Memoria Escrita Práctica 2

Programación Orientada a Objetos

Proyecto: Interfaz bancaria **Wunallet**

Grupo 1 Equipo 7

César Augusto Ospina Muñoz
Diego Ospina Ramírez
David Cardona Duque
Daniel Escobar David

Índice

1. Wunallet es un software interbancario inspirado en las aplicaciones bancarias convencionales, como por ejemplo Nequi y ahorro a la mano, pero que a diferencia de estas integra múltiples bancos en su portafolio; lo que permite, entre otras cosas, hacer transferencias entre bancos o solicitudes de crédito a diferentes entidades, desde una misma interfaz.

Esto está logrado a nivel de diseño al describir todo el entorno interbancario mediante 10 clases y una interfaz de gestión. Estas clases, en detalle, son:

- Banco: Cada instancia de esta clase representa uno de los bancos integrados al software, y en sus atributos almacenan toda la información sobre sus productos e identidad.

- Usuario: Como piedra angular del diseño, cada instancia de esta clase representa un usuario. Es su información y operaciones las que dan sentido y cohesión a todas las demás clases, y es el único parámetro que todas las funcionalidades deben garantizar.

- Credito: Son los objetos creados cuando a un usuario se le aprueba su solicitud de crédito. Su función va desde almacenar los detalles del préstamo y a qué banco pertenece, hasta ofrecer simulaciones de crédito.

- Banquero: De forma análoga al caso real, esta clase gestiona la interfaz gráfica, los inputs del usuario, la ejecución de funcionalidades, y los retornos en pantalla.

- PerfilCrediticio: Cada instancia está asociada a un usuario, y describe precisamente el perfil crediticio de éste. Es usada por los bancos para determinar si una solicitud de crédito se aprueba o rechaza.

- Transaccion: Es una clase que sólamente tiene atributos de tal forma que, en conjunto, permiten reconstruir a detalle la información sobre una operación bancaria.

- Cuenta: Como clase abstracta se encarga de dar el cuerpo, a nivel de atributos y métodos, a sus clases hijas: Ahorro, Corriente y BajoMonto. Siempre están asociadas a un banco y usuario.

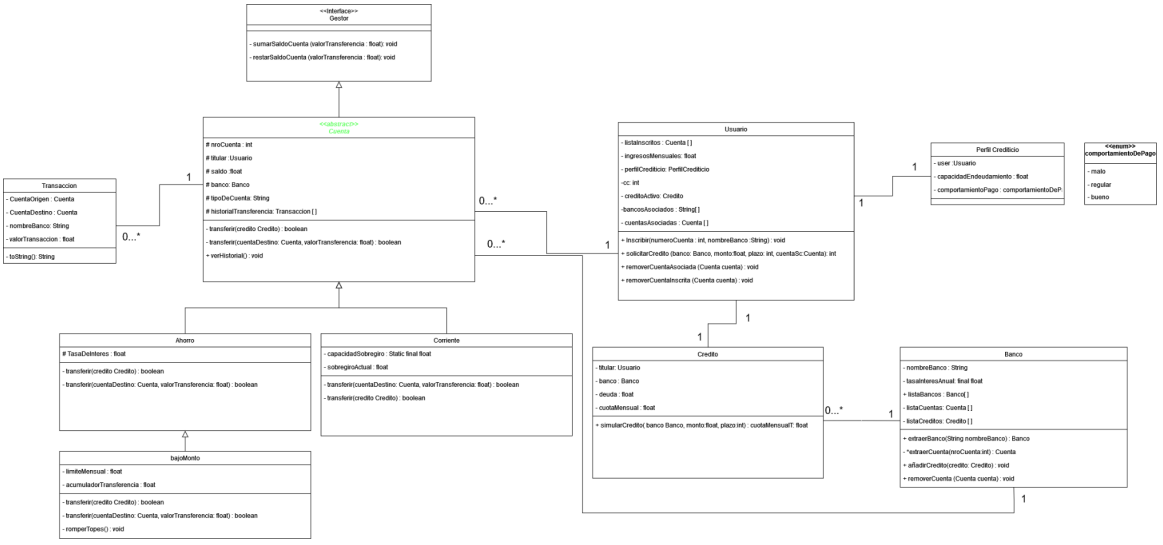
- Corriente: Siendo una clase hija de Cuenta, se diferencia por incluir la capacidad de sobregirarse, de tal forma que el usuario puede realizar transferencias de mayor cantidad que las de su saldo.

- Ahorro: Es el producto financiero más simple, diferenciándose de su clase padre Cuenta al ofrecer una tasa de interés que producirá ingresos en el tiempo.

- BajoMonto: Es el resultado de especializar las cuentas Ahorro, ya que si bien disponen de los mismos atributos, sus métodos están limitados a un tope máximo mensual que se puede transferir.

La implementación de la aplicación se hizo mediante una interfaz simple de consola, donde el usuario puede seleccionar una de las 5 funcionalidades disponibles y dentro de estas elegir entre más opciones que complementan o especifican su funcionamiento, de esta manera el usuario podrá acceder una amplia variedad de operaciones interbancarias desde una misma interfaz gráfica.

2. Imagen del diagrama UML



3.

1. La clase abstracta del diseño corresponde a la clase **Cuenta**, que a su vez contiene dos métodos abstractos y sobrecargados:

   abstract transferir(Cuenta cuentaDestino, float ValorTransferencia)

   abstract transferir(Credito credito)

   Estos métodos ejecutan las comprobaciones y actualizaciones de saldo correspondientes al realizar una transferencia. Este proceso depende del tipo de cuenta que lo ejecute, y por eso debe ser un método abstracto.

2. La interfaz **Gestor** es la encargada de solicitar que se declaren métodos para, precisamente, gestionar los productos bancarios. Es implementada en la clase **Cuenta** ya que sus clases hijas requieren hacer operaciones de ajuste de saldo.

3. Las clases **Corriente** y **Ahorro** heredan de la clase abstracta **Cuenta**, y a su vez **BajoMonto** hereda de **Ahorro**, ya que toda cuenta de BajoMonto es esencialmente una cuenta de Ahorro con limitaciones en sus métodos y atributos.

4. Los métodos sobrecargados **transferir***,* heredados desde la clase **Cuenta**, aplican ligadura dinámica al estar definida su invocación en base a si la instancia es de **Ahorro** o de **BajoMonto**.

   transferir(Cuenta cuentaDestino, float ValorTransferencia)

   transferir(Credito credito)

   Por las limitaciones impuestas a las cuentas de BajoMonto, los métodos deben hacer verificaciones diferentes para realizar las transferencias de pago de crédito y entre cuentas.

