Proyecto 3 - Entrega única

Integrantes:

• Juan Pablo Castro Idarraga – 202012399

• Joseph Eli Pulido Gómez – 202211365

• Santiago Martínez Castillo - 202124032

Actualización de diseño:

Esta vez no se le hizo ningún cambio sustancial al modelo del programa, todos los cambios se hicieron sobre la interfaz o sobre las herramientas auxiliares al modelo como lo son la persistencia y las pasarelas de pago. Por ende, en este documento nos centraremos en presentar los cambios a la interfaz y el paquete nuevo llamado “PasarelasDePago”.

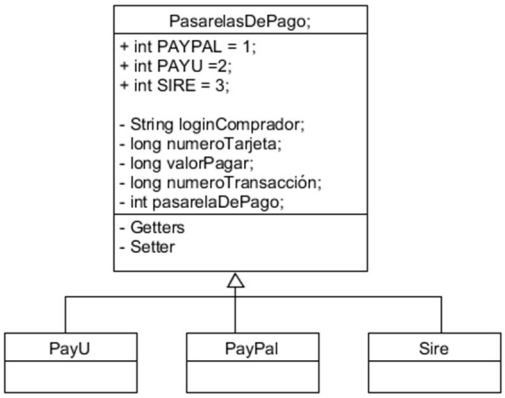
Correr Programa

El programa se corre desde el main de la clase MenuPrincipal1. Después, se tiene que elegir la opción de iniciar sesión. La mayoría de los requerimientos que se pidieron se puede comprobar que fueron implementados desde el MenuCajero, por ende, se tiene que iniciar sesión con el usuario y la contraseña de un cajero. En el archivo JSON llamado galería que está en la carpeta datos esta la información de los clientes y los empleados que se cargan.

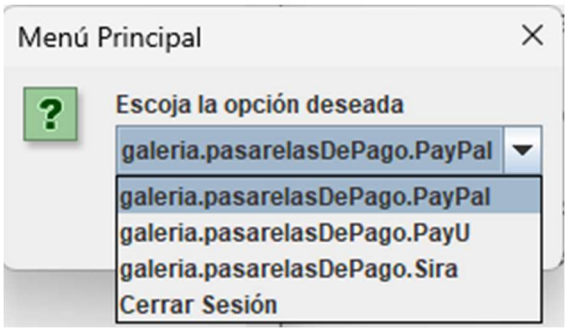
Ejemplo, para iniciar sesión como cajero utilice el usuario “caj” y la contraseña “b”.

Pasarela de Pagos:

Se creó un nuevo paquete dentro de src llamado pasarelasDePago. Este nuevo paquete contiene 4 clases. Primero está la clase abstracta PasarelaDePago. Esta clase tiene los atributos recomendados por la guía del proyecto 3, constantes para referirse a las subcalses, setters y getters (mostrados a continuación en el diagrama de clases UML). Esta clase abstracta tiene tres diferentes implementaciones: PayPal, PayU y Sire. Como su nombre lo indica estas son pasarelas de pago. Cada una con 10 líneas de código, lo que sugiere que la implementación de una nueva pasarela no es complicada. A continuación el diagrama UML del paquete pasarelasDePago.



En la figura anterior no se pusieron los constructores ni todos los getters y setters. IMPORTANTE las constantes que representan números que diferencian cuando se ha usado una pasarela u otra. Esto es importante por que al guardar una nueva transacción en el archivo transacciones, el primer digito del numeroTransacción indica que pasarela se usó en la transacción. Las pasarelas disponibles están en el archivo pasarelas.txt en el paquete galería.interfaz. Por ahora solo hay tres pasarelas. Si usted agrega una nueva pasarela en el archivo .txt en el formato en el que están las otras tres pasarelas. Notará que al iniciar sesión como cajero (según la información de arriba) y al elegir la opción de registrar un Pago, sea por venta directa o subasta, podrá ver la opción de la nueva pasarela de pagos que usted agrego. Aunque se puede elegir y colocar los datos de la tarjeta, si elige esta pasarela el problema fallara a menos que se haya implementado la clase correspondiente.



