## Proyecto 1- Entrega 2

Integrantes:

• Juan Pablo Castro Idarraga – 202012399

• Joseph Eli Pulido Gómez – 202211365

• Santiago Martínez Castillo - 202124032

Introducción:

El objetivo de esta entrega es diseñar e implementar un programa que logre sistematizar las operaciones requeridas por la galería y la casa de apuestas. Las cuales tienen tres grandes funcionalidades. En primer lugar, tendrá un inventario de las piezas disponibles para ser expuestas o ser subastadas. En segundo lugar, tendrá un sistema que gestionará el proceso de compra o subasta de una pieza o de una colección de piezas. En tercer lugar, manejará la información de los propietarios y compradores de las piezas. Estas son las tres secciones grandes que se identificaron. Entre estas tres secciones suman 14 funcionalidades particulares las cuales se tienen que asignar a los distintos componentes que vayamos encontrando en el camino del diseño. A esto hay que sumarle las restricciones y requerimientos no funcionales.

Modelo de dominio

Para la implementación se tuvo en cuenta el modelo de dominio entregado en la parte 1 del proyecto en el cual se modeló el entendimiento general que se tenía sobre el mundo del problema

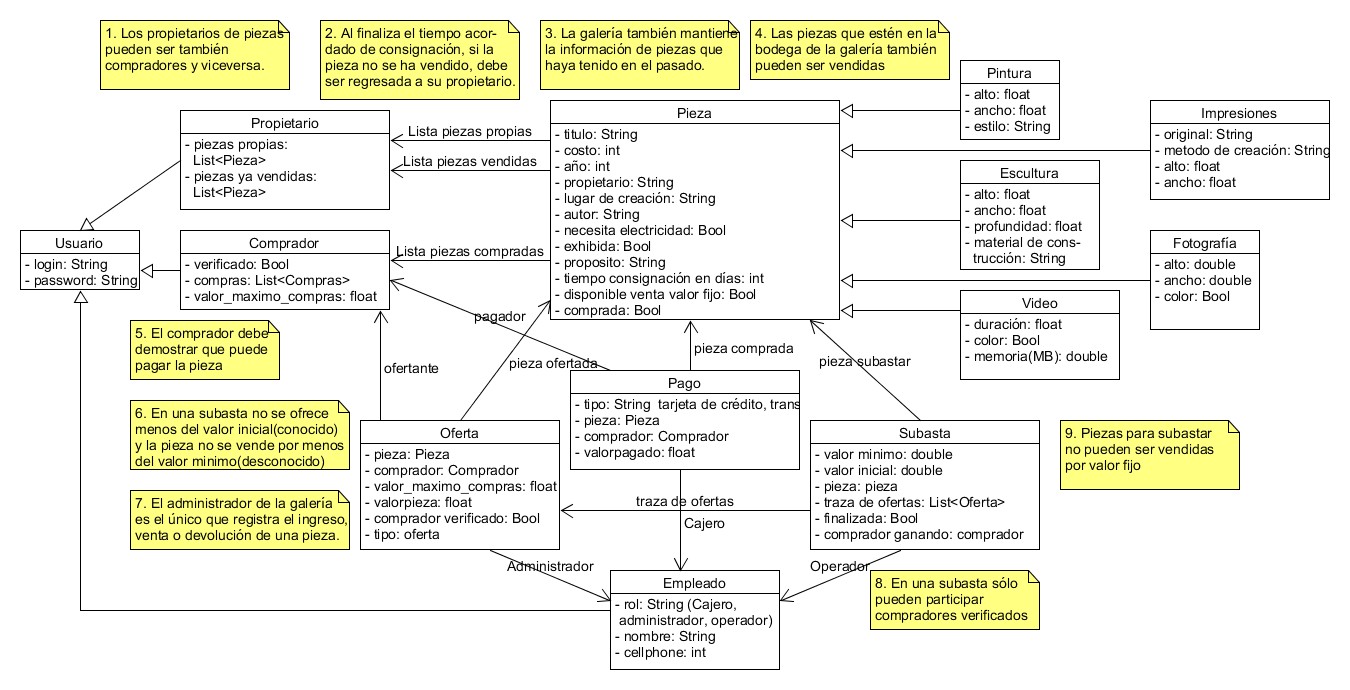


Figura 1. Modelo de dominio de galería y casa de subastas.