5. En la clase **Banco**, el atributo **listaBancos** es un método estático usado para almacenar los bancos que han sido creados y hacen parte del 'portafolio' de la aplicación.
   Adicionalmente, el método **extraerBanco(String nombreBanco)** también de la clase **Banco**, es estático ya que se usa para obtener del atributo anterior el objeto banco a partir de su nombre.

6. La constante *costoRomperTopes* implementada en la clase **BajoMonto** es la que permite verificar si se puede ejecutar o no la funcionalidad **Romper topes** en base al saldo de la cuenta.

7. **Protected:** Es usado en la clase **Cuenta** y **Ahorro** con todos sus atributos, ya que ambas clases heredan y se quería dar accesibilidad a sus clases hijas.

   Private: Es la visibilidad predilecta de los atributos de instancia, y se aplicó en (algunos o todos) los atributos de las clases: **BajoMonto, Corriente, Crédito, Banco, PerfilCrediticio, Transaccion y Usuario.**

   Public: En los atributos está presente en la clase **Banco** ya que por diseño el atributo *listaBancos* debe ser público y estático para determinar el portafolio de la aplicación. Sin embargo en cuanto a los métodos su visibilidad predilecta fue *public* y todos los métodos, en todas las clases, lo implementaron de esta manera.

8. 8.1. Sobrecarga del constructor: La primera sobrecarga del constructor se implementó en el módulo gestorAplicacion dentro del paquete infoClientes, en la clase transaccion, esto debido a que dentro de la lógica de la aplicación se tienen 2 tipos de transacciones posibles, la primera la transacción es de una cuenta bancaria a otra, la cual implica parámetros como la cuenta de destino, y el segundo tipo de transacción es la que se encarga de pagar créditos asociados al usuario, la cual implica parámetros como el banco con el cual se tiene el crédito.

   La segunda sobrecarga de constructor se implemento en el modulo gestorAplicacion dentro del paquete productosFinancieros, en la clase Ahorro, esto debido a que una cuenta de ahorro se crea con historial vacío y cuando se rompen topes se debe heredar el historial de la cuenta pasada.

Sobrecarga de métodos: La sobrecarga de métodos se implementa en el modulo gestorAplicacion, dentro del paquete productosFinancieros en todas las clases que heredan de Cuenta, esta sobrecarga se da mediante el método transferir debido a que dentro de la aplicación existen 2 tipos de transferencias, una que va dirigida a otra cuenta bancaria y por lo tanto necesita parámetros como la cuenta de destino y el otro tipo de transferencia que está enfocado a pagar créditos asociados con el usuario y necesita el parámetro crédito.

8.2. this de desambiguación: Los this que desambiguan se utilizaron en la gran mayoría de constructores de las clases con el objetivo de diferenciar el parámetro pasado al constructor con el atributo de la clase, ahora bien, para cumplir con este requisito se mencionan 2 casos específicos, estos se implementaron en el módulo gestorAplicacion, dentro del paquete productosFinancieros en el constructor de la clase bajoMonto, se desambigua el atributo limiteMensual con el parámetro limiteMensual y el atributo acumuladorTransferencia con el parametro acumuladorTransferencia.

this(): El uso de este this se implementó en el módulo gestorAplicacion, dentro del paquete productosFinancieros, en la clase ahorro en la sobrecarga de métodos, este this se utiliza para setear como un historial vacío con el otro constructor y crear una cuenta de ahorro desde 0.

8.3. Caso de enumeración: El caso de numeración fue implementado en el módulo gestorAplicacion dentro del paquete infoClientes, en la clase de enumeración comportamientoDePago, este enum consiste de los niveles de comportamiento de pago de cada usuario, los cuales están ranqueados desde bueno a malo con números del 1 al 3, se utilizó para tener un nivel de referencia a la hora de solicitar un crédito a una entidad bancaria. Se tiene dentro de esta clase el método comportamientoDePago el cual retorna de manera aleatoria una de las categorías anteriores para así tomar decisiones a la hora de aprobar un crédito.

4.

Inscribir cuenta

Funcionamiento:

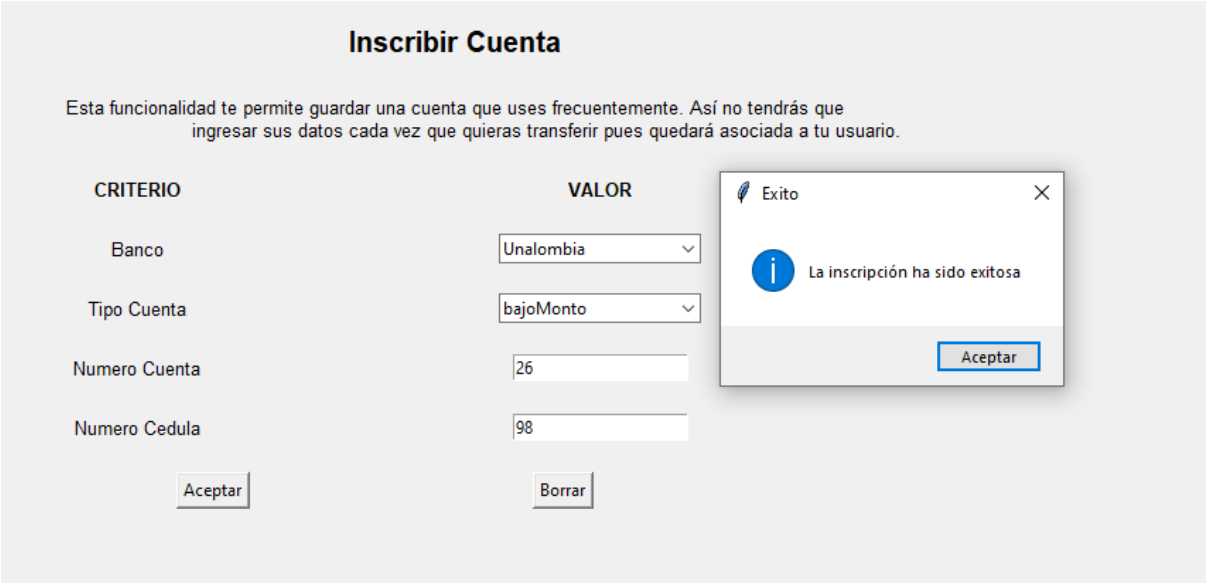
Esta funcionalidad le permite a un usuario inscribir una cuenta a la que transfiere frecuentemente para evitar ingresar los datos en cada ocasión, y, a su vez, aumentar la cantidad de dinero que puede enviar en cada transacción hacia dicha cuenta.

Está relacionada con las clases **Usuario, Cuenta** y sus clases hijas; y **Banco**.

Secuencia:

Una vez seleccionada se le pide al usuario que seleccione e ingrese de qué banco es la cuenta a inscribir, el tipo de cuenta y el número de cuenta así como la cédula del titular de dicha cuenta. En base a esta información se verifica en el **banco** seleccionado que exista una **cuenta** con todos los datos ingresados. De ser así la cuenta es extraída y vinculada al **usuario**.

