.

Diagrama de secuencia:

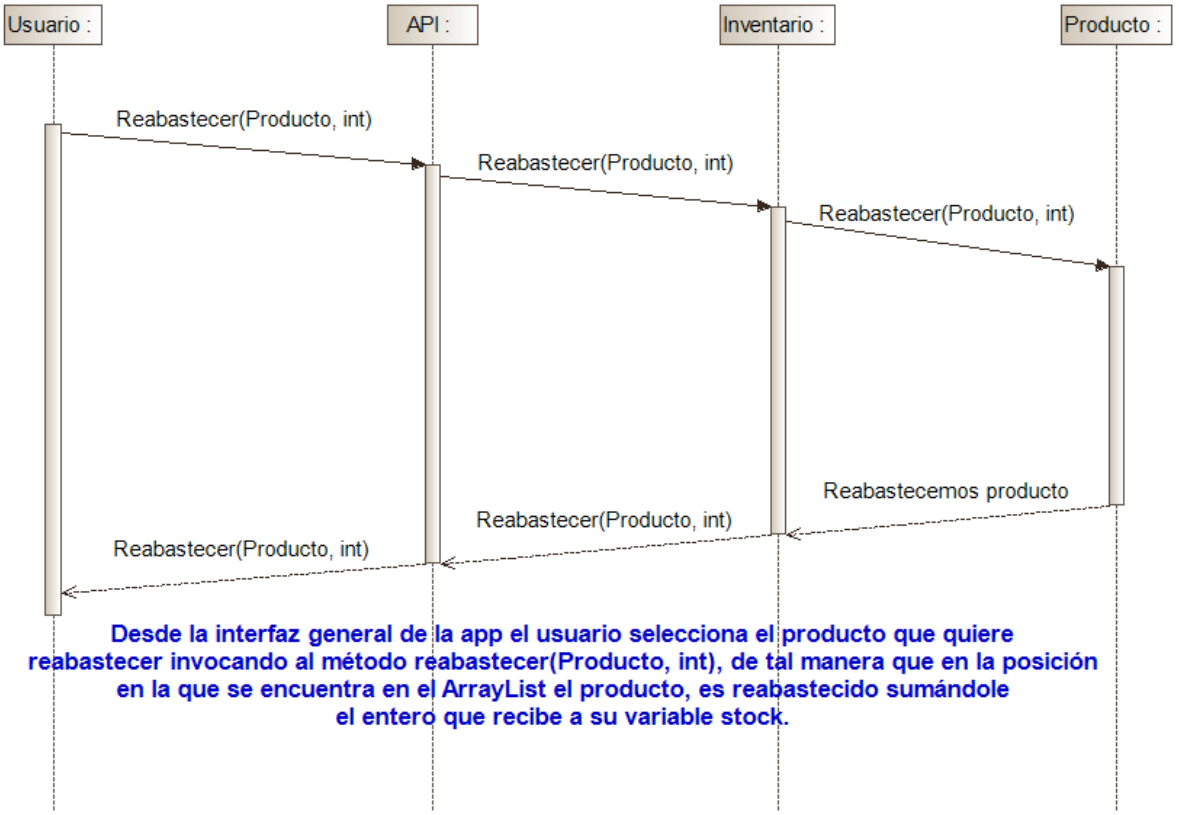
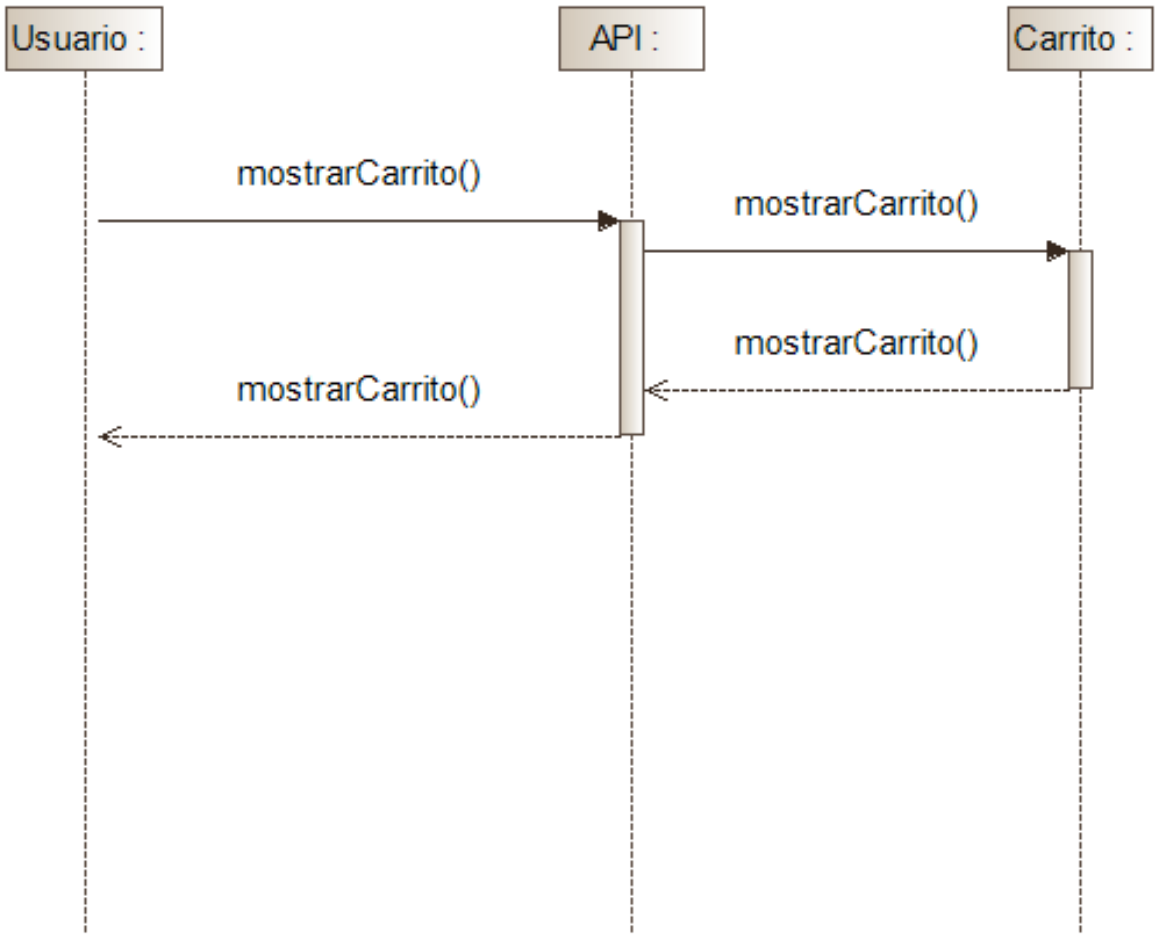


Diagrama de clases:
Mismo al de historia de usuario “Mostrar inventario”
No se ha llegado a implementar en la interfaz gráfica, solo está en el modelo

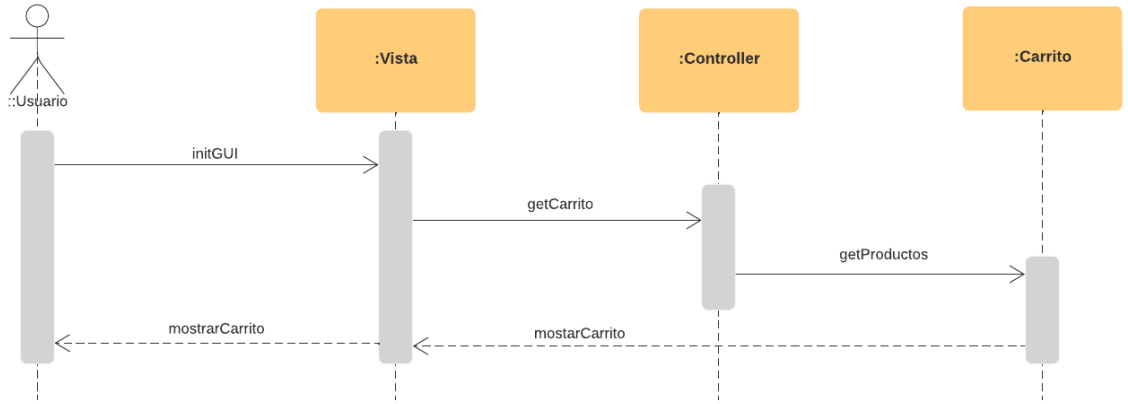
Historia de usuario Mostrar el Carrito

En el segundo sprint, antes de tener la interfaz gráfica creada, simplemente creamos un método mostrarCarrito() para mostrar una lista de productos del por pantalla, en formato textual.



Desde la interfaz general de la APP, selecciona el usuario el carrito invocando el método mostrarCarrito(), mostrando por pantalla los productos que estén añadidos en el carrito.

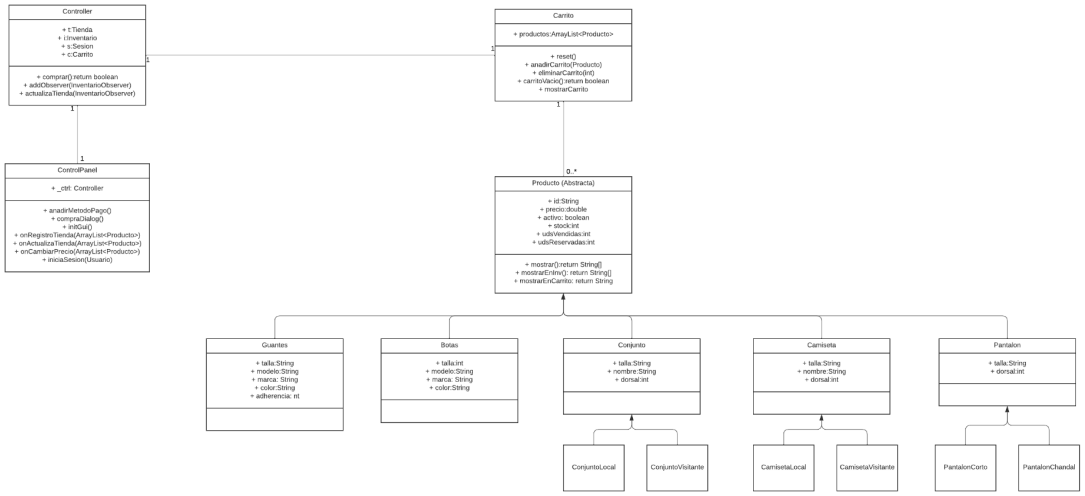
En el cuarto sprint, se ha modificado este diagrama de secuencias al introducir el modelo-vista-controlador:



A diferencia del anterior diagrama de secuencia, en este tenemos que para llegar a mostrar los productos en la interfaz hay que hacer uso del controller el cual tiene una instancia con

