

Paradigmas de Programación

Proyecto Integrador - Unidad 2

“Banco la Familia”



Perla Valerio, Gonzalo Padilla

Lic. Cs de la computación/Facultad de Ingeniería/UNCUYO

12 de septiembre de 2024

Abstract

En el presente trabajo se desarrolla el proceso de resolución del problema “Banco la Familia” correspondiente al proyecto integrador de la unidad 2: Paradigma Orientado a Objetos de la asignatura Paradigmas de Programación.

1. Introducción

Este proyecto implementa el Paradigma de Programación Orientada a Objetos para abordar el desarrollo de un programa en Java que simula el sistema del banco “La Familia” el cual, a pesar de su fachada legítima, ha estado bajo el radar de las autoridades debido a su conexión con la mafia local y actividades de lavado de dinero. Para proteger sus verdaderas actividades, cuenta con una estructura jerárquica compleja en la cual se ven representados varios tipos de empleados como gerente, cajeros, asesores financieros y agentes especiales, cada uno con un conjunto de responsabilidades y permisos bien definidos dentro del sistema, en cual se debe incluir la posibilidad de realizar operaciones encubiertas para los clientes de la mafia como por ejemplo transferencias no rastreables. A continuación se plantean los objetivos de desarrollo:

1. Modelar una solución que permita simular un banco con todos sus casos.
2. Generar al menos 50 movimientos en el banco teniendo en cuenta los posibles casos.
3. Imprimir el estado actual del banco.
4. Indicar cuando se realice una simulación de lavado de dinero.
5. Utilizar el sistema de copia de seguridad para mostrar los movimientos teniendo en cuenta lo explicado por el lavado de dinero.
6. Implementar al menos dos operaciones con la bolsa.

7. Implementar una interfaz en el diagrama de clases.

El presente proyecto se llevó a cabo con el fin de poner en práctica y afianzar los conocimientos adquiridos con respecto a la Programación Orientada a Objetos en la asignatura de Paradigmas de Programación, lo que conllevó al planteamiento de un sistema de banco que cumpliera con las características planteadas en la premisa del problema teniendo en cuenta los principios de abstracción, herencia, polimorfismo y encapsulación, además de todas las herramientas que nos brinda el paradigma para poder desarrollar un programa que a pesar de ser complejo pueda ser representado de una forma sencilla y organizada que permita la reutilización de código y la correcta estructuración de cada uno de los procesos del banco.

2. Metodología

Diseño

Siguiendo lo interpretado de la consigna del laboratorio, decidimos desarrollar un programa que simule un banco en donde se puedan realizar transferencias, depósitos, retiros, préstamos, y compra y venta de acciones, además de transacciones encubiertas para el lavado de dinero.

Por un lado, el usuario del programa debía poder tomar el rol de los distintos clientes y empleados, y realizar las operaciones exclusivas de cada uno, mediante la interacción manual con el programa. Por el otro, estas operaciones se deben poder simular, y que el usuario pueda observar cómo se ejecutan automáticamente.

Para cumplir con lo primero, decidimos armar un sistema de Login que acepte tanto a los clientes como a los distintos tipos de empleados, mediante el cual podamos realizar las operaciones permitidas por cada uno de ellos. Se excluyó el caso de los cajeros, ya que no habría mucha interacción por su parte a través de un sistema de login.

Para cumplir con el caso de la simulación, se implementó un sistema que programa a futuro una serie de distintos tipos de operaciones con los clientes del banco. Para ello, fue necesario implementar una simulación del paso del tiempo, para revisar constantemente si hay operaciones que ya es momento de realizar.

Se decidió implementar una interfaz de texto (se renderiza en una terminal, utilizando caracteres especiales para dibujar gráficos), para que la experiencia del usuario sea más amigable.

Etapas

Para alcanzar los objetivos propuestos, dividimos el problema en las siguientes etapas:

Etapa I: Modelado del sistema del banco con diagrama de clases en **UML**

1. Analizamos el funcionamiento del banco, sus componentes (empleados, clientes, operaciones) y las relaciones (agrupación, composición, y asociación) entre ellos.
2. Identificamos las necesidades del sistema y se determinaron los objetos que serían parte primordial del modelo solución.
 - Empleados, clientes, operaciones, copia de seguridad, inversiones en la bolsa, módulo de lavado de dinero.
3. Partimos del principio de abstracción para poder enfocar el diseño en las características y funciones fundamentales de los objetos antes identificados esto con el fin de filtrar solo los aspectos importantes para cumplir con los objetivos del proyecto.
4. Identificamos los casos en los que era útil la herencia, el polimorfismo, el uso de métodos abstractos e interfaces dentro de nuestro modelo para lograr la reutilización de código, simplificar el desarrollo.
5. Diseñamos el diagrama de clases usando **draw.io**, una aplicación en línea para diagramación que permite crear diagramas de flujo, diagramas en UML, etc.

Etapa II: Corrección de diagrama de clases, diseño de diagrama de flujo para los procesos más complejos del banco y buscar alternativas de interfaces gráficas

1. Verificamos el diagrama de clases y tratamos de lograr una mayor abstracción para hallar un modelo más simplificado.
2. Definimos los diagramas de flujo de los procesos más complejos como interacción cliente - agente especial (intermediario entre el banco y el mafioso) y el proceso completo de lavado de dinero.
3. Para la interfaz gráfica, elegimos implementar **Lanterna** que es una biblioteca de Java que permite crear interfaces de usuario basadas en texto y se utilizan en entornos de terminal o consola.

Etapa III: Desarrollo del proyecto basado en el modelo establecido en el diagrama de clases.

1. A partir del diagrama de clases, implementamos las clases principales del banco: cliente, operación, empleado así como las herencias y polimorfismos correspondientes
2. Una vez que teníamos las clases bases definidas, empezamos a desarrollar el flujo de los distintos procesos del sistema del banco.
3. Agregamos nuevas clases e interfaces que no estaban previstas en el modelo pero que eran necesarias para poder lograr que el sistema llevará a cabo cada uno de los procesos del banco, así como también agregamos nuevos métodos y atributos a algunas clases que ya se encontraban definidas en el modelo, todo esto teniendo en cuenta los principios del paradigma.
4. Todo lo anterior se realizó a la par con la interfaz gráfica para que pueda haber una coherencia entre lo que recibe el usuario por consola y cada uno de los procesos del banco.

Etapas IV: Implementamos la simulación del tiempo en el sistema, actualizamos el diagrama de clases e implementamos opciones de ir al banco para que los clientes puedan hacer depósitos y retiros.