Captura de pantalla:



Romper topes

Funcionamiento:

Esta funcionalidad es exclusiva de usuarios que cuentan con por lo menos una cuenta de BajoMonto. Su funcionalidad es romper las limitaciones en la cantidad de dinero que se puede usar cada mes con las cuentas de tipo BajoMonto al reemplazar dicha cuenta por una de Ahorro, migrando toda la información y saldos en el proceso.

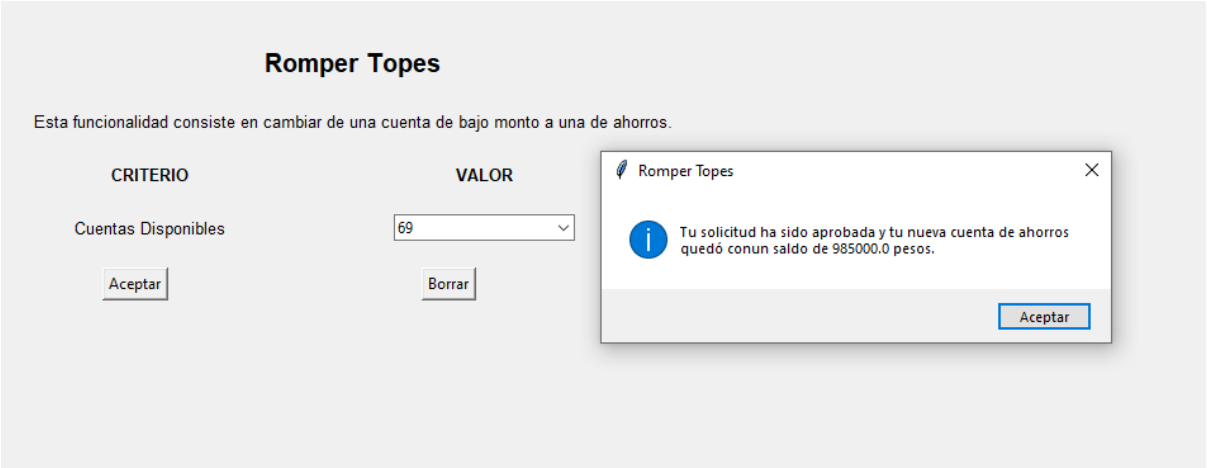
Está relacionada con las clases **Usuario**, **Cuenta**, **BajoMonto**, **Ahorro**, **Gestor (interface)** y **Banco.**

Secuencia:

Una vez escogida la funcionalidad desde la interfaz se realiza una comprobación de que el **usuario** tenga al menos una cuenta de **bajo monto**. De ser así se le indica al usuario en qué consiste el proceso que va a realizar y se le pide que seleccione la **cuenta** objetivo. Tras verificar que la cuenta cumpla las condiciones (con ayuda de la **interface**) para realizar la actualización, se procede a crear una cuenta de **ahorro** que se inicializa con los datos de

la cuenta anterior y finalmente se realizan las limpiezas y desasignaciones pertinentes de la cuenta anterior desde el **banco**.

Captura de pantalla:



Solicitar crédito

Funcionamiento:

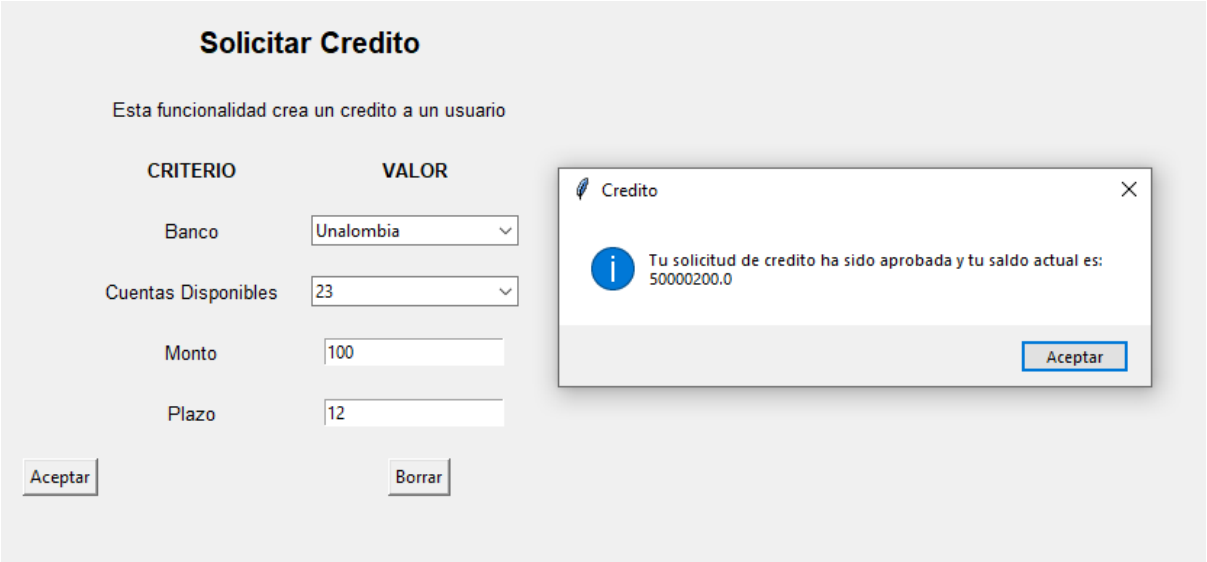
Permite que un usuario solicite un crédito con cualquier banco existente, brindando la oportunidad de escoger el monto y plazo del crédito. La solicitud puede ser aprobada o rechazada, y en caso de aprobación será depositado a la cuenta que el usuario indicó.

La funcionalidad está relacionada con la clase **Usuario**, **Banco**, **Cuenta** y sus hijas, **Credito** y **PerfilCrediticio**.

Secuencia:

Una vez seleccionada la funcionalidad desde la interfaz se realiza una comprobación de que el **usuario** no tenga ningún otro crédito activo. De ser así, procede a seleccionar el **banco** con el que desea adquirir el crédito, escoge la cuenta a la que será depositado en caso de ser aprobado, y por último ingresa el monto y plazo de interés. Con estos datos se procede a ejecutar el método *solicitarCrédito* del usuario, para que éste se encargue de analizar el **perfil crediticio** y la capacidad de endeudamiento. En caso de que el usuario cumpla los requisitos, se concede el **credito** y se hacen los ajustes de saldo apropiados a la **cuenta** seleccionada al inicio del proceso. Por última se registran los cambios hechos y se le notifica al usuario el éxito en la solicitud.

