Trabajo práctico 0

April 19, 2022

Dayana Xie Li 2022097967

Abel Gutiérrez Martínez 2022437323

Documentación Externa

En el presente documento, se detalla la información referente al trabajo práctico. En los siguientes párrafos se especificará a profundidad el funcionamiento del programa que va desde las variables, las funciones y la lógica que se emplea detrás del programa. A continuación, se presenta una lista con las secciones a cubrir dentro de este documento.

1. Ciclo de resolución de problemas
2. Diagramas usados para el trabajo práctico
3. Bibliografía

Resumen ejecutivo.

El programa consiste en un sistema bancario que sirve para depositar, retirar y consultar los montos de un determinado usuario. Además, hay otro usuario denominado banquero, este puede hacer uso de algunas funcionalidades como crear usuarios, crear cajeros, designar y agregar billetes a los cajeros, entre otras acciones. El sistema en la medida de lo posible, realiza cada requerimiento propuesto en el trabajo práctico. Se hace uso de los conocimientos desarrollados en el curso de "Taller de programación" como la recursividad, operadores numéricos, la creación de variables y funciones junto al uso de operadores lógicos y demás temas vistos en clases. Asimismo, se plantea el uso de buenas prácticas de programación y la elaboración de la documentación externa como recurso con la finalidad de apoyar a un mejor entendimiento del trabajo.

1 Ciclo de resolución de problemas

1.1 Especificación del problema:

Se desarrollará un sistema de cajero automático junto a la implementación de un sistema bancario. Habrá un archivo .txt, donde se almacenarán los usuarios con un número de pin aleatorio (los nombres de usuario deben ser creados por el banquero) y, además, se debe cumplir con una expresión regular propuesto por el profesor del curso. Por otro lado, se debe tener un identificador para cada cajero creado y se debe tener definido las denominaciones de los billetes para cada cajero, también se debe complementar con un historial de transacciones especificando los movimientos efectuados por el cliente. A grandes rasgos, esta es la especificación resumida de la definición del problema que se puede encontrar en el documento oficial entregado por el profesor del curso.

Análisis del problema:

En el análisis del problema se encuentran 3 módulos muy importantes y cada uno tiene sus propias entradas que se definirán a continuación:

Menú principal

- Correr menú principal:

  Entradas: no posee entrada alguna, es el inicio del sistema bancario

  Salidas: dependiendo de que si es banquero o usuario, se hace el llamado a la función con el menú de las acciones correspondientes.

Banquero

- **Creación de usuarios**

  Entradas: uno o más caracteres que pertenecen al abecedario (pueden ser mayúsculas o minúsculas) más cuatro números enteros y un carácter especial. Se debe cumplir con una expresión regular.

  Salidas: guardar la entrada en un archivo .txt y automáticamente se crea un pin de 4 números generados por el sistema.

  Restricciones: debe ser en el orden y estructura propuesto por la expresión regular, si no cumple con esto, se rechaza la entrada y no se continúa con el proceso.

- **Crear cajeros**

  Entradas: se debe ingresar por consola una identificación que cumpla con la expresión regular: tiene que tener 3 letras en mayúscula seguida de al menos un número entero o más. También se deberá de ingresar la cantidad de billetes por denominación que desea que el cajero disponga.

Salidas: se almacena la entrada en un archivo .txt y se debe agregar las cantidades por denominación al cajero creado.

Restricciones: se debe cumplir con la expresión regular, si no es así, se debe indicar al banquero que no cumple con esta condición y no seguir con el proceso. La cantidad de billetes debe de ser entera y positiva.

- **Agregar billetes a un cajero existente**

  Entradas: se debe ingresar por consola la identificación del cajero al que le desea agregar los billetes y deberá de ingresar la cantidad de billetes por denominación que desea agregar al cajero seleccionado.

  Salidas: se actualiza la cantidad de billetes por denominación.

  Restricciones: la cantidad de billetes debe de ser entera y positiva.

Cliente del banco

- **Retirar dinero**

  Entradas: nombre del usuario, su pin y cantidad de dinero que el cliente desea retirar.

  Salidas: registro del retiro en el historial, actualización del monto de los cajeros y en el saldo del usuario.

  Restricciones: no puede ser un número con decimales ni negativo, puede ser cualquier cantidad de dinero, siempre y cuando la cantidad de billetes se lo permita el cajero seleccionado.

- **Depositar dinero**

  Entradas: nombre del usuario, su pin y la cantidad de billetes por denominación que desea depositar.

  Salidas: registro del depósito en el historial, actualización del monto de los cajeros y en el saldo del usuario.

  Restricciones: la cantidad de billetes debe de ser entera y positiva, el nombre de usuario y el pin deben de ser correctos.

- **Consultar el monto disponible en la cuenta.**

  Entrada: debe de ingresar el nombre del usuario y el pin.

  Salidas: mostrar por pantalla el monto disponible de la cuenta.

  Restricciones: si no ingresa un nombre de usuario correcto y existente o su pin no se le permitirá ver el saldo.

- **Consultar el historial de transacciones**

  Entrada: nombre del usuario que desea consultar su historial.

  Salidas: mostrar cada movimiento realizado en la cuenta del usuario en forma tabular.

  Restricciones: si no ingresa un nombre de usuario correcto y existente o su pin no se le permitirá ver su historial.

1.2 Subproblemas

Menú principal

- **Correr menú principal**

    1. Verificar que el menú sea lo primero que se corra en el programa.
    2. Preguntar quién es el que está usando el programa. Si la persona usando el programa es banquero,
    3. Mostrar las opciones disponibles que puede realizar. Cargar los datos del cajero para ser mostrados.
    4. Si la persona usando el programa es cliente, mostrarle los cajeros disponibles.
    5. Permitir que el usuario ingrese su nombre de usuario y pin.
    6. Mostrarle al usuario las acciones que puede realizar.

Banquero

- **Crear usuarios**

    1. Se debe ingresar una hilera de texto por consola.
    2. Se debe recorrer el usuario e identificar sus caracteres en código Unicode y guardarlos en una lista.
    3. Se debe verificar si cumple con la expresión regular.
    4. Chequear si existe ese usuario en el archivo txt, sino crear el nuevo usuario.
    5. Se debe generar un pin de 4 números aleatoriamente.
    6. Guardar los datos en un archivo txt.

- **Crear cajeros**

    1. Se debe ingresar una hilera de texto por consola.
    2. Se debe recorrer el identificador de cajero e identificar sus caracteres en código Unicode, guardandolos en una lista.
    3. Verificar si el identificador ya existe o sino proceder a crearlo.
    4. Se debe de pedir al banquero la cantidad de billetes a agregar.
    5. Se debe agregar la denominaciones de billetes vinculados al identificador a crear.
    6. Chequear si las cantidades ingresadas son enteros y positivas.
    7. Guardar los datos del identificador junto a la cantidad de billetes por denominación en un archivo txt.

- **Agregar billetes a un cajero existente**

1. Saber a cuál cajero se desea agregar billetes.
2. Verificar que exista ese identificador.
3. Ingresar las cantidades de billetes por denominación.
4. Actualizar las cantidades de billetes por denominación.
5. Almacenar los datos en el archivo txt.

Clientes del bancos

- **Depositar dinero**
  1. Seleccionar el cajero. Buscar el cajero en el sistema.
  2. Preguntar las cantidades de billetes por denominación a depositar.
  3. Verificar si esas cantidades ingresadas son enteros.
  4. Almacenar los datos del depósito.
  5. Actualizar los datos del cajero.
  6. Buscar los datos del usuario y actualizarlos.
  7. Notificar al usuario del depósito exitoso o fallido.
- **Retirar dinero**
  1. Identificar y buscar el cajero seleccionado en el sistema.
  2. Preguntar por el monto de dinero que desea retirar.
  3. Establecer el mínimo de billetes a retirar en la transacción.
  4. Detectar las cantidades de billetes que tiene el cajero seleccionado.
  5. Determinar si se puede hacer el retiro del cajero.
  6. Buscar los datos del usuario y actualizarlos.
  7. Actualizar los datos del cajero.
  8. Almacenar los datos actualizados en sus respectivos archivos txt.
  9. Notificar al usuario del retiro exitoso o fallido.
- **Revisar el monto disponible del usuario**
  1. Identificar y buscar el usuario en el sistema.
  2. Confirmar que existe el usuario.
  3. Mostrar los datos por pantalla.
- **Consultar historial de transacciones**
  1. Identificar y buscar el usuario en el sistema.
  2. Confirmar si existe el usuario.
  3. Buscar datos del usuario en el sistema.
  4. Crear los títulos de las transacciones.
  5. Mostrar por pantalla de forma tabulada las transacciones realizadas por el usuario.

1.3 Pseudocódigo

El pseudocódigo se dividirá en varias partes como el menú principal, la sección de banquero y, por último, la sección de usuario, esto con la finalidad de llevar un orden y comprender mejor el programa.

Sección menú principal del sistema

Función correr_menu_principal()

1. Se imprimen por pantalla las 2 opciones que se pueden usar en el sistema.
   (a) Si el usuario elige la opción de Banquero se, retorna correr_menu_banquero().
   (b) Si el usuario elige la opción de Cliente del banco, se retorna correr_menu_usuario().
2. Si el usuario selecciona otra opción que no se encuentra dentro de las opciones, se hace el llamado recursivo correr_menu_principal().

