Grupo.o 2 — Diseño y programación orientado a objetos

Alejandra Fruto - 202311287 / José Romero - 202222606

Proyecto 2: Diseño

Este documento está basado en el documento del Proyecto 1 con las actualizaciones necesarias.

El desarrollo del diseño de este proyecto está basado en la metodología llamada Responsibility-Driven Design, esta metodología está compuesta de tres conceptos principales: roles, responsabilidades y colaboraciones.

A continuación, se explicará cómo se implementa esta metodología en el proyecto.

Roles

Cuando se habla de un rol, se hace referencia al conjunto de responsabilidades relacionadas.

Para hallar los roles en el proyecto, inicialmente se eligen una serie de candidatos y para que su diferenciación sea más sencilla, se le asignan estereotipos. Los estereotipos son roles comunes, muy simplificados.

Los elementos candidatos que se identificaron para el caso de estudio y sus respectivos estereotipos son los mostrados a continuación. (La agrupación mostrada está hecha de acuerdo a la perspectiva que se utilizó para encontrarlos):

- Elementos relacionados con el dominio del problema
    - Usuario <<information holder>>
    - Pieza <<information holder>>
    - Persistencia <<information holder

- Elementos relacionados con lo que hace el sistema
    - Subasta <<service provider>>
    - VentaPieza <<service provider>>
    - Consignación <<service provider>>
    - Fábrica <<service provider>>

- Agrupadores
    - Galería <<structurer>>

Estas asignaciones se dan debido a las siguientes razones:

Inicialmente, la galería es considerada un <<structurer>> debido a que no tiene una lógica compleja. Sino, que se encarga de mantener una organización y una relación entre los demás elementos mencionados.

Los elementos subasta, venta y fábrica son <<service provider>> debido a que se encargan de hacer un trabajo para los demás. El elemento de subasta, se encarga de manejar lo relacionado con la creación, desarrollo y finalización de las subastas. El candidato venta se encarga de lo relacionado con venta de piezas desde la galería hacia un usuario. El elemento consignación es un <<service provider>>, ya que se encarga de proveer el servicio relacionado a la consignación de piezas de un usuario a la galería. El elemento de fábrica se ocupa de la creación de piezas, usuarios y galerías. Gracias a estos tres elementos, se desarrolla todo lo necesario para que el sistema funcione correctamente.

Finalmente, usuario, pieza y persistencia son considerados <information holder>> ya que se encargan de mantener y entregar información. Pieza y usuario, tal como lo dice su nombre, mantienen información acerca de las piezas y los usuarios del sistema, respectivamente. (Usuario presentará también otras particiones más adelante). A su vez, persistencia se encarga de guardar y almacenar toda la información necesaria que el sistema necesita.

Responsabilidades

Cuando se habla de responsabilidad, se hace referencia a la obligación de hacer una tarea o de tener información. A continuación se nombraran las responsabilidades que tienen cada uno de los elementos anteriormente mencionados.

- Responsabilidades de galería.
    - Establecer conexiones entre los servicios y los usuarios(distintos tipos de usuarios que se ramifican más adelante).
    - Iniciar subastas.
    - Obtener y mostrar piezas.
    - Cambiar formatos de fechas según sea necesario.
    - Obtener usuarios, ya sea, compradores, empleados o artistas.
- Responsabilidades de pieza.
    - Almacenar información de las características principales de cada pieza.
    - Entregar información de las características de cada pieza.
- Responsabilidades persistencia.
    - Cargar la información de los usuarios y de las piezas.
    - Salvar la información de galería, usuarios y piezas.

    - Guardar y almacenar la información anteriormente nombrada.
- Responsabilidades de subasta.
    - Revisar ofertas (siendo una oferta la cantidad de dinero que ofrece un usuario durante una subasta por una pieza que está siendo subastada).
    - Registrar ofertas en la subasta.
    - Finalizar una subasta.
- Responsabilidades de venta (Venta hace referencia a la venta de piezas por parte de la galería hacia el usuario).
    - Verificar si una pieza se encuentra disponible para ser vendida o no.
    - Hacer efectiva una venta, registrar la venta tanto para la galería como para el usuario.
- Responsabilidades de consignación.
    - Generar una consignación hecha desde el usuario hacia la galería.
    - Revisar las condiciones que se deben cumplir para que se haga una consignación.
    - Dar una consignación por finalizada si se llega a la fecha límite acordada.
- Responsabilidades de fábrica (Fábrica hace referencia al encargado de crear y cargar en el sistema todos los usuarios y piezas).
    - Crear galería.
    - Crear usuarios (empleados y compradores).
    - Crear piezas.
    - Cargar la galería en el sistema.
- Responsabilidades de usuario. Antes de nombrar las responsabilidades del elemento usuario, es necesario hacer la salvedad, de que usuario al momento de llevar el diseño a

clases se divide en empleado, comprador y artista y empleado a su vez se divide en cajero, administrador y operador. Por lo cuál se nombraran las responsabilidades generales de un usuario (las responsabilidades generales aplican para todos los tipos de usuario) y luego las responsabilidades de cada tipo de usuario.

- Responsabilidades generales del usuario.
    - Almacenar la información de las características principales de un usuario. (características, como nombre, login, contraseña, entre otros).
    - Entregar información de las características principales de un usuario.
- Responsabilidades del comprador.
    - Almacenar y entregar su respectiva información.
    - Solicitar hacer una consignación de una pieza a la galería.
    - Solicitar conocer el estado de las piezas.
    - Solicitar la compra de una pieza.
    - Hacer una solicitud de oferta en una subasta.
- Responsabilidades de administrador.
    - Registrar piezas en la galería (ya sea por consignación o de otra forma).
    - Verificar usuarios para una subasta.
    - Verificar las compras hechas por un comprador.
- Responsabilidades de cajero.
    - Vender una pieza de la galería a un comprador.
- Responsabilidades del operador.
    - Mostrar piezas que serán partícipes de las subastas.
    - Registrar las ofertas que hagan los compradores en las subastas.