Ventana para elegir la pasarela

Una vez se elige la pasarela de pagos y digita la información para hacer la compra aparecerá en el archivo transacciones.txt una nueva transacción. Este archivo se ubica de último en la carpeta del proyecto:

Lugar en el que ponen las transacciones realizadas

La transacción mas reciente aparece abajo. Como se dijo anteriormente, el número de transacción está conformado por un digito que indica cual fue la pasarela de pagos (1,2,3) y tres números más específicos para reconocer la transacción.

Nueva Interfaz Grafica

Nuestra nueva interfaz gráfica se creó a partir de la interfaz por consola anterior. Entonces parece preciso, mostrar de nuevo el diseño de esta:

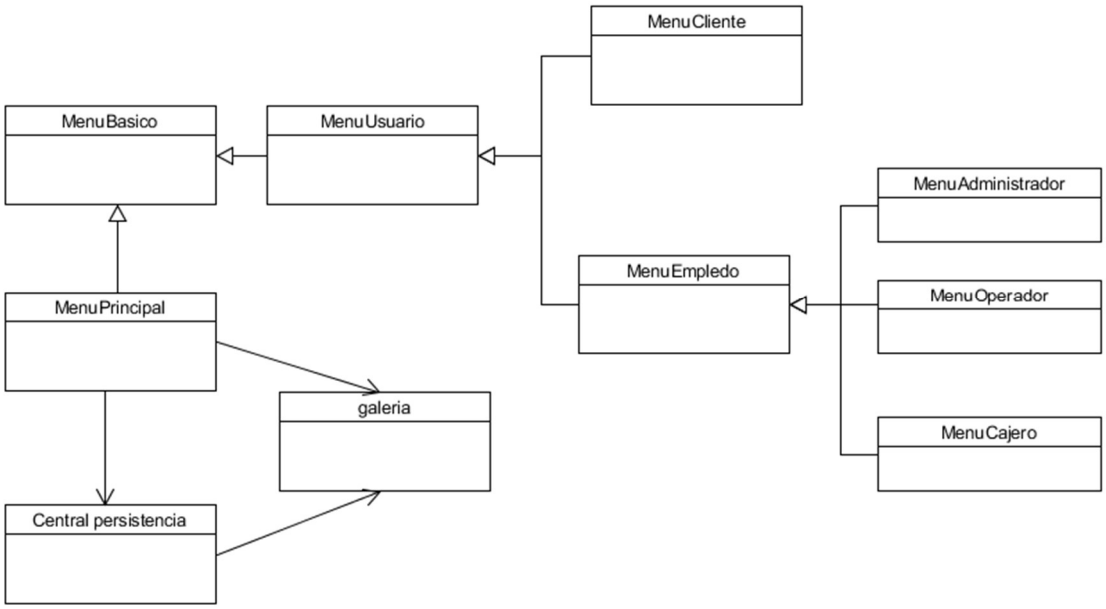
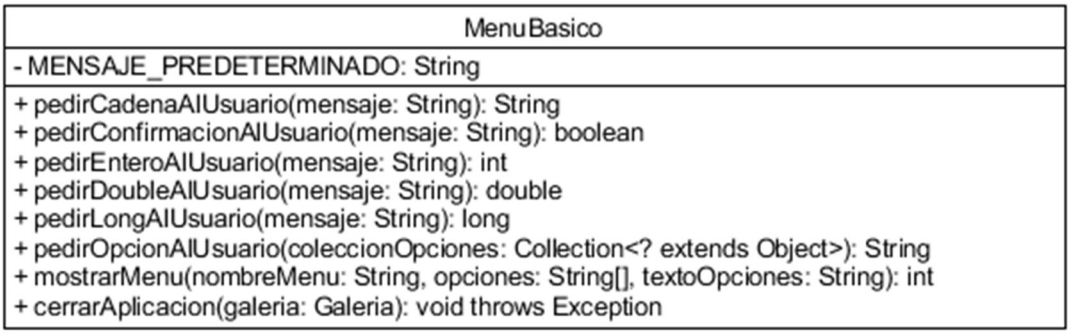