Función correr_menu_banquero()

Se muestran por pantalla 3 opciones, crear cajero, crear usuario y agregar billetes a un cajero existente.

1. Se muestran por pantalla las opciones que dispone el usuario del sistema.
   (a) Si el usuario selecciona la opción de crear cajero, se imprime por pantalla la opción seleccionada y se retorna crear_cajero().
   (b) Si el usuario selecciona la opción de crear usuario, se imprime por pantalla la opción seleccionada y se retorna crear_usuario().
   (c) Si el usuario selecciona la opción de agregar billetes a un cajero existente, se imprime por pantalla la opción seleccionada y se retorna agregar_billetes_cajero_existente().
2. En caso de seleccionar otra opción que no sea ninguna de las 3 anteriores, se retorna nuevamente la función correr_menu_banquero()

Función correr_menu_usuario()

La función correr_menu_usuario() imprime el título de "Lista de cajeros disponibles" e invoca la función auxiliar cargar_datos_cajero(). La función cargar_datos_cajero() necesita un parámetro numérico que es el 0 (es la posición donde inicia la búsqueda)

Función cargar_datos_cajero(linea_de_texto_cajero)

Se crea una variable local llamada archivos_cajeros_lectura() en modo lectura. Y luego se utiliza el método .readlines() (readlines() devuelve una lista con todo los elementos de un archivo) en la variable local llamada lineas_cajeros. Después, se cierra el archivo.

1. Sí linea_texto_cajero es menor es menor o igual tamaño de lista_hilera_cajero menos 1, en tal caso, se crea una variable local y se le asigna el valor del índice [1] de la variable lista_hilera_cajero (Se le asigna un uno porque esa es la ubicación fija del identificador del cajero en la lista).

   (a) Se crea una variable llamada archivo_cajero_disponible_escritura y se crea por primera vez el archivo llamado "Cajeros_disponibles.txt" con el parámetro "a" que añade texto. Este archivo almacenará los cajeros disponibles.

   (b) En la variable archivo_cajero_disponible_escritura se usa el método .write() para escribir los datos almacenados en cajeros_disponibles y se concatena con una coma.

   (c) Se cierra el archivo y se abre nuevamente con el parámetro "r" para la lectura del archivo.

   (d) Se imprime por pantalla "Cajeros disponibles" junto con el identificador del cajero.

   (e) Se establece una nueva variable llamada lineas_archivo_cajero_disponibles y se utiliza el método .read() y se cierra el archivo con el método .close().

   (f) Se retorna la función cargar_datos_cajero(linea_texto_cajero + 1) para que siga con la siguiente línea del archivo si es que existen más cajeros y se hace llamado a la función elegir_cajero(lineas_archivos_cajeros_disponibles)

Función elegir_cajero(lineas_archivos_cajeros_disponibles)

En función elegir_cajero() se toma un parámetro que contiene los datos de los cajeros.

1. Se establece una variable llamada cajero_seleccionado y dentro de esta variable se almacena la entrada digitada por el usuario (debe ingresar el cajero que desea utilizar).

2. Se retorna revisar_lineas(cajero_seleccionado, 0, lineas_archivo_cajeros_disponible) Para verificar que el cajero exista.

Función revisar_lineas(cajero_seleccionado, revisando_linea, linea_archivo_cajeros_disponibl

Esta función requiere de 3 parámetros, el cajero seleccionado, posición del dato en el archivo que inicia con el valor en 0 y el archivo a donde se encuentran los datos.

1. En la variable llamada archivos_cajero_lectura se abre el documento llamada "Lista_de_cajeros.txt".
2. En otra variable llamada lectura_lineas_cajero, se utiliza el método .readlines() para obtener todos los datos del archivo.
   (a) Si revisando_linea es menor o igual al tamaño de la variable lectura_linea_cajeros restando un -1.
   (b) Y además, si cumple que cajero_seleccionado se encuentra en lineas_archivo_cajeros_disponibles y el tamaño de la variable cajero_seleccionado es mayor o igual a 4. Entonces, retorna la función correr_menu_usuario_aux(cajero_seleccionado).
   (c) Sino, revisando_lineas se incrementa en 1 para revisar la siguiente línea y retorna otra vez más la función revisar_lineas(cajero_seleccionado, revisando_linea, lineas_archivo_cajeros_disponibles)
   (d) Ahora bien, si el cajero no existe o está siendo ocupado por otra persona, imprimir por pantalla "Estimado usuario, por favor seleccione un ccajero que esté". Y se regresa la función elegir_cajero(lineas_archivo_cajeros_disponibles).

Función correr_menu_usuario_aux(cajero_seleccionado)

Función que necesita una parámetro para solicitar el nombre de usuario y el pin para luego hacer la operación mediante el cajero seleccionado.

1. A través de la variable llamada usuario_usando_sistema capta el nombre de usuario.
2. Con la variable leer_base_de_datos_banquero, se abre el archivo "Lista_de_cajeros.txt" como lectura.
3. Con otra variable llamada lectura_base_datos, se utiliza el método .read() para leer el archivo.
   (a) Si usuario_usando_sistema se encuentra en la variable lectura_base_datos y el tamaño de usuario_usando_el_sistema es mayor o igual a 7 (que es el tamaño mínimo del nombre de usuario), solicitar el pin mediante la variable llamada pin_del_usuario.
      i. Si pin_del_usuario se encuentra en lectura_de_base_datos y el tamaño del pin es igual a 4, se vuelve la función acciones_usuario(cajero_seleccionado, usuario_usando_el_sistema)
   (b) Si el usuario digita mal el pin de su cuenta o su nombre de usuario, imprimir por pantalla un aviso notificando al cliente sobre el error y se retorna devuelta la función correr_menu_usuario_aux(cajero_seleccionado).

Función acciones_usuario(cajero_seleccionado, usuario_usando_el_sistema)

Esta función se utiliza para que el usuario elija la acción a realizar en el programa, esta función requiere 2 parámetros, el primero es el cajero donde va realizar la acción y el segundo es el nombre de usuario del cliente.

1. En la variable denominada elegir_accion_usuario, se mostrará por pantalla las opciones que dispone el cliente. la opción 1 es retirar dinero, la opción 2 es depositar dinero, opción 3 consultar saldos y por último la opción 4 consultar el historial de transacciones.

   (a) Si elegir_accion_usuario es igual a 1, se imprime por pantalla la acción a realizar y se retorna la función retirar_dinero(cajero_seleccionado, usuario_usando_el_sistema)

   (b) Si el usuario elige la opción 2, se imprime por pantalla la acción a realizar y se devuelve la función depositar_dinero(cajero_seleccionado, usuario_usando_el_sistema).

   (c) Si el usuario elige la opción 3, se imprime por pantalla la acción a realizar y se regresa la función consultar_saldo(cajero_seleccionado, usuario_usando_el_sistema).

   (d) Si el usuario elige la opcion 4, se imprime por pantalla la acción a realizar y se regresa la función historial_transacciones(cajero_seleccionado, usuario_usando_el_sistema).

   (e) Si el usuario no selecciona ninguna de las opciones anteriores o ingresa un valor incorrecto, se indica por pantalla un mensaje "por favor seleccione una opción de las disponibles por el sistema) y se retorna la función correr_menu_usuario_aux(cajero_seleccionado)

Sección banquero

Función crear_cajero()

Esta función no recibe ningún parámetro, también está ligada al menú principal con la función correr_menu_banquero().

1. En la variable identificador_del_cajero se almacenará el identificador ingresado por el banquero.

   (a) Si usando la función verificar_cajero con el identificador_del_cajero cumple con las condiciones, informar al usuario la creación exitosa del identificador del cajero y volver al menú principal.

   (b) Si no es posible cumplir con las condiciones del punto (a), notificar al usuario que el cajero no se pudo crear y se vuelve al menú principal.

Función verificar_cajero(identificador_del_cajero)

Esta función cumple el rol de verificar la longitud del identificador y necesita de un parámetro llamado identificador_del_cajero.

1. Si el identificador del sistema tiene una tamaño igual o mayor a 4, se retorna la función auxiliar llamada verificar_cajero_aux(identificador, 0)

2. De no cumplir con el punto 1, se notifica al usuario que la longitud del identificador_del_cajero no cumple con la condición y se retorna Falso.

Función convertir_a_unicode(elemento_a_convertir, posicion, lista_unicode)

Esta función convierte a código Unicode cada carácter del elemento ingresado (ya sea identificador o nombre de usuario), necesita de 3 parámetros. Número 1;el carácter a convertir, número 2; la posición del carácter incializada en cero y por último las lista donde se almacenan los código unicode.

1. La variable largo debe guardar el tamaño del parámetro elemento_a_convertir.
   (a) Si la posición es menor a la variable largo_cajero, se crear una variable llamada valor_unicode como una lista utilizando los datos elementos_a_convertir junto a la posición utilizando la función integrada llamada ord (devuelve el código unicode del argumento).
   (b) Lista_unicode se suma al valor de valor_unicode_cajero. Se incrementa en uno la variable posicion. y se hace el llamado recursivo de la función convertir_a_unicode(elemento_a_convertir, posicion, lista_unicode)
   (c) Por último, si posicion es igual largo_cajero, se devuelve la variable lista_unicode.