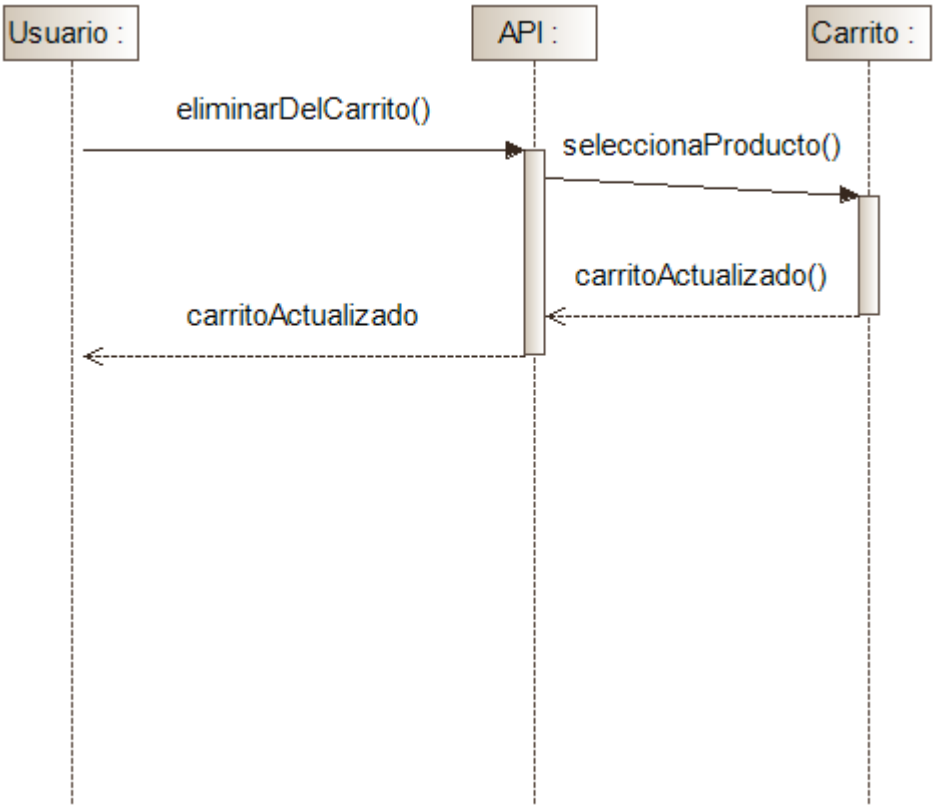
el estado del Carrito y por lo tanto a través de este podemos llamar a la función getProductos() de Carrito que nos devuelve la lista de productos que se mostrará en la interfaz gráfica.

Diagrama de clases:



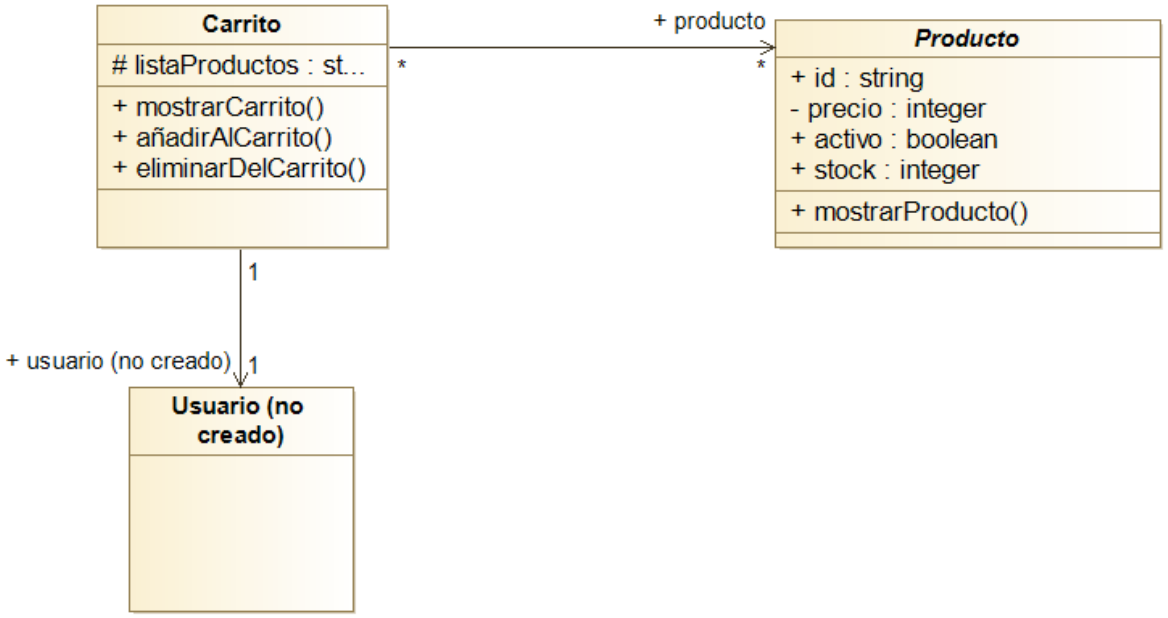
Historia de usuario Eliminar del Carrito

En el segundo sprint, hemos añadido una nueva clase que es Carrito, donde tenemos los productos que el cliente desea comprar. Dentro de esta clase tenemos diferentes métodos, entre los cuales está eliminarCarrito(int productoCarrito), con él lo que hacemos es eliminar un producto de la lista del carrito, para ello al método le pasamos la posición del array en la que se encuentra el producto que se desea eliminar.



Desde la interfaz general de la APP, el usuario selecciona la opción de eliminar del carrito, primero selecciona el producto que quiere eliminar, tecleando el número del producto,(con interfaz gráfica se seleccionará clickando). Si el producto es válido, el carrito se actualizará correctamente.

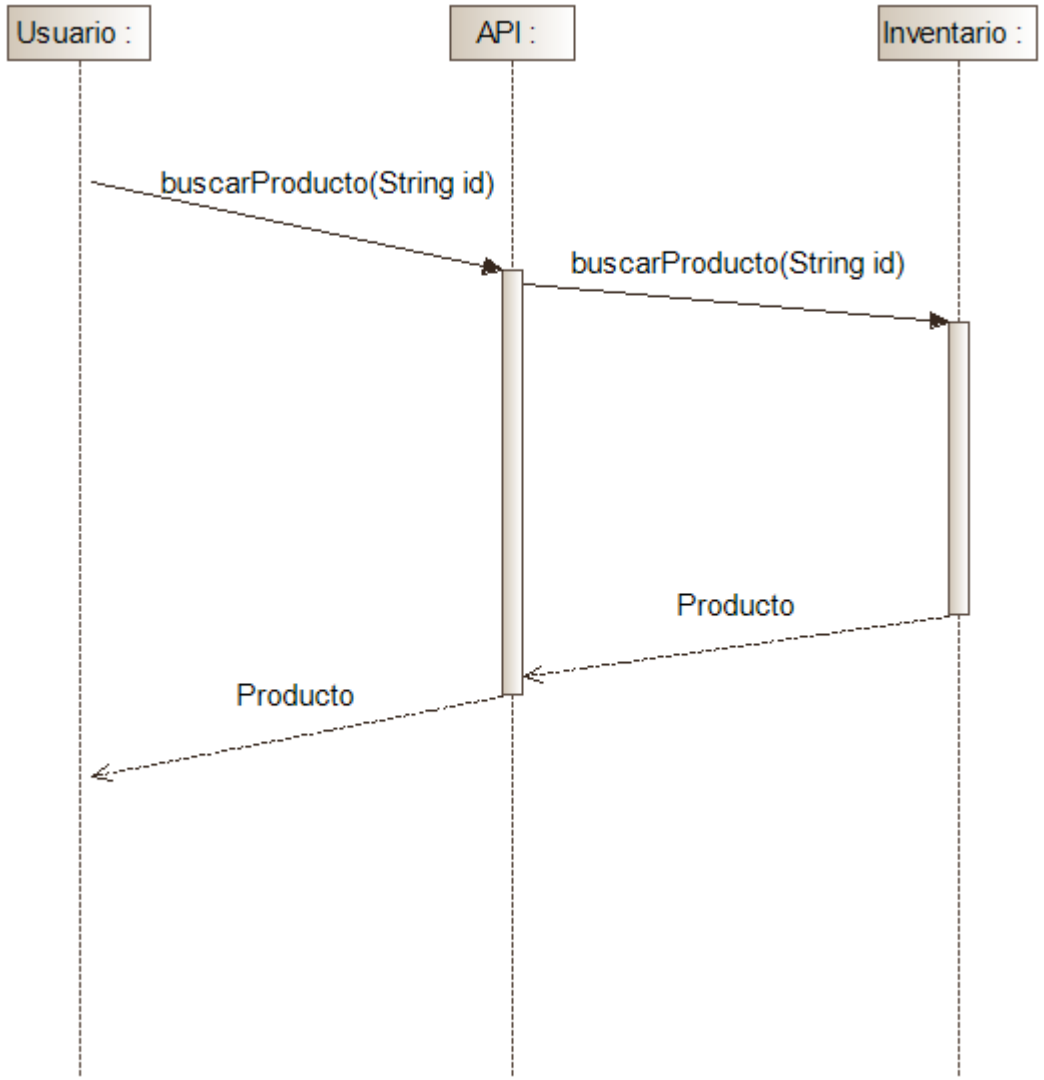
Diagrama de clases:



No se ha llegado a implementar en la interfaz gráfica, solo está en el modelo

Historia de usuario Buscar producto

En el segundo sprint, hemos añadido el método buscarProducto(string id) a la clase inventario, que recibe un id y devuelve el producto si está en el inventario y null si no está. De esta manera sabemos si un producto está o no está en el inventario.
Diagrama de secuencias:

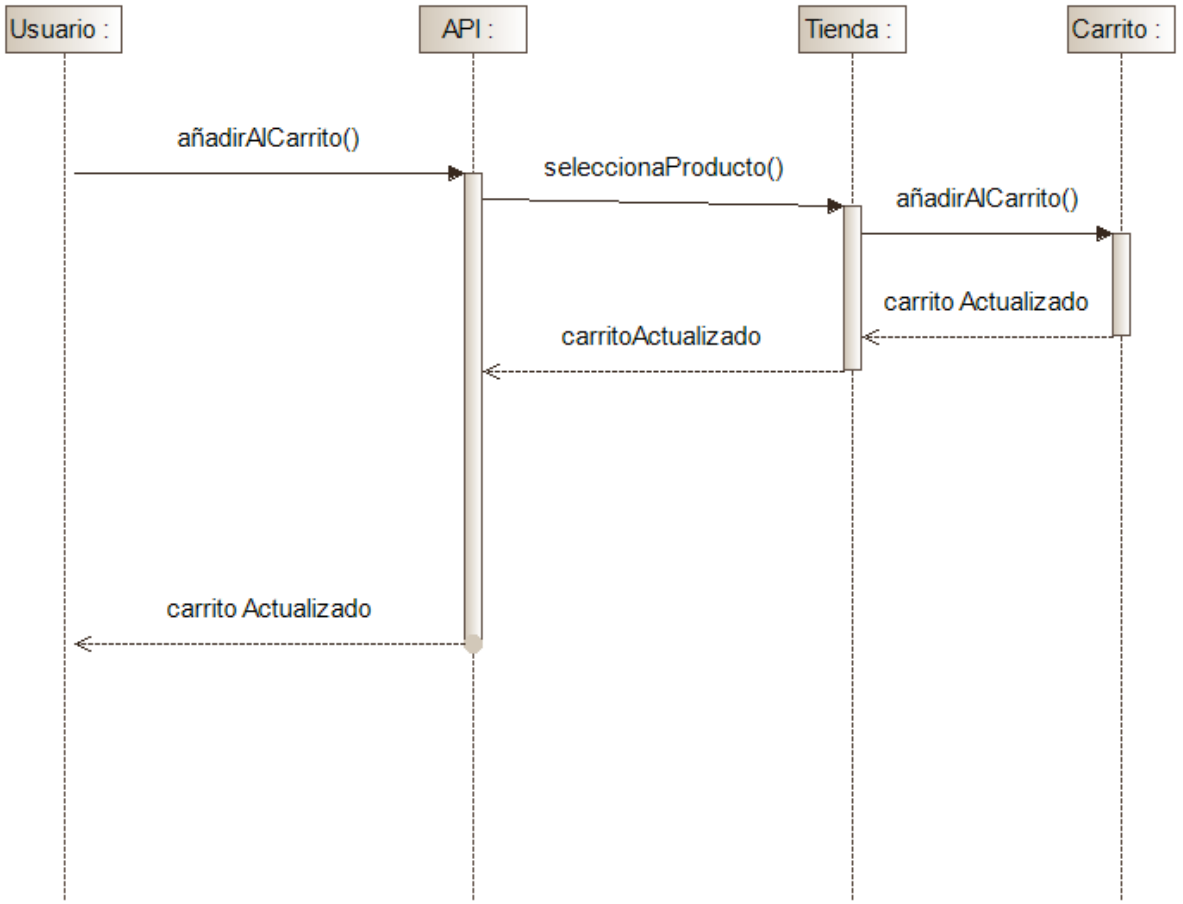


Desde la interfaz general de la app, el usuario (administrador) selecciona el id del producto que quiere buscar. Si el producto se encuentra entonces se devuelve el producto, sino se lanza un mensaje de que no se encontró el producto con ese id.

Diagrama de clases:
Mismo al de historia de usuario “Mostrar inventario”

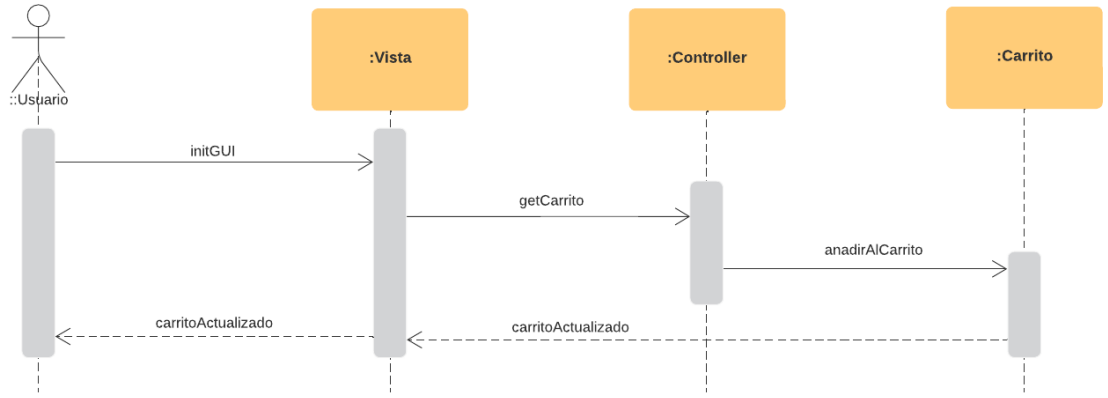
Historia de usuario Añadir al Carrito

En el segundo sprint, como ya hemos dicho, hemos creado la nueva clase carrito. En esta clase tenemos un método anadirCarrito(int productoCarrito)) que lo que hace es añadir a la lista de productos del carrito un nuevo producto de la tienda el cual es el que se pasa por el parámetro del método.



Desde la interfaz general de la APP, el usuario selecciona la opción añadir al carrito. Entonces, selecciona el producto a añadir, tecleando su número de producto (con interfaz gráfica se seleccionará clickando). Si el producto es válido, el carrito se actualizará correctamente.

En el cuarto sprint, se ha modificado este diagrama de secuencias al introducir el modelo-vista-controlador:



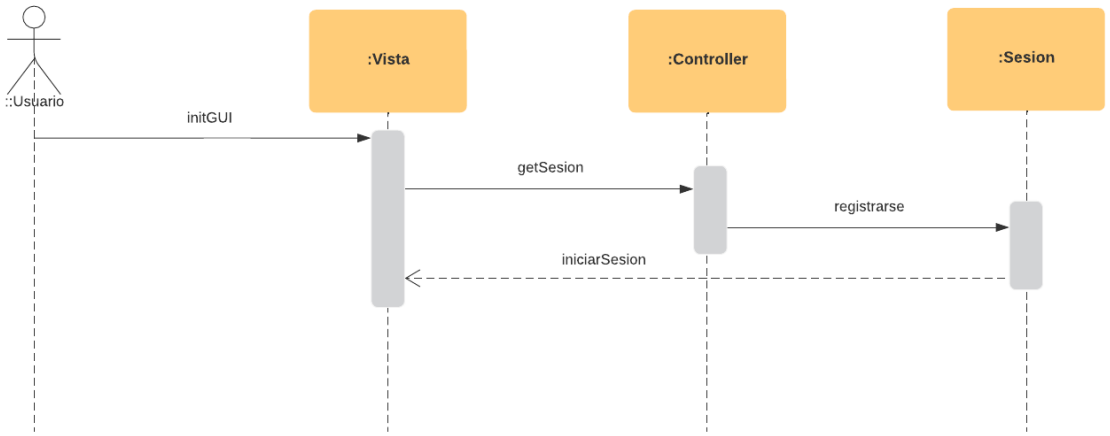
A diferencia del anterior diagrama de secuencia, en este tenemos que para llegar a añadir un nuevo producto al carrito hay que hacer uso del controller el cual tiene una instancia con el estado del Carrito y por lo tanto a través de este podemos llamar a la función anadirAlCarrito() de Carrito la cual añade a la lista de productos del carrito este nuevo producto que se mostrará en la interfaz gráfica.

Diagrama de clases:

Mismo al de historia de usuario “Mostrar Carrito”

Historia de usuario Crear Cuenta Usuario Cliente

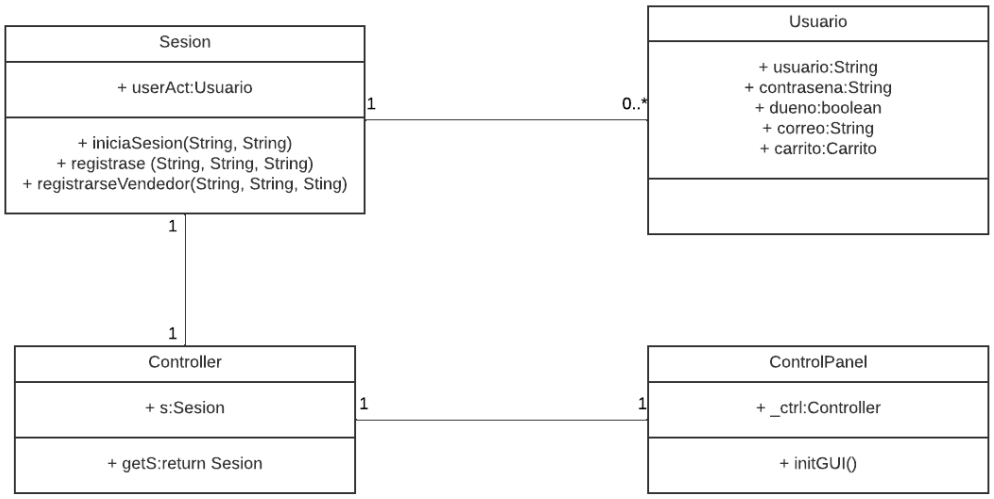
En el cuarto sprint, hemos creado una nueva clase Sesion, que nos permite guardar el usuario que está usando actualmente la aplicación, así como iniciar una nueva sesión con otro usuario o registrar a un nuevo usuario, diferenciando los usuarios entre Cliente y Vendedor.



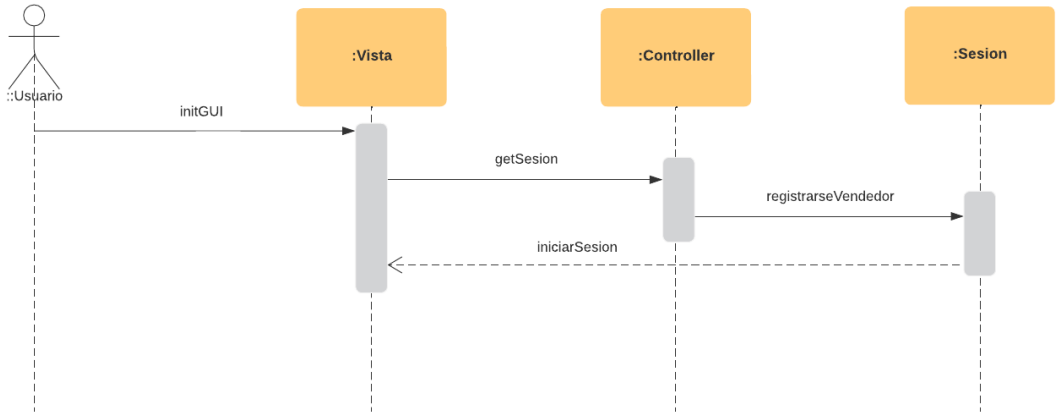
El usuario, en este caso un cliente, desde la interfaz puede registrarse introduciendo una serie de datos (usuario, mail y contraseña), por medio del controller, el cual tiene una

instancia con el estado de Sesion, se puede acceder a la clase Sesion que es la que se encargará de añadir al nuevo usuario.

Diagrama de clases:



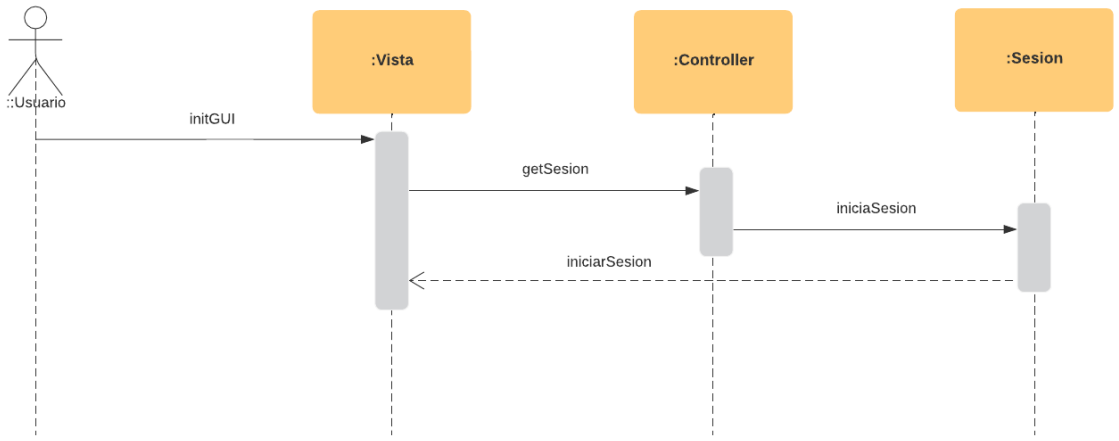
Historia de usuario Crear Cuenta Usuario Vendedor



El usuario, en este caso un vendedor, desde la interfaz puede registrarse introduciendo una serie de datos (usuario, mail y contraseña), por medio del controller,el cual tiene una instancia con el estado de Sesion, se puede acceder a la clase Sesion que es la que se encargará de añadir al nuevo usuario.

