Proyecto 1: Fase de diseño

# Contexto del problema

Gracias a lo realizado en la parte previa de análisis, se pudo generar unos requerimientos generales que tiene que satisfacer la aplicación con el objetivo de cumplir con lo dicho en el proyecto. A su vez, se generó un modelo de dominio que contiene los elementos pertenecientes al mundo de problem2.1a representados en clases con sus relaciones entre sí.

Los 6 usuarios a tener en cuenta en el desarrollo del análisis fueron administrador, empleado, comprador, operador, propietario y cajero. Respecto a los macro componentes modelados en este mundo el problema se definieron 3, uno de inventario , uno de ventas, y un último que maneja los usuarios.

# Nivel 1 de diseño

* 1. **Componentes candidatos y estereotipos**

Deben existir componentes que agrupen las siguientes funcionalidades:

1. Un componente que agrupe toda la información respecto a las piezas, incluyendo la diferenciación entre las mismas, dicho componente es un information holder.
2. Un componente que maneje todo el tema de usuarios y cómo van a interactuar con la aplicación, dicho componente es un controller e interfaz.
3. Un componente que maneje todo el proceso de la compra, de los propietarios y los compradores, Dicho componente es a su vez controller e information holder.
4. Otro componente que maneje el proceso previo al de la compra que es el de la venta, incluyendo la oferta hecha y la diferenciación que hay entre hoy los tipos de venta, dicho componente es un estructurador ya que remite este proceso posteriormente a la compra.

Diagrama

Descripción generada automáticamente

* 1. **Responsabilidades**

Se asignaron las responsabilidades a los diferentes componentes identificados, incluyendo responsabilidades generales necesarias para el funcionamiento del sistema.

|  |  |  |
| --- | --- | --- |
| **N°** | **Responsabilidad** | **Componente** |
| 1 | Iniciar el programa | Inventario |
| 2 | Registrar Pieza |
| 3 | Cambio de E­xhibida/Bodega |
| 4 | Confirmar devolución |
| 5 | Consultar comprador | Usuarios |
| 6 | Verificar comprador |
| 7 | Modificar comprador |
| 8 | Crear perfil comprador |
| 9 | Registrar pago |
| 10 | Consignar pieza | Compras |
| 11 | Historial piezas |
| 12 | Ofertar compra directa | Ventas |
| 13 | Confirmar venta |
| 14 | Ofertar compra en subasta |
| 15 | Realizar subasta |  |
| 16 | Registrar subasta |  |

* 1. **Colaboraciones**

-El inventario debe colaborar con comprar y ventas para las transferencias de piezas.

-Subasta debe interactuar con venta y compra al momento de realizar una subasta

-Compras y venta interactúan al momento de confirmar una venta, inicializando compra

-Al encontrar el historial de piezas, el elemento propietario de compras interactúa con inventario.

# Nivel 2 de diseño

* 1. **Componentes candidatos y estereotipos**

Inventario:

Se compone de la clase inventario, ya que fue la decisión tomada para el manejo de las piezas presentes en la galería por la facilidad para diferenciar el estado de las diferentes piezas. A su vez se tomó la decisión de manejar el hecho de que hay diferentes piezas mediante una clase piezas con los tributos generales y clases que hereden de la misma para diferenciar entre los tipos de piezas.

Usuarios:

Dentro de los usuarios, se tomó la decisión de diferenciar entre los usuarios presentes en el enunciado del problema, a través de la herencia, permitiendo a todos los usuarios gozar de características generales, mas diferenciarse en las específicas. El usuario interactúa con el programa a través de estar clases, haciendo de estos componentes controller e interfaz para el programa.