Función verificar_cajero_aux(identificador_del_cajero, posicion_cajero)

La función verificar_cajero_aux se usa para verificar que el identificador del cajero en código Unicode esté correcto cumpliendo con la expresión regular, y necesita 2 parámetros, primero el identificador del cajero y segundo la posición en donde se encuentra el cajero (el cual inicia en cero).

1. En la variable lista_unicode_cajero invoca a la función convertir_a_unicode y le pasa por parámetro el identificador del sistema, la posición del cajero y la lista unicode. Lo que hace esta función es devolver una lista con el identificador a crear en valores unicode.
2. En otra variable llamada rango_letras_mayus con el rango de 65 a 91 que corresponden a las letras del abecedario en mayúscula.
3. Se crea otra variable que almacena el rango de los números y su código Unicode del 48 al 58.
4. En longitud_cajero se guarda el tamaño de lista_unicode_cajero.
5. En la variable local numeros_posicion_identificador se le asigna el valor de 3.
6. Se utilizará el primer número de identificador como referencia. Si la posición_cajero es menor a numeros_posicion_identificador
   (a) Se verifica que lista_unicode_cajero junto posicion_cajero como índice se encuentra en rango_letras_mayus, se debe incrementar en 1 la posición del identificador y hacer el llamado recursivo verificar_cajero_aux(identificador_cajero, posicion_cajero)

(b) Si posicion_cajero es mayor o igual numeros_posicion_identificador y, además, posicion_cajero es menor a longitud_cajero y también si lista_unicode_cajero con posicion_cajero como índice se encuentra en rango_numeros, sumar un uno a posicion_cajero. Se hace el llamado recursivo verificar_cajero_aux(identificador_del_cajero, posicion_cajero).

(c) Ya habiendo revisado carácter por carácter, si posicion_cajero es igual a longitud_cajero, retornar la función chequear_identificador(identificador_del_cajero).

(d) Luego de todos los pasos anteriores, si no se cumplen se debe informar al usuario que el identificador no cumple con la expresión regular solicitada por el sistema y retornar un falso.

Función chequer_identificador(identificador_del_cajero)

Esta función comprueba si existe el identificador del cajero y solo recibe un parámetro.

1. En la variable archivo, se abre el archivo "Lista_de_cajero.txt" en modo append (es para añadir datos a un archivo).
2. Se abre nuevamente la variable llamada archivo, solo que en este caso en modod lectura con el parámetro "r".
3. En lineas_cajero, se utiliza el método .read() para leer los datos del archivo.
4. Se cierra el archivo con el método .close()
5. Si el identificador_del_cajero se encuentra en lineas_cajero, entonces informar al usuario que ya existe ese identificador y que debe proceder a crear otro. En esta parte retorna falso.
6. En caso contrario, si el identificador_del_cajero no está en lineas_cajero, se procede a llamar la función agregar_billetes_a_cajero_nuevo(identificador_del_cajero) y retorna un verdadero.

Función agregar_billetes_a_cajero_nuevo(identificador_del_cajero)

Esta función se utiliza para agregar los billetes que desea por denominación al nuevo cajero creado, precisa de un parámetro que es el identificador del cajero creado recientemente.

1. Se muestra por pantalla una pregunta al usuario sobre la cantidad de billetes por denominación que desea agregar al nuevo cajero.
2. En una variable llamada billetes_100_cajero_nuevo, se almacena la cantidad de billetes de esa denominación y se convierte en una dato tipo entero. Asimismo, se invoca una función llamada verificar_entrada_cajero_nuevo()

3. En otra variable llamada billetes_50_cajero_nuevo se guarda la cantidad de billetes de esa denominación y se utiliza la función verificar_entradas_cajero_nuevo, luego se convierte a un dato tipo entero.

4. En billetes_20_cajero_nuevo se guarda la cantidad de billetes de esa denominación y se utiliza la función verificar_entradas_cajero_nuevo, luego se convierte a un dato tipo entero.

5. En billetes_10_cajero_nuevo se guarda la cantidad de billetes de esa denominación y se utiliza la función verificar_entradas_cajero_nuevo, luego se convierte a un dato tipo entero.

6. En billetes_5_cajero_nuevo se guarda la cantidad de billetes de esa denominación y se utiliza la función verificar_entradas_cajero_nuevo, luego se convierte a un dato tipo entero.

7. En billetes_2_cajero_nuevo se guarda la cantidad de billetes de esa denominación y se utiliza la función verificar_entradas_cajero_nuevo, luego se convierte a un dato tipo entero.

8. En billetes_1_cajero_nuevo se guarda la cantidad de billetes de esa denominación y se utiliza la función verificar_entradas_cajero_nuevo, luego se convierte a un dato tipo entero.

9. Se invoca la función guardar_datos_del_cajero para guardar los datos de todas las variables anteriores.

Función verificar_entradas_cajero_nuevo(billetes_depositados)

La función se utiliza para verificar si las entradas son enteras y de no ser así se tira el error al usuario. Se requiere un parámetro que es la cantidad de billetes depositados.

1. Con la palabra reservada try, se manejan los errores que se pueden dar en determinada situación. En la variable verificar_billetes_depositados se intenta convertir a entero el parámetro de la función.

   (a) Si verificar_billetes_depositados fue posible convertirlo a un entero, se revisa si verificar_billetes_depositados es es distinto del valor absoluto de verificar_billetes_depositados (esto quiere decir que el dato ingresado no sea de valor númérico).

   (b) Indicar al usuario mediante la función de manejo de errores ValueError, indicar al usuario que la cantidad es un valor inválido.

2. Si no es posible realizar el punto uno, se alza un error al usuario.

Función guardar_datos_del_cajero(identificador_del_cajero, billetes_100, billetes_50, billetes_20, billetes_10, billetes_5, billetes_2, billetes_1)

La función se encarga de almacenar todos los datos relacionados con los billetes agregados a un cajero nuevo. Se requiere de 8 parámetros, el identificador del cajero y todos los datos relacionados con las cantidades de billetes por denominación.

1. En lista_diccionario_cajero, se almacenan los datos del identificador, billetes_100, billetes_50, billetes_20, billetes_10, billetes_5, billetes_2, billetes_1 en una lista.
2. En la variable archivo, se abre el archivo "Lista_de_cajeros.txt" con el parámetro "a" que funciona para agregar datos.
3. Se crea una variable local con el nombre de diccionario_hilera y se transforma a datos tipo hilera la variable lista_diccionario_cajero.
4. Con el método .write(), se escribe diccionario_hilera en el archivo "Lista_de_cajeros".
5. Se cierra el archivo con el método .close()

Función crear_usuario()

La función crear_usuario() no necesita parámetros y se llama desde el menú principal del programa.

1. En la variable llamada nombre_usuario se almacena el nombre de usuario digitado por el usuario.
2. Si verificar_longitud con el argumento nombre_usuario cumple con la condiciones preestablecidas, se indica al usuario que el usuario ha sido creado exitosamente y se vuelve al menú principal.
3. De no cumplir con las condiciones de la función verificar_longitud, se informa al usuario que el no se pudo crear y se vuelve al menú principal.

Función verificar_longitud(nombre_usuario)

En esta función como lo indica su nombre, verifica la longitud de nombre de usuario ingresado por el usuario.

1. Si el tamaño del nombre_usuario es mayor o igual a 6, se retorna la funcion verificar_unicode_usuario(nombre_usuario, 0)
2. De no cumplir con la condición del punto 1, se informa al cliente que el nombre de usuario no cumple con la longitud deseada y retorna un Falso.

Función verificar_unicode_usuario(nombre_usuario, posicion_a_verificar)

En esta función, se verifica los código unicode del usuario digitado por el usuario del sistema. Necesita dos parámetros. Primero, el nombre de usuario y, segundo, la posición a verificar.

1. Con la variable lista_unicode_usuario, se invoca la función llamada convertir_a_unicode(nombre_usuario, 0, [])
2. Se almacena un rango de 97 a 123 con la letras minúsculas en la variable letras_minus_rango.
3. En otra variable llamada letras_mayus_rango, se guarda el rango de 65 a 91 que pertenecen a los códigos unicode.
4. Con numero_rango, se almacena el rango de 48 a 58 con los código unicode los números.
5. En la variable simbolo_rango, se guardan el número 33,35, 36, 38 y 63 con los caracteres especiales que se requieren para cumplir con la expresión regular.
6. Con longitud_lista, se calcula el tamaño de lista_unicode_usuario.
7. A longitud_lista se le resta 1 y se guarda en la variable simbolo_posicion. Ya que la posición del símbolo siempre será la última.
8. A longitud_lista se resta 5 y se almacena en la variable numeros_posicion_usuario.
9. Si posicion_a_verificar es menor a numeros_posicion_usuario y si lista_unicode_usuario con posicion_a_verificar como índice se encuentra en letras_minus_rango o lista_unicode_usuario con posicion_a_verificar como índice se halla en letras_mayus_rango. Entonces, posicion_a_verificar se le suma un 1 y retorna la función verificar_unicode_usuario(nombre_usuario, posicion_a_verificar)
10. Si posicion_a_verificar es mayor o igual numeros_posicion_usuario y posicion_a_verificar es menor a simbolo_posicion y también si lista_unicode_usuario con posicion_a_verificar como índice entre corchetes se encuentra en numeros_rango, en tal caso, posicion_a_verificar se incrementa en uno y se retorna la función verificar_unicode_usuario(nombre_usuario, posicion_a_verificar)
11. Si posicion_a_verificar es igual a simbolo_posicion y si lista lista_unicode_usuario con posicion_a_verificar como índice se halla en simbolo_rango, regresar la función chequear_usuario(nombre_usuario).
12. Si no cumple con las condiciones del punto 9 a 11, se detalla por pantalla que el nombre de usuario no cumple con la expresión regular. y retorna un Falso.