- Responsabilidad del artista
    - Crear piezas para la galería y solicitar el registro de las piezas a través del administrador.

Colaboraciones

Cuando se habla de colaboración, se hace referencia a la interacción entre objetos y roles. Las colaboraciones se pueden mostrar a partir de diagramas de secuencia. Estos diagramas muestran cómo colaboran unos objetos con otros para lograr llevar a cabo una funcionalidad del sistema. Dentro de un proyecto, se presentan una gran cantidad de colaboraciones, por lo cuál en este documento solo se mostraran las colaboraciones más esenciales. (Todos los diagramas se encontrarán en la carpeta llamada Diagramas en GitHub). También se hace necesario recalcar que dentro de los diagramas de secuencia hay colaboraciones que requieren parámetros muy largos, por lo cuál serán denotados por números dentro de los diagramas y consecuentemente se nombrará después a que hacen referencia dichos números.

Colaboraciones: 1. Cargar una galería en el sistema.

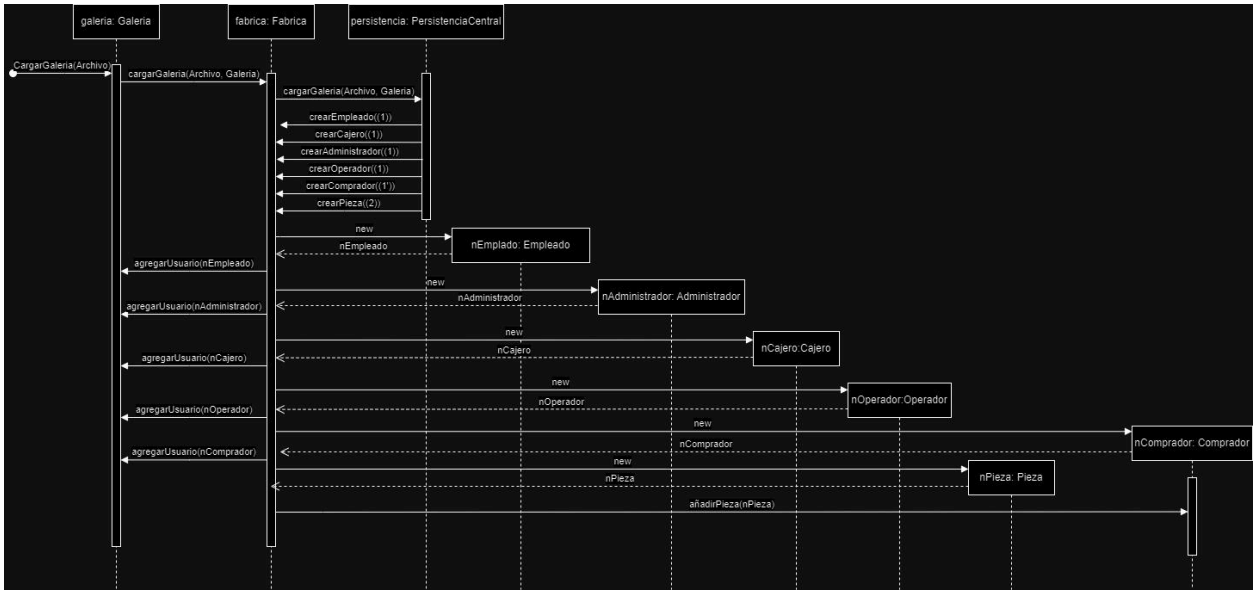


Diagrama de secuencia: Cargar galería.

(1)= String login, String password,int telefono, String nombre .
(1')= String login, String password,int telefono, String nombre para los empleados y String login, String password, String nombre, int valorMaximoCompras, ArrayList<Pieza> historialPiezas, ArrayList<Pieza> piezasActuales, int dinero, int telefono, Galeria galeria para los compradores.
(2) String titulo, int año, int valor, String lugar, String autores, boolean exhibida, Usuario propietario, String exhibaVendaoSubasta, boolean consignación, String fecha, boolean dispsubasta, boolean dispventa.

Colaboraciones: 2. Guardar una galería en el sistema.

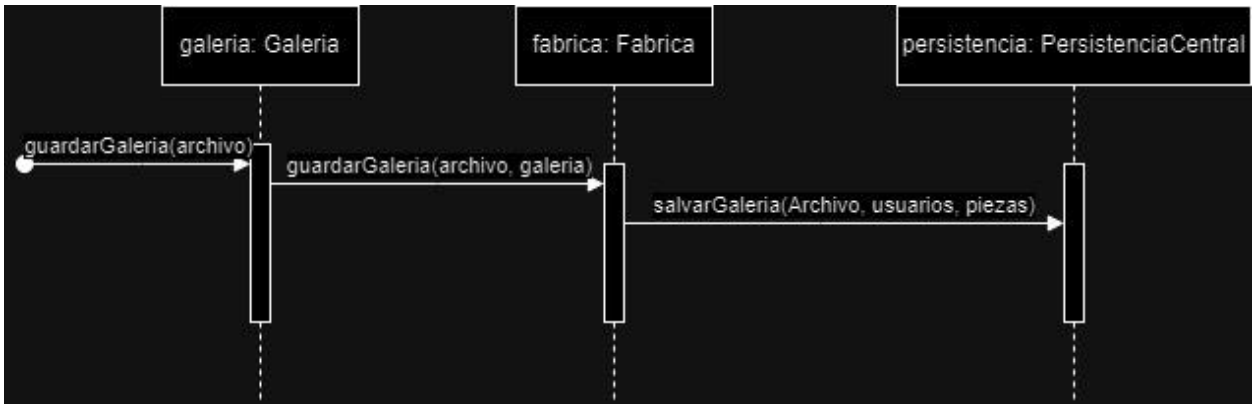


Diagrama de secuencia: Guardar galería .

Colaboraciones: 3. Crear un empleado.