Captura de pantalla:



Ver historial de transacciones

Funcionamiento:

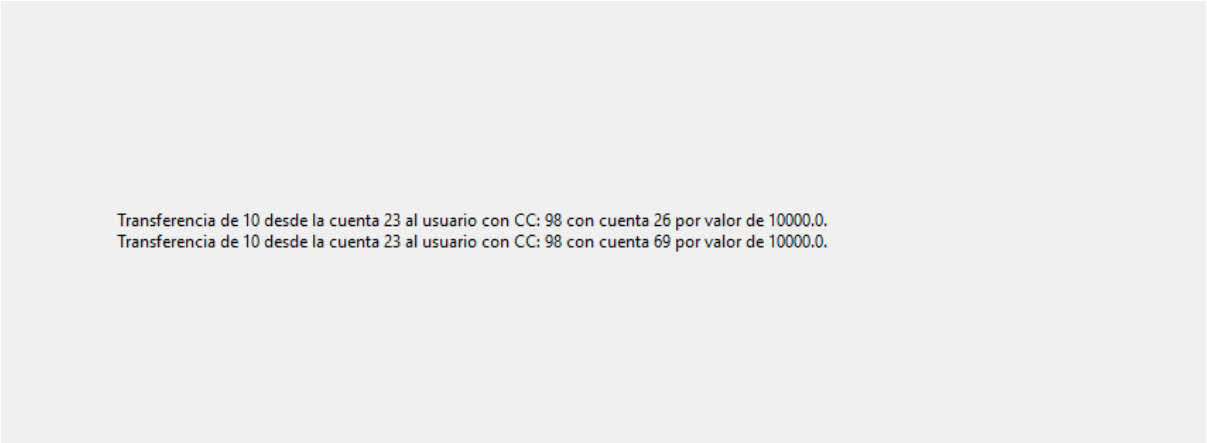
Mediante esta funcionalidad se obtiene el reporte de todas las transacciones realizadas con una cuenta desde el momento de su creación.

Se relaciona con las clases **Usuario, Cuenta** y sus hijas, **Transaccion.**

Secuencia:

En base a los atributos del **usuario** se despliega en la interfaz su lista de **cuentas** asociadas para que se seleccione la cuenta de la que se quiere ver el historial. Posteriormente se accede a los atributos de dicha **cuenta** para extraer cada **transacción** que ha realizado y así, finalmente, desplegar un texto formateado que describa cada transacción.

Captura de pantalla:



Transferir

Funcionamiento:

Con esta funcionalidad se realizan transferencias entre dos cuentas o directamente a un banco en el caso de querer pagar un crédito. Ya que el diseño es interbancario, **NO** es necesario que las transferencias sean entre el mismo banco.

Se ve relacionada con las clases **Usuario**, **Cuenta** y sus hijas, **Banco**, **Transaccion** y **Credito, Gestor (interface)**.

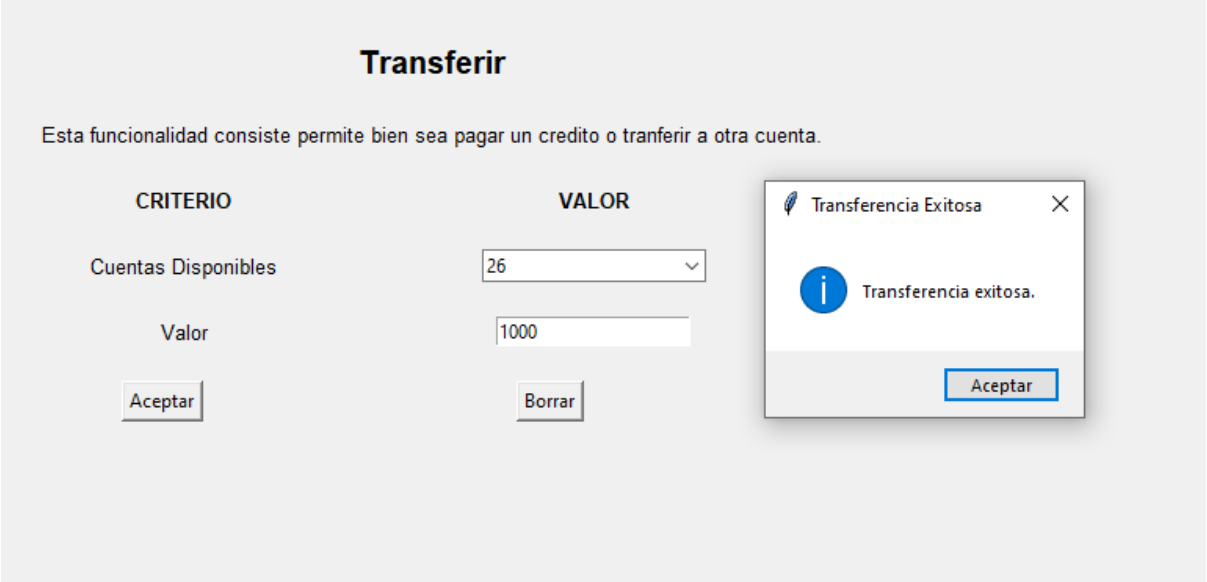
Secuencia:

El **usuario** debe seleccionar la **cuenta de origen** desde la que realizará la transferencia y el tipo de transferencia que hará, de donde se desprenden dos secuencias:

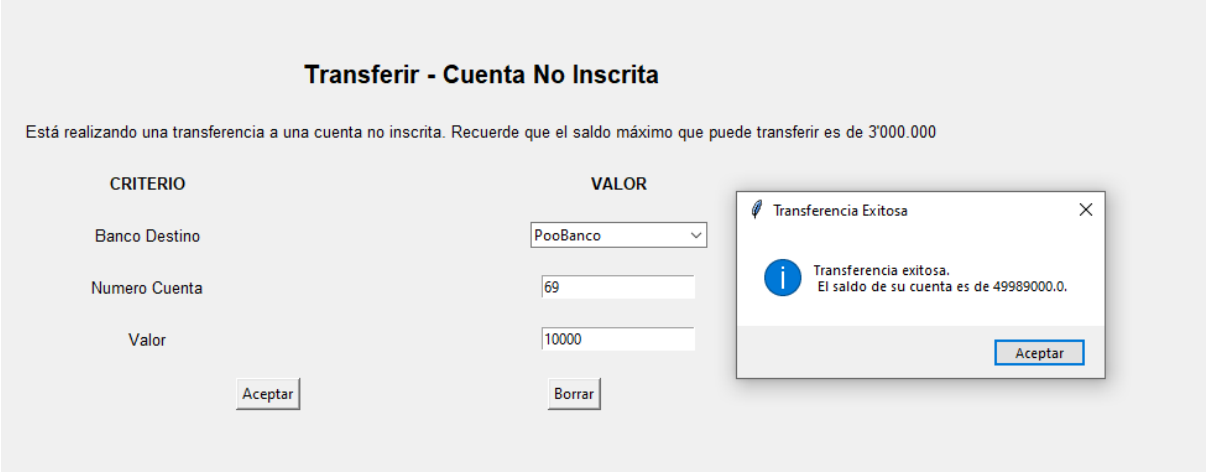
1. Si selecciona pagar **crédito** y cuenta con dicho crédito, se usarán los atributos para verificar la información y notificar al usuario sobre las condiciones de la operación. Si éste decide continuar se creará una **transacción** y mediante los métodos de la **interface** se harán los ajustes a la cuenta origen. Finalmente se hacen los registros pertinentes.