Muchas de los conceptos mostrados en el modelo del dominio se usarán como clases en las categorías que explicaremos a continuación.

Proceso de partición en los componentes del programa

Iteración 1 y 2

Tomando como ejemplo diseños realizados en clase dividimos el programa general (cuadro 1) en tres programas. El enunciado nos da directamente dos piezas: **La casa de apuestas** y la **Galería,** añadimos una división mas llamada **Sistema** enfocada en todo el ambiente que rodea estas dos otras divisiones como las personas, el programa y su persistencia, interfaz, etc. Esto se demarca en la figura 1 cuadro 2.

Después subdividimos el cuadro 2 en cinco secciones grandes.

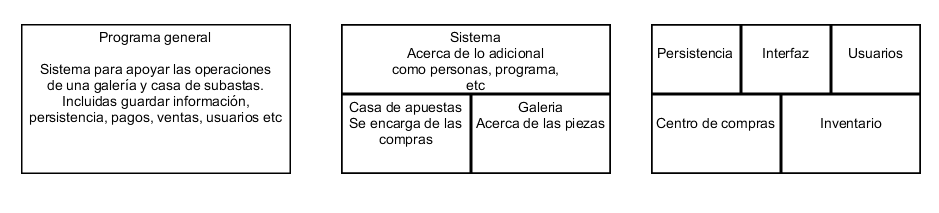
**Inventario**: Encargado de conocer la información de todas las piezas.

**Centro de compras**: Encargado de las compras por valor fijo y las subastas, aquí tendrá lugar los requerimientos más importantes (compras).

**Usuarios:** Encargado de saber la información de los usuarios y administrar que usuario es de cada tipo.

**Interfaz:** Encargado de recibir y dar la información al usuario.

**Persistencia:** Encargado de la persistencia del programa, es decir, piezas, usuarios y movimientos.



1

2

3

Figura 2: Visualización de las primeras dos iteraciones del proceso de partición y obtención de los componentes del programa.

En figura 2 aún no se tienen clases ni se hace uso de ningún concepto descrito en el mundo del problema.

Una vez tenemos la primera subdivisión importante podemos empezar a establecer que partes se usaran en los requerimientos funcionales, que secciones se conectan entre si cual es si estereotipo. Es decir, además de reconocer las funcionalidades y delimitar el alcance de nuestro código en este paso se asignarán responsabilidades a los cinco diferentes componentes. Sin embargo, estas responsabilidades no se describirán en detalle puesto que aún no conocemos los componentes de estos cinco componentes.

Los requerimientos según el documento de análisis son:

* Registrar una nueva “consignación”
* Consultar el estado de una pieza
* Registrar un nuevo propietario
* Registrar la salida de una pieza
* Registrar un nuevo comprador
* Como propietario, ofrecer una nueva consignación
* Como comprador, comprar una pieza por valor fijo
* Como propietario, ver el historial de piezas que he vendido
* Como comprador, ver mi historial de compras
* Como comprador, proponer una puja en una subasta
* Registrar venta por valor fijo
* Crear subasta
* Gestionar subasta
* Cerrar subasta

Usuario e interfaz:

Antes de entrar a ningún requerimiento en específico, se ve que las operaciones posibles dependen del tipo de usuario. Para administrar toda la información relacionada con los usuarios se tiene la sección **usuario**. Esta guardara todos los logins, paswords y la descripción de los tipos de usuario (empleado, propietario, comprador) y sus posibilidades. Esta sección estará directamente conectada a la interfaz para recibir el login y pasword del usuario y dirigirlo al menú correcto con las funcionalidades que este pueda ejecutar. Esta sección puede ser

Primera decisión: Se tendrán menus diferentes para los distintos tipos de usuarios enfocados en sus requerimientos específicos. A un menú dado solo se podrá acceder con un usuario del tipo correcto y su contraseña.

Lo que se espera evitar es hacerle seleccionar al usuario que tipo de usuario es. Debido a que los usuarios compradores pueden ser propietarios y viceversa. Ambos menus se mostrarán por defecto cuando se introduzca un login de tipo cliente. Algunas subdivisiones de este grupo **usuario** pueden ser **empleado** y **cliente**. El grupo **usuario** deberá poder pasarle todos sus datos a la **persistencia** de manera ya ordenada para que esta pueda guardarlos. En muchas ocasiones la información de un usuario tiene que ser entregada al **centro de compras** para validar que esta compra si se puede hacer.