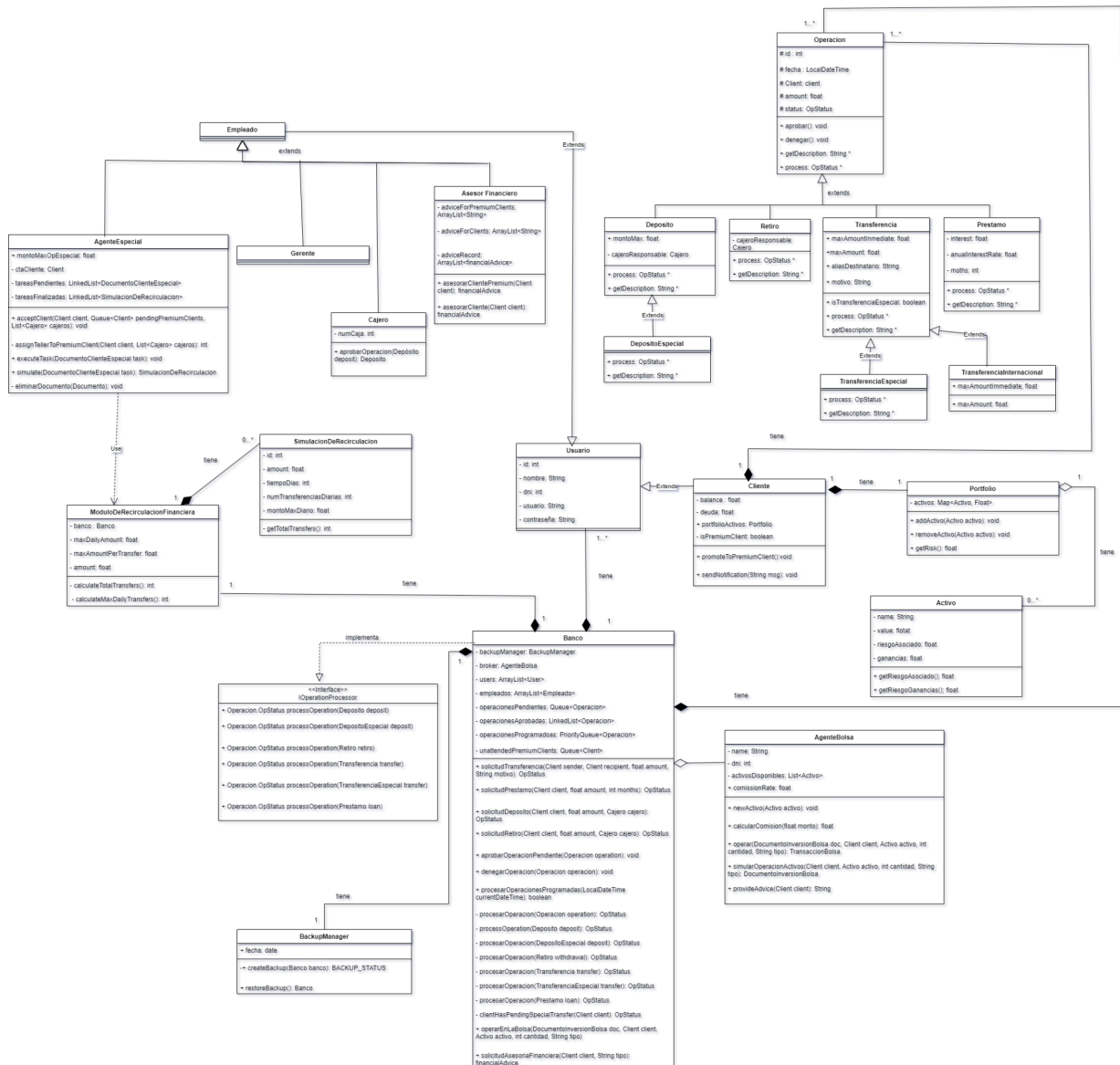
Herramientas utilizadas

Las herramientas adicionales utilizadas para el desarrollo del proyecto fueron las siguientes:

Java (lenguaje de programación estático y fuertemente tipado, versión 11), **intelij** y **vscode** (IDEs), **git** y **github** (para control de versiones y trabajo en equipo), **Maven** (herramienta para gestionar proyectos en Java que facilita la compilación, pruebas y manejo de dependencias por medio del archivo pom.xml). Se desarrolló la aplicación en el sistema operativo **Windows 11** bajo el entorno de emulación de Linux **WSL (Windows Subsystem for Linux)**.

Modelado

A continuación se presenta el Diagrama de Clase utilizado para modelar el sistema del banco:



Decisiones tomadas en el modelado

1. Partimos de una clase “Banco” como clase principal del desarrollo, ya que nos ayudaría a mantener la jerarquía de la entidad y como consecuencia llevar un control de todas las actividades que se realicen, lo que a su vez facilita el proceso de recolectar toda la información para mostrar en el estado del banco, por lo que todas las relaciones del diagrama de clase están en torno al banco.
2. Teniendo en cuenta que tanto los clientes como los empleados, independientemente de que tuvieran roles diferentes, compartían características en común, creamos una clase usuario de la cual heredarán las clases empleado y cliente.
3. Utilizamos polimorfismo para representar los diferentes tipos de empleados dentro del

banco (gerente, cajero y asesor financiero), porque si bien todos son empleados y tienen métodos en común, cada uno tiene sus responsabilidades bien definidas.

4. Para la clase Operacion tambien usamos polimorfismo para definir las distintas operaciones (transferencias, préstamos, retiros, depósitos) y además implementamos los métodos abstractos process() el cual procesa la operación y nos proporciona el estado de la misma y getDescription() que nos da una breve descripción de la operación para saber que proceso tiene que seguir dentro del banco. Estos métodos son abstractos porque se implementan diferente para cada clase que hereda de operación por ejemplo:

```
public Retiro(LocalDateTime fecha, Client client, float monto, Cajero cajeroResponsable) { 1 usage
    super(fecha, client, monto);

    this.cajeroResponsable = cajeroResponsable;
}

@Override
public String getDescription() {
    return "Realizado en sucursal";
}

@Override
public OpStatus process(IOperationProcessor processor) {
    return processor.processOperation( retiro: this);
}
```

```
public Transferencia(LocalDateTime fecha, Client client, Client recipient, float monto, String motivo)
{ 3 usages
    super(fecha, client, monto);
    this.recipient = recipient;
    this.motive = motivo;
}

@Override 2 overrides
public String getDescription() {
    return "Ordenante: " + getClient().getNombre()+ "\nBeneficiario: " + recipient.getNombre();
}

@Override 2 overrides
public OpStatus process(IOperationProcessor processor) { return processor.processOperation( transfer: this); }
```

