**Documento de diseño del Proyecto 1 entrega 2**

**Por Grupo 7**

**Introducción y Justificación del método de control:**

En términos generales, los requerimientos funcionales son traducibles en manejar (crear, acceder, editar y mantener) tipos abstractos de datos. Se propuso una división primaria en tres partes view, controller, model. En primera instancia, un *modelo* compuesto solo de *data Holders* con únicamente getters y setters  representando los tipos abstractos de datos que se deben manejar (ej Pieza y Venta) en Pieza se permitió una división de herencia dependiendo del tipo de pieza tal como se pide en el proyecto , es decir una clase si la pieza es una pintura, una escultura, fotografía, etc que tienen como super clase Pieza, esto ayuda a manejar la complejidad del contenido que puede tener una pieza. En segunda instancia, un *controller* compuesto en principio de 6 clases, 3 de ellas representando cada una a un usuario (administrado, cliente y empleado) con estereotipo de *controllers*, en el sentido de que deben definir cómo manejar los datos según lo que quiera hacer cada usuario. Además, en este nivel se incluye varias clases con denominadas; BasesDatosGaleria, BasesDatosInventario y BaseDatosEmpresa,compuesta de mapas de objetos,informationHolder y metodos delegadores, que debe cargar, descargar y mantener los datos, como los métodos writer (que convierten de objeto a string las instancias de los objetos), los métodos Reader hacen lo opuesto. La idea era que cada miembro del equipo se encarga de un usuario y desarrolla las clases que puedan apoyar a su controller o dejar toda la implementación en esta dicha clase únicamente, a criterio de cada uno. Por último, el programa cuenta con una consola general y 4 auxiliares, una por cada usuario, que regulan la interacción con los usuarios mediante login y menús de acciones.

**Persistencia:**

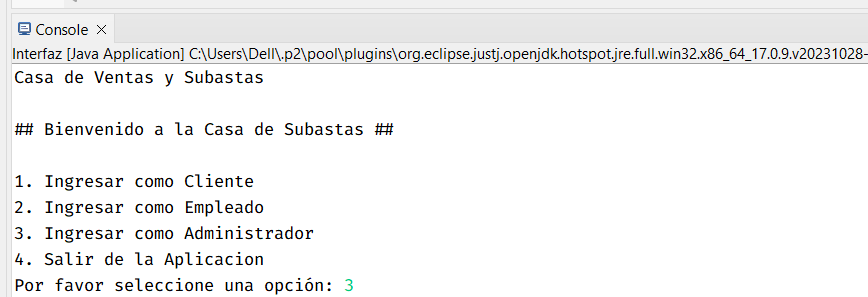
La persistencia se maneja conjuntamente entre las tres clases BasesDeDatos y a través de archivos .txt. Hay un archivo txt por cada tipo abstracto de datos definido en el modelo, (ej pieza.txt, venta.txt). Cada vez que un usuario cierra sesión, se reescriben los archivos que enlistan las instancias de los objetos a través de Strings que permitan su posterior descarga y conversión en objetos. Por ejemplo, para cargar un Cliente, se transforman todos sus atributos individuales en string y se juntan en un orden determinado separados por punto y coma.

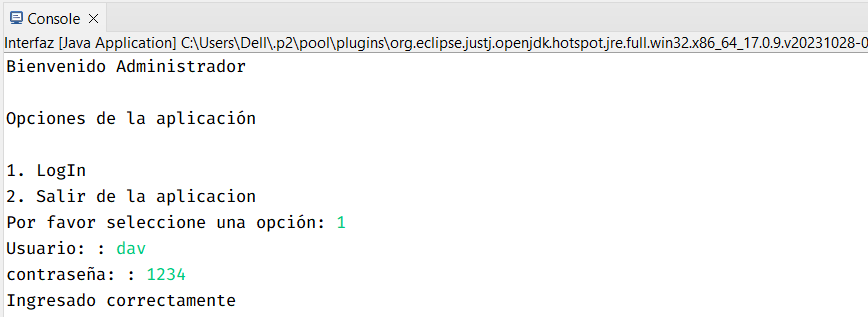
Cada vez que se inicializa la aplicación se descargan los datos, se crean los objetos y se añaden estos a los mapas. Se descargan todas las instancias de algún objeto y se almacenan en el map correspondiente antes de pasar al siguiente. Para esto se utiliza los métodos reader hallados en cada clase BaseDeDatos, que a partir de las String devuelve instancias de los objetos. Por ejemplo, al descargar el cliente, primero se crea este a partir de sus atributos primitivos, con el nombre de usuario en el mapaClientes y se agrega. Para que funcione correctamente es necesario que los constructores funcionen solo con datos primitivos y que los demás se puedan agregar con sets.