Fig1: Interfaz anterior

Recordemos como funciona esta interfaz. La función main se encuentra en la clase MenuPrincipal, este main muestra el MenuPrincipal que es iniciar sesión o salir de la aplicación. El menúPrincipal puede imprimir esto por consola gracias a que tiene una función llamada mostrarMenu(). Esta función es heredada. Es heredada de la clase MenuBasico, que cuenta con muchas otras funciones. A continuación, el UML de la clase Menu Básico:



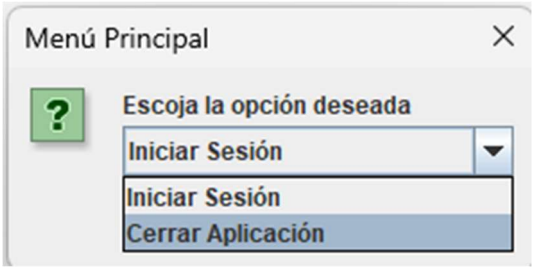
Como se puede ver, menú básico contiene todo tipo de métodos que nos sirven para Pintar cosas que interactúen con el usuario. Como pedir cadenas, enteros, doubles entres otros. También tiene el método mostrarMenu que recibe un grupo de opciones y hace que el usuario escoja entre ellas.

De este Menu básico, como se ve en el Diagrama UML de alto nivel de la interfaz anterior hereda MenuPrincipal y MenuUsuario. De MenuUsuario hereda todas las demás clases que interactúan con los usuarios. Es decir MenuBasico es el pilar fundamental de nuestra interfaz.

Como una buena práctica de programación, las clases que heredan de MenuBasico no les importa como estos métodos están implementados, solo les importa que cumplan con la función de mostrar las opciones o pedir información, dependiendo de la clase. Por esta razón, la forma en que cambiamos la interfaz basada en Consola a una interfaz grafica fue simplemente cambiar Menú Básico. Solo fue necesario cambiar la implementación de los métodos de MenuBasico asegurándonos de cumplir el principio SOLID numero 3: Principio de sustitución de Liskov.

Por ende, aplicando el principio de sustitución de Liskov, sustituimos todos los métodos de la clase MenuBasico, para que recibieran y devolvieran los mismos parámetros. Cambiando la implementación.

Debido a esto nuestro programa es particular. No usa solo una ventana que se repinta de acuerdo con lo que se necesite mostrar si no que hace aparecer y desaparecer la ventana una y otra vez. Además, las ventanas no tienen un diseño personalizado pues todas son pintadas por el mismo método. Esto explica por qué cada vez que se requiere elegir una opción sale el mismo JObject para presentarla.



Nuestro modelo para la interfaz puede ser el mas sencillo de implementar, si se tiene el diseño de interfaz correcto y la idea.

El diseño básico de la interfaz no sufrió ningún cambio drástico, pero si fue necesario agregar dos clases para implementar el nuevo requerimiento MenuPago y MenuPasarela. MenuPasarela aparece tan pronto el cajero elige la opción de registrar un Pago y es el que se encarga de mostrar las pasarelas que hay. El cuadro UML de esta clase es:

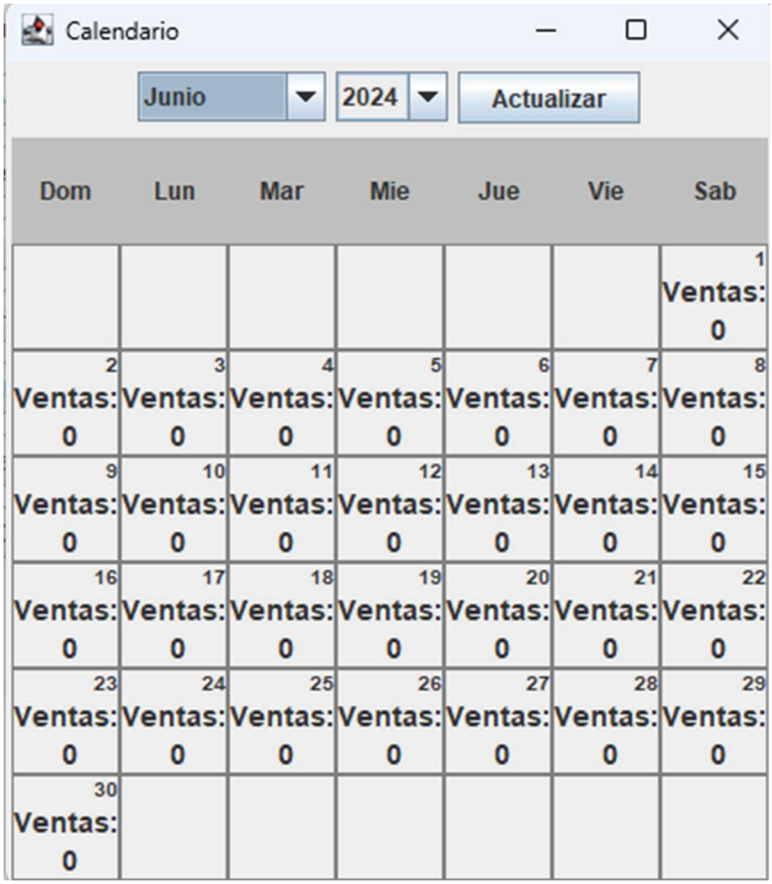
MenuPasarela
- pasarelas: List<String> - pasarelaElegida: String
+ MenuPasarela() + getPasarela(): String + getNumeroPasarelas(): int - mostrarMenuPasarela(): void - leerPasarelas(): void