5. Implementamos la interfaz IOperationProcessor en la cual se definen los métodos para procesar cada una de las operaciones que se realizan en banco. Estas tienen diferentes flujos de procesos y condiciones de aprobación.

```

public interface IOperationProcessor extends Serializable { 1implementation
    public Operacion.OpStatus processOperation(Deposito deposit); 1usage 1implementation
    public Operacion.OpStatus processOperation(DepositoEspecial deposit); 1usage 1implementation
    public Operacion.OpStatus processOperation(Retiro retiro); 1usage 1implementation
    public Operacion.OpStatus processOperation(Transferencia transfer); 1usage 1implementation
    public Operacion.OpStatus processOperation(TransferenciaEspecial transfer); 1usage 1implementation
    public Operacion.OpStatus processOperation(TransferenciaInternacional transfer); 1usage 1implementation
    public Operacion.OpStatus processOperation(Prestamo loan); 1usage 1implementation
}

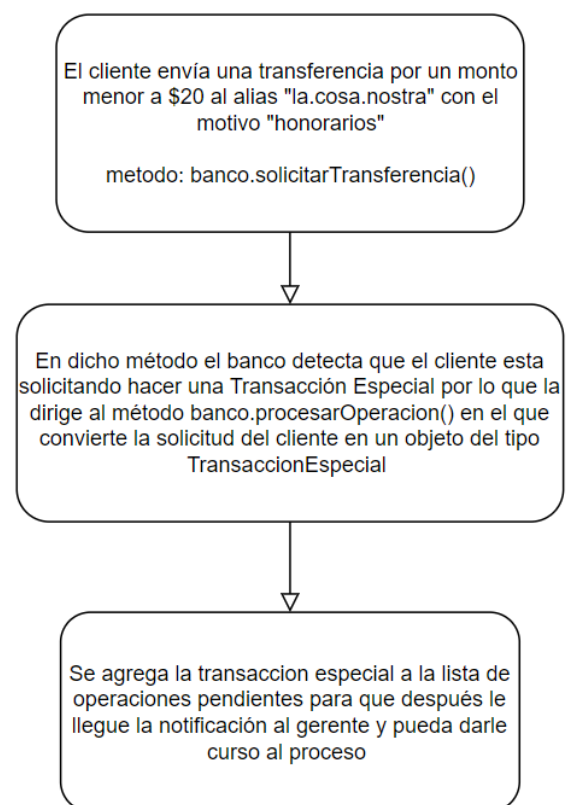
```

6. Para el módulo de lavado de dinero definimos la clase MóduloDeRecirculacionFinanciera el cual hace la simulación del lavado de dinero por medio de la clase SimulacionDeRecirculacion, recolecta toda la información correspondiente a la operación incluyendo las indicaciones de cada una de las transacciones que se deben llevar a cabo.
7. Definimos la relación entre el banco y cualquier tipo de usuario, ya sea cliente o empleado, como una relación de composición ya que dentro de nuestro modelo el empleado y el cliente no pueden existir independientemente del banco, lo mismo hicimos con las relaciones banco-Operacion, banco-BackupManager, y banco-ModuloDeRecirculacionFinanciera.
8. La relacion banco-AgenteBolsa, la definimos como una relación de agregación porque según la consigna, el agente de bolsa es un agente externo al banco que brinda asesoría a los clientes sobre cómo invertir en la bolsa por lo que puede existir fuera del banco.

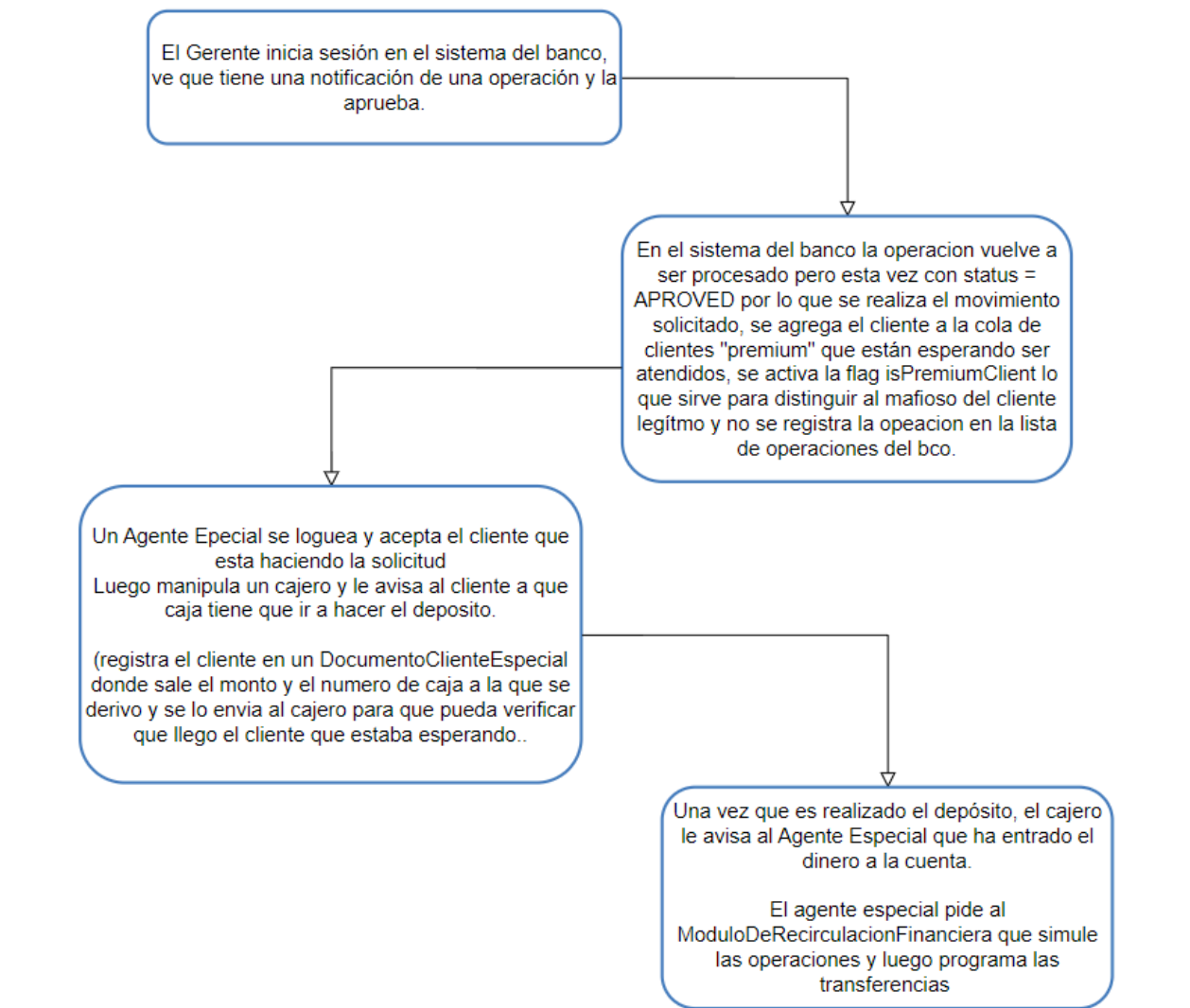
- Diagrama de flujo del proceso de Lavado de Dinero

A continuación se muestran los diagramas de flujo que nos permitieron definir la lógica del proceso de lavado de dinero en el banco.