La interacción con el usuario se maneja a través de la consola. La implementación de estas interfaces se divide en 4 partes principales. Para ingresar se le pedirán al usuario los datos que aparecen y se buscará si coinciden, si no, no dejara ingresar. Cada una de estas tiene los requerimientos de cada usuario; Administrador , Cliente, Empleado. Al administrador se le pedirá que ingrese una nueva pieza o que registre una compra, confirme una venta, verificar el comprador y crear una nueva subasta;

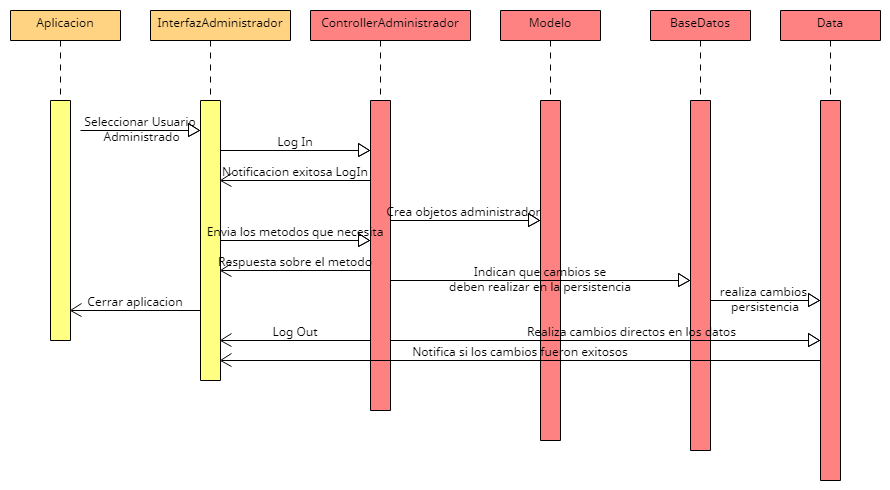
El cliente tiene la opción de comprar ingresando el titulo de la pieza , el administrador también tiene de ser un cajero y recibir las compras que se hagan , muchos de estos métodos quedaron incompletos por obvias razones, O incluso no se implementaron.

Aquí se puede observar consola cuando se ejecuta desde eclipse y se escoge la opción del LogIn:

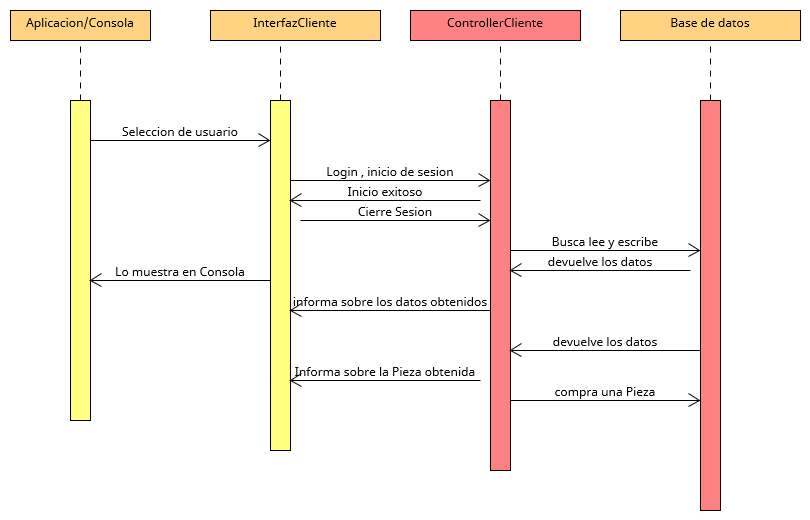




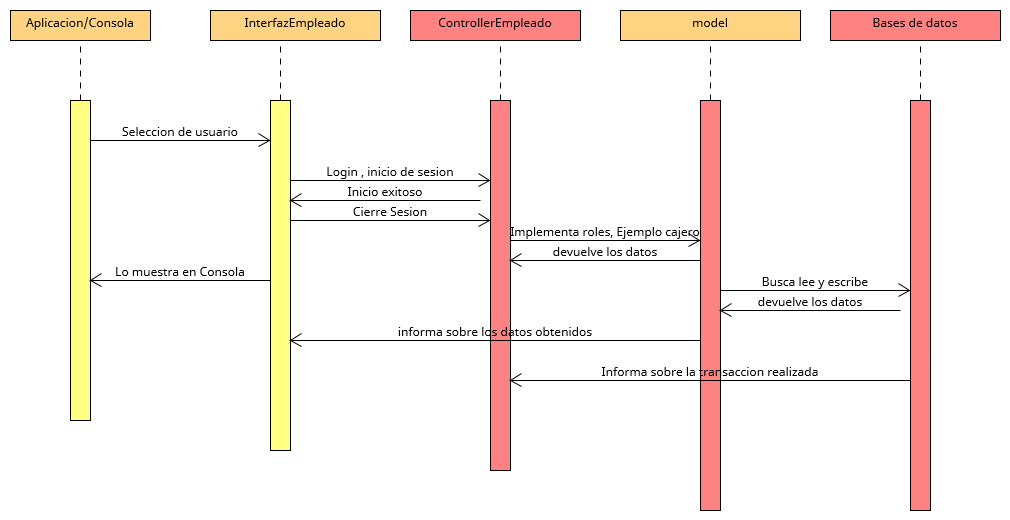
**Historia de Usuario**

**Administrador:** 

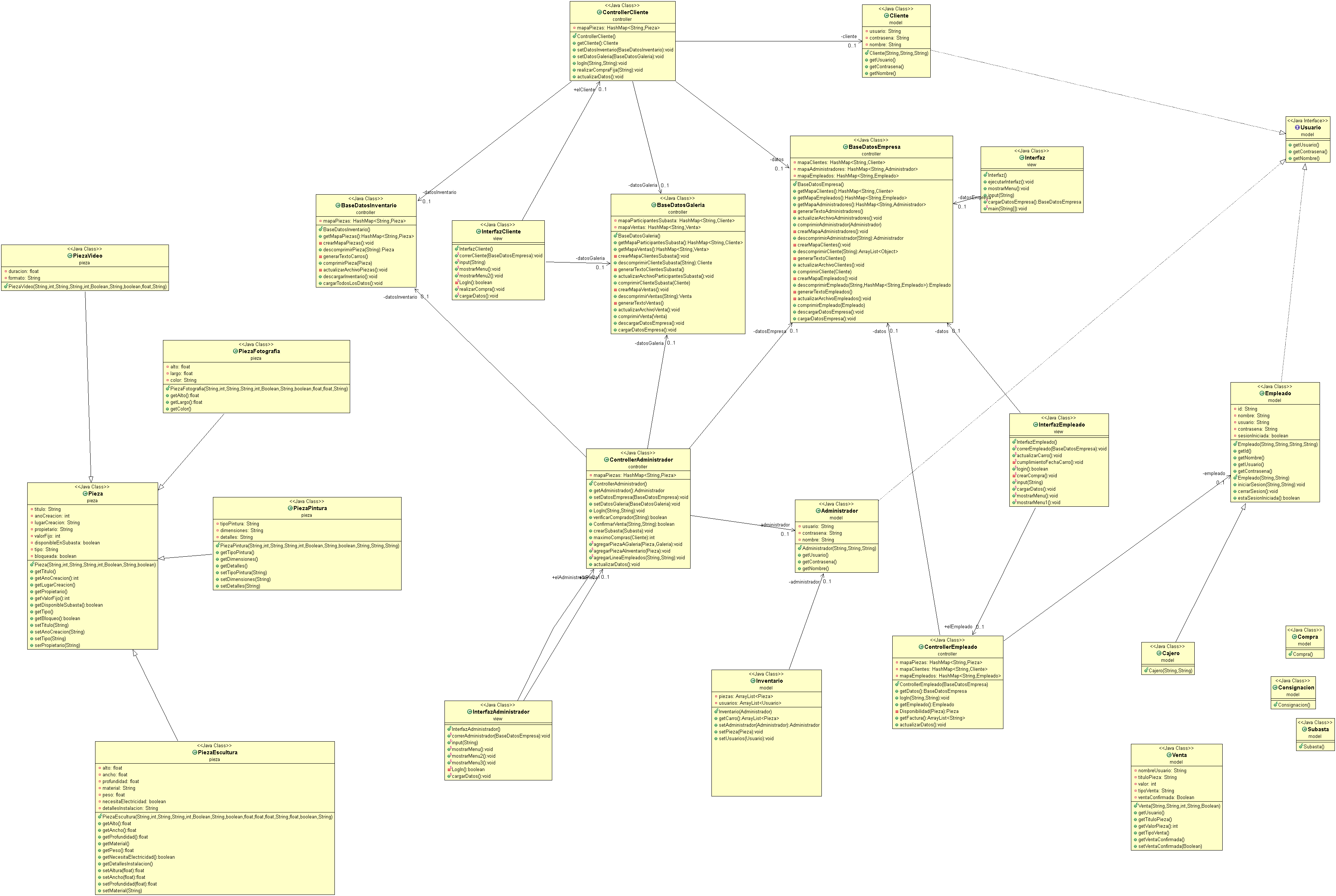
**Comprador, Cliente**

****

**Historia Empleado**

****

**Diagrama de Dominio (Incompleto)**

****

**Galería y casa de subastas**

**Historias de usuario**

1. **Administrador**
   1. **Gestión de ingreso, venta y devolución de una obra:**

|  |
| --- |
| **Como un:** Administrador |
| **Quiero:** gestionar los ingresos y salidas del inventario de piezas |
| **Para:** ponerlos venderlos o subastarlos |
| **Detalles Adicionales:** el inventario estará compuesto por piezas diferentes piezas de arte como esculturas, pinturas, videos, entre otras. |
| **Criterios de aceptación:** Todos los campos de registro deben ser obligatorios. |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | ingresar\_pieza\_nueva | |
| **Descripción de la función** | Esta función agregará al inventario disponible piezas nuevas | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| titulo | String | Es el nombre de la pieza por la cual se debe verificar la compra. |
| anio\_cracion | int | No: la pieza estará disponible, sí: la pieza estará vendida |
| artista | String | Nombre del creador o creadores de la obra |
| lugar\_creacion | String | Lugar en el que fue creada la obra |
| propietario | String | Quien es el dueño actual |
| valor\_fijo | int | Valor de la obra |
| disponible\_subasta | Boolean | True si esta destinada a subasta, de lo contrario False |
| tipo | String | Si es una escultura, pintura, video, etc. |
| caracteristicas | String | Las características según el tipo de obra |
| **Retorno** | void | Agrega al inventario una obra nueva |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | confirmar\_venta | |
| **Descripción de la función** | Cambia el estado de la obra a vendida, por ende, la saca del inventario disponible | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| titulo | String | Es el nombre de la pieza por la cual se debe verificar la compra. |
| comprador | String | El nombre de quien compró la obra. |
| **Retorno** | void | Cambia el estado del inventario |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | devolucion | |
| **Descripción de la función** | Si el cliente la regresa esta debe devolverse al inventario y cambiar al dueño anterior | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| titulo | String | Nombre de la pieza que devolvió el cliente |
| **Retorno** | void | La obra estará disponible y a nombre del dueño anterior. |