Compra:

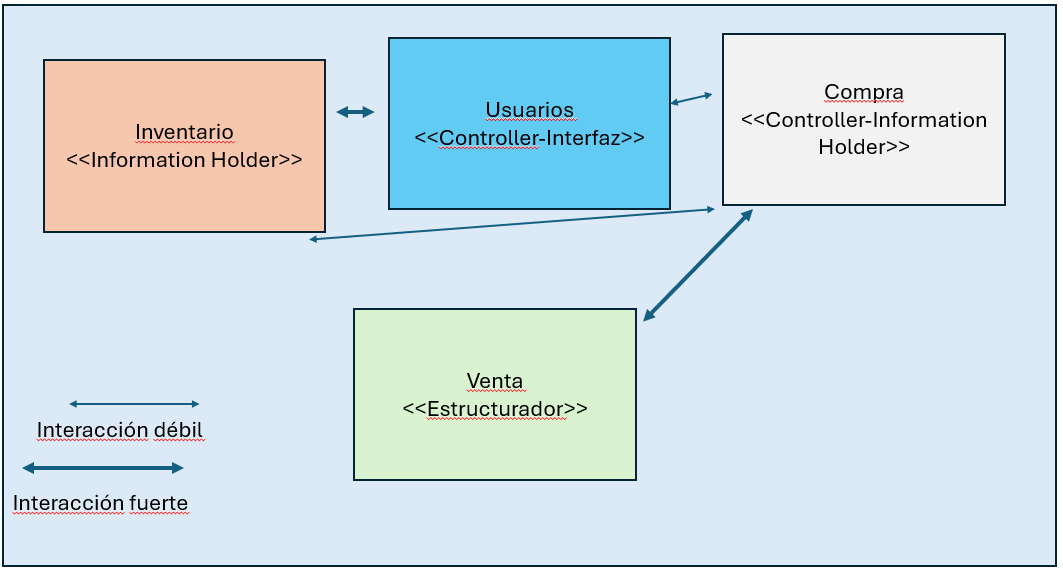
Compra es un elemento con clases propietario, comprador y compra. Se tomó esta decisión porque en este proceso de compra, son los elementos que controlan esta transacción, y a su vez la mejor forma de modelar la información presente en los mismos elementos, haciendo de estos elementos controller e information holder.

Venta:

Finalmente, se tomó la decisión de componer el elemento venta con las clases oferta, lista de subasta, subasta individual , venta y venta fija viene del hecho que se busca un alto acoplamiento entre estas clases, más no sobrecargar una sola con varias responsabilidades.

* 1. Colaboraciones:

Respecto a las colaboraciones, se toma la decisión ed tratar el core del programa, las ventas y subastas, a través de los componentes de venta y compra. Por lo tanto, las clases en las mismas van a tener una alta interacción entre ellas. A su vez, usuarios va a interactuar con inventario.



# Nivel 3 de diseño

Al llegar a este punto, se definieron las responsabilidades como métodos, las colaboraciones como relaciones entre clases y se mantuvieron los estereotipos encontrados en el anterior nivel. De esta manera, el proceso ya puede ser aterrizado a elementos como los que se van a mostrar a continuación. Adicional se encuentran ciertas descripciones de métodos importantes para nutrir los resultados de las decisiones tomadas.

**1 Ventas**

* 1. **Venta**

Clase con un metodo para crear una instancia de la clase Compra, Crea una

copia de la instancia de clase Comprador en una instancia clase Propietario en

caso de que el usuario no tenga una instancia de dicha clase; y en caso de que ya

sea Propietario se agrega la instancia de la Compra a una lista con el historial de

compras. Finalmente saca la pieza del Inventario y asigna nuevo propietario por

medio del metodo setPropietario en la clase Pieza de la pieza correspondiente.

La clase Venta tiene asociadas una o varias piezas

**Atributos**

• pagoHecho Atributo tipo booleano que es modificado por la clase Cajero,

se verifica que sea verdadero antes de llamar el método realizarCompra

Metodos

• crearCompra(): Este método crea una nueva instancia de la clase Compra

y registra los detalles de la transacción.