Inventario:

El inventario guarda toda la información relacionada con las piezas, sus características y movimientos. Por ende, es natural que el inventario cuente con la clase **pieza** y sus tipos. El **inventario** deberá poder pasarle todos sus datos a la **persistencia** para que esta pueda guardarlos. En muchas ocasiones información acerca de las piezas debe ser entregada al grupo **centro de compras**, para validar si una compra se puede hacer.

Segunda decisión: El grupo persistencia recibe la información lista para cargar por parte de todas las otras clases, además cuando la devuelve es trabajo de la clase en si interpretar los datos.

Centro de compras:

El centro de compras maneja todas las operaciones relacionadas con un usuario comprando una pieza sea por valor fijo o por subasta. Por ende, tiene sentido que esta categoría cuente con las **clases subasta**, **oferta** y **pago.** Además, tiene que poder comunicarse con categoría **usuarios** para conocer el usuario que hace la compra, y debe saber conectarse con la categoría **inventario** para saber la pieza que se compra.

Persistencia:

Se encarga de la persistencia del programa. Recibe la información ya preparada y la guarda en una locación fuera del programa.

Iteración 3

En esta iteración se dividen los componentes encontrados en la fase anterior en categorías aún mas pequeñas. Estas categorías más pequeñas ya son los componentes candidatos que utilizara nuestro programa. A estas clases se les asignaran responsabilidades especificas posibles de implementar computacionalmente de manera directa. Se representará los componentes finales y su responsabilidad en la siguiente tabla 1. Además, se añadirán estereotipos para entender el papel de los componentes en el programa.

Usuario e interfaz:

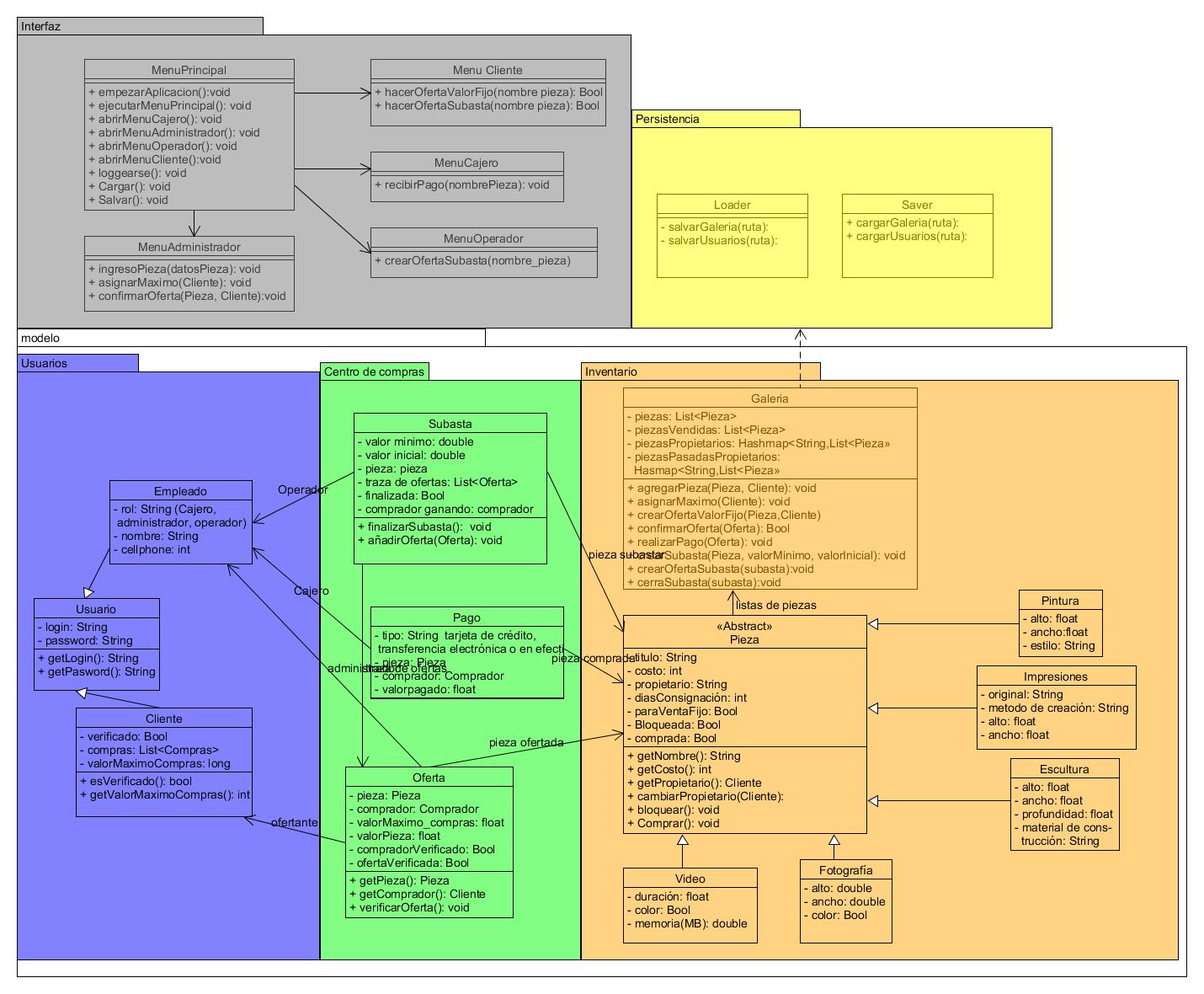
Mencionado en la sección anterior, necesitamos de varios menús dependiendo del usuario del programa, por ende, necesitamos el **Menú Principal** que recibe a todos los usuarios y maneja el proceso de ‘loggearse’. Después el **menú cliente** para los clientes, y el **menú empleado** para los empleados. El menú empleado debe tener diferentes funcionalidades dependiendo si lo maneja un cajero, un empleado cualquiera o el gerente. Estas diferentes funcionalidades del menú empleado se implementarán en el proyecto 2 en donde se enfatiza en la interfaz con el usuario.