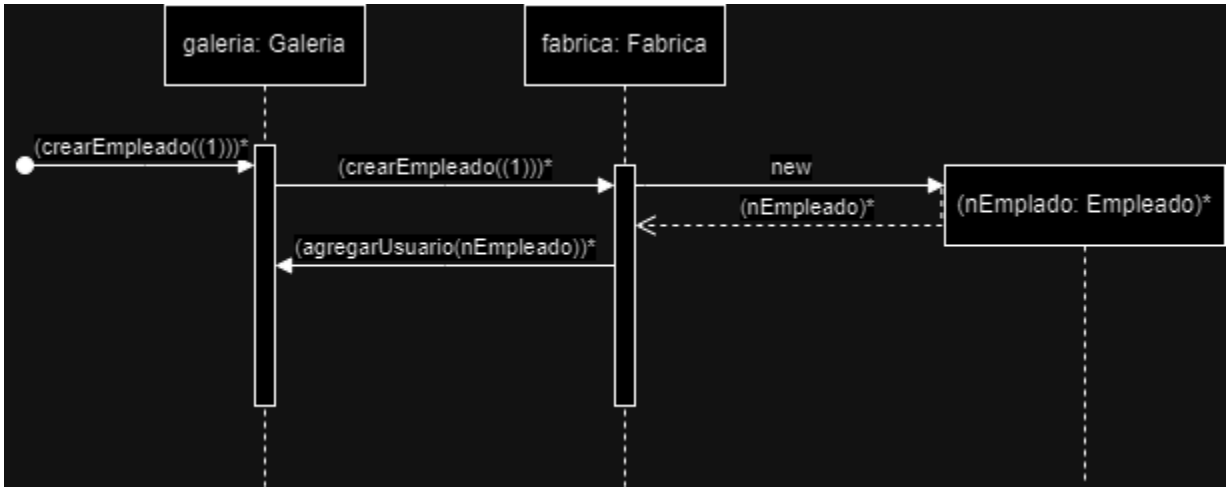


Diagrama de secuencia: Crear empleado (La fábrica genera un nuevo empleado).

(1)= String login, String password,int telefono, String nombre .
():Para crear los otros tipos de usuarios es la misma secuencia, basta cambiar los nombres de los métodos y los objetos sin modificar los parámetros.*

Colaboraciones: 4. Crear un comprador.

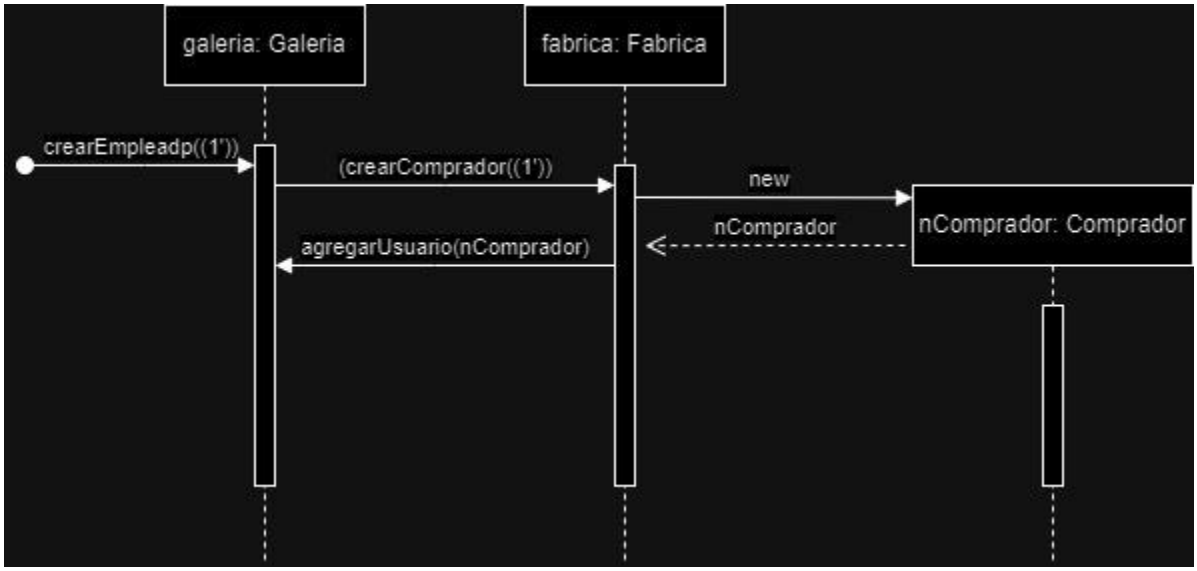


Diagrama de secuencia: Crear comprador (La fábrica genera un nuevo comprador).

(1')= String login, String password,int telefono, String nombre para los empleados y String login, String password, String nombre, int valorMaximoCompras, ArrayList<Pieza> historialPiezas, ArrayList<Pieza> piezasActuales, int dinero, int telefono, Galeria galeria para los compradores.

Colaboraciones: 5. Crear una pieza

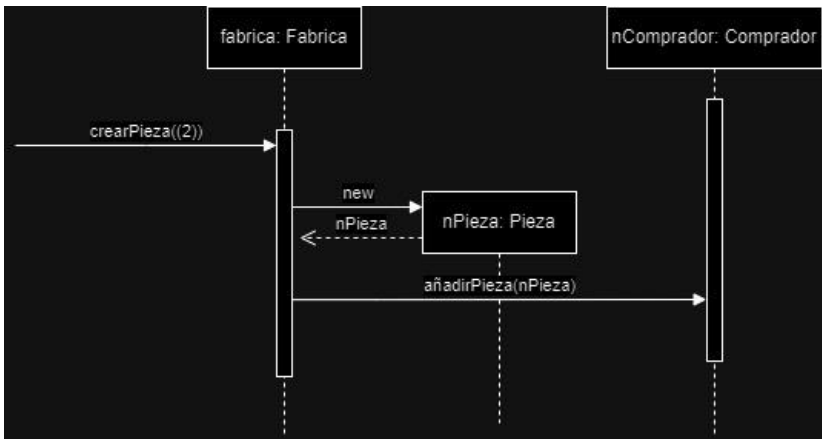


Diagrama de secuencia: Crear pieza (La fábrica genera un nuevo pieza, la cuál es asignada a un Usuario)

(2) String titulo, int año, int valor, String lugar, String autores, boolean exhibida, Usuario propietario, String exhibaVendaoSubasta, boolean consignacion, String fecha, boolean dispsubasta, boolean dispventa.