• reclasificarComprador(): Verifica si el comprador es un Propietario

o no. Si no lo es, crea una instancia de Propietario y actualiza las

referencias adecuadas.

• agregarHistorialCompra(Compra compra): Agrega la compra al

historial de compras del Propietario si ya existe como tal.

• sacarPiezaDelInventario(Pieza pieza): Llama al m etodo SacarDelInventario

del Inventario para remover una pieza y la asocia al nuevo propietario

utilizando setPropietario() en la clase Pieza.

• realizarVenta: Tras verificar que el atributo pagoHecho es True, llama

a todos los metodos anteriores para realizar una venta.

* 1. **ListaDeSubastas**

Esta clase es instanciada por el operador, y est ́a compuesta por las diferentes

subastas de cada pieza en especifico, presentes en una lista.

**1.3 SubastaIndividual**

Esta clase se encarga de la subasta de cada pieza en especifico, en la cual tiene

como atributo la pieza en cuesti ́on, un HashMap con llaves como el monto de

las ofertas y valores el comprador que hizo la oferta. A su vez, una lista con las

ofertas asociadas a esa pieza en subasta.

**1.4 Oferta**

Esta clase se encarga de registrar y llevar el historial de cada oferta que se haga

a una SubastaIndividual. Este historial se plantea mediante un hashMap donde

la llave sea el valor de la oferta y el valor la instancia de la clase Comprador.

• nuevaOferta(): Este m ́etodo crea una nueva tupla que corresponde al

valor de la oferta y el propietario, esta tupla se agrega al hashmap donde

el valor de la oferta es la llave del hashmap y el valor asociado a dicha

llave es la instancia de Comprador

**1.5 VentaFija**

Venta fija contiene como atributo la oferta hecha para venta directa por un

Comprador

**2 Inventario**

**2.1 Inventario**

Clase con varios atributos tipo lista donde se guardan todas las piezas depen-

diendo del estado de las piezas (InventarioPiezasVendidas, InventarioPiezasEn-

Venta, etc.) posee dos metodos, AgregarAInventario y SacarDelInventario para

respectivamente agregar o remover una pieza de la lista Inventario correspon-

diente. Un atributo PiezasBloqueadas donde se almacenan en una lista todas

las piezas que esten bloqueadas, es decir, que tengan el atributo ’disponible’ en

False. Un atributo PiezasDispoibles, donde se almacenan todas las piezas que

esten disponibles para subastarse o venderse en VentaFija al tener el atributo

’disponible’ en True.

• PiezasDisponibles

• PiezasBloquedas

**2.2 Pieza**

Esta es una clase abstracta que se instancia seg ́un el tipo de obra. Por este

motivo, una pieza debe tener atributos generales validos para cada tipo. Estos

son: Un t ́ıtulo, un a ̃no de creaci ́on, un lugar de creacion, una lista de autores,

un booleano que diga si la pieza est ́a exhibida. Adem ́as, debe tener atributo

booleano ’disponible’ en caso de que la pieza este bloqueada o disponible debido

a una venta fija. Por ́ultimo, la clase pieza tiene un atributo propietario de la

clase Propietario, que indica el usuario de quien posee la pieza.

Metodos

• getter de cada atributo

• setPropietario(Propietario)A ̃nade un valor de clase Propietario al atrib-

uto propietario de la pieza en caso de ser vendida.

• setExhibida()Cambia el estado del atributo ”exhibida” a True.