Diagrama de clases:
Mismo al de historia de usuario “Crear Cuenta Usuario Cliente”

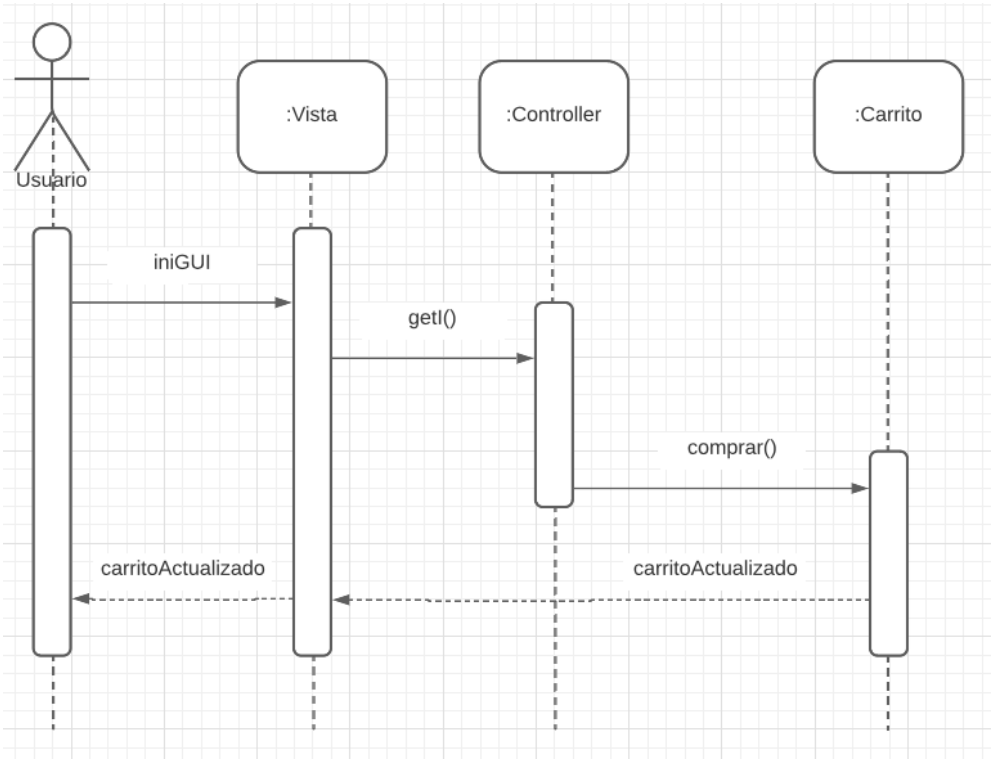
Historias de usuario Iniciar Sesión Usuario Cliente / Vendedor



El usuario (vendedor o cliente), desde la interfaz inicia sesión introduciendo una serie de datos (usuario y contraseña), por medio del controller, el cual tiene una instancia con el estado de Sesion, se puede acceder a la clase Sesion que es la que se encargará de iniciar la sesion.

Diagrama de clases:
Mismo al de historia de usuario “Crear Cuenta Usuario Cliente”

Historia de usuario Comprar

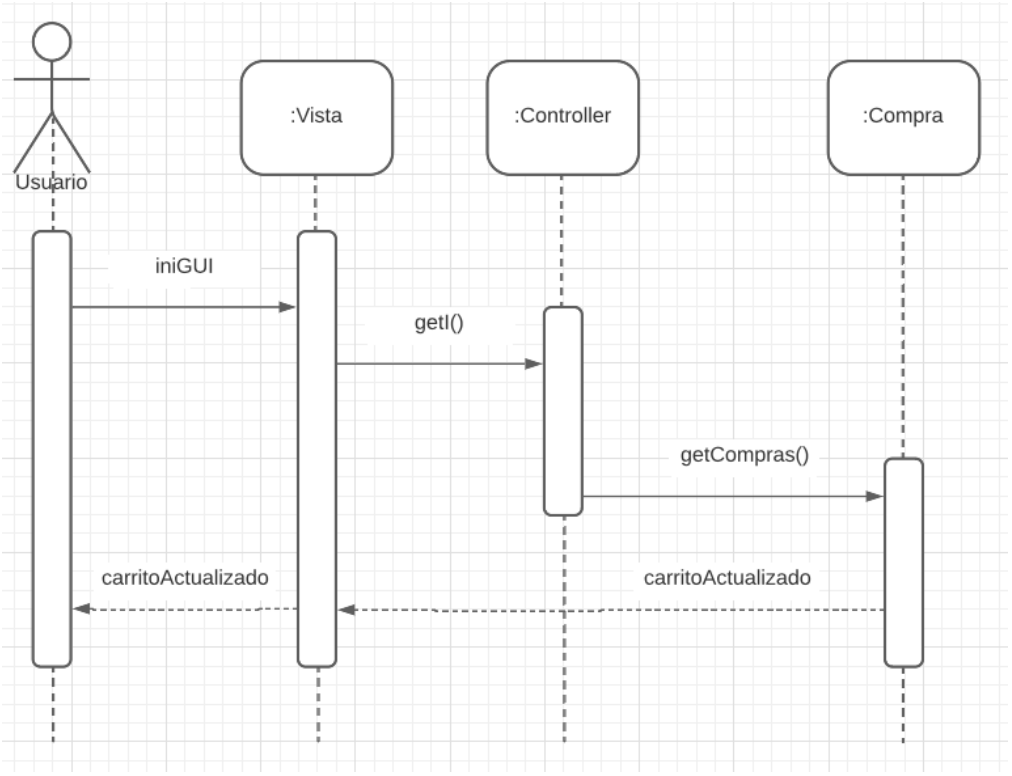


En el **cuarto sprint**, hemos añadido un nuevo método (comprar(Carrito carrito)) a la clase Inventario la cual se encarga de disminuir el stock de los productos del carrito del usuario.
Diagrama de clases:

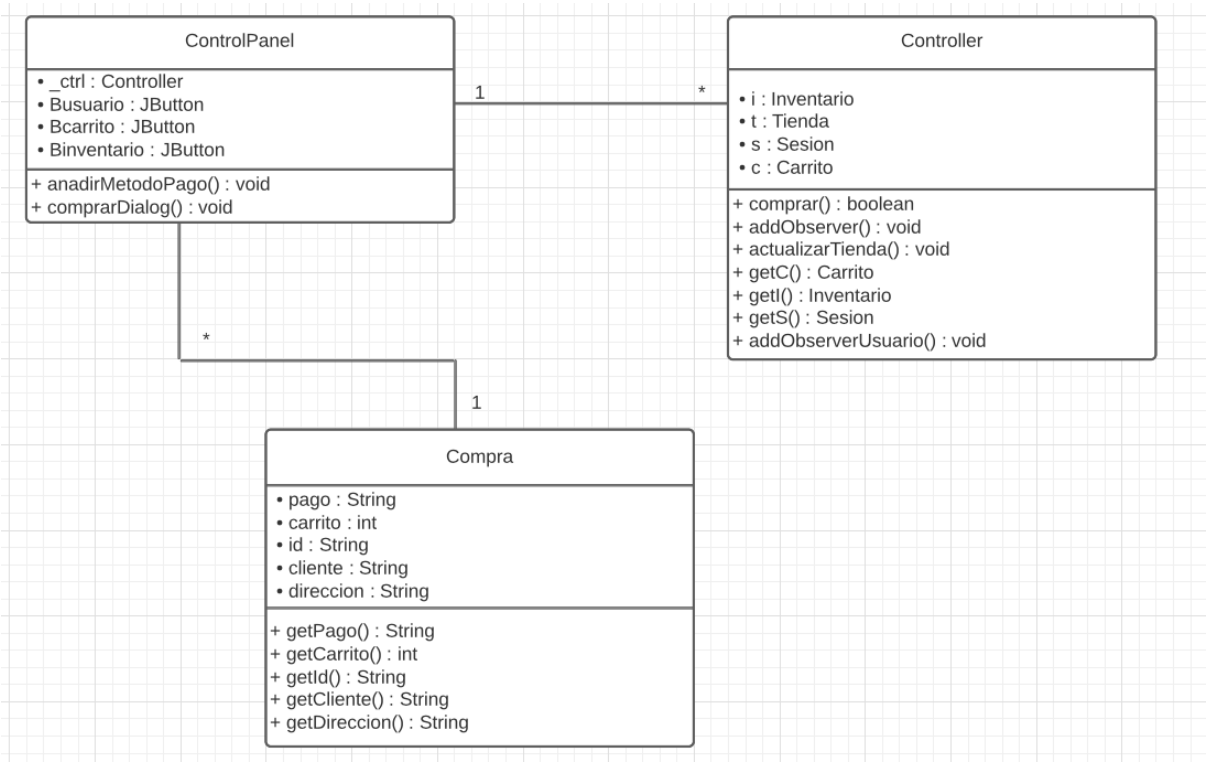
Mismo al de historia de usuario “Mostrar Inventario”

De igual manera se ha añadido un método (comprarDialog()) en la clase ControlPanel en el cual se deben introducir una serie de datos (nombre y dirección, y método de pago) para hacer efectiva la compra.

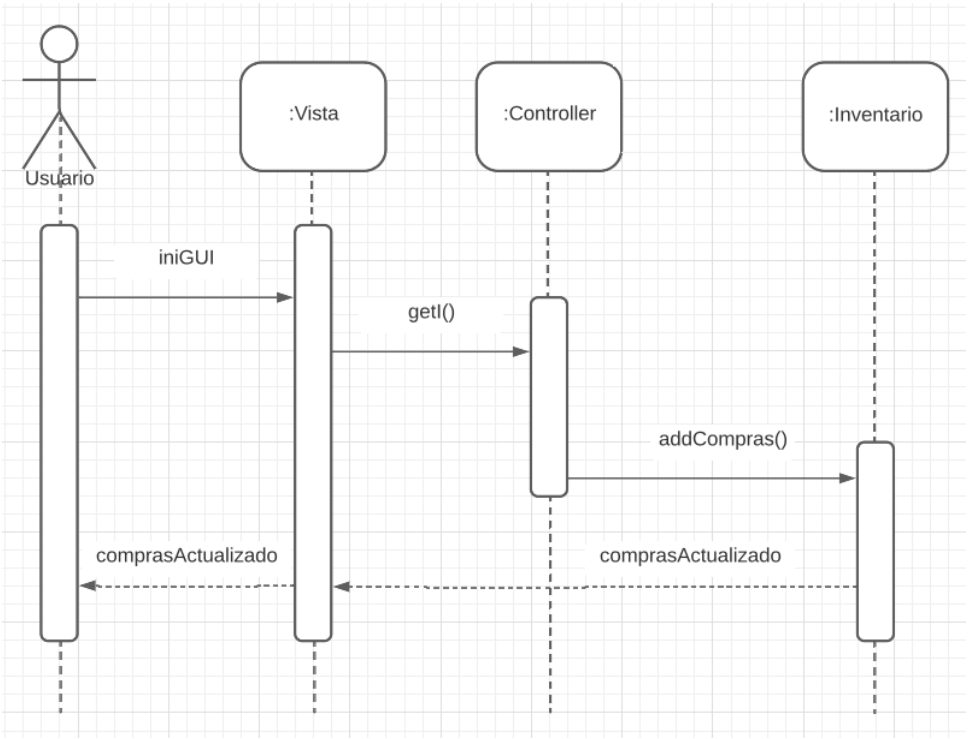
Historia de usuario Añadir método de pago