Colaboraciones: 6. Generar una nueva consignación

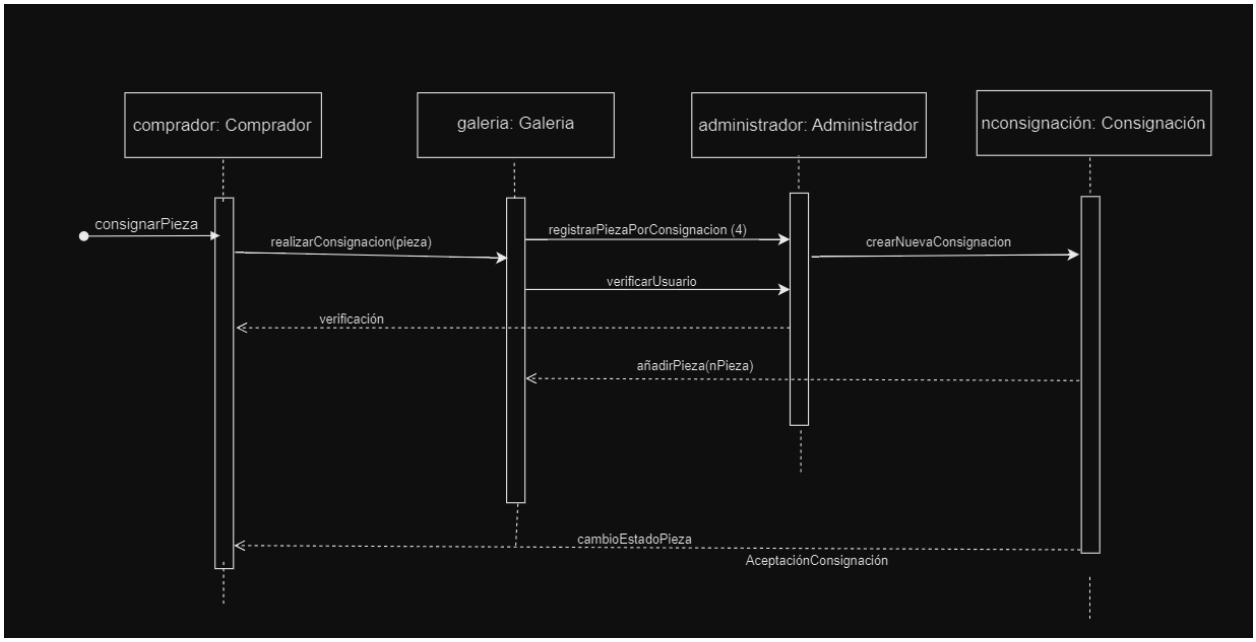


Diagrama de secuencia: Consignación (El usuario entrega una pieza temporalmente a la galería)

(4)Usuario propietario, Pieza piezaAConsignar, String fechaLimite, Galeria galeria, String exhibaVendaoSubasta

Colaboraciones: 7. Crear nueva subasta

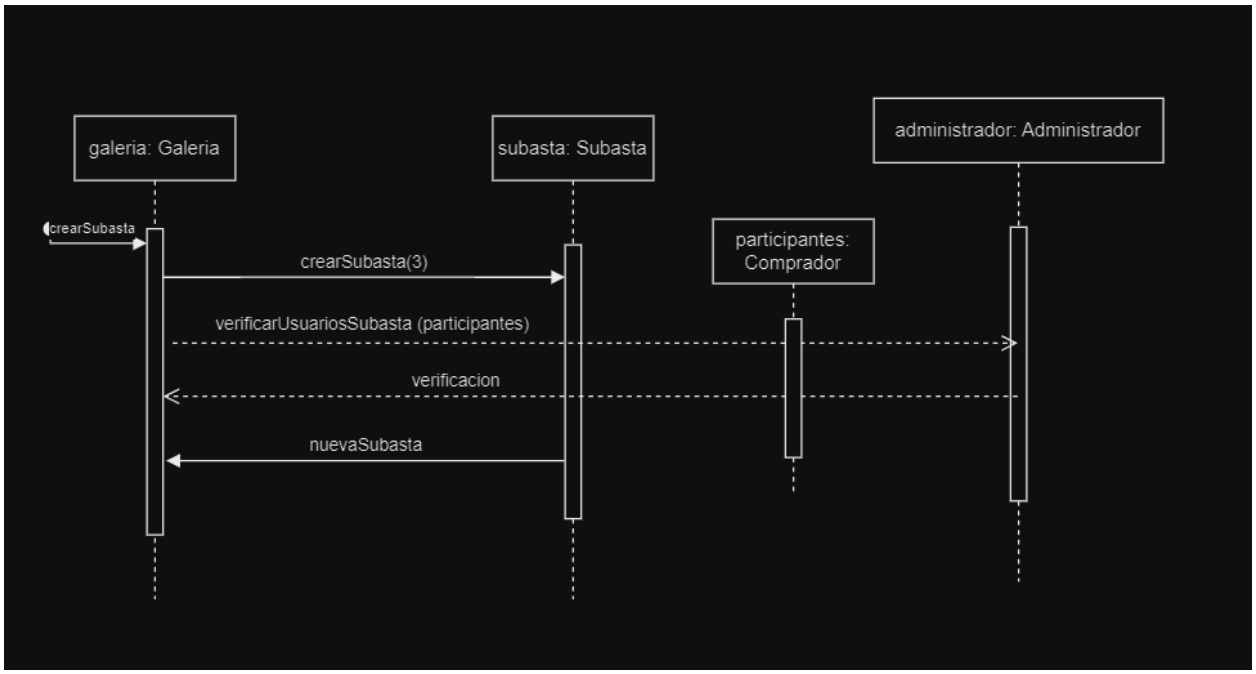


Diagrama de secuencia: Crear subasta (Se crea una nueva subasta perteneciente a la Galeria)