**2.3 Pintura**

Tipo de pieza que hereda la clase Pieza. Cuneta con sus atributos particu-

lares que son: ancho, alto, tecnica y estilo, además de los atributos heredados.

métodos

• Metodo constructor

• getter de cada atributo particular de esta pieza

**2.4 Impresion**

Tipo de pieza que hereda la clase Pieza. Cuneta con sus atributos particulares

que son: ancho, alto y tecnica, además de los atributos heredados. métodos

• Metodo constructor

• getter de cada atributo particular de esta pieza

2.5 Video

Tipo de pieza que hereda la clase Pieza. Cuneta con sus atributos particulares

que son: idioma y duración, además de los atributos heredados. métodos

• Metodo constructor

• getter de cada atributo particular de esta pieza

**2.6 Fotografia**

Tipo de pieza que hereda la clase Pieza. Cuneta con sus atributos particulares

que son: ancho, alto y camara, además de los atributos heredados. métodos

• Metodo constructor

• getter de cada atributo particular de esta pieza

**2.7 Escultura**

Tipo de pieza que hereda la clase Pieza. Cuneta con sus atributos partic-

ulares que son: ancho, alto, profundidad, material, peso, necesitaElectrici-

dad(booleano) y detallesInstalaci ́on, adem ́as de los atributos heredados. métodos

• Metodo constructor

• getter de cada atributo particular de esta pieza

**3 Compradores**

**3.1 Compra**

Clase que posee la informacion acerca de una compra en especifico con atributos

de montoPagado, idDeCompra y LocalDate. Clase de tipo Information Holder,

solo retiene informacion.

**3.2 Propietario**

Es una clase que hereda de Comprador. Esta posee un atributo de tipo lista

con todas sus piezas compradas. Clase de tipo Information Holder, retiene

informacion pertinente acerca del Propietario

**3.3 Comprador**

Clase para un usuario que efectua el rol de comprador de una pieza o coleccion,

retiene informacion pertinente acerca del Propietaro como atributos que son:

nombre, identificador, tel ́efono, login, password y un booleano que indica si esta

verificado.

metodos

• Metodo constructor

• getter de cada atributo

**4 Usuarios**

**Usuarios es una superclase y las siguientes clases son subclases de la misma, por**

**lo tanto heredan las propiedades basicas de login y contraseña.**

**4.1 Admin**

Metodos

• RegistrarPieza()

• ConfirmarVenta(Directa/Subasta)

• ConfirmarDevolucion()

• ConsultarComprador()

• VerficarComprador()

• ModificarComprador()

**4.2 Empleados**

Metodos

• CambioEstadoABodega() Cambia el estado de una pieza de exhibida

a bodega.

• CambioEstadoAExhibida() Cambia el estado de una pieza de bodega

a exhibida.

**4.3 Comprador**

Metodos

• OfertarCompraDirecta() Reliza una oferta para una pieza por por

medio de la clase VentaFija, es decir, llama el metodo venta de la subclase

VentaFija.

• OfertarCompraSubasta() Reliza una oferta para una pieza por por

medio de la clase Subasta, llama el metodo venta de la subclase Subasta.

• ConsignarPieza() Crea una instancia de la clase Propietario tras fi-

nalizar una compra, saca la pieza del inventario piezasDisponibles de la

clase Inventario y lo agrega a la lista piezasActuales de la clase Propietario

**4.4 Operador**

Metodos

• RealizarSubasta()

• RegistrarSubasta()

**4.5 Propietario**

Atributos

• piezasActuales Lista de las instancias de la clase Pieza con todas las

piezas que posee actualmente un Propietario.

• piezasVendidas Lista de las instancias de la clase Pieza con todas las

piezas que ha vendido un Propietario.