PARTE I: Este proceso inicia cuando el mafioso le hace saber al banco que quiere hacer una operación ilegítima (lavar dinero) y desea comunicarse con el agente especial (que es el intermediario entre la mafia y el banco).



PARTE II: este proceso inicia desde el momento que el gerente recibe la notificación del banco y finaliza cuando el agente especial recibe los fondos para empezar el proceso de lavado, en el que le solicita al módulo de lavado de dinero que haga una simulación y le envíe un documento con las indicaciones de los movimientos que debe hacer el agente especial para lavar el dinero sin levantar sospechas. Una vez recibida esta información el agente programa las transferencias.



Instrucciones para ejecutar el programa

Se debe tener instalado Java 11, o alguna versión más reciente, en el sistema donde se ejecutará el programa. Para ejecutarlo, hay que abrir una terminal y escribir el comando:

```
java -jar banco-la-familia.jar
```


Si se está utilizando Linux, el programa se ejecutará en la terminal en una interfaz de texto. En Windows, se abrirá una nueva ventana con un emulador de consola.

Es necesario que la terminal tenga el tamaño suficiente para renderizar correctamente la interfaz. Recomendamos que se maximice la ventana de la terminal utilizada para que no tenga problemas.

Instrucciones de uso

Página de inicio

Al iniciar el programa, nos encontraremos en la página de inicio. En la parte superior, se encuentran la fecha y hora simuladas por el programa, que avanzan a un ritmo igual o mayor al real para poder simular los movimientos del banco en el futuro. También se observa la velocidad de la simulación.



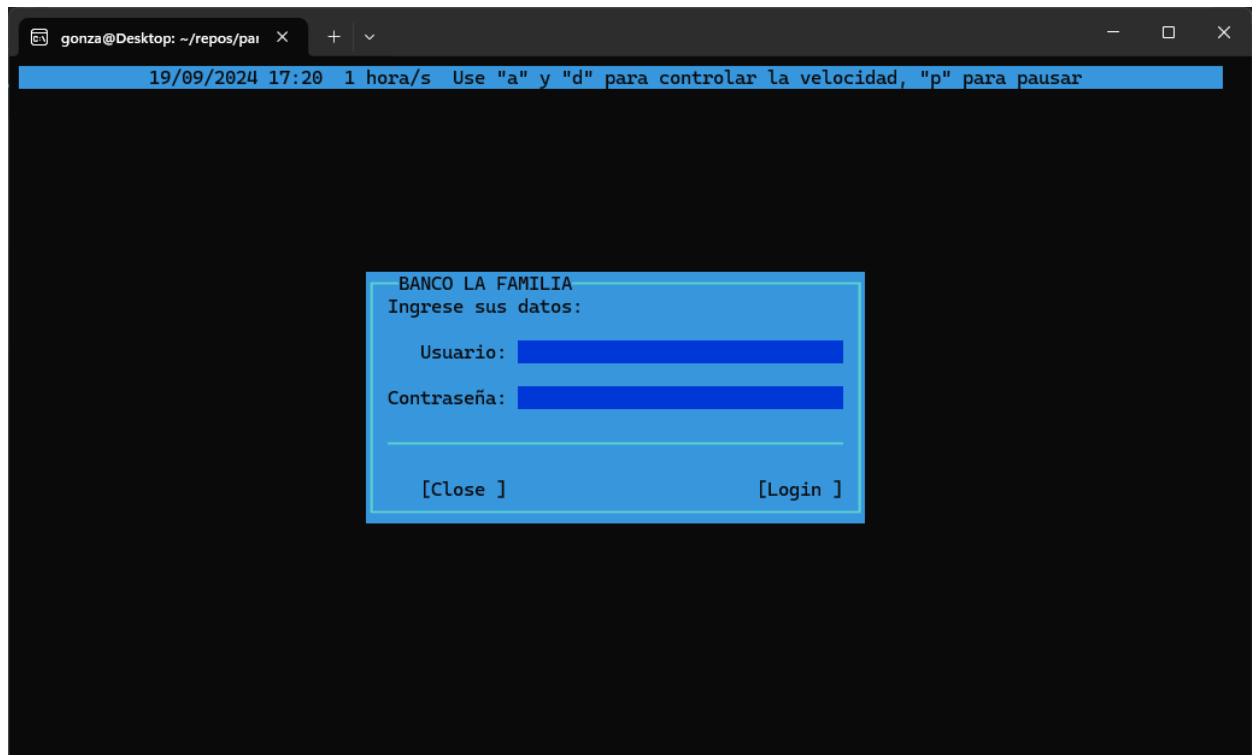
Menú principal

En todo momento, salvo que el cursor se encuentre enfocado sobre un campo de texto, podemos controlar la velocidad de la simulación con las letras “a” y “d”. Las mismas desaceleran y aceleran el tiempo, respectivamente.

En el centro de la pantalla, se encuentran las distintas opciones que podemos tomar.

1. “Iniciar Sesión”. Nos permite acceder al sistema de login del banco para poder realizar operaciones, ya sea tomando el rol de un cliente, o de alguno de los distintos tipos de empleados.
2. “Ir al Banco”. Mediante esta opción tomamos control de uno de los clientes para ir a realizar retiros o depósitos.
3. “Ver Estado del Banco”. Sirve para observar el estado actual del banco (reservas, depósitos, operaciones realizadas, etc.).
4. “Copia de Seguridad”. Esta opción permite crear y restaurar las copias de seguridad del banco
5. “Simular Movimientos”. Programa una serie de operaciones que irán siendo procesadas por el banco a medida que avance la simulación del tiempo.
6. “Salir”. Salir del programa.