(3)ArrayList<Usuario> participantes, Usuario operador, HashMap<Pieza,HashMap<Usuario, Integer>> registroSubasta, HashMap<Pieza, ArrayList<Integer>> piezasSubastadas

Colaboraciones: 8. Generar oferta en una Subasta

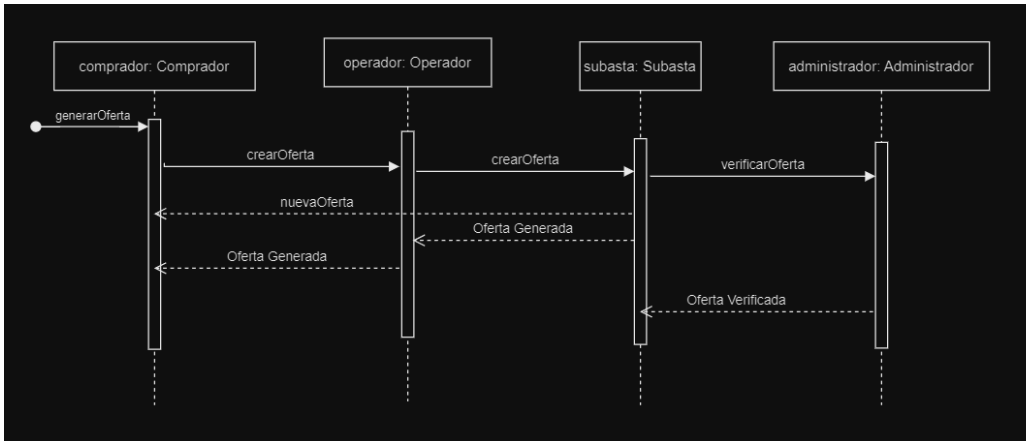


Diagrama de secuencia: Generar oferta en Subasta (Un comprador desea generar una oferta dentro de una subasta)

Colaboraciones: 9. Comprar y venta de una pieza

La compra y la venta de una pieza es considerada una misma colaboración. Se debe ver como compra lo que pide el comprador y como venta el retorno de esta solicitud por parte de la galería.

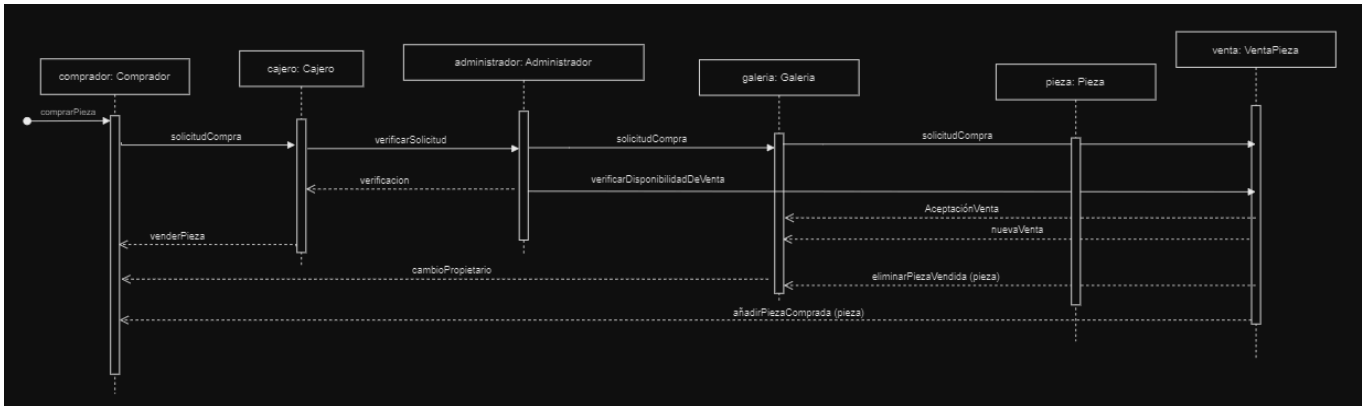


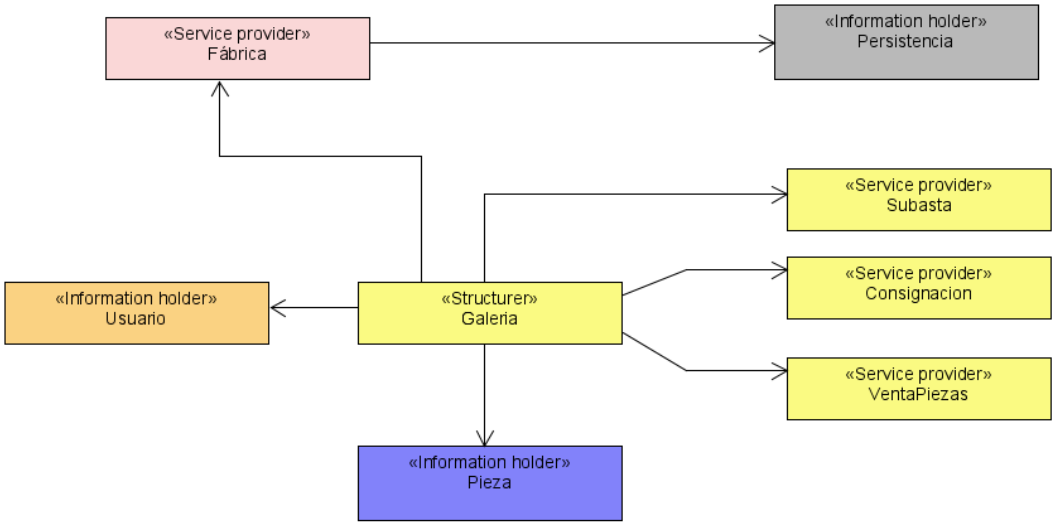
Diagrama de secuencia: Compra de una pieza (Un comprador desea adquirir una nueva pieza de la galería).