2. En caso de que seleccione transferencia a otra cuenta, se le preguntará si quiere transferir a cuentas inscritas o no inscritas. Si es a no inscritas se le solicitará los datos de identificación de la cuenta destino, e internamente con los datos ingresados se extraerá de **banco** el objeto **cuenta** correspondiente a la cuenta de destino. Tras esto se realizan las verificaciones para hacer los ajustes de saldo en cada cuenta que describen la **transaccion** y finalmente se deja registro en las cuentas involucradas.

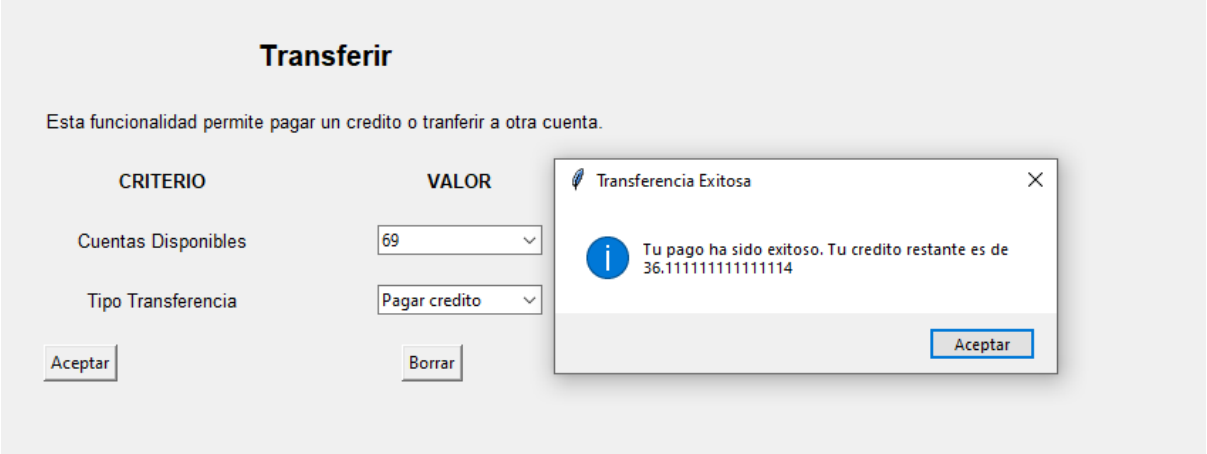
Captura de pantalla: Transferencia - A otra cuenta inscrita



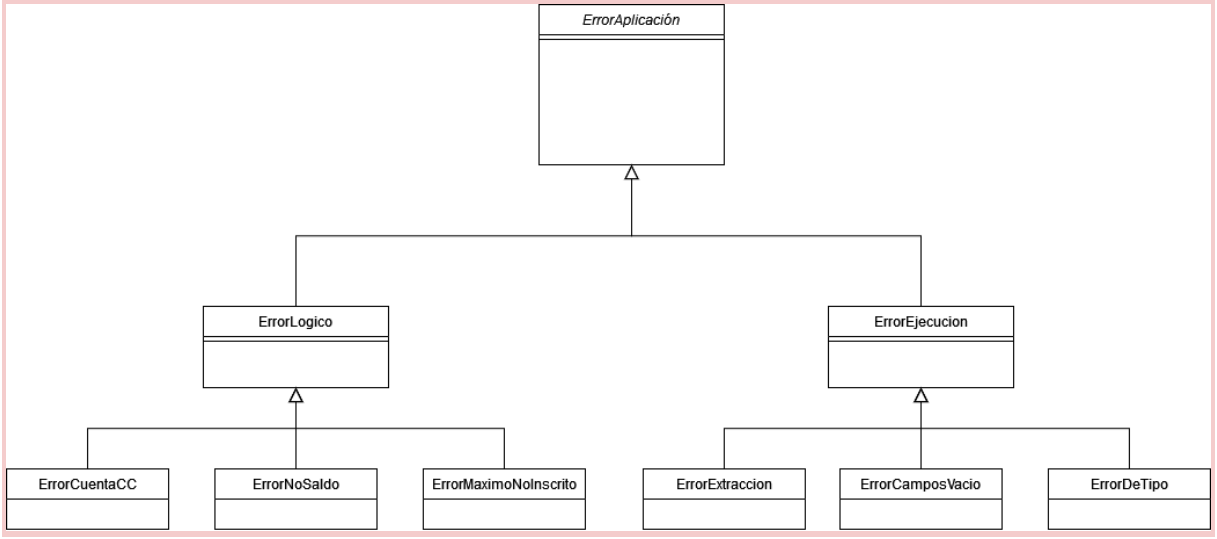
Transferencia - A otra cuenta no inscrita



Transferencia- Pago crédito



5. **Excepciones**



Exception ErrorAplicacion

Esta clase/exception siempre es llamada al ser la clase padre de todas las demás clases Exception definidas en el proyecto. En nuestro sistema su funcionalidad consiste en aportar el título característico "Manejo de errores de la aplicación" a las ventanas de error, así como producir y almacenar los atributos y métodos para el mensaje de error específico que se construye desde sus clases hijas.