Persistencia:

Para realizar la tarea de manejar la información debe tener por lo menos dos subdivisiones: loader and saber los cuales se encargan de cargar y salvar los datos respectivamente. Claro estas dos componentes tendrán varias instancias o métodos dependiendo del tipo de dato a tratar.

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Sub-componenete | Estereotipo | Responsabilidad |
| Inventario | Galeria | <<Controller>>  <<Structurer>> | Se ejecutaran las funciones principales |
| Pieza | <<Information  holder>> | Almacenar información de  Las piezas |
| Persistencia | loader | <<Service provider>> | Cargara todos los archivos |
| Saber | <<Service provider>> | Salvar todos los archivos |
| Interfaz | MenuPrincipal | <<Interfacer>> | Deja loggearse y abre los Menus |
| Menus | <<Interfacer>> | Muestra las funcionalidades para ejecutar |
| Centro de compras | Subasta | <<Information  Holder>> | Guarda información de las subastas |
| Pago | <<Information hoder>> | Guarda información de los pagos |
| Usuarios | Empleado | <<Information holder>> | Guarda información de los empleados |
| Cliente | <<Information holder>> | Guarda información de los clientes |

Diagrama UML final



Justificación de Diseño Implementado

Ya teniendo en cuenta el diagrama UML final, en conjunto con sus clases, atributos y métodos a continuación se detallan las principales decisiones críticas que se tuvieron en cuenta.

* Se tomó la decisión de que los usuarios clientes no conocieran las piezas de las que son dueños, se prefiere que toda la información este concentrada en nuestra galería. Esto para evitar que se tenga información distribuida sin organización por el programa.
* Se tomó la decisión de crearse la clase galería, aunque este no es un objeto del mundo problema sino más bien el mundo en sí, esta clase sirve para tener un lugar desde el cual se dirigen las funciones grandes.
* Se toma la decisión de que cada diferente pieza es una clase que hereda de la clase abstracta pieza.Ya que el programa debe tener persistencia, se agregaron las clases loader y saber.
* Ya que el programa y sus funcionalidades cambia dependiendo quien lo maneje, hay cuatro diferentes menús dependiendo del usuario.
* Se decidió que comprador y propietario conceptos diferentes en el modelo del dominio se unan en una solo clase.
* Para manejar todo el sector de compras solo se necesitan tres clases: subasta, oferta y pago. La interacción entre estas tres clases se especificará adelante en los diagramas de secuencia.

Factores adicionales

La lista de traza de ofertas será una linkedList, esto para que se pueda ingresar las ofertas nuevas en las primeras posiciones