Desarrollo del diseño

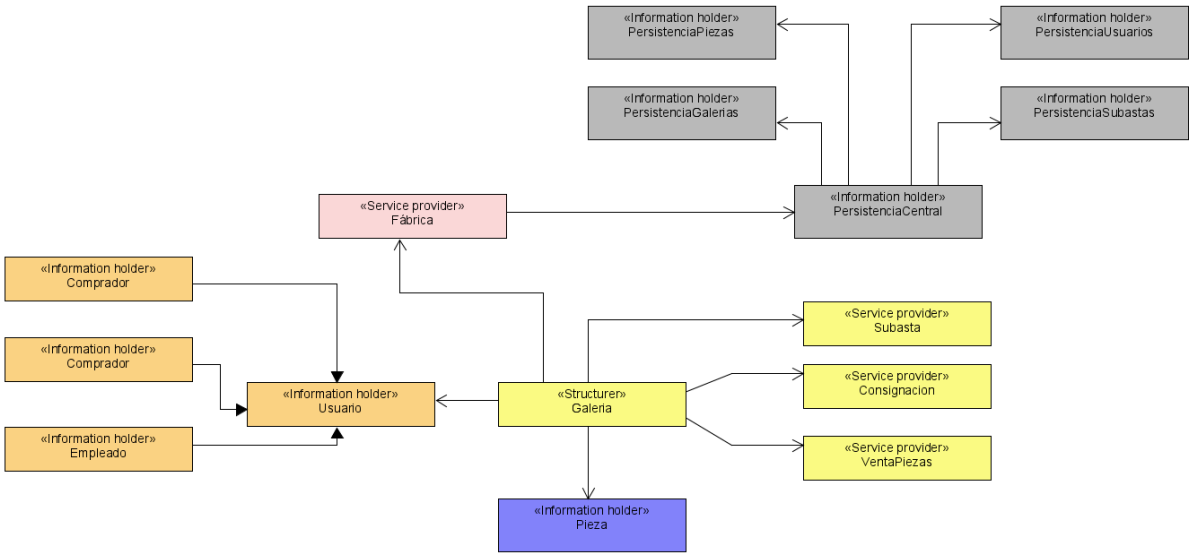
Diagramas de evolución

El correcto desarrollo del diseño explicado anteriormente, se basa en la evolución de los diagramas mostrados a continuación hasta llegar al diagrama de clases, que cuenta con todos los métodos y atributos. (Todos los diagramas se encuentran en la carpeta llamada Diagramas en GitHub).

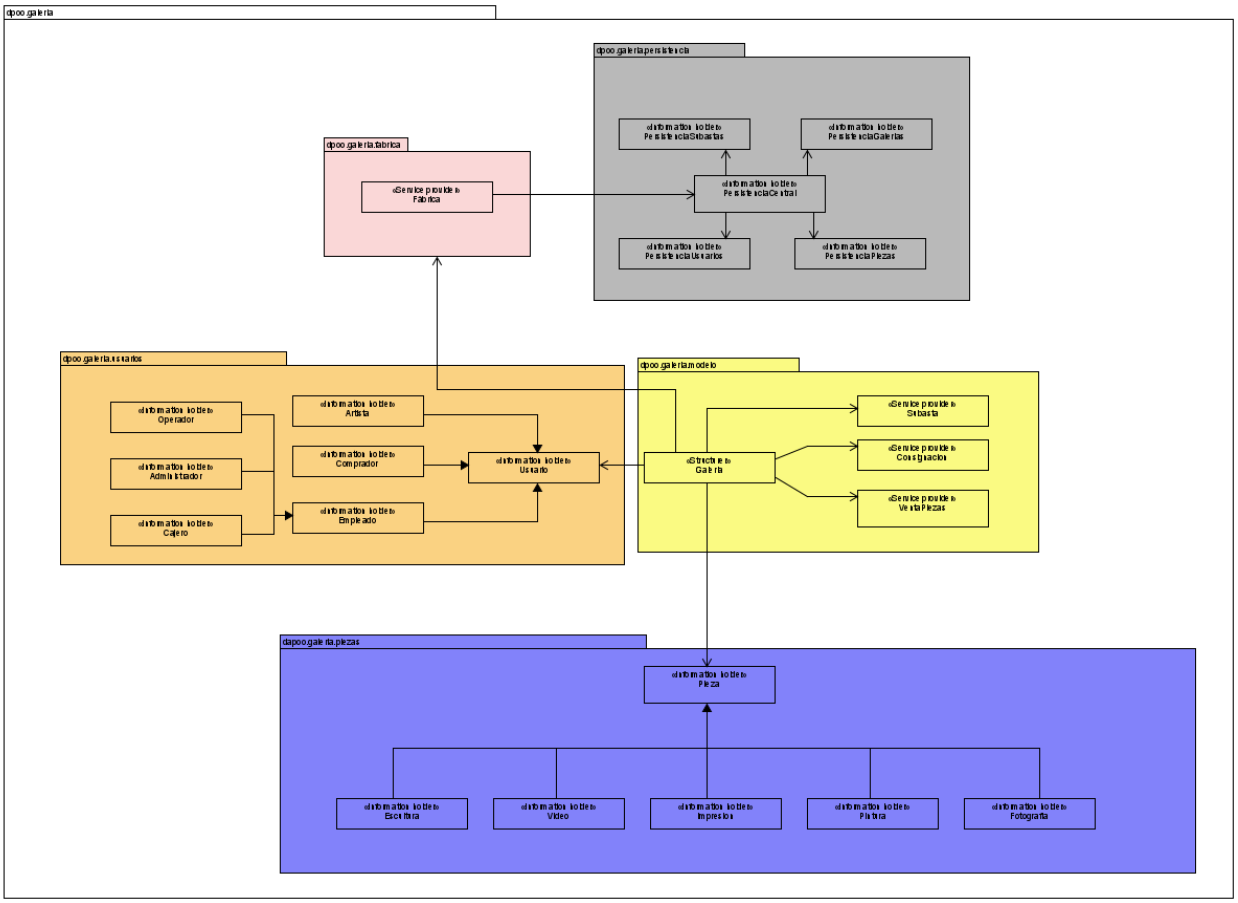
El primer diagrama muestra el primer nivel de las relaciones con los candidatos mencionados más abstractos.