Exception ErrorLogico

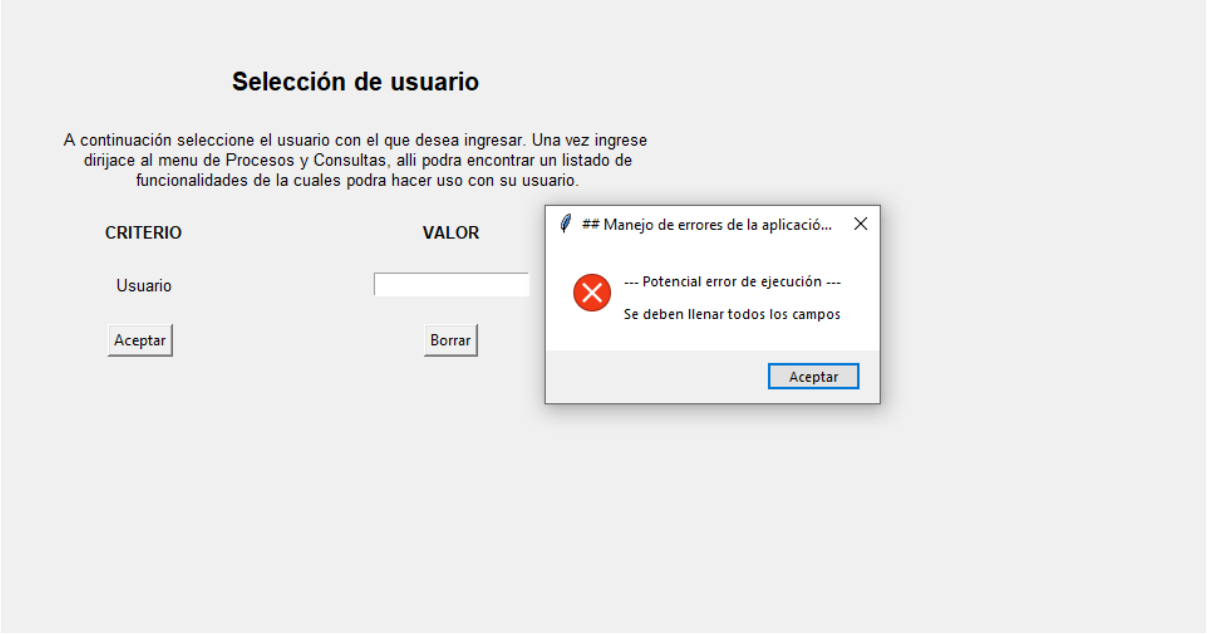
Esta clase/exception es la clase padre de todos los errores que sin ser perjudiciales para la ejecución correcta del programa, suponen un error para la lógica que impone nuestro diseño del sistema bancario.

Exception ErrorEjecucion

Esta clase/exception es la clase padre de todos los errores que producen (o pueden producir más adelante) un error que fuerce a detener la ejecución del programa por errores sintácticos o semánticos.

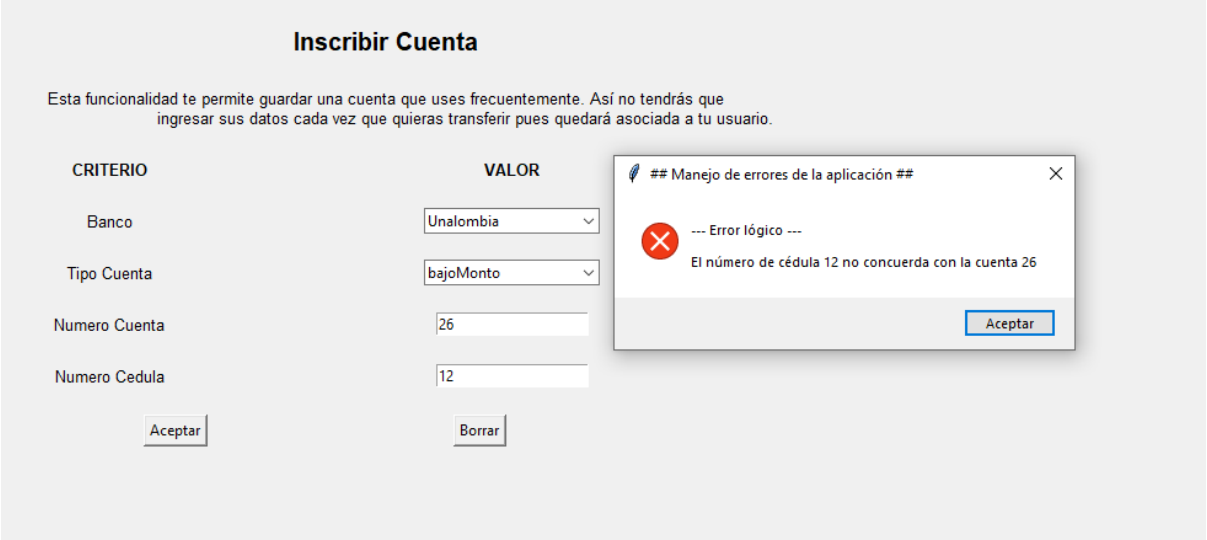
Exception ErrorCamposVacios

Esta clase/exception es lanzada cuando en un formulario el usuario no ingresa los campos solicitados y obligatorios. Es hija de ErrorEjecucion ya que si el usuario no suministra dichos datos, el procesamiento de las funcionalidades procurarían hacer operaciones con objetos que no están determinados o son del tipo incorrecto (None), generando así que el programa, eventualmente, detenga su ejecución.



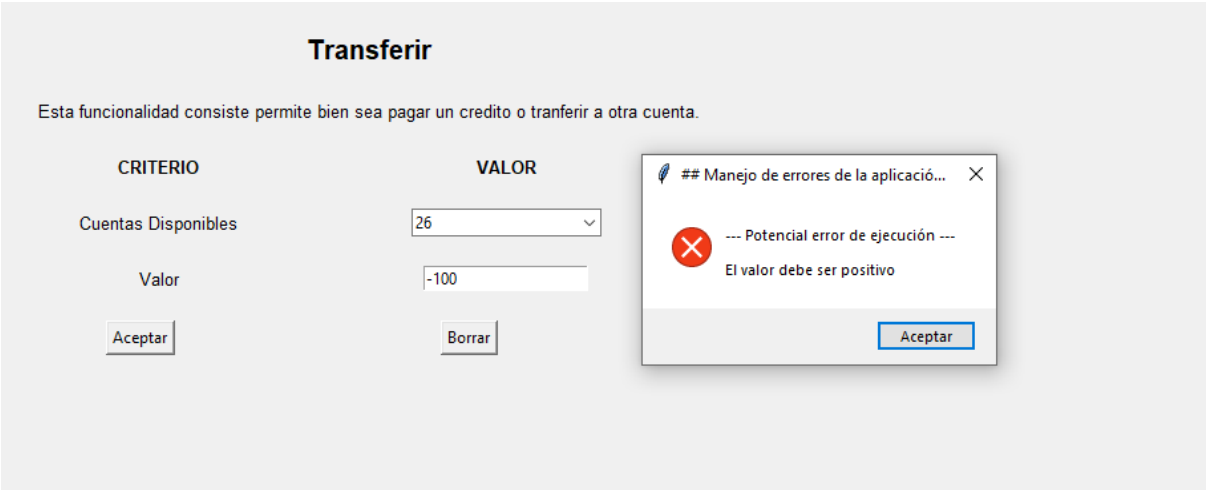
Exception ErrorCuentaCC

Esta clase/exception es lanzada cuando no se puede realizar una verificación exitosa de la identidad de un usuario (mediante su cédula), con la cuenta que se piensa que tiene dicho usuario, en la funcionalidad de Inscripción. Extiende de ErrorLógico ya que si bien se podría realizar la inscripción del objeto cuenta al usuario, sería indebido desde el diseño que un usuario inscribiera una cuenta (obteniendo transferencias inmediatas que no piden formularios de datos), si la información del destino no es acertada y consistente.