Notar que esta clase esta conectada con la clase PasarelDePagos. La segunda clase que agregamos es MenuPago. Esta clase pide los datos necesarios para crear la transacción. Por facilidad, no nos importó aclarar la pieza que se esta vendiendo, el sistema debe saberlo ya. Este MenuPago queda así:

MenuPago
- miPasarela: PasarelaDePago
+ MenuPago(pasarela: String) - mostrarMenuPago(pasarela: String): void - salvarTransaccion(pasarelaObjeto: PasarelaDePago): void - ObtenerNumeroTransaccion(pasarelaDePagoNumero: int): long - getPasarela(pasarelaNombre: String, loginComprador: String, numeroTarjeta: long, valorPagar: long, numeroTransaccion: long, pasarelaDePagoNumero: int): PasarelaDePago - extraerNombrePasarela(linea: String): String

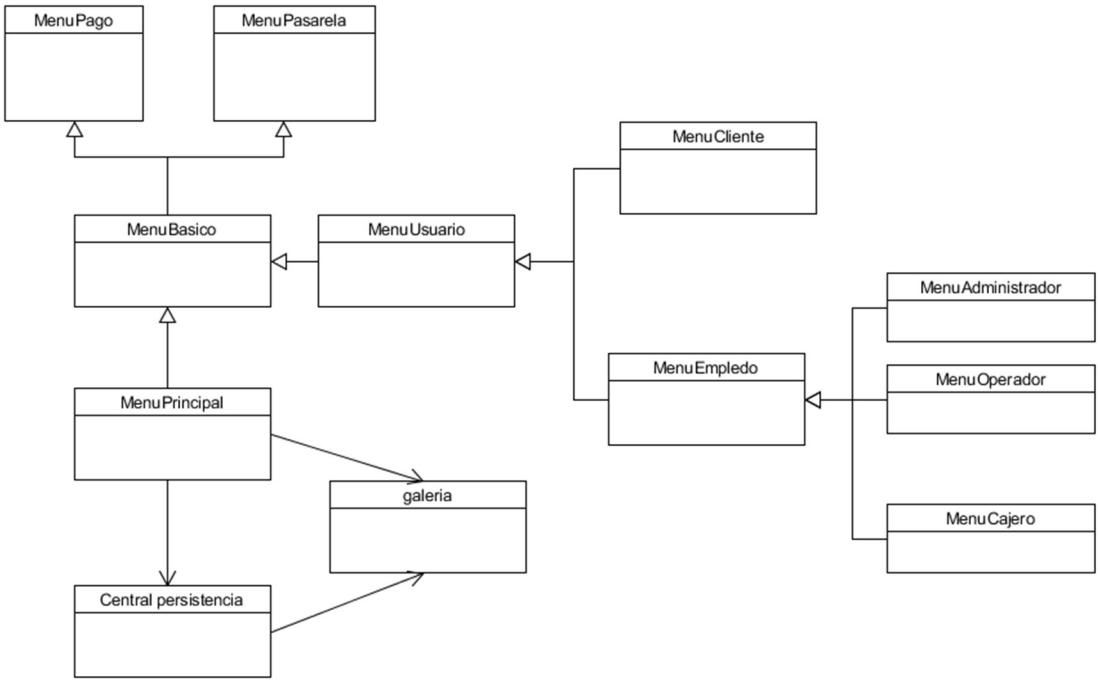
También se programo una clase que sirve de soporte para crear los pequeños frames de algunas interfaces la cual se llama InternalFrameManager que aparece en el diagrama de clase dado que es de soporte y no afecta tanto a la parte funcional dado que perfectamente se habría podido incluir en los mismos códigos de los menús.