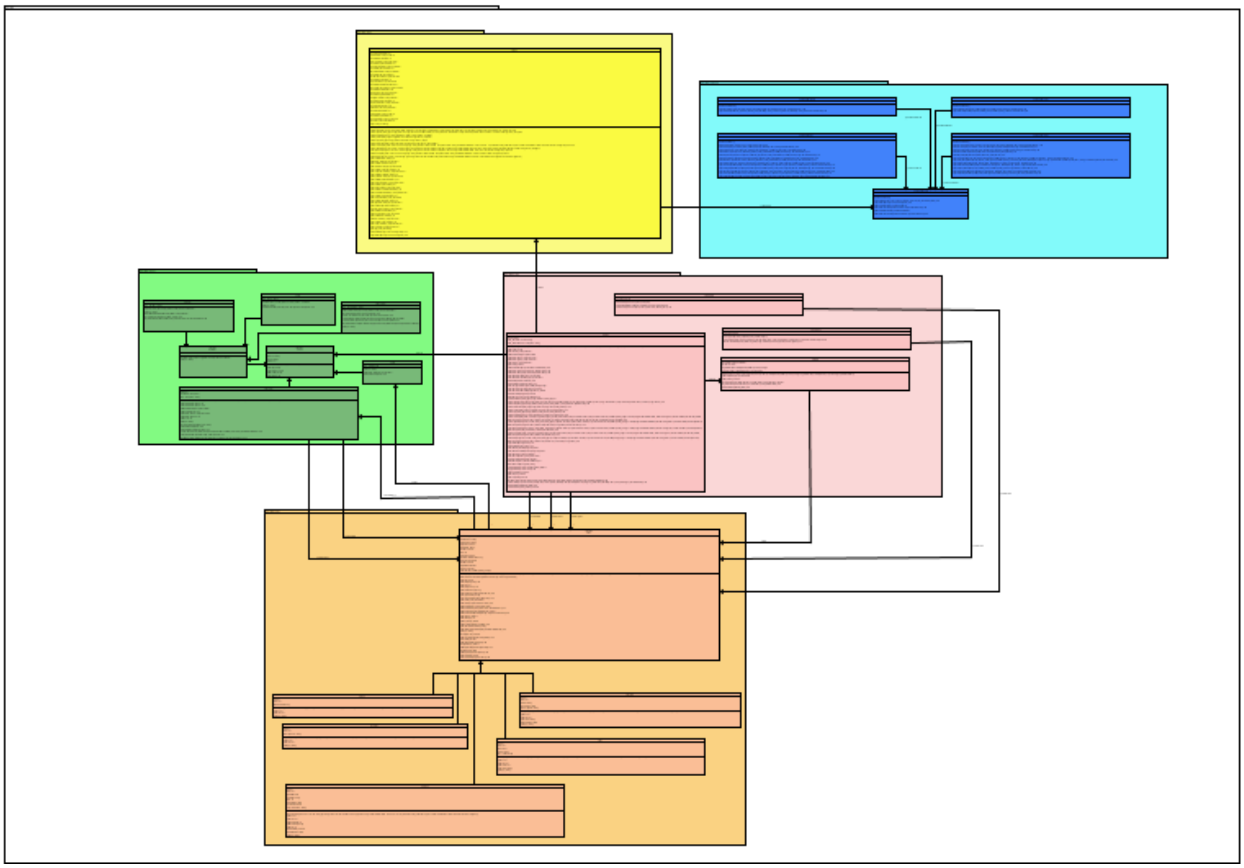
Por consiguiente, en el segundo nivel el diagrama muestra cómo estos candidatos se dividen, para obtener elementos más acordes a las responsabilidades planteadas.



Finalmente, se obtiene el último nivel para obtener las clases que serán implementadas.



De igual forma, se presenta el diagrama UML de alto nivel, con todos los atributos, métodos y cardinalidades. (Se recomienda que se observe desde el PDF encontrado en la carpeta de Diagramas de GitHub debido al tamaño de este).



Explicaciones y justificaciones

Persistencia

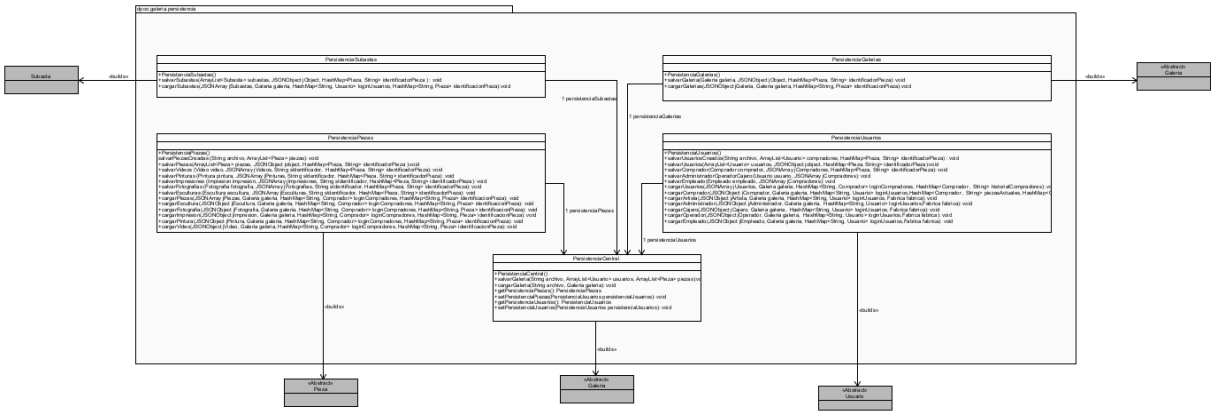
Para almacenar la información de forma persistente se decidió usar JSON en vez de archivos planos o binarios por la versatilidad que el primer método ofrece a la hora de almacenar y recuperar la información almacenada.

Para guardar y cargar la información almacenada se crearon métodos para guardar y cargar las piezas, los usuarios, las subastas y los historiales de la galería.