Página de inicio de sesión



Página de inicio de sesión

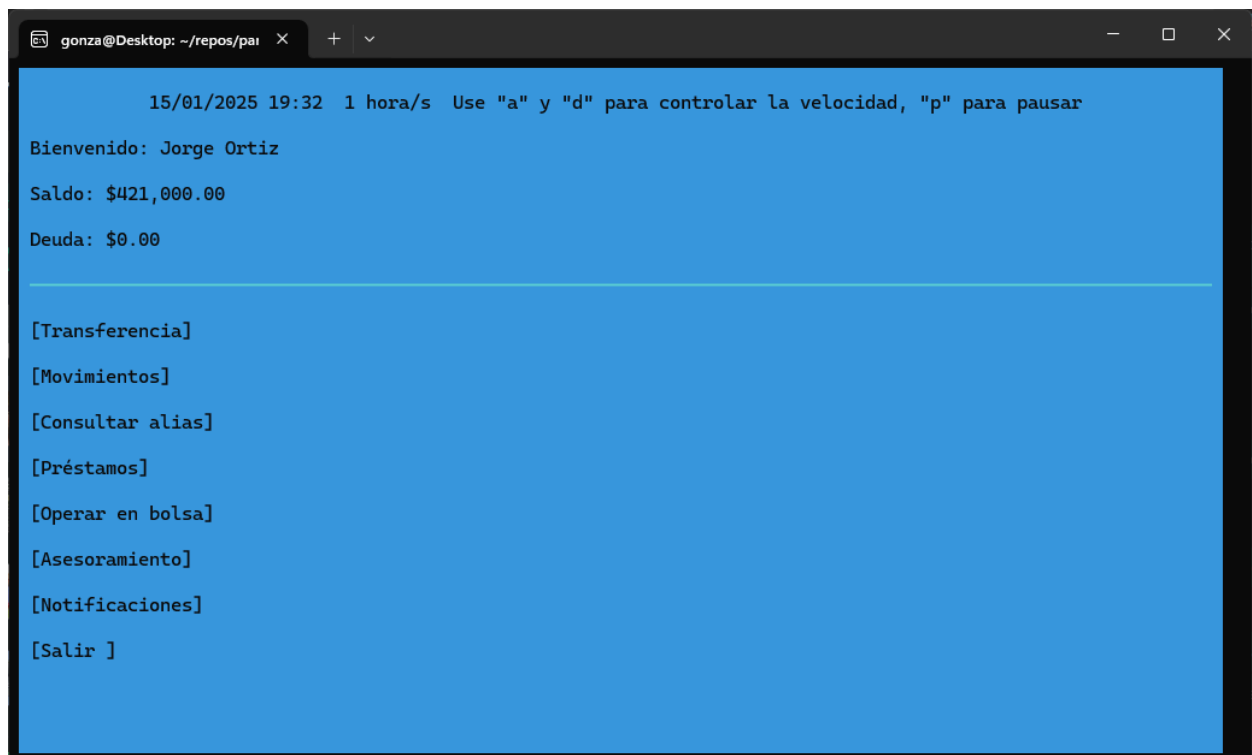
En esta página, debemos ingresar el usuario y contraseña de un usuario registrado en el sistema. El sistema trae varios usuarios pre cargados (ver Anexo) para poder experimentar con

los distintos tipos de operaciones.

Dependiendo de qué credenciales se ingresen, si las mismas son correctas se accederá al menú de un cliente, gerente, asesor financiero o agente especial.

Menú del cliente

En el menú del cliente podemos ver su nombre, saldo y deuda. Además se pueden realizar transferencias, siempre que se sepa el alias de la cuenta de destino (ver Anexo), y se tengan los fondos suficientes.

A screenshot of a terminal window titled 'gonza@Desktop: ~/repos/pai'. The terminal output shows a timestamp '15/01/2025 19:32', a speed indicator '1 hora/s', and instructions 'Use "a" y "d" para controlar la velocidad, "p" para pausar'. Below this, it says 'Bienvenido: Jorge Ortiz', 'Saldo: \$421,000.00', and 'Deuda: \$0.00'. A horizontal line separates the header from a list of menu options: '[Transferencia]', '[Movimientos]', '[Consultar alias]', '[Préstamos]', '[Operar en bolsa]', '[Asesoramiento]', '[Notificaciones]', and '[Salir]'.

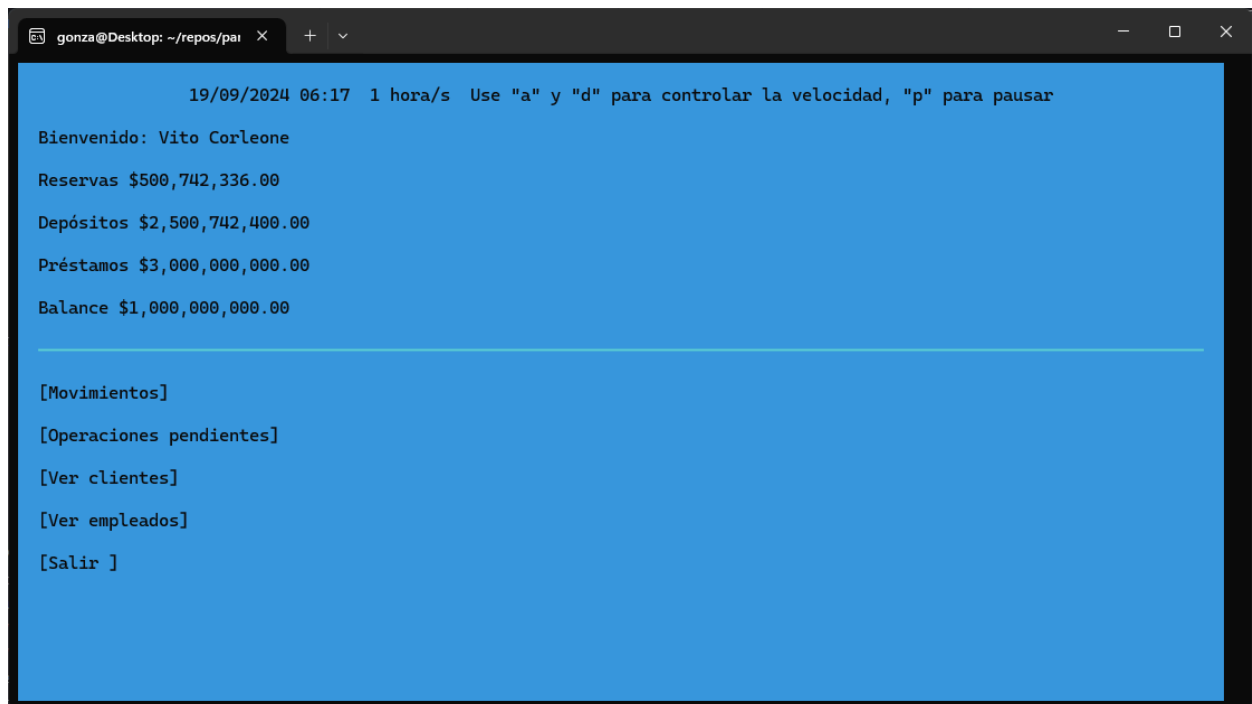
Menú del cliente

Se pueden consultar los movimientos que involucran a la cuenta y el alias de la misma. También se pueden solicitar préstamos, cuya disponibilidad está sujeta a las finanzas del cliente, y del banco (el banco trata en todo momento cubrir cierto porcentaje de los depósitos con sus reservas).