En el **cuarto sprint**, hemos añadido un nuevo método (anadirMetodoPago(String nombre, String Direccion)) a la clase ControlPanel el cual se encarga de permitir rellenar una serie de datos (elegir el método de pago e introducir el titular y número de cuenta) para poder efectuar la compra antes iniciada.
Diagrama de clases:

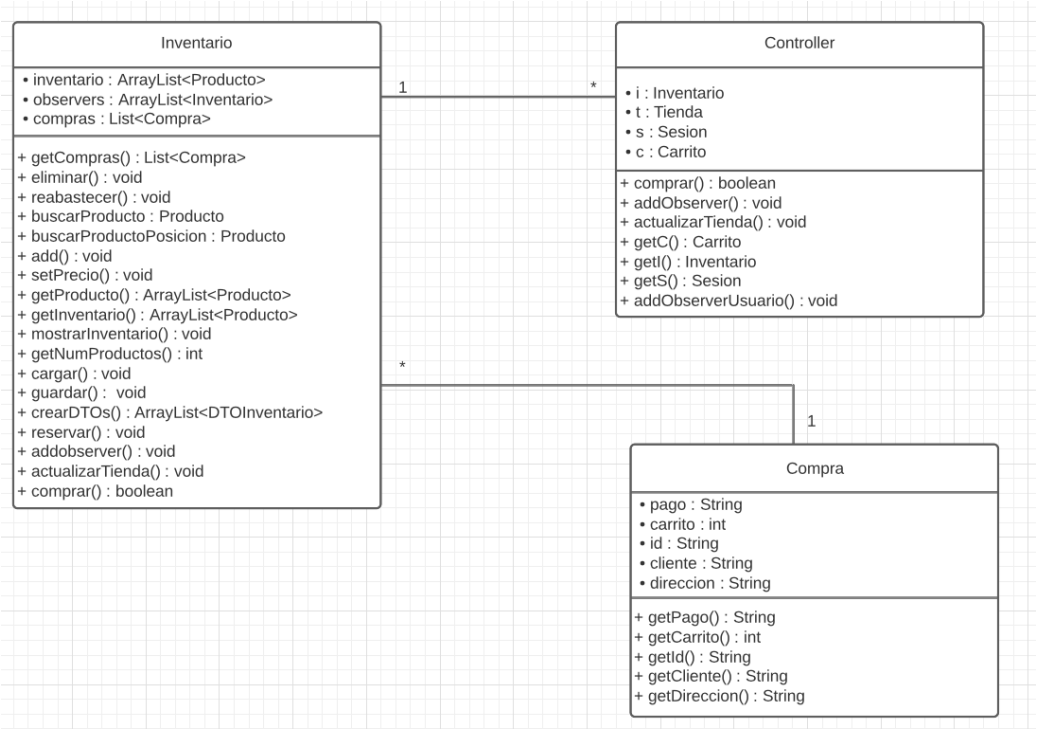


Historia de usuario Resumen de compras



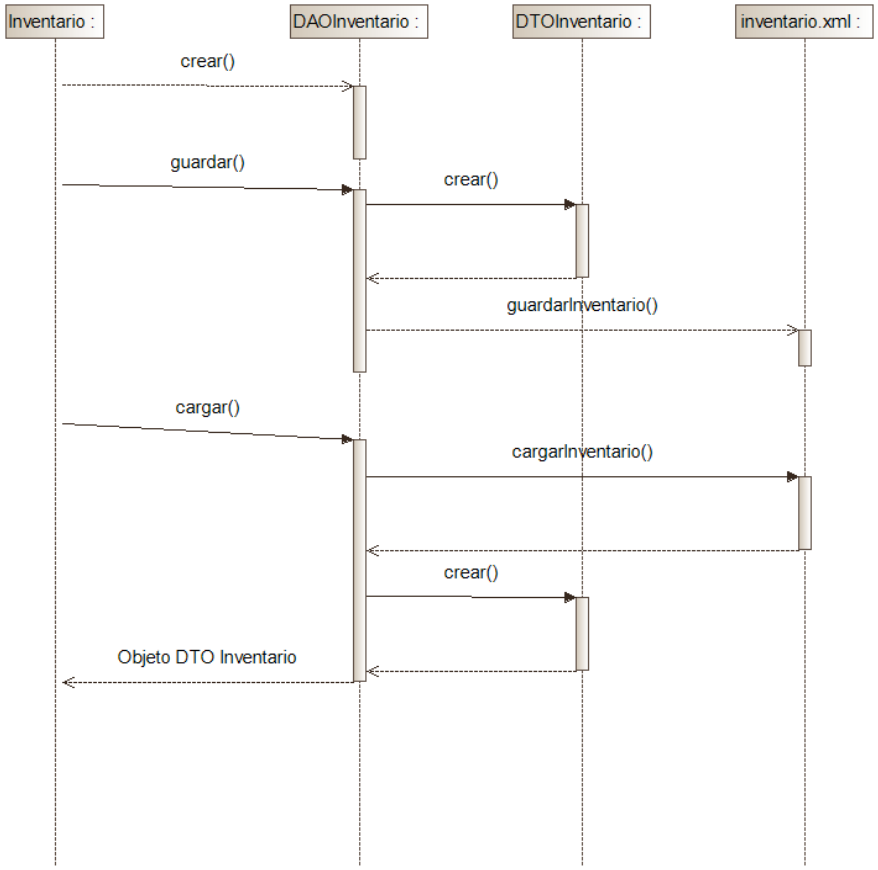
En el **cuarto sprint**, hemos creado la clase Compra la cual guarda la información de cada compra realizada durante la sesión actual.
De igual manera, hemos añadido un nuevo método (resumenCompra()) en la clase ControlPanel para guardar y mostrar el resumen de las compras efectuadas en la sesión actual.

Diagrama de clases:



Patrones DAO (DAOInventario)

En el tercer sprint, hemos creado dos clases nuevas (DAOXMLInventario y DTOInventario) para poder leer y guardar en una base de datos en XML los productos de la clase Inventario (ya creada en anteriores sprints).



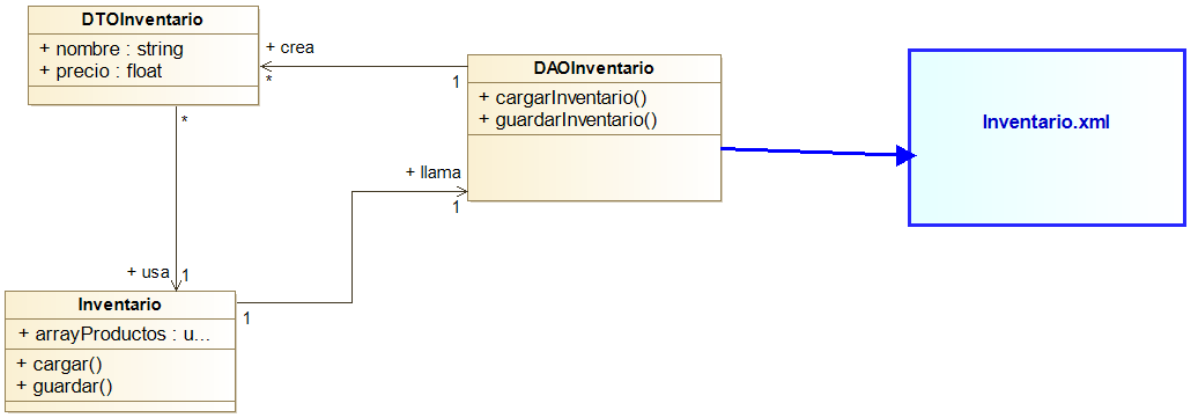
Guardar en la base de datos:

El Inventario actualiza algún valor del DTOInventario y solicita al DAOInventario el guardado de los datos actualizados y este ya se encarga de guardar los datos en el archivo inventario.xml.

Cargar de la base de datos:

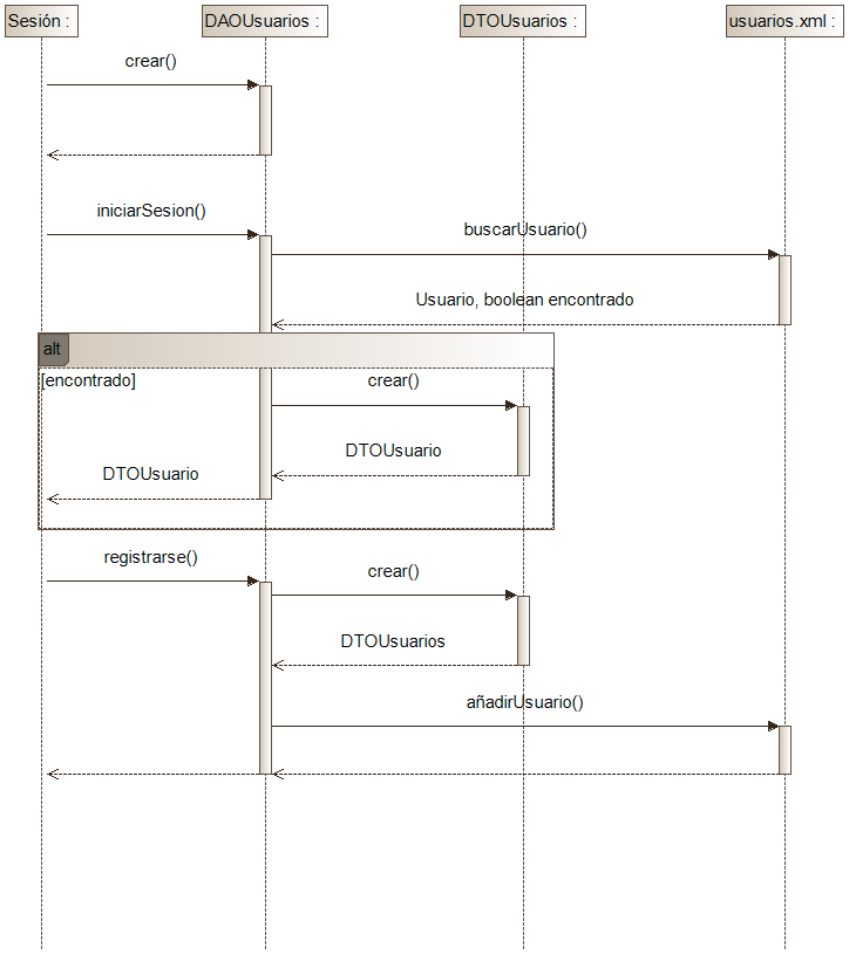
El inventario crea una referencia al DAOInventario, luego el Inventario solicita información al DAOInventario y este solicita la información a inventario.xml, a su vez, desde DAOInventario se crea una instancia de DTOInventario con los datos recuperados de inventario.xml, el DAOInventario responde con el Inventario creada en los pasos anteriores.