Función chequear_usuario(nombre_usuario)

El objetivo de esta función es comprobar que el nombre de usuario no haya sido creado y, si es así, continuar con el proceso de creación de usuario.

1. Con archivo_usuario se abre el archivo llamado "Lista_usuario.txt" en modo añadir datos al archivo.
2. Con archivo_usuario_lectura se abre nuevamente el archivo "Lista_usuario.txt" en modo letura.
3. En lineas_usuario, se usa el método .read() para leer todo el archivo.
4. Se cierra el archivo_usuario con el método .close().
5. Si nombre_usuario se halla en lineas_usuario, se informa al usuario que no se puede crear ese usuario porque ya existe y retorno un Falso.
6. Si no se cumple la condición del punto 5, se llama a la función generar_pin(nombre_usuario).

Función generar_pin(nombre_usuario)

Cuando ha cumplido con cada función anterior y todas sus condiciones, se procede a crear un pin único al nombre de usuario.

1. Se crea una variable pin y se almacena una función llamada randrange(1000,9999) que funciona para combinar cuatro dígitos del 1000 a 9999.
2. Se muestra por pantalla, el pin generado por la función al usuario del sistema.
3. Retornar la función guardar_usuarios(nombre_usuario, pin)

Función guardar_usuarios(nombre_usuario, pin)

Esta función almacena los datos tanto el nombre de usuario como el pin generado aleatoriamente en un archivo txt.

1. Si pin es un entero, se crea una variable llamada datos_del_usuario. datos_del usuario almacena los datos en una lista.
2. Mediante la variable archivo, se abre el archivo llamado "Lista_de_usuarios" en modo "a" para añadir datos al archivo.
3. En diccionario_hilera, la variable datos_del_usuario se transforma en una dato de tipo hilera.
4. Con el método .write(), se escribe en el archivo los datos almacenados en la variable diccionario_hillera y se contatena con una salto de línea ("\n").
5. Se cierra el archivo con el método .close() y se retorna Verdadero.

Función agregar_billetes_cajero_existente()

Esta función le permite al usuario del sistema agregar billetes a un cajero previamente creado o un cajero existente desde hace tiempo.

1. En una variable llamada ingresar_identificador se almacena el identificador del cajero.
2. Con el archivo "lista_de_cajero.txt", se le asigna a la variable llamada lista_cajeros.
3. Sí ingresar identificador se encuentra en lista_cajero con el método .read() y el tamaño de ingresar_identificador es mayor o igual a 4. Entonces, muestra por pantalla que el identificador se encuentra en el sistema e inmediatamente se solicitan las cantidades de billetes por denominación para agregar al cajero existente.
4. Se cierra el archivo con el método .close()
5. En la variable billetes_100_a_agregar se guarda la información y se convierte en un dato de tipo entero.
6. En la variable billetes_50_a_agregar se guarda la información y se convierte en un dato de tipo entero.
7. En la variable billetes_20_a_agregar se guarda la información y se convierte en un dato de tipo entero.
8. En la variable billetes_10_a_agregar se guarda la información y se convierte en un dato de tipo entero.
9. En la variable billetes_5_a_agregar se guarda la información y se convierte en un dato de tipo entero.
10. En la variable billetes_2_a_agregar se guarda la información y se convierte en un dato de tipo entero.
11. En la variable billetes_1_a_agregar se guarda la información y se convierte en un dato de tipo entero.
12. Se define la variable posicion y se le asigna el valor de 0. Se retorna la función guardar_billetes_cajero_existente(ingresar_identificador, billetes_100_a_agregar, billetes_50_a_agregar, billetes_20_a_agregar, billetes_10_a_agregar, billetes_5_a_agregar, billetes_2_a_agregar, billetes_1_a_agregar)
13. Si no cumple con las condiciones anteriores, se utiliza la función ValueError para indicar al usuario su error.

Función guardar_billetes_cajero_existente(ingresar_identificador, billetes_100_a_agregar, billetes_50_a_agregar, billetes_20_a_agregar, billetes_20_a_agregar, billetes_10_a_agregar, billetes_5_a_agregar, billetes_2_a_agregar, billetes_1_a_agregar)

Función encargada de agregar y actualizar los billetes a un cajero existente. Necesita de 8 parámetros, el identificador y las cantidades de todos los billetes que se desean agregar.

1. Con guardar_biiletes, se abre el archivo "Lista_de_cajeros.txt" en modo lectura y con la posibilidad de agregar más datos al archivo.
2. En leer_lineas_cajero_existente, se almacena guardar_billetes con el método readlines() (método que regresa una lista con todos los elementos del archivo.
3. Se cierra el archivo con el método .close()
4. Se crea una variable llamada linea_cajero_existente que almacena la información de leeer_lineas_cajero_existente con posicion como índice.
5. Sí ingresar identificador se halla en lineas_cajero_existente con posición.
   (a) En variable linea_cajero_existente se utiliza el método replace() para reemplazar "[" con un espacio nulo.
   (b) En variable linea_cajero_existente se utiliza el método replace() para reemplazar "]" con un espacio nulo.
   (c) En variable linea_cajero_existente se utiliza el método replace() para reemplazar " ' " (comilla simple) con un espacio nulo.
   (d) En variable linea_cajero_existente se utiliza el método replace() para reemplazar " " (espacio) con un espacio nulo. En variable linea_cajero_existente se utiliza el método replace() para reemplazar "\n" (salto de línea) con un espacio nulo.
   (e) Se crea una nueva variable denominada lista_hileras_cajero_actualizado y se almacena el resultado de linea_cajero_existente junto el método .split(",") para separar todos los elementos por una coma.
   (f) Se buscan los índices 3,5,7,9,11,13,15 para sumar los valores numéricos a los ya existentes.
   (g) Se retorna la función rescatar_lineas_cajero(0, '', lista_hilera_cajero_actualido, ingresar_identificador)
6. Si no se cumple con las condiciones del punto 5, la variable posicion se incrementa en uno para seguir buscando la hilera correcta.
   (a) Como retorno se hace el llamado recursivo.

Función rescatar_lineas_cajero(posicion_cajero, datos_cajero_actualizados, lista_hileras_cajero_actualizado, cajero_seleccionado)

Esta función se encarga de rescatar las líneas de información que no se van a modificar, necesita 4 parámetros. Primero; se inicia con 0 y luego se recorre para leer las demás líneas del archivo, segundo; en este parámetro con un espacio vacío para guardar los datos, tercero; se almacena la hilera con la información actualizada. Y por último, el cuarto parámetro, es el identificador con el que se vinculan los datos guardados.

1. Con rescatar_linea_archivo_cajero, se abre el archivo "Lista_de_cajero.txt" en modo lectura.
2. En la variable leer_lineas_texto_cajero, se guarda los datos de rescatar_linea_archivo_cajero utilizando el método .readlines(), dicho método regresa una lista con todos los elementos del archivo.
3. Se cierra el archivo con el método .close()
4. Si posicion_cajero es menor a la longitud de leer_linea_texto_cajero, entonces:
   (a) En la variable linea_a_revisar_cajero se asigna el valor del índice de posicion_cajero contenido en la variable linea_a_revisar_cajero.
   (b) Con el método .replace(), se reemplaza el salto de línea con un espacio nulo.
   (c) Con el método .replace(), se reemplaza la doble comilla con un espacio nulo.
   (d) Con el método .replace(), se reemplaza los espacios con un espacio nulo.
      i. Si cajero_seleccionado se halla en linea_a_revisar_cajero, se incrementa en uno la variable posicion_cajero.
      ii. Se retorna rescatar_lineas_cajero(posicion_cajero, datos_cajero_actualizados, lista_hileras_cajero_actual, cajero_seleccionado).
      iii. Si no se cumple con la condición del punto i, entonces:
         A. datos_cajero se le asigna la información de linea_a_revisar_cajero + un salto de línea.
         B. Y, además, datos_cajero_actualizados se suma a datos_cajero.
         C. La variable posicion_cajero se incrementa en uno.
         D. Se retorna la función rescatar_linesa_cajero(posicion_cajero, dato_cajero_actualizados, lista_hileras_actualizado, cajero_seleccionado).
5. De no cumplirse la condición del punto 4, entonces:
   (a) datos_cajero_actualizados se concatena con lista_hileras_cajero_actualizado
   (b) Se retorna actualizar_datos_cajero(datos_cajero_actualizados)

Función actualizar_datos_cajero(datos_cajero_actualizados)

Esta función se encarga de actualizar los datos del archivo "Lista_de_cajeros.txt", se requiere un parámetro con la información actualizada con otras fuciones desarrolladas.

1. En archivo_cajero_original, se abre el archivo "Lista_de cajeros.txt" con el modo de agregar o elminar datos de un archivo.
2. Con el método .truncate(0), se eliminan todos los datos.
3. Se reescribe la información con el método .write(), utilizando la variable archivo_cajeros_actualizados.
4. Se cierra el archivo con el método .close() y se vuelve al menú principal.