También se puede operar en la bolsa, pudiéndose comprar y vender distintos tipos de acciones. La opción de asesoramiento, ofrece consejos precargados. Finalmente, la opción de notificaciones juega un papel crucial en el proceso de lavado de dinero.

Menú del gerente

El menú de un gerente, muestra el estado actual del banco (reservas, depósitos y préstamos otorgados), así como también su balance, que viene de sumar activos y restar los pasivos. Con el botón movimientos, el gerente puede observar todos los movimientos del banco, excepto algunos que nunca se registran. También puede ver la lista de clientes y empleados, con los datos de cada uno.

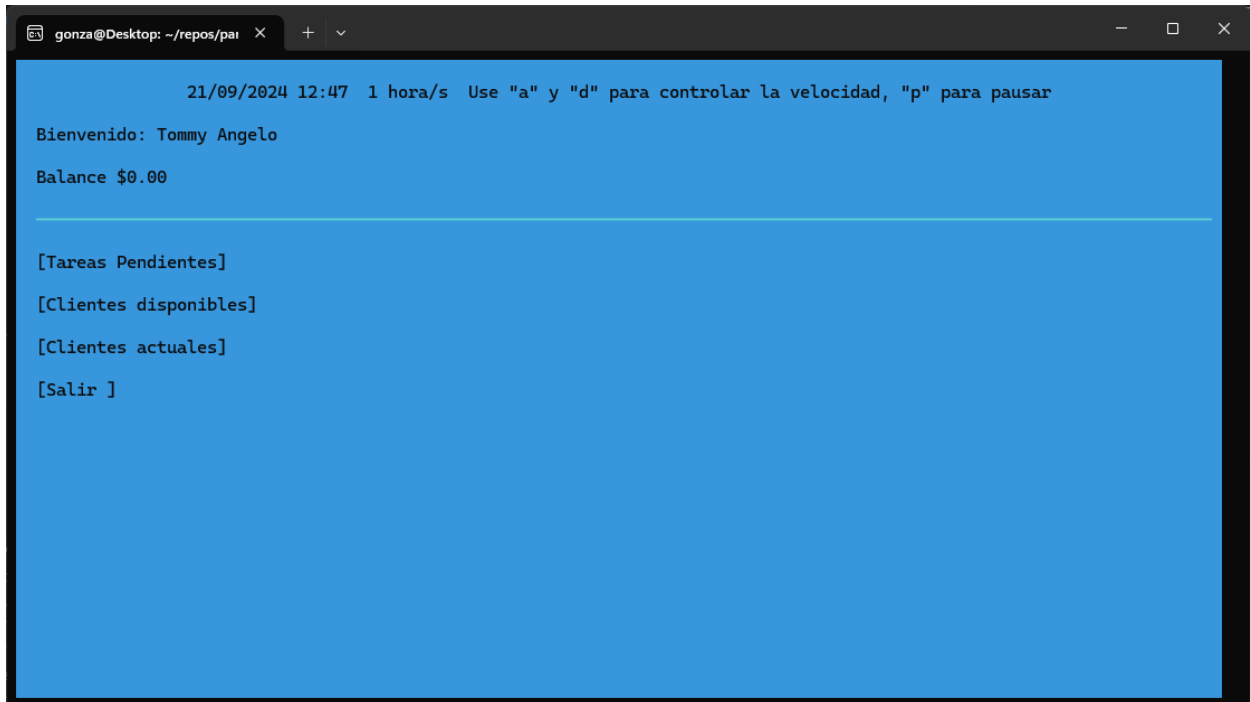
A screenshot of a terminal window with a blue background. The window title is 'gonza@Desktop: ~/repos/pai'. The terminal output shows a date and time '19/09/2024 06:17', a speed indicator '1 hora/s', and instructions 'Use "a" y "d" para controlar la velocidad, "p" para pausar'. Below this, it says 'Bienvenido: Vito Corleone'. Then, it lists financial data: 'Reservas \$500,742,336.00', 'Depósitos \$2,500,742,400.00', 'Préstamos \$3,000,000,000.00', and 'Balance \$1,000,000,000.00'. A horizontal line separates this from a list of menu options: '[Movimientos]', '[Operaciones pendientes]', '[Ver clientes]', '[Ver empleados]', and '[Salir]'.

Menú del gerente

Principalmente, puede aceptar transacciones que hayan quedado pendientes de aprobación, como las transacciones grandes y las “especiales” involucradas en el lavado de dinero.

Menú del agente especial

El agente especial posee un balance, que utilizará para lavar dinero. También puede ver la lista de “tareas pendientes”, que no son más que fondos a lavar de distintos clientes. La sección “clientes disponibles”, tiene una lista de clientes “premium” a los cuáles no se les ha asignado aún un agente especial. Por último, la sección “clientes actuales”, muestra los clientes asignados al agente especial en particular.



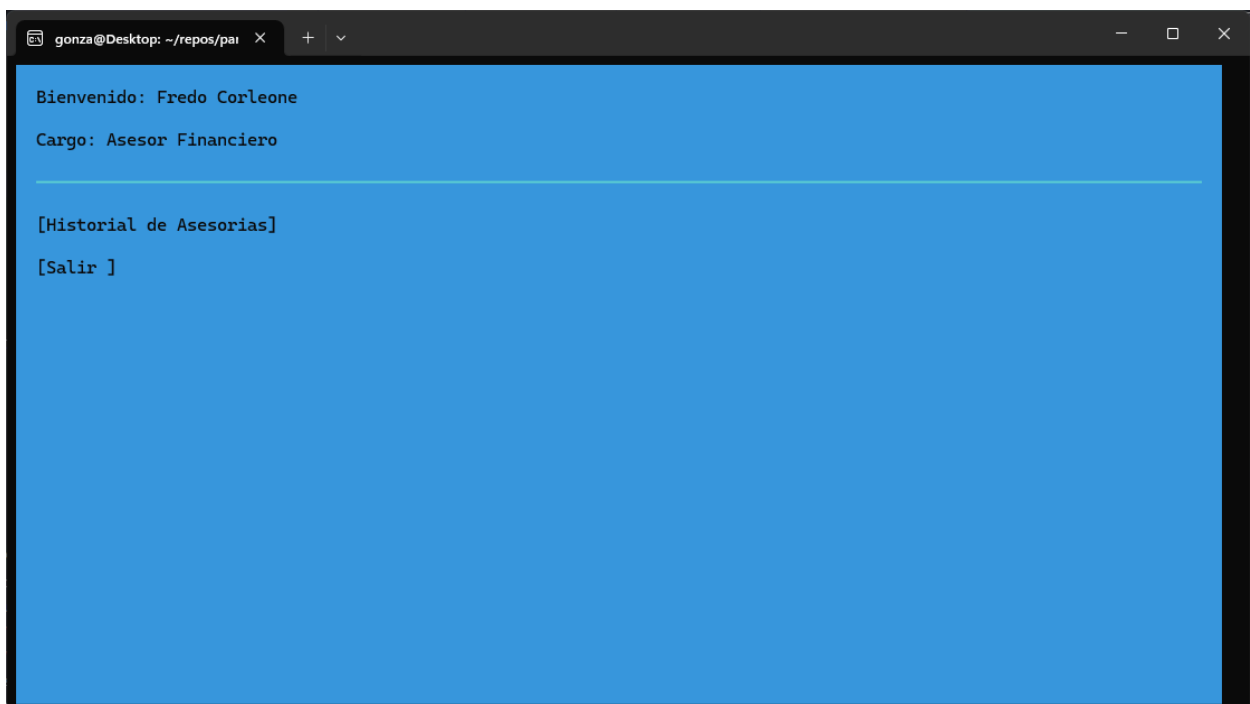
```
gonza@Desktop: ~/repos/pai X + v
21/09/2024 12:47 1 hora/s Use "a" y "d" para controlar la velocidad, "p" para pausar
Bienvenido: Tommy Angelo
Balance $0.00

[Tareas Pendientes]
[Clientes disponibles]
[Clientes actuales]
[Salir ]
```