* 1. **Confirmación de la seriedad de una venta a precio fijo:**

|  |
| --- |
| **Como un:** Administrador |
| **Quiero:** verificar la seriedad de una compra |
| **Para:** asegurarme que no existan piezas separadas por las que el comprador no vaya a pagar. |
| **Detalles Adicionales:** la pieza separada por un comprador a precio fijo estará bloqueada hasta que yo confirme que efectivamente se realizó esa compra, de lo contrario debo poder regresar esa pieza al inventario disponible para subasta o venta. |
| **Criterios de aceptación:** La pieza que no sea vendida debe aparecer nuevamente en el inventario disponible y la que se vendió debe aparecer en los datos de ventas. |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | verificar\_compra | |
| **Descripción de la función** | Permitirá cambiar el estado de una pieza a vendida o disponible, a su vez se debe reflejar esto en la base de datos | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| pieza | String | Es el nombre de la pieza por la cual se debe verificar la compra. |
| vendida | String | Sí la respuesta es no: el estado de la pieza será disponible. Si la respuesta es sí: la pieza tendrá un estado como pago pendiente. |
| **Retorno** | N/A | Sólo actualiza el estado del inventario. |

* 1. **Verificación de un comprador:**

|  |
| --- |
| **Como un:** Administrador |
| **Quiero:** verificar y asignarle un montó de compras a un comprador |
| **Para:** asegurarme que sus propuestas de compras sean serias y efectivamente pueda pagar. |
| **Detalles Adicionales:** sólo podrán comprar las personas que tengan un estado verificado. |
| **Criterios de aceptación:** Debo poder ingresar el monto aprobado para el comprador y modificar su estado como verificado o no. |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | Verificar comprador | |
| **Descripción de la función** | Se podrá modificar el estado del comprador y el monto máximo de sus compras. | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| comprador | String | Nombre del comprador |
| autorizado | boolean | True en caso de sí y False en caso de no |
| monto\_autorizado | int | El monto que estima el administrador para autorizarle. 0 en caso de no. |
| **Retorno** | N/A | Sólo actualiza los datos de un comprador. |

* 1. **Ver la historia de un comprador: REQUERIMIENTO NUEVO**

|  |
| --- |
| **Como un:** Administrador |
| **Quiero:** Ver el historial de transacciones de un comprador en nuestra galería |
| **Para:** para conocer las transacciones de su tienda y su perfil de compras. |
| **Detalles Adicionales:** Deseo conocer sus datos de contacto e información relacionada con sus compras en la galería. |
| **Criterios de aceptación:** Debe tener datos de contacto, el historial de piezas que ha comprado, cuando las compró, de cuales es dueño y por cuanto las compró. |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | obtener\_historial\_comprador | |
| **Descripción de la función** | El administrador podrá ver la historia de un comprador: qué piezas ha comprado y cuándo, de cuáles piezas es dueño, y cuál es el valor de su colección. | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| comprador | String | Nombre del comprador de quien queremos ver la historia. |
| Historial\_ventas | HashMap | El mapa donde estará almacenada la información de las transacciones de la galería. |
| **Retorno** | String | Él String mostrará las piezas que ha comprado, cuando y de cuales piezas es actualmente dueño, además de el valor total de su colección. |

1. **Empleados**

**2.1 Operador – actualizar información subasta**

|  |  |  |
| --- | --- | --- |
| **Como un:** Operador | | |
| **Quiero:** registrar la información de cómo va la subasta | | |
| **Para:** para que el cajero pueda cobrar el dinero de la obra y el administrador retirar la obra del inventario disponible. | | |
| **Detalles Adicionales:** La obra pasará a un estado pendiente de pago | | |
| **Criterios de aceptación:** El monto que ofrece el comprador en la subasta debe ser superior al monto inicial y la pieza no se venderá a menos que llegue al valor mínimo | | |
| **Nombre de la función** | actualizar\_subasta | |
| **Descripción de la función** | Permitirá al operador actualizar datos de una puja como el nombre del mayor postor y el precio que ofreció. | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| pieza | String | Nombre de la pieza subastándose. |
| comprador | String | Nombre del comprador que ha ofrecido el monto mayor. |
| precio | HashMap | El mapa donde estará almacenada la información de las transacciones de la galería. |
| **Retorno** | String | Él String mostrará las piezas que ha comprado, cuando y de cuales piezas es actualmente dueño, además del valor total de su colección. |