Sección cliente del banco

En esta sección, se explicarán todas las funciones vinculadas a todas las acciones que se pueden realizar como cliente del banco.

Función depositar_dinero(cajero_seleccionado_usuario_usando_el_sistema)

Esta función consiste en poder ejecutar la acción de depositar dinero por parte de un cliente. Son necesarios 2 parámetros para hacer posible la ejecución del programa.

1. Se imprime por pantalla la indicación de ingresar el usuario.
2. En depositar_billetes_100, se almacena la cantidad de billetes de dicha denominación, se utiliza una función llamada verificar_entero() para garantizar que el usuario ingresa un dato numérico de tipo entero.
3. En depositar_billetes_50, se almacena la cantidad de billetes de dicha denominación, se utiliza una función llamada verificar_entero() para garantizar que el usuario ingresa un dato numérico de tipo entero.
4. En depositar_billetes_20, se almacena la cantidad de billetes de dicha denominación, se utiliza una función llamada verificar_entero() para garantizar que el usuario ingresa un dato numérico de tipo entero.
5. En depositar_billetes_10, se almacena la cantidad de billetes de dicha denominación, se utiliza una función llamada verificar_entero() para garantizar que el usuario ingresa un dato numérico de tipo entero.
6. En depositar_billetes_5, se almacena la cantidad de billetes de dicha denominación, se utiliza una función llamada verificar_entero() para garantizar que el usuario ingresa un dato numérico de tipo entero.

7. En depositar_billetes_2, se almacena la cantidad de billetes de dicha denominación, se utiliza una función llamada verificar_entero() para garantizar que el usuario ingresa un dato numérico de tipo entero.

8. En depositar_billetes_1, se almacena la cantidad de billetes de dicha denominación, se utiliza una función llamada verificar_entero() para garantizar que el usuario ingresa un dato numérico de tipo entero.

9. Se llama a la función actualizar_montos_cajeros(cajero_seleccionado, depositar_billetes_100, depositar_billetes_50, depositar_billetes_20, depositar_billetes_20, depositar_billetes_10, depositar_billetes_5, depositar_billetes_2, depositar_billetes_1)

10. Se retorna la función sumar_monto_depositado(depositar_billetes_100, depositar_billetes_50, depositar_billetes_20, depositar_billetes_20, depositar_billetes_10, depositar_billetes_5, depositar_billetes_2, depositar_billetes_1, cajero_seleccionado, usuario_usando_el_sistema)

Función verificar_entero(cantidad_de_billetes_depositoados)

Función encargada de verificar si el dato ingresa corresponde a un número entero, se requiere de un parámetro que es obtenido de la función depositar_dinero().

1. Con la palabra reservada try: se utiliza la variable cantidad_de_billetes_depositados y se almacena como un dato de tipo entero.

2. Invoca a la función verificar_cantidad_billetes(cantidad_de_billetes(cantidad_de_billetes_depositados

3. Con except: se usa para la manipulación de error. De la ayuda de raise junto con ValueError, se indica el error al usuario del sistema.

Función verificar_cantidad_billetes(cantidad_de_billetes_depositados)

La utilidad de esta función consiste en verificar que sean números de tipo entero y sean positivos. hace uso de un parámetro para almacenar la información digitada por el usuario.

1. En la variable cantidad_de_billetes_depositados, se convierten los datos de cantidad_de_billets_depositados en datos tipo entero.

2. Si cantidad_de_billetes_depositados es distinto del valor absoluto de cantidad_de_billetes_depositados de lo contrario, se hace uso de raise con ValueError para notificar al usuario de su posible error digitando la información.

Función buscar_linea_cajero(cajero_seleccionado, posicion_del_archivo)

Esta función es de utilidad para encontrar los datos dentro de un archivo utilizando el cajero digitado por el usuario. Se usa un primer parámetro que almacena el cajero elegido y la posición de la línea que por defecto es 0.

1. Con buscar_linea_archivo_cajero, se abre el archivo llamado "Lista_de_cajeros.txt".
2. En otra variable llamada lectura_linea_texto_cajero, se obtienen los elementos del archivo mediante el método .readlines(). S
3. e cierra el archivo con .close()
4. Con lineas_texto_cajero, se almacena la información de lectura_linea_texto_cajero utilizando posicion_del_archivo_cajero como índice.
5. Sí cajero_seleccionado se encuentra en lineas_texto_cajero.
   (a) Se devuelve la función posicion_del_archivo_cajero
6. Si no se cumple con la condición del punto 5, se usa la variable posicion_del _cajero y se incremente en uno.
   (a) Se retorna la función buscar_linea_cajero(cajero_seleccionado, posicion_del_archivo_cajero.

Función actualizar_montos_cajeros(cajero_seleccionado, cantidad_billetes_100, cantidad_billetes_50, cantidad_billetes_20, cantidad_billetes_10, cantidad_billetes_5, cantidad_billetes_2. cantidad_billetes_1)

Esta función se encarga de sumar y actualizar los montos del cajero luego de un depósito.

1. Con posicion_del_archivo_cajero, se alamacena el llamado a la función buscar_linea_cajero(cajero_seleccionado, 0).
2. Se abre el archivo "Lista_de_cajeros.txt con la variable actualizar_linea_cajero.
3. En leer_linea_texto_cajero, se usa la variable actualizar_linea_cajero junto al método .readline().
4. Con la variable linea_cajero, se busca la posición de la línea donde se encuentra los datos necesitados.
5. En linea_cajero se usa el método . replace(), para sustituir los corchetes, las comillas simples y los espacios con un espacio vacío.
6. Luego, con la variable lista_hilera_cajero_actualizado, se guarda la información de linea_cajero y se hace uso del método split() para convertir en lista y que sean separadas por comas.
7. Se hace uso de los índices 3, 5,7,9,11,13 y 15 en la variable lista_hileras_cajero_actualizado para sumar los billetes depositados a los ya existentes dentro del cajero.
8. Se cierra el archivo.
9. Y se retorna la función rescatar_lineas_cajero_cliente(o, '', lista_hilera_cajero_actualizado, cajero_seleccionado)

Función rescatar_lineas_cajero_cliente(posicion_cajero, datos_cajero_actualizados, lista_hileras_cajero_actualizado, cajero_seleccionado)

Se encarga de rescatar las líneas del archivo que no necesitan actualizar o modificar información.

1. Se abre el archivo a través de una variable rescatar_linea_archivo_cajero_cliente.
2. Usando readlines() se almacenan todos los elementos en la variable leer_linea_texto_cajero.
3. Se cierra el archivo.
4. Si posicion_cajero es menor a la longitud de leer_linea_texto_cajero, se hace los siguiente:
   (a) Se usa el valor de posicion_cajero como índice.
   (b) Mediante el método .replace() se reemplazan el salto de línea, las dobles comillas y el espacio con una espacion nulo.
      i. Si cajero_seleccioando se halla dentro de linea_a_revisar_cajero.
      ii. posicion_cajero se incrementa en uno.
      iii. Se retorna rescatar_lineas_cajero_cliente(posicion_cajero, datos_cajero_actualizados, lista_hileras_cajero_actualizado, cajero_seleccionado).
   (c) Si no se cumple las condiciones de punto i,
      i. datos_cajero_actualizados se suma con lista_hilera_cajero_actualizado.
      ii. Se regresa la función actualizar_datos_cajero_cliente(datos_cajero actualizados)

Función actualizar_datos_cajero_cliente(datos_cajero_actualizados)

Como lo menciona el nombre de la función esta se utiliza para actualizar los datos de los cajeros y utiliza como parámetro los datos almacenados por otra función.

1. Se abre el archivo "Lista_de_cajeros.txt".
2. Se usa el método .truncate(0) para vaciar el archivo.
3. Con el método .write() se escriben los nuevos datos al archivo.
4. Se cierra el archivo.

Función sumar_monto_depositado(depositar_billetes_100, depositar_billetes_50, depositar_billetes_20, depositar_billetes_10, depositar_billetes_5, depositar_billetes_2, depositar_billetes_10)

Como lo indica su nombre, suma todas las cantidades de billetes ingresados por el usuario.

1. En la variable monto_depositado, se guardará toda la información de la cantidad de billetes por denominación. Se multiplica por 100 la cantidad de billetes de 100, por 50 la cantidad de billetes de 5 y así consecutivamente hasta llegar a la cantidad de billetes de 1.
2. Se regresa la función actualizar_monto_usuario(usuario_usando_el_sistema, monto_depositado, cajero_seleccionado)

Función buscar_linea_usuario(usuario_usando_el_sistema, posicion_del_archivo_usuario)

Esta función encuentra donde están los datos en el archivo.

1. Se abre el archivo llamado "Lista_de_cajeros.txt".
2. Se utiliza el método readlines() para convertir en una lista todos los elementos del archivo.
3. Se cierra el archivo con close()
4. Se usa posicion_del_archivo_usuario como índice en la variable lectura_lineas_texto.
5. Si usuario_usando_el_sistema se encuentra dentro de lectura_lineas_texto.
   (a) Se retorna posicion_del_archivo_usuario
6. Si no se cumple con la condición del punto 5, se incrementa en uno la posicion_del_archivo.
   (a) Se retorna buscar_linea_usuario(usuario_usando_el_sistema, posicion_del_archivo_usuario)

Función actualizar_monto_usuario(usuario_usando_el_sistema, monto_depositado, cajero_seleccionado)

Se encarga de actualizar los montos que se vinculan con el usuario dentro del sistema.