Diagrama de clases:



Patrones DAO (DAOUsuarios)

En el cuarto sprint, hemos creado dos clases nuevas (DAOXMLUsuarios y DTOUsuarios) para poder leer y guardar en una base de datos en XML la información de los diferentes usuarios del programa, para ello también hemos tenido que crear una nueva clase Sesion que tiene el usuario actual y varios métodos para iniciar sesión y registrar a los usuarios.



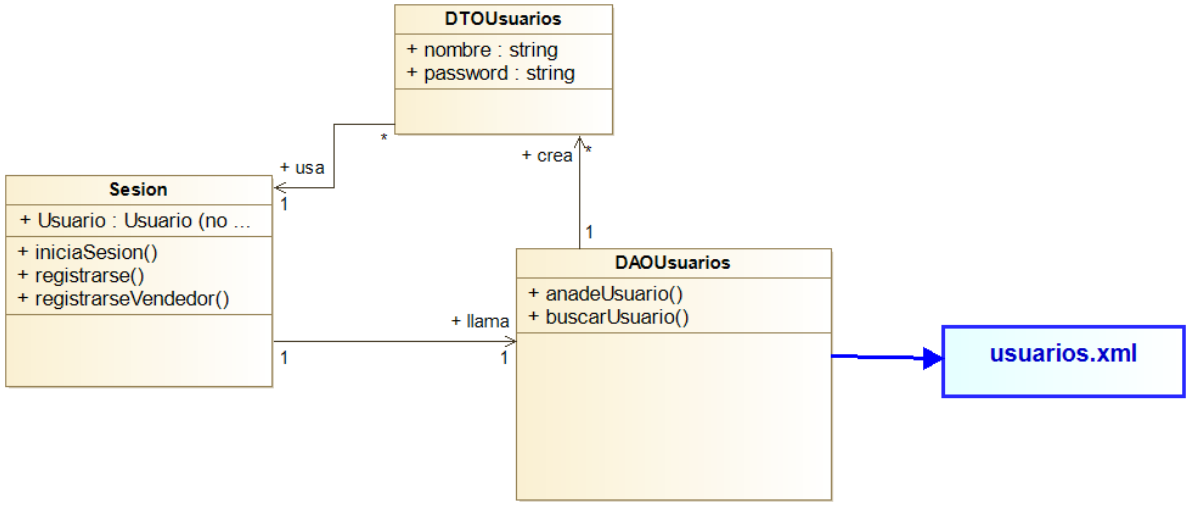
Registrarse:

La sesión actualiza algún valor del DTOUsuarios y solicita al DAOUsuarios el guardado de los datos actualizados y este ya se encarga de guardar los datos en el archivo usuarios.xml.

Iniciar Sesión:

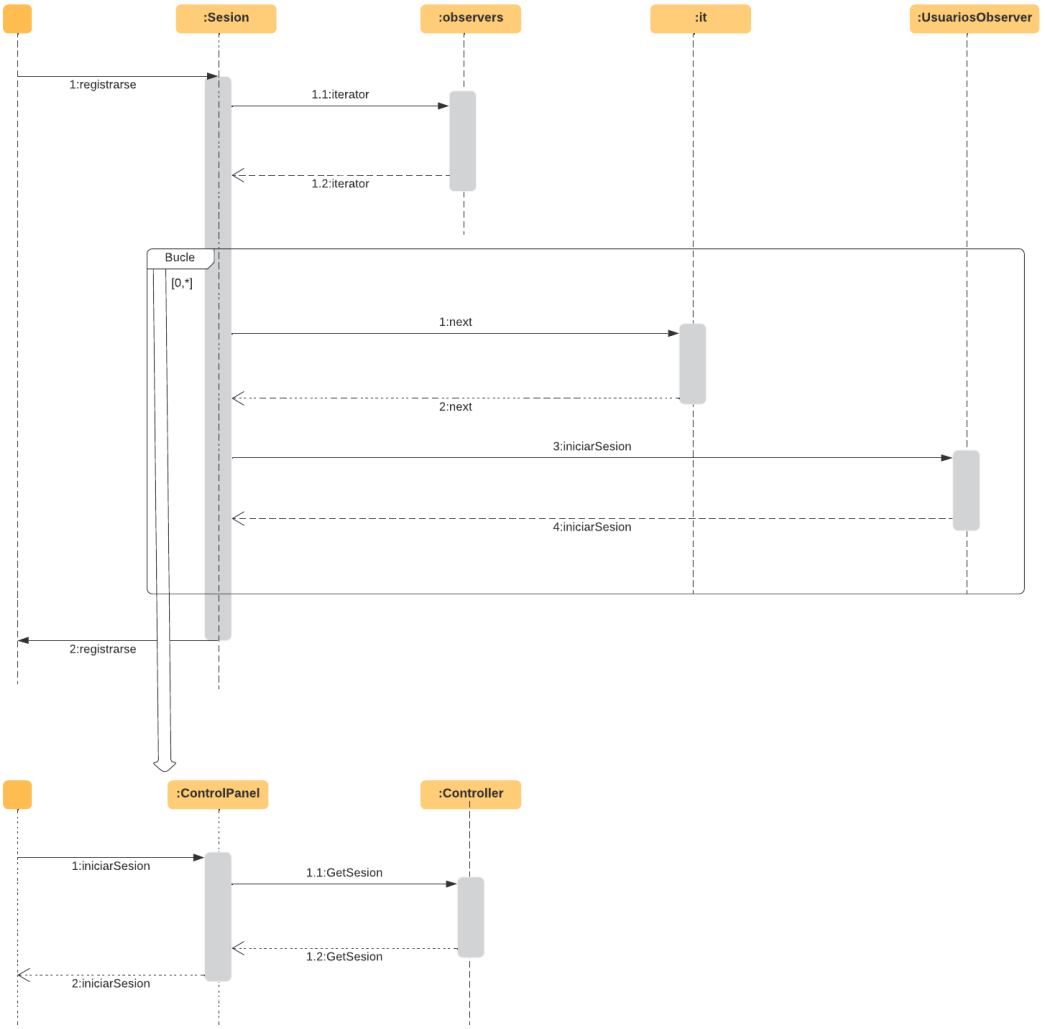
La sesión crea una referencia al DAOUsuarios, luego la se solicita información al DAOUsuarios y este solicita la información a usuarios.xml, a su vez, desde DAOUsuarios se crea una instancia de DTOUsuarios con los datos recuperados de usuarios.xml, el DAOUsuarios responde con la sesion creada en los pasos anteriores.

Diagrama de clases:



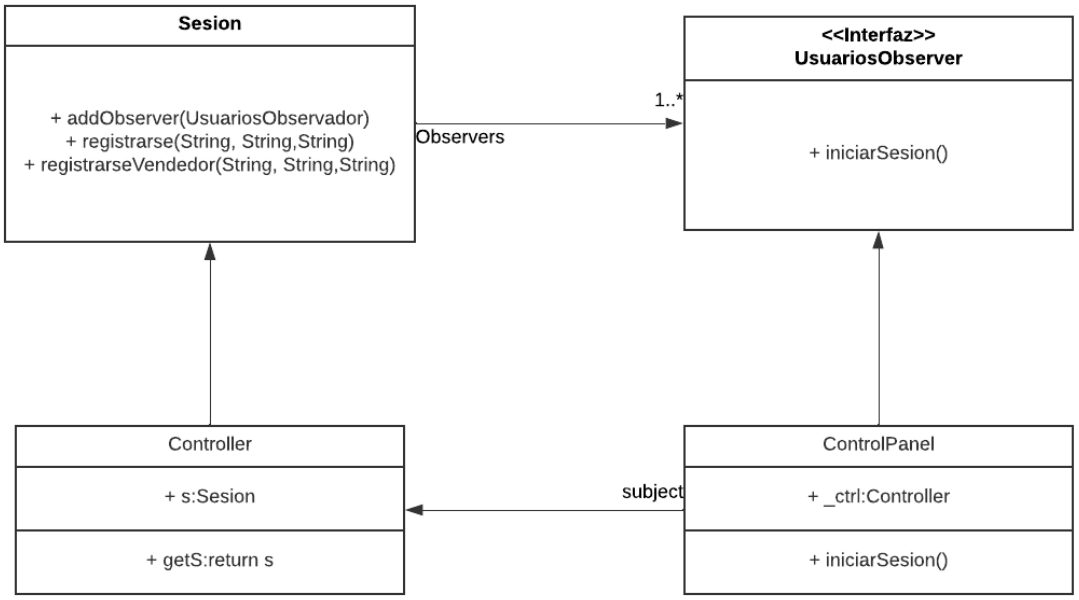
PATRÓN OBSERVER (UsuariosObserver)

En el cuarto sprint, hemos añadido un nuevo patrón de diseño de software, el patrón Observer, para ello hemos creado una interfaz UsuariosObserver, la cual la implementa ControlPanel para realizar los cambios correspondientes en la vista, para ellos en la clase Controller hemos añadido un atributo de la clase Sesion.



Controller notifica a sus observadores cualquier cambio que se produzca. Una vez notificado el cambio, el ControlPanel puede notificar al Sujeto el mismo, actualizando el estado en el Controller.

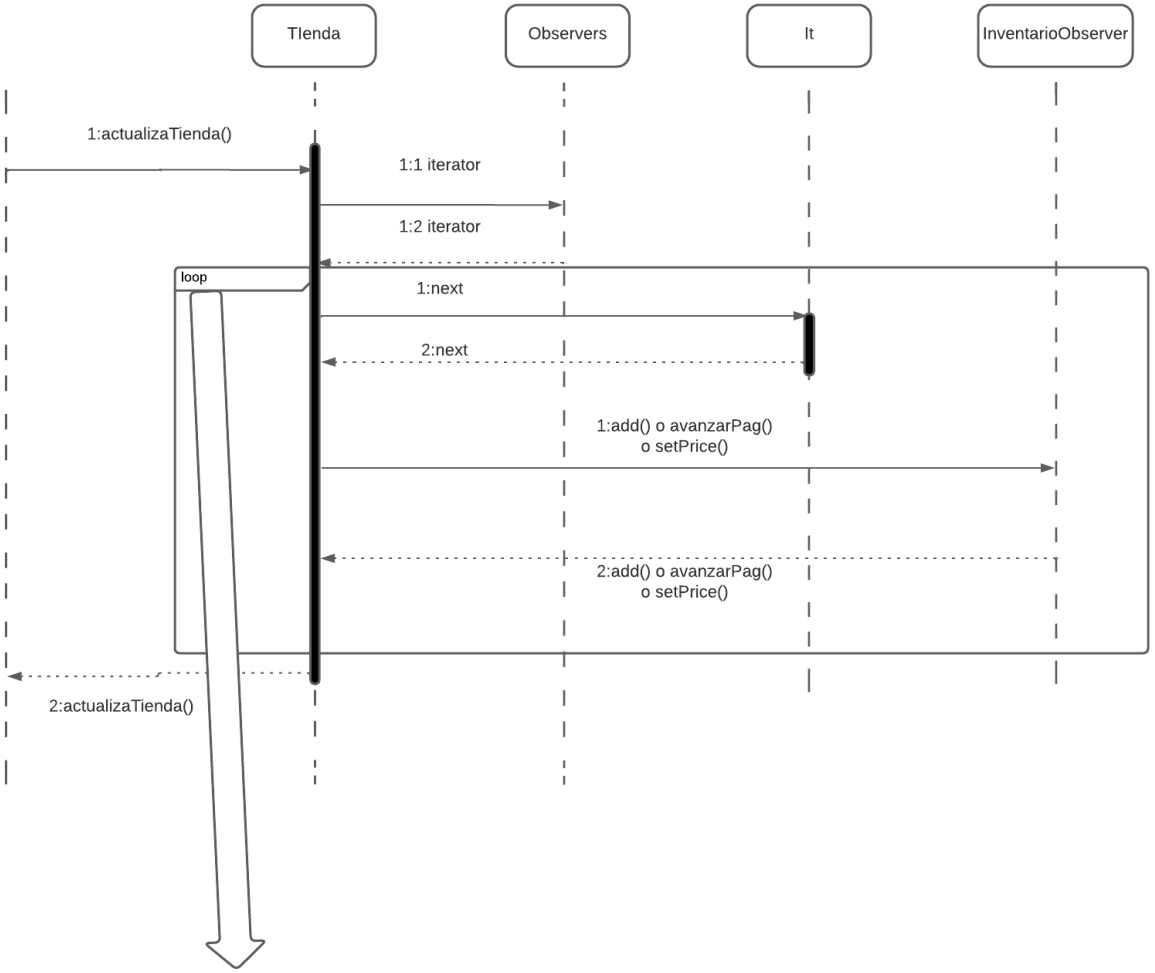
Diagrama de clases:



PATRÓN OBSERVER (InventarioObserver)

En el cuarto sprint, mediante el patrón Observer, hemos creado otra interfaz llamada InventarioObserver, la cual la implementa PantallaTienda para realizar los cambios correspondientes en la vista.

Diagrama de secuencias:



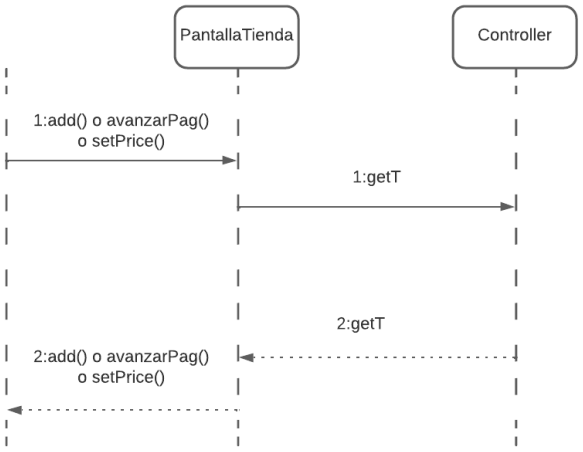


Diagrama de clases:

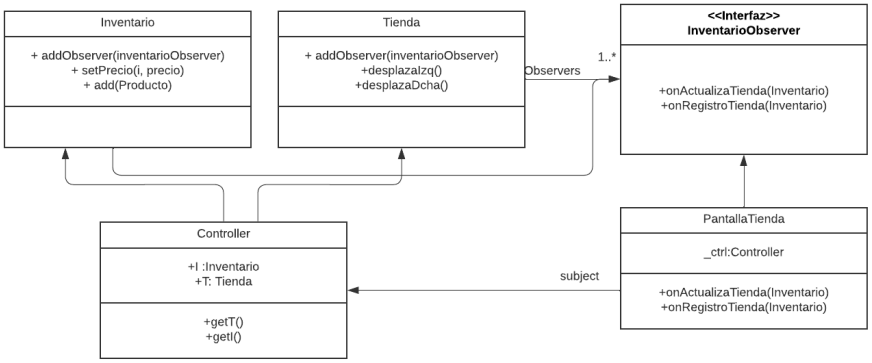


DIAGRAMA DE ACTIVIDADES

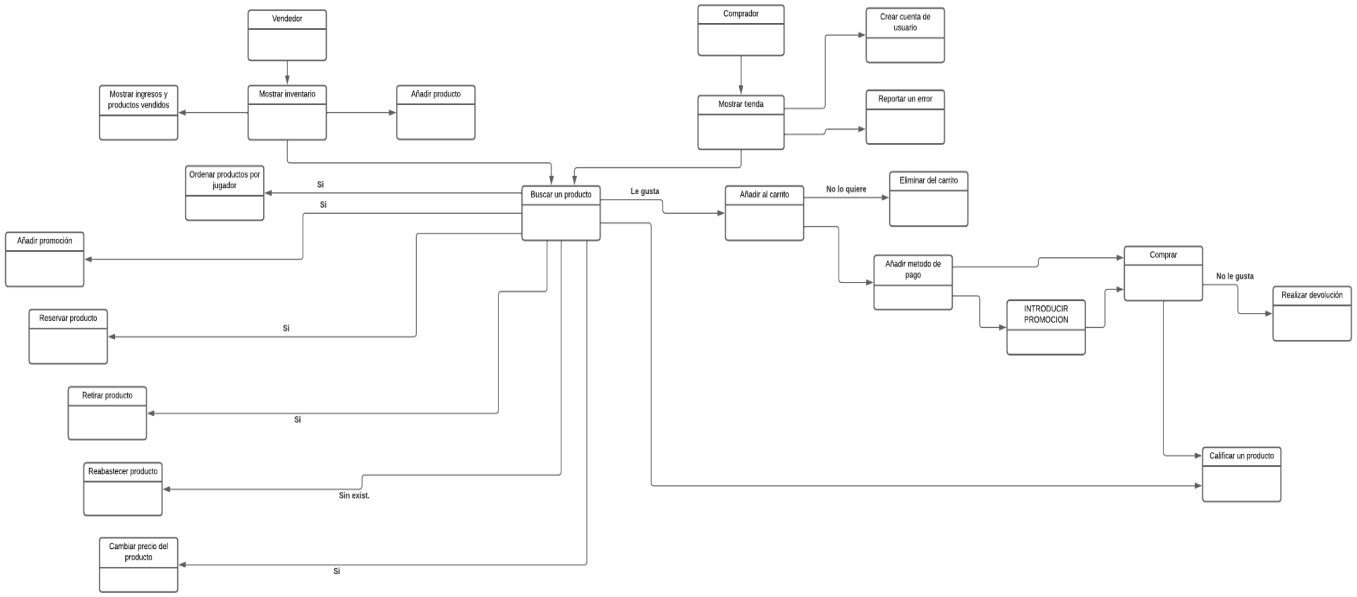


DIAGRAMA DE DOMINIO

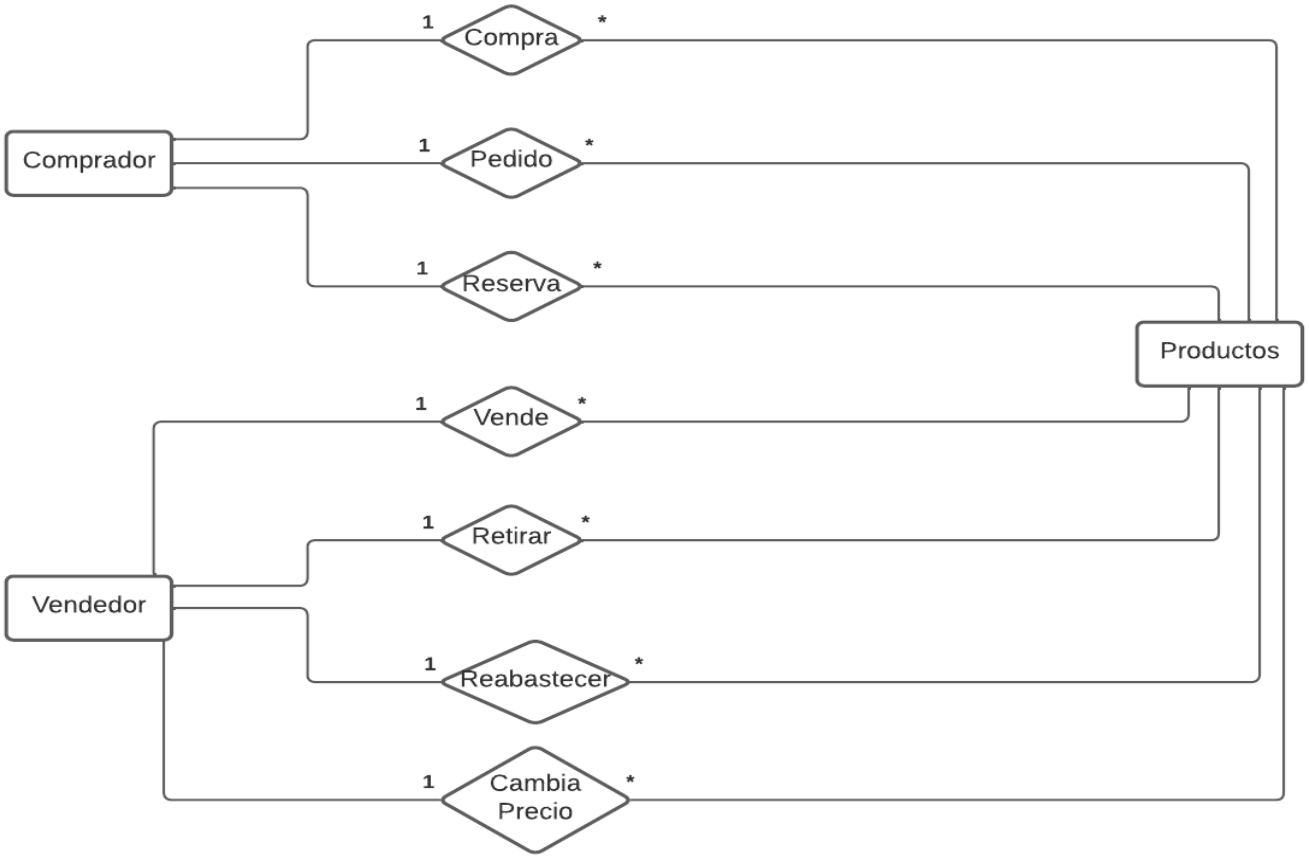


DIAGRAMA DE CASO DE USO

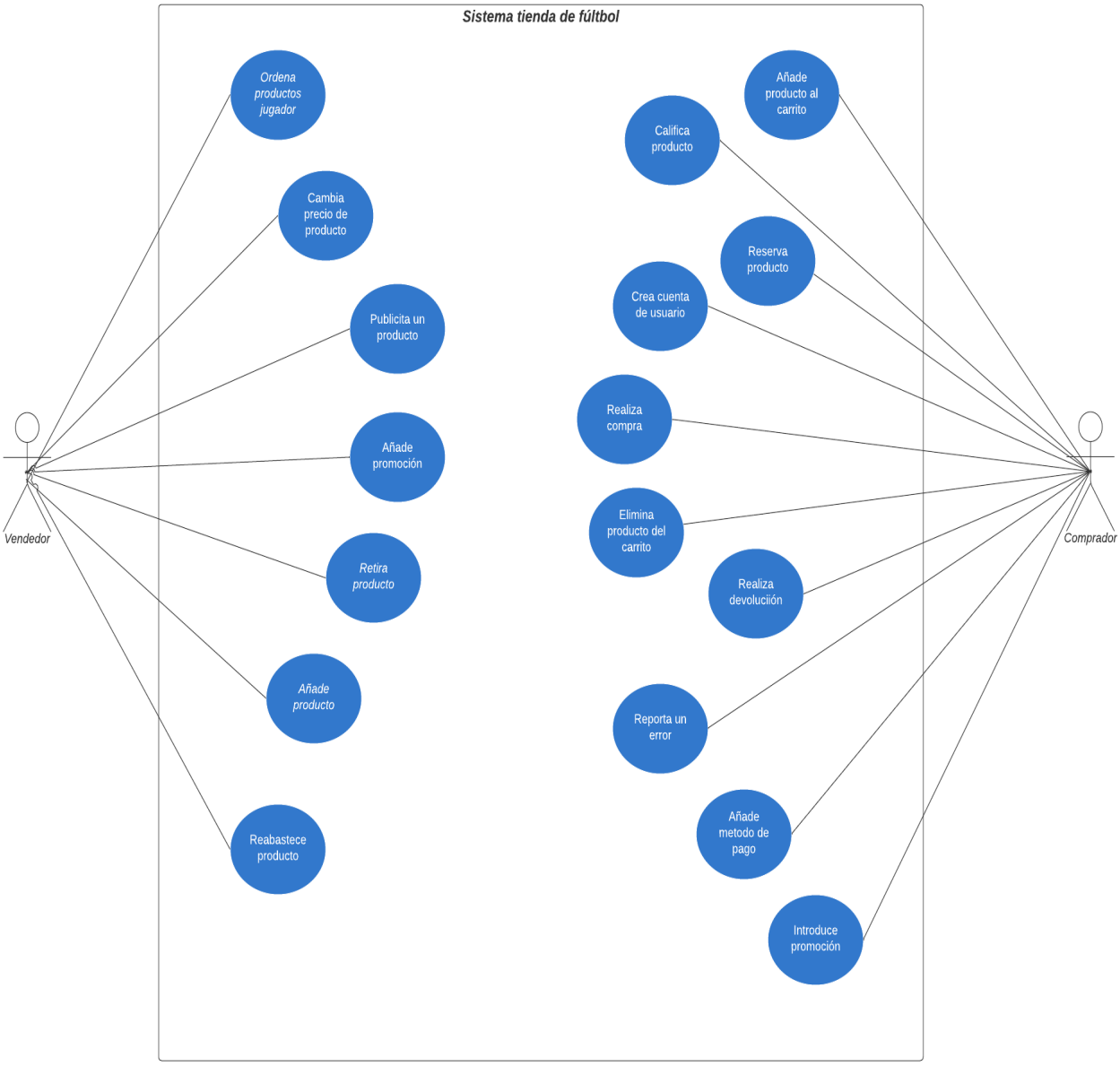

Sistema tienda de fúltbol
Ordena productos jugador
Cambia precio de producto
Publicita un producto
Añade promoción