**2.2 Cajero – Recibir pagos**

|  |
| --- |
| **Como un:** Cajero |
| **Quiero:** recibir el pago de un cliente |
| **Para:** para que el administrador entregue la obra al nuevo dueño |
| **Detalles Adicionales:** La obra será parte del historial vendido luego de que reciba el pago. |
| **Criterios de aceptación:** Debe ingresar un método de pago: Transferencia, tarjeta o efectivo. Además de ingresar el nombre de la pieza. |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | recibir\_pago | |
| **Descripción de la función** | En esta función el pago se hará efectivo y el cliente será el nuevo dueño de la obra. | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| pieza | String | Nombre de la pieza por la que se hará el pago. |
| Método de pago | String | Transferencia, Tarjeta o Efectivo. |
| **Retorno** | void | Se cambia el estado de la pieza al historial vendido y se actualiza el nombre del propietario de la pieza. |

1. **Clientes**

**3.1 Comprar por un precio fijo:**

|  |
| --- |
| **Como un:** Cliente |
| **Quiero:** separar una obra que tenga un precio fijo |
| **Para:** comprarla y que no la vayan a subastar antes que yo pague. |
| **Detalles Adicionales:** La obra pasará de disponible a separada. |
| **Criterios de aceptación:** Debo ingresar el nombre de la obra que quiero comprar, está bajo ningún motivo puede ser subastada ya que estará separada. |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | comprar\_obra | |
| **Descripción de la función** | En esta función separará una obra que el cliente quiere separa por un monto fijo | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| pieza | String | Nombre de la pieza por la que se hará el pago. |
| **Retorno** | void | El estado de la obra estará cambiado |

1. **En común todos los usuarios:**

**4.1 Ver la historia de los artistas: REQUERIMIENTO NUEVO**

|  |
| --- |
| **Como un:** Administrador/Empleados/Cajero |
| **Quiero:** conocer el historial de un artista específico |
| **Para:** para conocer cuales son sus obras y el precio de estas |
| **Detalles Adicionales:** |
| **Criterios de aceptación:** La información debe mostrar: piezas que ha hecho, cuándo, cuándo han sido vendidas y por cuánto. |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | obtener\_historia\_artista | |
| **Descripción de la función** | Muestra la información de un artista recibido por parámetros | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| artista | String | Nombre del artista del cual se desea conocer sus obras y el valor de las mismas. |
| obras | HashMap | El mapa que contiene la información básica de las obras. |
| **Retorno** | String | El string tendrá las piezas que ha hecho el artista, cuando las hizo, cuando han sido vendidas y por cuanto se vendieron. |

**4.2 Ver la historia de una pieza: REQUERIMIENTO NUEVO**

|  |
| --- |
| **Como un:** Administrador/Empleados/Cajero |
| **Quiero:** conocer el historial de una pieza específica |
| **Para:** para conocer el historial de transacción de dicha obra y su precio |
| **Detalles Adicionales:** Puede ser una obra que ha sido vendida o está disponible en el inventario. |
| **Criterios de aceptación:** Se deben conocer los siguientes datos: datos generales, quién ha sido el dueño, por cuánto ha sido vendida y cuando. |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función** | obtener\_historia\_pieza | |
| **Descripción de la función** | Muestra la información de una pieza específica | |
| **Parámetros** | | |
| **Nombre** | **Tipo** | **Descripción** |
| obra | String | Nombre de la obra sobre la cual se va a consultar |
| obras | HashMap | El mapa que contiene la información básica de las obras. |
| **Retorno** | String | El String tendrá la siguiente información: datos generales, quién ha sido el dueño, por cuánto ha sido vendida y cuando |

**Pruebas unitarias**

**Prueba Base de Datos Inventario**

* Obtener historia artista

**Prueba Controller Administrador**

* Probar LogIn
* Probar Crear Subasta
* Probar confirmación de venta

**Prueba Pieza Escultura**

* Probar creación escultura
* Probar modificación de escultura

**Prueba Pieza**

* Probar creación de pieza
* Probar modificación de pieza
* Verificar historial transacciones sobre una pieza