1. Se hace el llamado a la función buscar_linea_usuario(usuario_usando_el_sistema, 0).
2. Se abre el archivo llamado "Lista_de_cajero.txt"
3. Se usa readlines() para acceder a los datos del archivo.
4. Con la variable linea, se usa el método replace para sustituir la comilla simple, los corchetes, el salto de línea con una espacio vacío.
5. En lista_hileras_usuario_actualizado, se guardan los datos en una lista y separados por coma usando elmétodo split().
6. En nuevo_monto, se usa la variable del punto 5 y se convierte en entero y a través del índice 5, se suma monto_depositado.

7. Se hace el cambio con los datos actualizado en la variable lista_hilera_usuario_actualizados propiamente en el índice 5 nuevamente.

8. En la variable deposito se concatena un espacio en blaco, el símbolo + y la información de la variable monto.

9. Con hora, se hace una concatenación de una espacio en blanco junto con time, importado de la librería time y hace uso del método asctime para guardar la fecha de esa transacción.

10. En cajero_seleccionado se hace otra concatenación de un espacio en blanco, la hilera "Acción realizada" y los datos de la variable cajero.

11. Si lista_hilera_usuario_actualizado en el índice 7 es igual a un espacio en blanco, se agregan los valores de deposito, hora y cajero e imprime por consola la variable lista_hileras_usuario_actualizado.

12. De no cumplirse el punto 11, la variable lista_hileras_usuario_actualizado en el índice 7 se sumarán la información de las variables depositos, hora, cajero_seleccionado.

13. Se cierra el archivo.

14. Se retorna la función rescatar_lineas_usuario(0. '', lista_hilera_usuario_actualizado, usuario_usando el sistema)

Función rescatar_lineas_usuario(posicion, datos_usuario_actualizados, lista_hilera_usuario_actualizado, usuario_usando_el sistema)

Función encargada de rescatar las líneas que no van a ser modificadas.

1. Se abre el archivo "Lista_de_cajero.txt"

2. Se usa readlines() para acceder a los datos del archivo.

3. Si posicion es menor al tamaño de leer_linea_texto_usuario.

   (a) linea_a_revisar utiliza como índice el valor de posicion.

   (b) Además, linea_a_revisar utiliza el método replace, para sustuir el salto de línea, las dobles comillas por una espacion vacío.

   (c) Se cierra el archivo.

      i. Si usuario_usando_el_sistema se encuentra en linea_a_revisar.

      ii. posicion aumenta en uno

      iii. Y se retorna rescatar_lineas_usuario(posicion, datos_usuario_actualizados, lista_usuario_actualizado, usuario_usando_el_sistema).

4. Si no se cumplen las condiciones del punto 3, la variable llamada datos_usuario_actualizados se concatena con lista_hileras_usuario_actualizado

   (a) Se regresa la función actualizar_datos_usuario(datos_usuario_actualizados)

Función actualizar_datos_usuario(datos_usuario_actualizados)

Actualiza los datos de los usuario en el archivo correspondiente.

1. Se abre el archivo "Lista_de_cajero.txt".
2. Se vacía el archivo con el método truncate()
3. Se reescribe los datos actualizados con el método writelines()
4. Se imprime por pantalla el estado de la transacción en este caso que la transacción ha sido exitosa.
5. Se cierra el archivo con close()
6. Se importa menu_principal y hace el llamado a la función correr_menu_principal.
7. Se imprime por pantalla que se está regresando al menú principal
8. Se retorna la función correr_menu_principal().

Función retirar_dinero(cajero_seleccionado, usuario_usando_el_sistema)

Función que se utiliza para realizar el retiro de dinero en una cajero del sistema.

1. Se muestra por pantalla la solicitud del usuario.
2. Con try, se gestionan las excepciones o errores que se puedan presentar en una situación determinada.
3. Con retirar_monto, se convierte en dato de tipo el entero la variable retirar_monto.
4. Si retirar_monto es un entero y además es distinto al valor absoluto de retirar_dinero, se indica el error con ValueError.
5. Si no se cumplen las condiciones del punto 4, se retorna la función verificar_posibilidad_del_retirar(cajero_seleccionado, retirar_monto, usuario_usando_el_sistema)
6. De cumplirse ni el punto 4, ni el 5, se hace uso de except para efectuar y mostrar por pantalla el error cometido por el usuario del sistema.

Función verificar_posibilidad_del_retiro(cajero_seleccionado, retirar_monto, usuario_usando_el_sistema)

Es la encargada de chequear si es posible realizar el retiro deseado por el cliente.

1. se hace el llamado a la función buscar_linea_cajero(cajero_seleccionado, 0)
2. Se abre el archivo "Lista_de_cajeros.txt"

3. Se hace uso de readlines() para tomar los datos que se encuentran dentro de los archivos.
4. Se cierra el archivo.
5. En la variable cajero_a_retirar, se utiliza la posicion_cajero_retirar como índice.
6. Con el método replace() se reemplazan los corchetes, las comillas simples, los espacios y el salto de línea con el espacio vacío.
7. A cajero_a_retirar se le aplica el método split() para convertir en una lista y separar los elementos por comas.
8. Se imprimen por pantalla todos las cantidades disponibles del cajero seleccionado.
9. En la variable dinero_disponible, se multiplican la cantidad de billetes elegidos para retirar y se multiplican por sus respectivos pesos.
10. Se abre nuevamente el archivo "Lista_de_cajero.txt"
11. Con readlines() se accede a los datos del archivo.
12. Se cierra el archivo.
13. Se invoca a la función buscar_linea_usuario(usuario_usando_el_sistema, 0)
14. En la variable verificar_saldo_a_retirar se emplea como índice la posicion_usuario_actual.
15. Con replace() se sustituyen los corchetes,las comillas simples y los espacios en blanco junto al salto de línea con el espacio nulo.
16. En hilera_del_saldo_usuario se guarda la información usando split() que convierte en lista y separa con comas.
17. Si dinero_disponible es mayor o igual a retirar_monto.
    (a) Y hilera convertido en entero es mayor o igual a retirar_monto.
    (b) Se retorna realizar_el_retiro(cajero_seleccionado, retirar_monto, hilera_del_cajero, 3, usuario_usando_el_sistema, retirar_monto, hilera_del_cajero)
18. Si no se cumple lo establecido en el punto 17, se informa al usuario la insuficiencia del monto que dispone a retirar e informar que no se puede realizar el retiro. Se regresa al menú principal.

Función obtener_denominacion(posicion_de_billetes)

Determina la denominación correspondiente de billetes.

1. monto_denominacion es igual a 0.
2. si posicion_de_billetes es igual 3.
3. monto_denominacion es igual a 100.
4. Si posicion_de_billetes es igual a 5.
5. monto_denominacion es igual a 50.
6. Si posicion_de_billetes es igual a 7.
7. monto_denominacion es igual a 20,
8. si posicion_de_billetes es igual a 9.
9. monto_denominacion igual a 10.
10. Si posicion_de_billetes es igual 11.
11. monto_denominacion es igual 5.
12. si posicion_de_billetes es igual a 13.
13. monto_denominacion igual a 2.
14. Si posicion_de_billetes es igual a 15.
15. monto_denominacion es igual a 1.
16. Se retorna monto_denominacion.

Función realizar_el_retiro(cajero_seleccionado, retirar_monto, hilera_del_cajero, posicion_de_billetes, usuario_usando_el sistema, retirar_monto_original, hilera_modificada)

Con esta función se permite realizar el retiro del usuario en el sistema.

1. Se invoca a la función obtener_denominacion con la posición.
2. Si retirar_monto es mayor a 0,
   (a) Y si la posicion_de_billetes es menor o igual a 15.
   (b) Se almacena en cantidad los datos de hilera_del_cajero_de_billetes junto a posicion_de_billetes.
      i. Si la division entera de retirar_monto y monto_denominacion es distinta de cero y cantidad_billetes es distanta de 0.
      ii. Se hace la resta a la denominación correspondiente.

iii. Se actualiza el monto de billetes por denominación.

iv. Si monto_restante es mayor a 0.

v. posicion_de_billetes se incrementa en 2 unidades.

vi. Se retorna realizar_el_retiro(cajero_seleccionado, monto_restante, hilera_del_cajero, posicion_de_billetes, usuario_usando_el_sistema, retirar_monto_original, hilera_modificada)

vii. Si no se cumple con el punto A, se invoca a la función rescatar_lineas_cajero_cliente(0, '', hilera_modificada, cajero_seleccionado)

viii. Retornar actualizar_monto_usuario_con_retiro()

3. En caso de no cumplir con las condiciones del punto 2.2.3.1, se imprime por pantalla que la operación no pudo realizarse.

4. Se devuelve al menú principal en caso de no poder completarse la operación.

5. Se regresa al menú principal si el usuario desea hacer un retiro de 0 pesos.

Función actualizar_monto_usuario_con_retiro(usuario_actual, monto_retirado, cajero_seleccionado)

Con esta función se actualiza la hilera que posee los datos del usuario con la información del retiro.

1. Se abre el archivo de Lista_de_usuarios.txt como lectura y escritura.

2. Se hace llamado a la función buscar_linea_usuario ya previamente creada, para averiguar la hilera en donde se encuentra el usuario actual.