Exception ErrorDeTipo

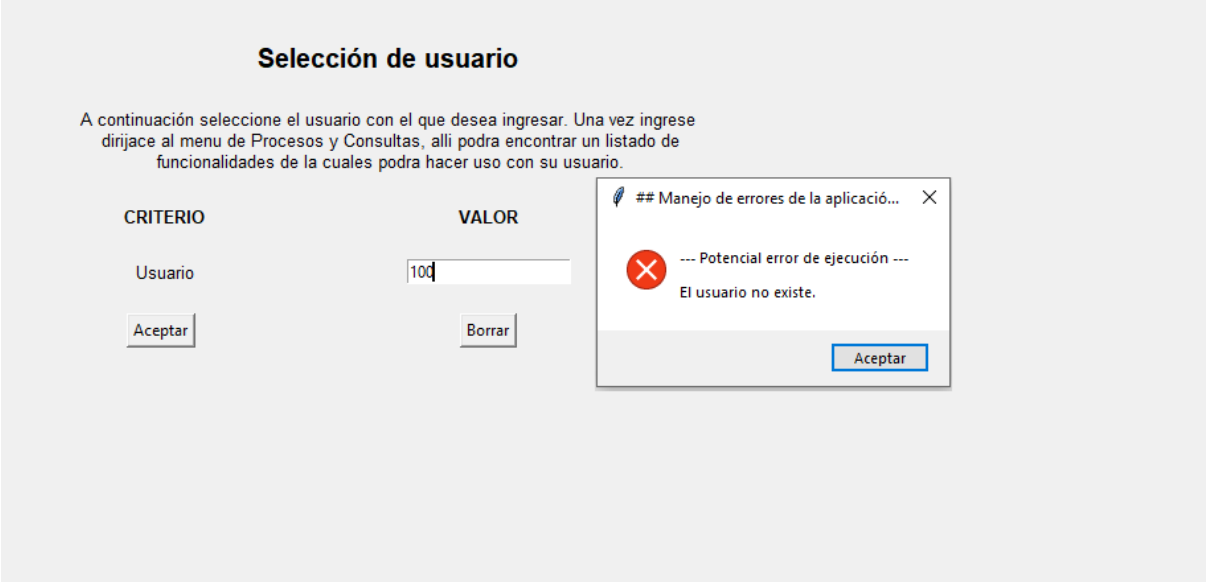
Esta clase/exception es llamada cuando el tipo de dato que ingresa un usuario no corresponde al tipo de dato esperado o requerido por el sistema. Corresponde a un error de ejecución ya que sin esta excepción se podrían hacer operaciones enter tipos incompatibles que resultarían en la detención del programa.



Exception ErrorExtraccion

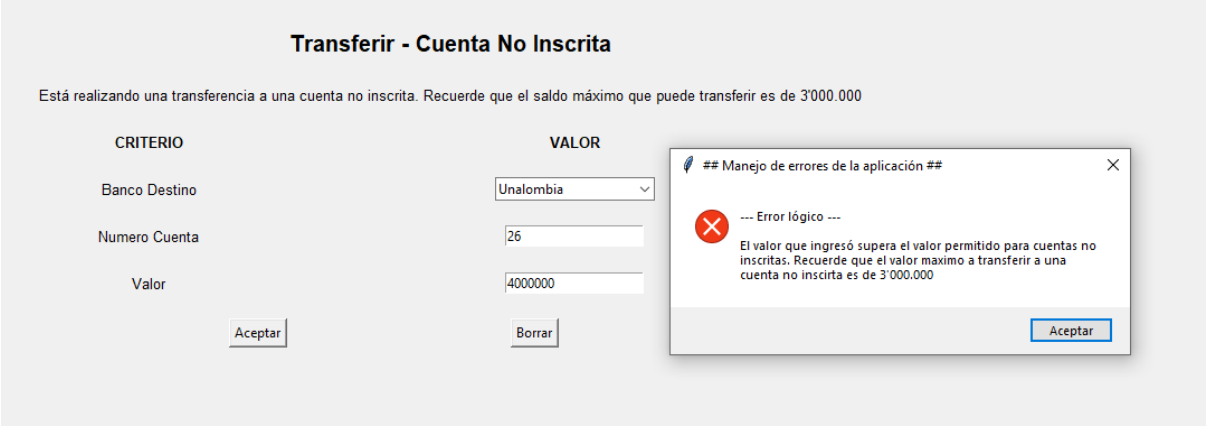
Esta clase/exception es llamada cuando se intenta extraer un objeto de alguna estructura de datos, y dicho objeto no está. Extiende de error de ejecución ya qué, si el programa

continuase su ejecución, se realizarían operaciones entre tipos potencialmente indeseados que forzarían la detención del programa.



Exception ErrorMaximoNoInscrito

Esta clase/exception es llamada cuando un usuario quiere enviar a una cuenta no inscrita una cantidad superior a la cantidad máxima permitida. Extiende de ErrorLogico ya que las operaciones se podrían ejecutar pero dañarían los comportamientos esperados y presupuestados del sistema.



Exception ErrorNoSaldo

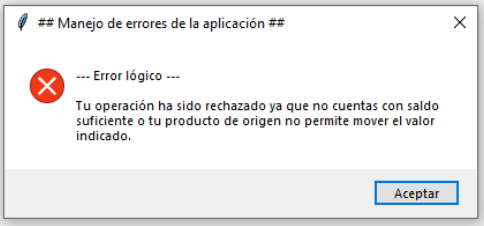
Esta clase/exception es llamada cuando una cuenta no tiene saldo o capacidad para concluir una operación. Es hija de la clase ErrorLogico ya que si bien las operaciones que se deben realizar son sintácticamente correctas, carecen de sentido bajo la lógica del programa (i.e., aunque tiene sentido sintáctico y semántico restar 1.000 - 2.000, no tiene sentido lógico bajo nuestro objetos ya que esto nos daría cuentas con saldos negativos, lo que, en general, está prohibido).