Menú del agente especial

Menú del asesor financiero

Cómo las tareas del asesor financiero están automatizadas, en este menú solamente podemos ver todas las ocasiones en las que los distintos clientes solicitaron consejos.



```
gonza@Desktop: ~/repos/pai X + v
Bienvenido: Fredo Corleone
Cargo: Asesor Financiero

[Historial de Asesorias]
[Salir ]
```

Menú del asesor financiero

Opción “Ir al Banco”

Al seleccionar esta opción, nos preguntará ¿cuál de los clientes será el que irá al banco? Al seleccionar uno de los clientes, tendremos que elegir una caja en particular, y finalmente el tipo de operación (retiro o depósito) y el monto correspondiente.

Opción “Ver Estado del Banco”

Es muy similar a la del gerente, mostrándonos todos los datos del banco, pero muestra también en tiempo real las operaciones que se están realizando (normalmente ninguna, para ello hay que tomar otras acciones que se explicarán más adelante).

```

gonza@Desktop: ~/repos/pai
20/10/2024 14:53 1 hora/s Use "a" y "d" para controlar la velocidad, "p" para pausar

ESTADO ACTUAL BANCO LA FAMILIA

Reservas $500,742,336.00
Depósitos $2,500,742,400.00
Préstamos $3,000,000,000.00
Balance $1,000,000,000.00

[Ver clientes]
[Ver empleados]
[Salir ]

ID Fecha Hora Tipo Cliente Monto Detalle
8 17/09/2024 10:41 Deposito bancofamilia.banco.Deposito $124,587.00 Realizado en caja Nro. 1
7 17/09/2024 10:41 Deposito bancofamilia.banco.Deposito $121,452.00 Realizado en caja Nro. 1
6 17/09/2024 10:41 Deposito bancofamilia.banco.Deposito $45,872.00 Realizado en caja Nro. 1
5 17/09/2024 10:41 Deposito bancofamilia.banco.Deposito $12,405.00 Realizado en caja Nro. 1
3 17/09/2024 10:41 Deposito bancofamilia.banco.Deposito $4,587.00 Realizado en caja Nro. 1
2 17/09/2024 10:41 Deposito bancofamilia.banco.Deposito $12,452.00 Realizado en caja Nro. 1
1 17/09/2024 10:41 Deposito bancofamilia.banco.Deposito $421,000.00 Realizado en caja Nro. 1
  
```

Menú “Estado del Banco”

Opción “Copia de Seguridad”

Esta opción nos permite guardar el estado actual del banco, para luego poder restaurarlo desde la última copia realizada.

Opción “Simular Movimientos”

Con esta opción, podemos simular cierta cantidad de operaciones que se realizarán de forma automática, y podremos ver como cambia el estado del banco en tiempo real tanto en los menús de los usuarios (clientes, gerentes, etc.) como en la página de “Estado del Banco”.

Simulará al menos un lavado de dinero, que se verá reflejado en la página de “Estado del

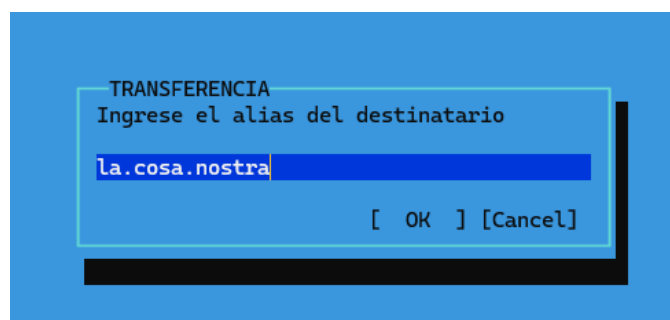
Banco”, como muchas “transferencias internacionales”, bajo el concepto de “dividendos”, provenientes de “Corleone S.A.” y dirigidas al cliente mafioso.

Al realizar la simulación, se nos redirigirá a la página “Estado del Banco” para observar lo que ocurre. Es conveniente ajustar la velocidad de la simulación a una deseada, si es que las operaciones ocurren demasiado rápido o lento.

Ejemplo de lavado de plata de forma manual

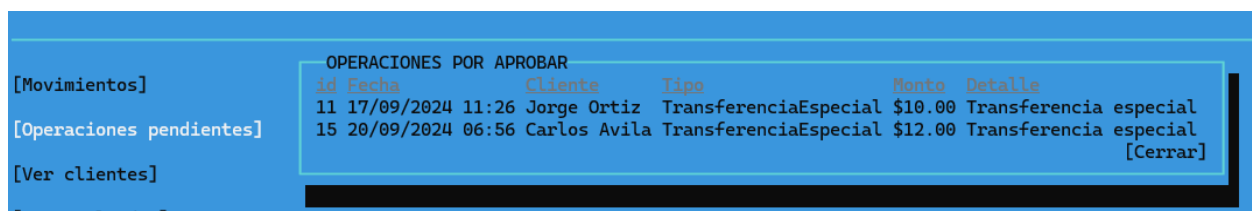
Para utilizar el sistema de lavado de dinero, de principio a fin, debemos elegir a un cliente en particular que todavía no sea considerado como cliente “premium” (ver Anexo), e iniciar sesión en el sistema con sus credenciales.