3. Se hace el reemplazo correspondiente de los elementos que no nos interesan del archivo y se hace un split para obtenerlo como lista.

4. Se indexa la posición en donde se encuentra el saldo del usuario y se le hace la resta correspondiente con el monto retirado. Y se hace actualización del saldo del usuario,

5. Se crea las variables que van a guardar los elementos importantes a incluir en el historial de transacciones, desde el monto retirado (-), la fecha y hora (usando la función de la librería time) y el enunciado que comenta sobre la ubicación donde se hizo la transacción.

6. Se hace un retorno a la función previamente creada rescatar_lineas_usuario para ya finalmente, actualizar el archivo txt.

Función consultar_saldo(usuario_usando_el_sistema)

Con esta función se muestra el saldo disponible del usuario.

1. Se abre el archivo de Lista_de_usuarios.txt como lectura y escritura.

2. Se hace llamado a la función buscar_linea_usuario ya previamente creada, para averiguar la hilera en donde se encuentra el usuario actual.
3. Se hace el reemplazo correspondiente de los elementos que no nos interesan del archivo y se hace un split para obtenerlo como lista.
4. Se hace impresión del título "Monto Disponible:" y se indexa la posición en donde se encuentra el saldo del usuario.
5. Luego de ser mostrado el monto disponible se vuelve al menú principal.

Función historial_transacciones(usuario_usando_el_sistema)

Función que se encarga de buscar sólo la información del historial del usuario actual usando el sistema.

1. Se abre el archivo de Lista_de_usuarios.txt como lectura y escritura.
2. Se hace llamado a la función buscar_linea_usuario ya previamente creada, para averiguar la hilera en donde se encuentra el usuario actual.
3. Se hace el reemplazo correspondiente de los elementos que no nos interesan del archivo y se hace un split para obtenerlo como lista.
4. Se indexa la posición donde se encuentra el historial en la hilera.
5. Si el historial está vacío.
   (a) Se imprime un mensaje notificando al usuario que el historial está vacío por lo que no lo puede revisar.
   (b) Se vuelve al menú principal.
6. De no cumplirse el punto 5
   (a) Se hace llamado a la función que imprime los títulos que van a abarcar la tabla del historial.
   (b) Luego de esto se hace llamado a la función que se encarga de separar los datos del historial a su respectivo título, enviándole como parámetro la hilera con el historial y un cero (posición inicial a revisar).

Función imprimir_titulos()

Función encargada de imprimir los títulos de la tabla del historial de transacciones.

1. Con los parámetros recibidos se utilizan los métodos ljust(), cente()r y rjust() para acomodar los datos a su columna correspondiente con medidas para que quede simétrico.
2. Luego de ser acomodado se imprimen estos valores para que tomen su lugar en la tabla.

Función separar_transaccion(hilera_con_historial, posicion_str_historial)

Función encargada de separar los datos del historial de transacciones para que sea asignado a su posición en la tabla.

1. Con hilera_con_historial hacemos una separación de los datos por espacios.
2. Con la función len se saca la longitud de la lista_con_historial.
3. Si la posición de la hilera es menor o igual a la longitud de la hilera menos uno.
   (a) Se crea y se asigna a la variable cantidad la hilera indexada en la primera posición. (La primera posición es la que posee el monto del retiro o depósito)
   (b) Se crea y se asigna a la variable fecha_y_hora las cinco posiciones siguientes (los datos de la fecha y hora siempre abarcan estas 5 posiciones), cada valor separado por espacios.
   (c) Se crea y se asigna la variable ubicación las dos siguientes valores después de los valores que poseen la fecha y hora. Separándolas con espacios.
   (d) Se hace llamado a la función previamente creada imprimir_datos con los valores recién encontrados.
      i. Se verifica si la posición (posición que posee el primer valor de la transacción) más siete es menor que la longitud de la hilera menos uno. (Se hace más siente ya que cada transacción tiene 8 elementos, contando el cero) (Se hace hilera menos uno para ver si aún quedan transacciones por recorrer)
      ii. Se suma a la posición inicial ocho, para seguir con la siguiente transacción a imprimir,
   (e) De no cumplirse el punto i, se termina de mostrar el historial de transacciones, se vuelve al menú principal.

2 Pruebas

2.1 Pruebas para la creación de usuarios

2.1.1 Prueba de creación correcta de usuarios

1. El sistema pregunta si el usuario es un cliente del banco, o un banquero.
   (a) El usuario selecciona la opción de banquero.
2. El sistema despliega las opciones disponibles para un banquero.
   (a) El banquero selecciona la opción crear usuario.

3. El sistema pregunta al banquero el nombre de usuario que se desea crear.
   (a) El banquero ingresa el nombre de usuario daya2004!
4. El sistema verifica el nombre de usuario ingresado, acepta el usuario, y genera un pin con 4 dígitos, de forma aleatoria. El sistema informa al banquero del pin generado en pantalla. Además, se guarda los datos del usuario en el archivo Lista_de_usuarios.txt
5. Se vuelve al menú principal.

Resultado esperado de la prueba: Verificar que el archivo con la información de los usuarios Lista_de_usuarios.txt contenga la información del nuevo usuario agregado. Además el sistema debe haber notificado que el usuario fue creado exitosamente.

2.1.2 Pruebas para la creación y manipulación de cajeros

1. El sistema despliega la opción de indicar si es un cliente de banco o un banquero.
   (a) El usuario indica que es un banquero.
2. El sistema despliega las opciones disponibles para un banquero.
   (a) El banquero selecciona la opción de crear cajero.
3. El sistema despliega un mensaje solicitando el nombre de usuario que desea ingresar al sistema.
   (a) El banquero ingresa el identificador del cajero del cajero: AAA0
4. El sistema verifica si el identificador del cajero cumple con los requisitos y que sea único. Lo acepta.
5. El sistema solicita la cantidad de billetes por denominación.
   (a) El banquero ingresa 1 billete de 100 pesos, 1 billete de 50 pesos, 1 billete de 20 pesos, 1 billete de 10 pesos, 1 billete de 5 pesos, 1 billete de 2 pesos, 1 billete de 1 peso.
6. El sistema al haber aceptado el identificador del cajero, guarda en un archivo txt. llamado Lista_de_cajeros el identificador del cajero con la cantidad de billetes de cada denominación ingresada por el banquero.
7. El sistema le avisa al banquero que el cajero fue creado exitosamente.
8. El sistema vuelve al menú principal.

Resultado esperado de la prueba: Que el archivo Lista_de_cajeros.txt contenga la información del nuevo cajero agregado (identificador AAA0 y con la cantidad de billetes por denominación de 1 billete de 100 pesos, 1 billete de 50 pesos, 1 billete de 20 pesos, 1 billete de 10 pesos, 1 billete de 5 pesos, 1 billete de 2 pesos, 1 billete de 1 peso.).

2.1.3 Prueba de agregar billetes a un cajero existente de forma correcta

Pre-requisitos de la prueba: Se debe ejecutar la Prueba 2.1 de creación correcta de cajeros.

1. El sistema despliega la opción de indicar si es un cliente del banco o un banquero.
   (a) El usuario indica que es un banquero.
2. El sistema despliega las opciones disponibles para un banquero.
   (a) El banquero selecciona la opción de agregar billetes a un cajero existente.
3. El sistema despliega un mensaje solicitando el identificador del cajero que desea agregarle billetes.
   (a) El banquero ingresa el identificador AAA0
4. El sistema pregunta por la cantidad de billetes a depositar por cada denominación.
5. El banquero ingresa 2 billetes de 100 pesos. El sistema notifica al usuario que los billetes fueron depositados con éxito al cajero.
6. Se vuelve al menú principal.

Resultado esperado de la prueba: El archivo Lista_de_cajeros.txt contiene una línea al menos donde se haya actualizado la información de la cantidad de billetes por denominación del cajero con identificador AAA0. La cantidad de billetes por denominación debe ser: 3 billetes de 100 pesos, 1 billete de 50 pesos, 1 billete de 20 pesos, 1 billete de 10 pesos, 1 billete de 5 pesos, 1 billete de 2 pesos, 1 billete de 1 peso.

2.2 Prueba de agregar billetes a un cajero existente de forma correcta

Pre-requisitos de la prueba: Se debe ejecutar la Prueba 2.1 de creación correcta de cajeros.

1. El sistema despliega la opción de indicar si es un cliente del banco o un banquero.
   (a) El usuario indica que es un banquero.
2. El sistema despliega las opciones disponibles para un banquero.
   (a) El banquero selecciona la opción de agregar billetes a un cajero existente.

3. El sistema despliega un mensaje solicitando el identificador del cajero que desea agregarle billetes.

   (a) El banquero ingresa el identificador AAA0

4. El sistema pregunta por la cantidad de billetes a depositar por cada denominación.

   (a) El banquero ingresa 2 billetes de 100 pesos.

5. El sistema notifica al usuario que los billetes fueron depositados con éxito al cajero.

6. Se vuelve al menú principal.

Resultado esperado de la prueba: El archivo Lista_de_cajeros.txt contiene una línea al menos donde se haya actualizado la información de la cantidad de billetes por denominación del cajero con identificador AAA0. La cantidad de billetes por denominación debe ser: 3 billetes de 100 pesos, 1 billete de 50 pesos, 1 billete de 20 pesos, 1 billete de 10 pesos, 1 billete de 5 pesos, 1 billete de 2 pesos, 1 billete de 1 peso.