Ahora, se explicará la manera en la que se guarda la información de la galería. En primer lugar, se crean estructuras de datos auxiliares que nos ayudarán a almacenar la información. Se crea un HashMap con llaves de tipo Pieza y valores de tipo String. La razón para usar dicha estructura de datos es que no se pueden almacenar objetos en el formato JSON, así, para poder guardar el historial de piezas del usuario el historial de piezas del artista y los historiales de piezas de la galería, a cada pieza se le asigna un identificador de tipo String para que dichos identificadores se almacenen en el historial de piezas del usuario, el historial de piezas del artista y los historiales de piezas de la galería. Después, se obtiene la información de los usuarios registrados en la galería, de las piezas creadas, de las subastas y de las piezas de la galería. Para lograr lo anterior, se obtiene el ArrayList que almacena objetos de tipo Usuario que contiene todos los usuarios creados y registrados en la galería junto a las piezas creadas que son almacenadas en el objeto fábrica de la galería y los ArrayLists de subastas y piezas de la galería almacenados en la galería. Una vez se tiene lo anterior, se crea el archivo JSON y se llaman los métodos que recorren los ArrayLists de los objetos previamente mencionados y los guarda . Es para esta parte que cada pieza y usuario tiene un atributo de tipo String llamado tipo que almacena de qué tipo son, pues de lo contrario no se podría saber a la hora de leer el archivo.json de qué tipo es el objeto que se está leyendo. Además, el atributo tipo nos permite llamar un método específico que se encarga de guardar cada tipo de pieza y usuario. Por ende, el archivo en formato JSON resultante se divide entre los usuarios, las piezas, las subastas y los historiales de piezas de la galería. Por último, se decidió almacenar la información en un solo archivo ya que de esta forma es mucho más fácil leer y cargar la información almacenada por la manera como se guarda la información.

Para terminar la explicación de la persistencia se explicará la manera en la que se carga la información almacenada. Como se hace para guardar la información, se crean estructuras de datos de tipo HashMap que nos permiten que los usuarios se recuperen con su login, que las piezas se recuperen con su identificador asociado en la carga y que a cada comprador se le asocie el historial de piezas que tiene. Posteriormente, se recupera la información de las piezas, los usuarios, las subastas y los historiales de piezas de la galería almacenada en el archivo y se llama un método encargado de cargar los objetos presentes en el archivo .json.

La persistencia se puede entender de una manera más sencilla, a partir del siguiente diagrama:



Decisiones tomadas para la implementación.

A continuación se presentará la explicación de las decisiones tomadas con respecto a la implementación relacionada con inventario de piezas, proceso de compra y manejo de compradores.

En un principio, al diseñar las clases la galería estaba compuesta de cuatro grandes elementos: piezas, usuarios, galerías y subastas.

El primer elemento se trataba de todos los tipos de piezas a los que se les asignó el estereotipo de information holder y se decidió crear una clase abstracta llamada Pieza que tuviera como atributos las características que todas las piezas poseen (título, autor, año, lugar, si está exhibida, el valor, el propietario, si se quiere vender exhibir o subastar y su disponibilidad) y una clase de cada tipo (Video, Escultura, Fotografía, Impresión y Pintura) con sus características particulares que herede de la clase abstracta.

El segundo componente se trataba de los usuarios que interactúan con la galería, los cuales son administradores, cajeros, operadores, empleados, compradores, vendedores, participantes de subastas y propietarios. Como los administradores, cajeros y operadores son empleados con más responsabilidades, se decidió que existieran las clases Administrador, Cajero y Operador que heredan de una clase llamada Empleado con los atributos que todos los empleados tienen (login, password, nombre, teléfono) y los métodos que permiten encargarse de manipular y controlar el inventario de piezas de la galería.

Por otro lado, estaban los compradores, participantes de subastas y los propietarios de las piezas. Se decidió juntarlos en una sola clase llamada Comprador porque las funciones que cumple cada elemento es muy reducida puesto que el propietario se refiere a tener piezas, el comprador a adquirir pieza de la galería y el participante de subastas a ofrecer dinero en las subastas que le sean permitidas. A su vez, si un comprador compra una pieza se convierte en propietario y si quiere participar en una subasta se convierte en participante de una subasta. Como el mismo razonamiento es válido para los otros tres elementos quiere decir que se comportan de forma similar y es más fácil que estén agrupados en solo una clase.

El tercer elemento era la galería al que se le asignó el estereotipo de structurer para que mantuviera las relaciones entre los demás elementos y la información del funcionamiento del programa. Por lo anterior, se decidió que galería no necesitaba dividirse en más clases, pues no tiene responsabilidades complejas en el funcionamiento de la aplicación.

El cuarto elemento eran las subastas que se llevarían a cabo en la galería. A este elemento se le asoció el estereotipo de service provider y al evaluar sus responsabilidades a la hora de realizar una subasta, se llegó a la conclusión de que sería buena idea recurrir a varios service providers que faciliten las conexiones y realización de las funcionalidades críticas de la galería. De dicha reflexión se decidió crear cinco clases que proveen servicios a las otras clases para poder llevar a cabo las funcionalidades de forma pertinente. Estas clases son Subasta que se encarga de gestionar lo relacionado a las subastas, VentaPiezas que se encarga de gestionar la compra y venta de piezas, Consignación que se encarga de la consignación de piezas a la galería por parte del usuario, Fábrica que es un patrón de diseño en la programación orientada a objetos encargado de crear todos los objetos que intervienen en el funcionamiento de la aplicación (piezas, usuarios y galerías) y PersistenciaCentral que se encarga de que la información de la galería sea persistente.

Por último, para implementar los nuevos requerimientos del proyecto se creó la clase de Artista. Esta clase sigue el estereotipo de information holder, pues su única función es tener su información y la de las piezas que ha creado. Así mismo, se decidió que esta clase herede de Usuario porque la clase de Usuario se sabe cómo se comporta y permite que esta nueva clase se agregue a la aplicación sin modificar mucho la lógica y los métodos que ya existen.