Una vez dentro, enviaremos una transferencia muy específica. La misma debe estar dirigida al alias “la.cosa.nostra”, con motivo/descripción “honorarios”, y un monto menor a \$20. Una vez hecho esto, se nos indicará que esta transacción “especial” quedó pendiente de aprobación.



Envío de la transacción especial

Luego, elegimos a un gerente e iniciamos sesión con sus credenciales. En la sección de operaciones pendientes, aparecerá nuestra transacción especial, la cual debemos aprobar para continuar.



El gerente aprueba la transacción

Una vez hecho esto, el cliente se considera premium, por aún no está asignado a un agente especial. Hay que loguearse al sistema con las credenciales de uno de ellos, y en la sección de clientes pendientes, aceptar al que nos interesa. Esto enviará automáticamente una notificación al cliente, diciéndole que utilice cierta caja para realizar su depósito “especial”.

[Clientes disponibles]
[Clientes actuales]
[Salir]

CLIENTES DISPONIBLES						
Id	Nombre	DNI	Usuario	Alias	Balance	Deuda
18	Armando Perez	54,231,564	user1	alias.user.uno	\$12,452.00	\$0.00

[Cerrar]

Cliente aparece como disponible en el menú del agente y lo acepta.

<p>[Operar en bolsa] [Asesoramiento] ▶Notificaciones◀ [Salir]</p>	<p>Notificaciones Utilice la caja Nro. 3 [Cerrar]</p>
--------------------------------------------------------------------------------	---------------------------------------------------------------

Notificación en el menú del cliente

Una vez que revisamos la notificación en la cuenta del cliente, vamos a realizar el depósito en esa caja. Normalmente, solo podríamos ingresar una pequeña cantidad de dinero, pero si se realizó correctamente el proceso, ahora deberíamos poder ingresar cientos de millones.

TAREAS PENDIENTES

id	Cliente	Monto
5	Carlos Avila	\$200,000,000.00

[Cerrar]

CONFIRMAR REALIZAR TAREA

Desea realizar la tarea?

Detalles:

Monto a procesar: \$200,000,000.00

Nro. de transacciones: 400

Nro. de transacciones diarias: 10

Plazo total en días: 40

[No] [Yes]

Aparece la tarea en el menú del agente especial

Una vez hecho esto, el dinero no estará disponible en la cuenta del cliente, ni tampoco aparecerá en las reservas o depósitos del banco. Sin embargo, al agente especial asignado al cliente le aparecerá una nueva “tarea” a realizar. En su menú, buscamos esta tarea. Al seleccionarla, nos dará información acerca del proceso que se llevará a cabo para lavar el dinero. Finalmente, si aceptamos, el dinero comenzará a lavarse de forma automática durante los siguientes días.

ESTADO ACTUAL BANCO LA FAMILIA

Reservas \$502,442,336.00

Depósitos \$2,502,217,216.00

Préstamos \$3,000,000,000.00

Balance \$1,000,225,024.00

[Ver clientes]	ID	Fecha	Tipo	Cliente	Monto	Det
[Ver empleados]	27	29/09/2024	11:25	TransferenciaInternacional	bancolafamilia.banco.TransferenciaInternacional	\$500,000.00
[Salir]	22	29/09/2024	06:37	TransferenciaInternacional	bancolafamilia.banco.TransferenciaInternacional	\$500,000.00
	20	29/09/2024	04:13	TransferenciaInternacional	bancolafamilia.banco.TransferenciaInternacional	\$500,000.00
	17	23/09/2024	08:19	Deposito	bancolafamilia.banco.Deposito	\$200,000.00
	8	17/09/2024	11:31	Deposito	bancolafamilia.banco.Deposito	\$124,587.00
	7	17/09/2024	11:31	Deposito	bancolafamilia.banco.Deposito	\$121,452.00
	6	17/09/2024	11:31	Deposito	bancolafamilia.banco.Deposito	\$45,872.00
	5	17/09/2024	11:31	Deposito	bancolafamilia.banco.Deposito	\$12,405.00
	3	17/09/2024	11:31	Deposito	bancolafamilia.banco.Deposito	\$4,587.00
	2	17/09/2024	11:31	Deposito	bancolafamilia.banco.Deposito	\$12,452.00
	1	17/09/2024	11:31	Deposito	bancolafamilia.banco.Deposito	\$421,000.00

Comienza el lavado de dinero

Podemos monitorear el proceso desde cualquiera de las páginas relevantes, y veremos como múltiples transacciones provenientes de “Corleone S.A.” empiezan a llegar al cliente bajo el concepto de “Dividendos”.

3. Conclusión

En conclusión, los resultados obtenidos cumplen con la consigna del problema y los objetivos de desarrollo haciendo uso del Paradigma de Programación Orientado a Objetos, el cual nos permitió inicialmente modelar el problema (a pesar de su grado de complejidad) de una forma más simple y limpia gracias a los principios de abstracción, herencia, polimorfismo y encapsulamiento, lo que al final fue muy útil cuando afrontamos algunas dificultades con el flujo del proceso de lavado de dinero o algunos casos en los que encontramos que los procesos eran más complejos de lo inicialmente habíamos planteado y gracias a que el paradigma permite ir desarrollando el programa de forma clara y ordenada pudimos agregar nuevas clases, expandir las que ya teníamos y crear nuevas relaciones manteniendo la coherencia en el programa.

4. Anexo

Se deja disponible una tabla con distintos empleados y clientes precargados en el banco, junto con sus credenciales, para poder experimentar las distintas opciones que ofrece el programa

Tipo	Nombre	Usuario	Contraseña	Estado
------	--------	---------	------------	--------

Gerente	Vito Corleone	admin0	admin	-
Gerente	Michael Corleone	admin1	admin	-
Gerente	Tony Soprano	admin2	admin	-
Agente Especial	Tommy Angelo	agente0	admin	-
Agente Especial	Frank Costello	agente1	admin	-
Agente Especial	Vito Scaletta	agente2	admin	-
Asesor Financiero	Fredo Corleone	asesor0	admin	-
Cliente	Jorge Ortiz	user0	1234	Pendiente de aprobación como "Premium"
Cliente	Armando Perez	user1	1234	"Premium" pero sin asignar agente especial
Cliente	Martín Gimenez	user2	1234	"Premium" asignado al A.E. "Tommy Angelo"
Cliente	Pedro García	user3	1243	Cliente normal
Cliente	Carlos Ávila	user4	1324	Cliente normal
Cliente	Sofía Núñez	user5	1234	Cliente normal
Cliente	Milagros Pérez	user6	1234	Cliente normal
Cliente	Juan Agüero	user7	1234	Cliente normal
Cliente	Sol Perez	user8	1234	Cliente normal
Cliente	Agustina Solares	user9	1234	Cliente normal