**4.6 Cajero**

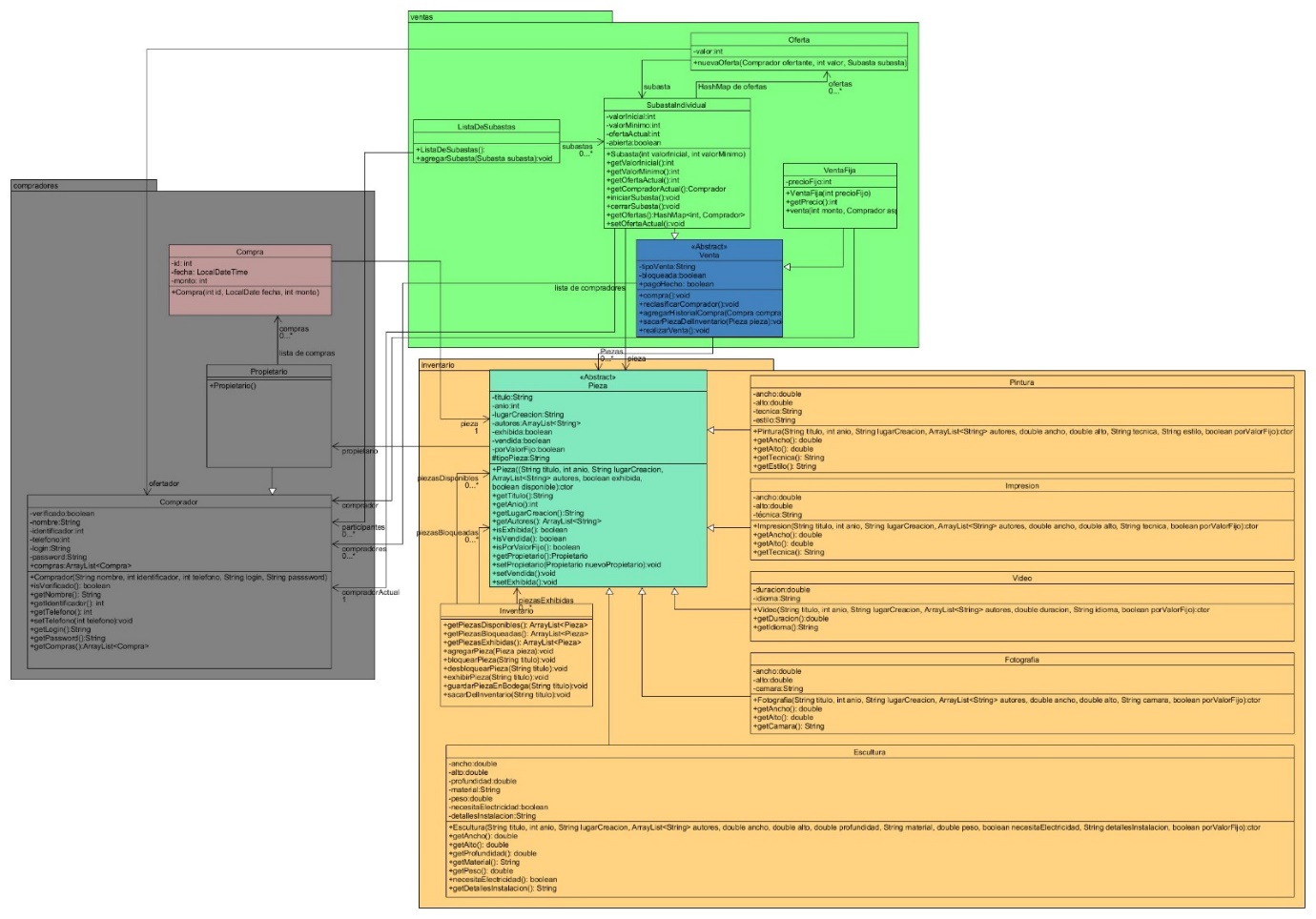
Metodos

• RegistrarPago() Registra un pago una vez realizada una compra al cam-

biar el atributo pagoHecho en la clase Venta.

# Fase final

Finalmente, se llega a la representación por diagrama de UML del diseño, de manera completa incluyendo todos los atributos, métodos, clases y relaciones entre clases, dentro de los componentes. Dicho diagrama se muestra a continuación. Elementos adicionales, como la persistencia y el manejo de ciertas excepciones se encuentran en los archivos adjuntos.



Diagrama, Esquemático

Descripción generada automáticamente

Adicional se encuentran diagramas de secuencia para ilustrar funcionalidades del programa.

Diagrama

Descripción generada automáticamente