2.3 Pruebas de la interacción del usuario del banco con los cajeros

2.3.1 Prueba de retirar dinero incorrectamente.

Pre-requisitos de la prueba: Se debe ejecutar la Prueba 2.1 de creación correcta de cajeros y se debe ejecutar la prueba de Prueba 1.1 de creación correcta de usuarios.

1. El sistema despliega la opción de indicar si es un cliente del banco o un banquero.

   (a) El usuario indica que es un cliente del banco.

2. El sistema le solicita ingresar un identificador de cajero en donde desea hacer su retiro.

   (a) El usuario indica que AAA0

3. El sistema verifica que es un cajero existente.

4. El sistema le solicita ingresar su nombre de usuario

   (a) El usuario ingresa daya2004!

5. El sistema verifica que el nombre de usuario exista y esté correcto.

6. El sistema le solicita ingresar su pin.

   (a) El usuario ingresa 1234

7. El sistema verifica que el pin esté correcto.
8. El sistema despliega las opciones disponibles para un cliente del banco.
   (a) El cliente selecciona la opción de retirar dinero.
9. El sistema le pregunta el monto a retirar.
   (a) El cliente ingresa -5
10. El sistema verifica que sea un número entero y positivo, ve que no cumple y tira error informando de que la cantidad es inválida.

Resultado esperado de la prueba: el sistema tira el error informando que la entrada es inválida, ya que no es una cantidad válida de dinero a retirar.

2.4 Prueba correcta de depositar dinero

Pre-requisitos de la prueba: Se debe ejecutar la Prueba 2.1 de creación correcta de cajeros y se debe ejecutar la prueba de Prueba 1.1 de creación correcta de usuarios.

1. El sistema despliega la opción de indicar si es un cliente del banco o un banquero.
   (a) El usuario indica que es un cliente del banco.
2. El sistema le solicita ingresar un identificador de cajero en donde desea hacer su retiro.
   (a) El usuario indica que AAA0
3. El sistema verifica que es un cajero existente.
4. El sistema le solicita ingresar su nombre de usuario
   (a) El usuario ingresa daya2004!
5. El sistema verifica que el nombre de usuario exista y esté correcto.
6. El sistema le solicita ingresar su pin.
   (a) El usuario ingresa 1234
7. El sistema verifica que el pin esté correcto.
8. El sistema despliega las opciones disponibles para un cliente del banco.
   (a) El cliente selecciona la opción de depositar dinero.
9. El sistema le pregunta la cantidad de billetes por denominación a depositar.

   (a) El cliente ingresa 1 billete de 100 y 1 billete de 50.

10. El sistema verifica que sea un número entero y positivo.

11. El sistema ve que sí cumple y procede.

12. El sistema actualiza el monto de billetes del cajero y actualiza el saldo del usuario y su historial de transacciones.

13. Se realiza de forma correcta el depósito del dinero, mostrando un mensaje al usuario sobre esto.

Resultado esperado de la prueba: El archivo Lista_de_cajeros.txt contiene una línea al menos donde se haya actualizado la información de la cantidad de billetes por denominación del cajero con identificador AAA0. La cantidad de billetes por denominación debe ser: 4 billetes de 100 pesos, 2 billetes de 50 pesos, 1 billete de 20 pesos, 1 billete de 10 pesos, 1 billete de 5 pesos, 1 billete de 2 pesos, 1 billete de 1 peso. El archivo Lista_de_usuarios debe de tener una línea al menos donde se haya actualizado la información del saldo del cliente daya2004!. El saldo debe de ser 150.

2.5 Prueba correcta de ver el monto disponible

Pre-requisitos de la prueba: Se debe ejecutar la Prueba 2.1 de creación correcta de cajeros y se debe ejecutar la prueba de Prueba 1.1 de creación correcta de usuarios.

1. El sistema despliega la opción de indicar si es un cliente del banco o un banquero.

   (a) El usuario indica que es un cliente del banco.

2. El sistema le solicita ingresar un identificador de cajero en donde desea hacer su retiro.

   (a) El usuario indica que AAA0

3. El sistema verifica que es un cajero existente.

4. El sistema le solicita ingresar su nombre de usuario

   (a) El usuario ingresa daya2004!

5. El sistema verifica que el nombre de usuario exista y esté correcto.

6. El sistema le solicita ingresar su pin.

   (a) El usuario ingresa 1234

7. El sistema verifica que el pin esté correcto.

8. El sistema despliega las opciones disponibles para un cliente del banco.

(a) El cliente selecciona la opción de consultar saldo.

9. El sistema despliega un mensaje diciendo "Monto disponible: 150"

Resultado esperado de la prueba: El sistema debe de mostrar el saldo disponible en la cuenta del usuario.

2.6 Prueba incorrecta de ver el historial de transacciones

Pre-requisitos de la prueba: Se debe ejecutar la Prueba 2.1 de creación correcta de cajeros y se debe ejecutar la prueba de Prueba 1.1 de creación correcta de usuarios.

1. El sistema despliega la opción de indicar si es un cliente del banco o un banquero.
   (a) El usuario indica que es un cliente del banco.
2. El sistema le solicita ingresar un identificador de cajero en donde desea hacer su retiro.
   (a) El usuario indica que AAA0
3. El sistema verifica que es un cajero existente.
4. El sistema le solicita ingresar su nombre de usuario
   (a) El usuario ingresa daya2004!
5. El sistema verifica que el nombre de usuario exista y esté correcto.
6. El sistema le solicita ingresar su pin.
   (a) El usuario ingresa 1234
7. El sistema verifica que el pin esté correcto.
8. El sistema despliega las opciones disponibles para un cliente del banco.
   (a) El cliente selecciona la opción de consultar historial de transacciones.
9. El sistema despliega el historial de transacciones del usuario de forma tabulada, con los títulos "Transaccion realizada", "Fecha y hora de la transaccion" y "Cajero donde se hizo la transaccion". con la información respectiva del historial de transacciones del usuario.

Resultado esperado de la prueba: El sistema debe de mostrar de forma tabulada el historial de transacciones del usuario.

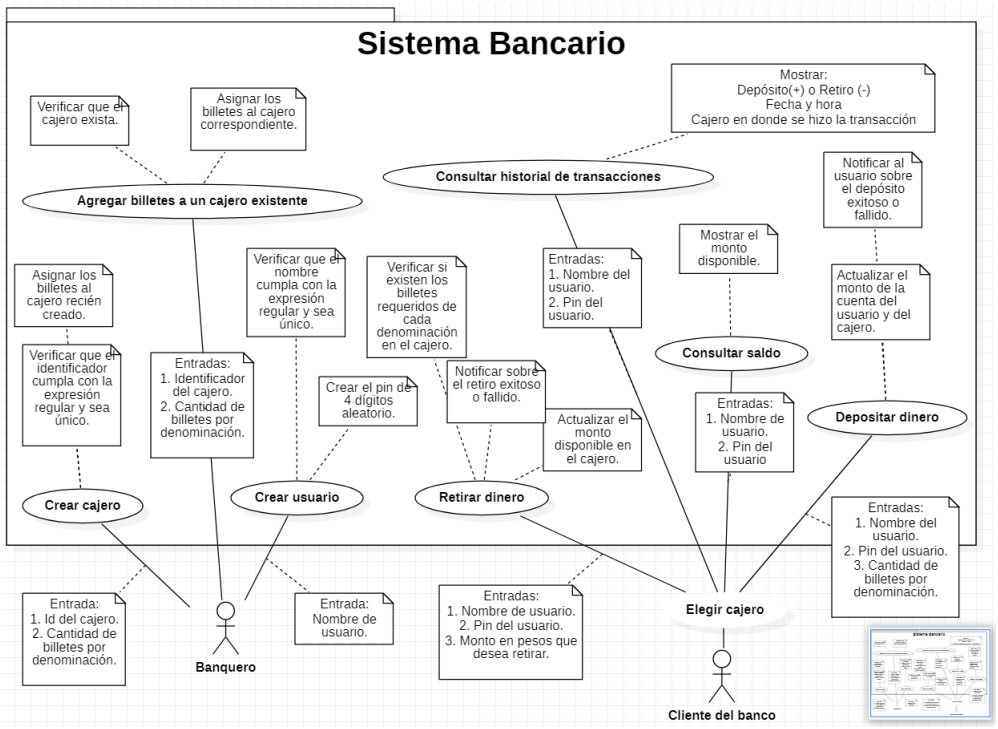

Figure 1: Diagrama UML del trabajo práctico.

3 Diagrama usado para el trabajo práctico

Se creó por medio de la aplicación Star Uml un diagrama de casos de uso del sistema bancario a implementar. En este diagrama intentamos abarcar todas las acciones posibles a realizar de parte del banquero, del cliente y roles del sistema. Se agregó todo lo que vimos oportuno de recordar, siempre manteniendo una idea general de lo que requería el programa.

4 Bibliografía

Singh. (2018). Python String | ljust(), rjust(), center(). GeeksforGeeks.Recuperado de https://www.geeksforgeeks.org/python-string-ljust-rjust-center/

Python Software Fundation (2022). Referencias del lenguaje python. https://docs.python.org/es/3/reference/index.html

W3School. (2022), Python tutorial. https://www.w3schools.com/python/default.asp