En el apartado de MenuAdministrador, que hace parte de MenuAdministrador se añadió una interfaz gráfica que se despliega cuando se le da a la opción “Número de ventas por día” en la que se muestra un calendario donde se le pueden cambiar el mes y el año. Además, en cada día muestran los números de ventas que se realizaron en ese día. Para el desarrollo de este requerimiento se creó la clase CalendarFrame.



En esta imagen se ve el calendario diseñado para mostrar las ventas.

Finalmente, el diagrama de alto nivel que representa la interfaz es este:



Diseño de la aplicación

Como el diseño de la aplicación no se cambió, excepto por las clases que ya mencionamos anteriormente, no es necesario volver a comentar las decisiones de diseño que se tomaron. Sin embargo, ponemos el diagrama UML de alto nivel para refrescar las conexiones del modelo. En este diagrama no aparece la interfaz (ya mostrada), la persistencia ni la pasarelasDePagos.

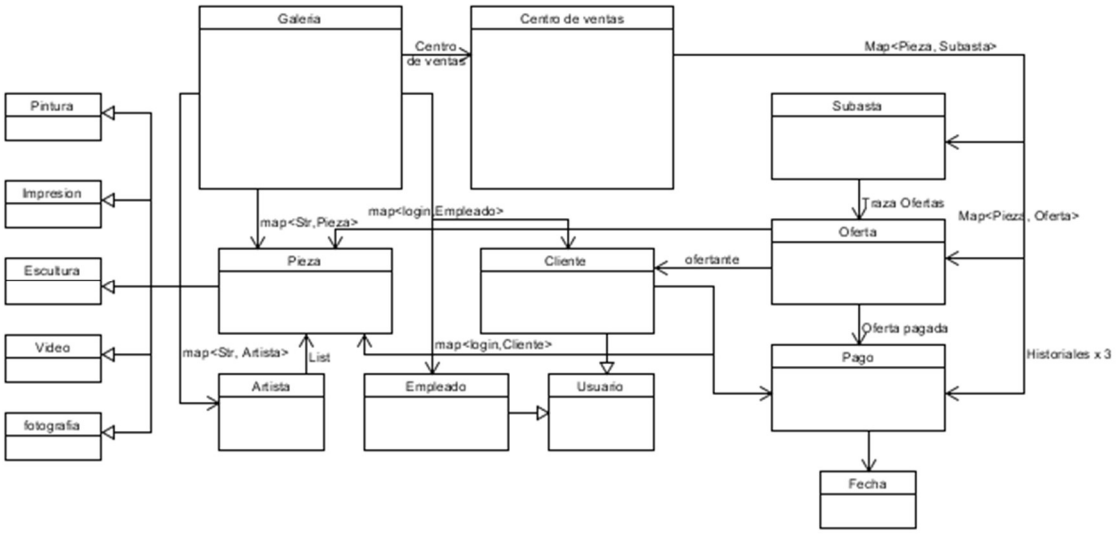


Diagrama UML

La conexión que va desde centro de ventas hasta pago “Historialx3” hace referencia a tres atributos de tipo HashMap que hacen de historiales:

```
private Map<String, List<Pago>> historialDePagosPorPieza;
private Map<String, List<Pago>> historialComprasComprador;
private Map<String, List<Pago>> historialVentasPropietario;
```

Las flechas que no tienen label se asume que solo tienen uno de ese objeto y que la relación es trivial. Por ejemplo, pago->fecha, es evidente que un pago tiene una fecha y esta fecha es cuando se realiza el pago.