Retira producto
Añade producto
Reabastece producto
Añade producto al carrito
Califica producto
Reserva producto
Crea cuenta de usuario
Realiza compra
Elimina producto del carrito
Realiza devolución
Reporta un error
Añade metodo de pago
Introduce promoción
Vendedor
Comprador

DIAGRAMA CASO DE USO (Comprador - Sistema)

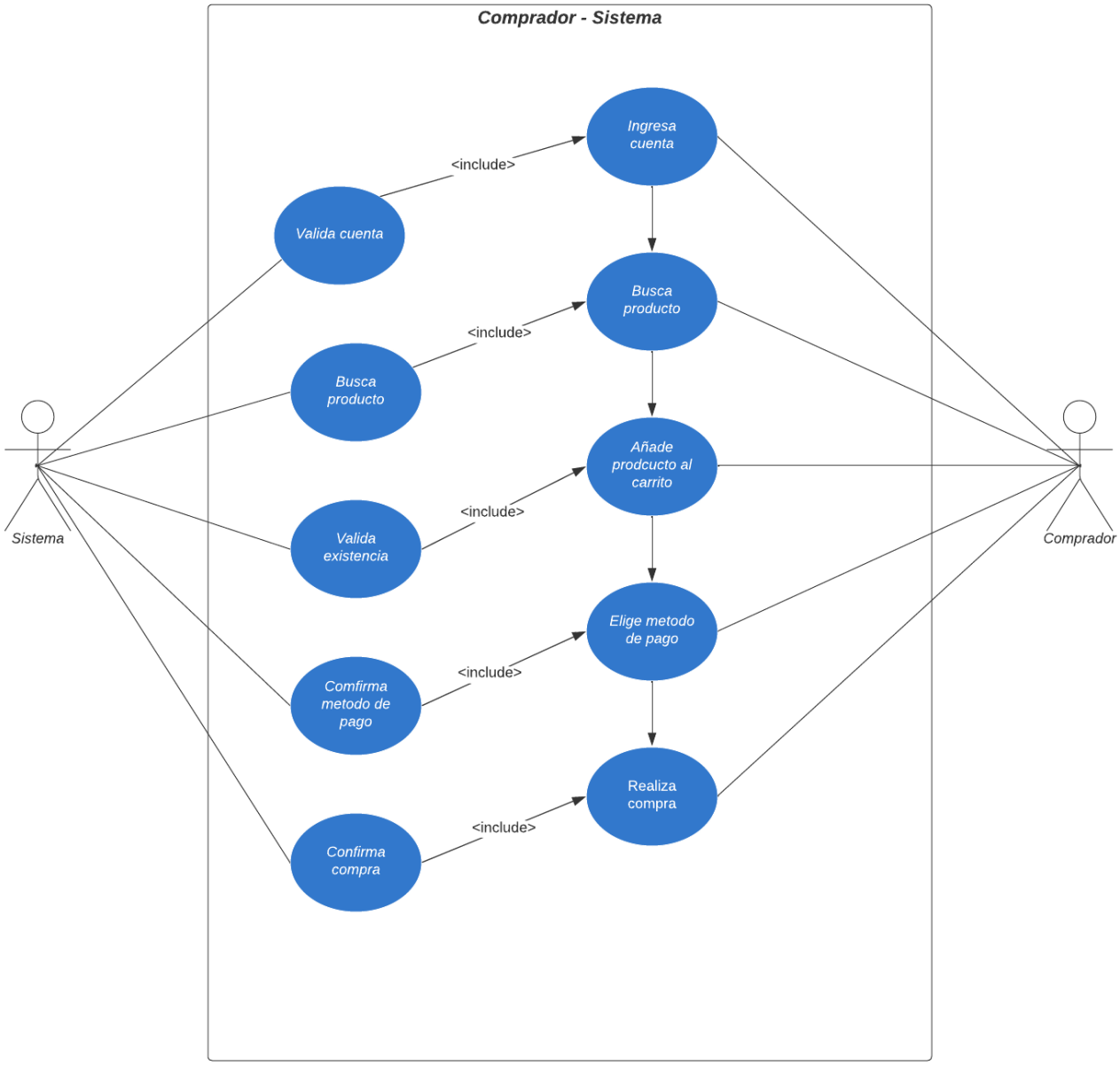
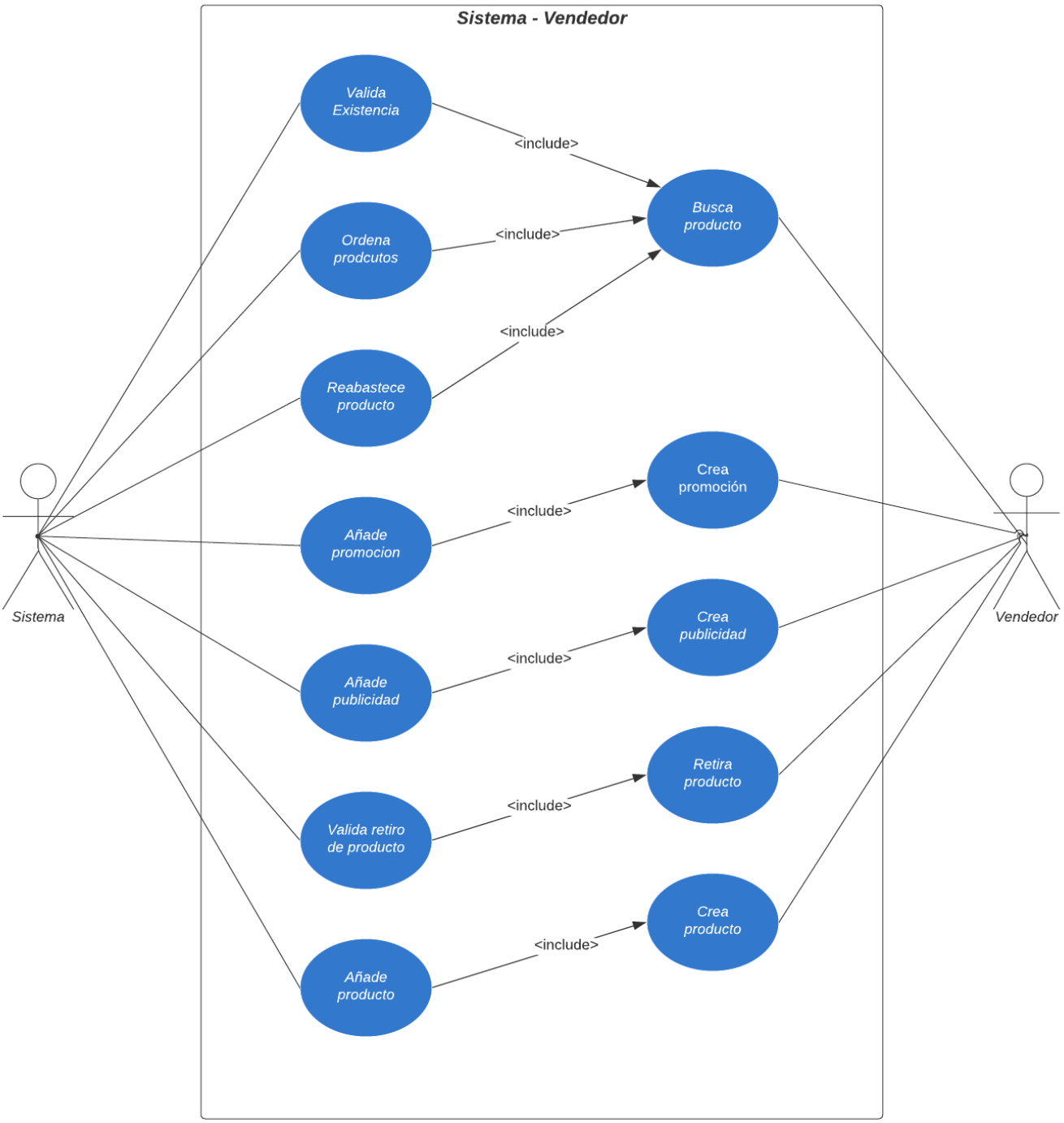


DIAGRAMA CASO DE USO (Vendedor - Sistema)


Sistema - Vendedor
Valida Existencia
Busca producto
Ordena prodcutos
Reabastece producto
Crea promoción
Añade promocion
Crea publicidad
Añade publicidad
Retira producto
Valida retiro de producto
Crea producto
Añade producto
<include>
Sistema
Vendedor