Transferir - Cuenta No Inscrita

Está realizando una transferencia a una cuenta no inscrita. Recuerde que el saldo máximo que puede transferir es de 3'000.000

CRITERIO	VALOR
Banco Destino	Unalombia
Numero Cuenta	26
Valor	100000

Aceptar Borrar



Manual de Usuario

Ventana Inicio

La ventana de inicio está compuesta por los siguientes elementos:

- En el menú de Inicio se puede acceder a las opciones Salir y Descripción
    - Al seleccionar el menú descripción se actualizará el lado derecho de la ventana mostrando información referente al sistema. En caso de seleccionarlo nuevamente, se regresará a la ventana original con las presentaciones e imágenes de los desarrolladores
    - Al seleccionar el menú Salir se detendrá la ejecución del sistema y se cerrará la ventana
- Imágenes del sistema a la izquierda de la ventana. Estas imágenes se pueden cambiar ingresando y retirando el cursor del área de las mismas.
- En la parte superior derecha se encuentra una breve presentación de los desarrolladores. Estas pueden cambiar al dar click sobre el texto.
- En la parte inferior derecha encontrará imágenes asociadas al desarrollador activo en la presentación de la parte superior derecha.
- Finalmente en la parte inferior izquierda estará el botón ‘Ingresar’, mediante el cual se puede acceder a la ventana de Funcionalidades.

Ventana de Funcionalidades

La ventana de funcionalidades presenta el siguiente flujo de operación:

Tras seleccionar en la ventana previa se mostrará en pantalla un recuadro para seleccionar el usuario con el que se desea acceder al sistema, ingresando su cédula. Este usuario será persistente durante toda la ejecución del programa.

Las dos cédulas válidas para el testeo de funciones son:

Cédula: 10 -> Da ingreso al usuario: Juan Perez (***RECOMENDADA***)
Cédula: 98 -> Da ingreso al usuario: Hernesto Perez

Una vez realizado el login se da la indicación de ingresar al menú ‘Procesos y Consultas’ en la parte superior de la ventana. Desde este menú el usuario puede acceder a las 5 funcionalidades:

Nota: Para el desarrollo del manual de usuario se supondrá que se escogió la cuenta recomendada 10. Sin embargo, al final del manual encontrará las propiedades de cada usuario para testear comportamientos con otras cuentas y usuarios.

1. **Inscribir cuenta** Desde esta opción se puede realizar la inscripción de una cuenta para evitar que se solicite información adicional cada vez que se desea transferir a dicha cuenta. Para testear esta funcionalidad se procede con los siguientes datos:

   Banco: Unalombia
   Tipo de cuenta: bajoMonto
   Número de cuenta: 26
   Número de cédula: 98

2. **Ver historial de transacciones** Con esta funcionalidad se puede visualizar todas las transacciones que se han realizado desde una cuenta. Para testearla se procede con los siguientes datos:

   Se mostrará en las opciones: 23 y 89. Una vez seleccionada la cuenta se da en el botón Aceptar que mostrará en pantalla las transacciones realizadas por dicha cuenta.

   Nota: En caso de que la cuenta no tenga ninguna transacción para mostrar se le notificará al usuario. Sin embargo puede realizar una transferencia de prueba para verificar posteriormente esta funcionalidad. (Para ver cómo realizar la transferencia diríjase al numeral 5).

3. **Solicitar crédito** Esta funcionalidad permite hacer una solicitud de crédito, que en caso de ser aceptada añadirá el capital aprobado al saldo de la cuenta indicada. El proceso es el siguiente:

   Banco: QuitaVivienda
   Cuenta: 89
   Monto: 10000
   Plazo (en meses): 12

   Nota: Ya que al usuario se le realiza una inspección de su perfilCrediticio, el cual se puntúa de manera aleatoria, ingresar estos datos dará en algunos casos una solicitud aprobada y en otros una solicitud rechazada.

4. **Romper topes** Mediante esta funcionalidad se transforma una cuenta de tipo bajoMonto a una cuenta de Ahorros (por lo tanto la funcionalidad sólo es válida para usuarios que tienen en sus cuentas asociadas, alguna cuenta de bajoMonto). El proceso de testeo es el siguiente:

Nota: En este caso el usuario 10 no tiene asociadas cuentas de bajoMonto, por lo tanto si se quiere testear la funcionalidad es necesario ingresar con el usuario 98.

Con usuario 98:

Cuentas disponibles: 69 o 26

En ambas selecciones se mostrará una pantalla de confirmación. Si la selección fue la cuenta 69 el proceso será exitoso, pero si fue la cuenta 26, el proceso fallará por falta de saldo.

5. **Transferir** Mediante esta funcionalidad se pueden realizar 3 tipos de transacción. Transferencias a Bancos, las cuales corresponden al pago de la cuota de un crédito; transferencias a cuentas inscritas y transferencias a cuentas no inscritas.

- **Transferencias a bancos (Pagar crédito)** Al seleccionar esta opción se paga el crédito desde la cuenta seleccionada por el usuario. El testeo se puede realizar mediante la selección de cualquiera de las cuentas, *SIN EMBARGO* es necesario haber realizado previamente una solicitud de crédito y que esta haya sido aprobada. (Ver numeral 3). Una vez hecha la selección se pedirá confirmación del usuario para pagar el crédito y se le notificará el resultado de la operación.

- **Transferencias a cuentas inscritas** Tras haber inscrito una cuenta (numeral 1), se selecciona la cuenta a la que se desea transferir y el valor. Se pedirá confirmación y posteriormente se notifica el resultado de la operación.

- **Transferencias a cuentas no inscritas** En este caso se deben ingresar los siguientes datos para obtener una transferencia exitosa:

  Banco: Poobanco
  Número cuenta: 69
  Valor: 10000

Lista objetos adicionales

Usuarios

- Juan Perez
  - Sin perfil crediticio
  - Ingresos mensuales de 1.000.000
  - Cédula: 10
  - Sin crédito activo
  - Cuentas asociadas: 1 y 3

- Hernesto Perez
    - Sin perfil crediticio
    - Ingresos mensuales: 2.000.000
    - Cédula: 98
    - Sin crédito activo
    - Cuentas asociadas 2 y 4

Cuentas

- Cuenta 1
    - Nro Cuenta: 89
    - Titular: Juan Perez
    - Saldo: 10.000
    - Banco: QuitaVivienda
    - Tipo cuenta: Ahorro

- Cuenta 2
    - Nro Cuenta: 69
    - Titular: Hernesto Perez
    - Saldo: 1.000.000
    - Banco: Poobanco
    - Tipo cuenta: Bajo Monto

- Cuenta 3
    - Nro Cuenta: 23
    - Titular: Juan Perez
    - Saldo: 50.000.000
    - Banco: Unalombia
    - Tipo cuenta: Corriente

- Cuenta 4
    - Nro Cuenta: 26
    - Titular: Hernesto Perez
    - Saldo: 14.000
    - Banco: Unalombia
    - Tipo cuenta: Bajo monto

Adicionalmente al menú de Procesos y Consultas se encuentran los menús:

- Archivo: Contiene los submenús

    - Descripción: Brinda información sobre lo que hace el sistema mediante una ventana emergente.

- Salir: Seleccionando este submenú se regresa a la ventana de Inicio

- Ayuda: Se encuentra un submenú ‘Acerca de’ que contiene los nombres de los autores de la aplicación.