**Instituto Tecnológico de Costa Rica**

**Tarea 02: Transacciones Bancarias - SML**

Curso:

IC4700 – Lenguajes de programación

Profesor:

Allan Rodríguez Dávila

Estudiante:

**Mary Paz Álvarez Navarrete**

Carné:  
2023138604

II Semestre 2024

**Manual de usuario**

Para compilar el program se escribe la ruta donde esta el archivo “/programa/” y se escribe sml “programa.sml” después se escribe “main();” y listo el programa se ha iniciado.

**Menú Principal**

Al iniciar el programa, se mostrará el menú principal con tres opciones:

**1. Creador**: Para agregar o limpiar registros en un archivo CSV.

**2. Analizador**: Para analizar y obtener reportes sobre las transacciones.

**3. Salir**: Para salir del programa.

Elija una opción ingresando el número correspondiente.

**Menú Creador**

Desde el menú Creador, puede realizar las siguientes acciones:

**1. Añadir nuevo registro**: Permite agregar una nueva transacción. Se solicita la siguiente información:

* + Ruta del archivo CSV.
  + Número de cuenta.
  + Tipo de transacción: Depósito, Retiro o Transferencia.
  + Monto de la transacción.
  + (En caso de ser transferencia) Número de cuenta de destino.

Tras ingresar los datos, el registro se añade al archivo.

**2. Limpiar índice**: Elimina los registros.

**3. Volver**: Retorna al menú principal.

**Menú Analizador**

El menú Analizador permite revisar y generar informes sobre las transacciones. Incluye las siguientes opciones:

**1. Mostrar top**: Permite ver las transacciones donde el monto esté dentro de un rango específico. El sistema ordena las transacciones de mayor a menor monto y las presenta en formato tabular.

* + Se solicita la ruta del archivo CSV.
  + Ingrese el monto máximo y mínimo para el rango.

**2. Informe de actividades sospechosas**: Genera un reporte de actividades consideradas sospechosas. (Actualmente no sirve).

**3. Transacciones por cuenta**: Filtra y muestra todas las transacciones asociadas a una cuenta específica, ya sea como cuenta de origen o de destino.

* + Se solicita la ruta del archivo CSV.
  + Ingrese el número de cuenta a analizar.

**4. Cantidad de transacciones por tipo**: Muestra la cantidad de transacciones según el tipo: depósitos, retiros o transferencias.

* + Se solicita la ruta del archivo CSV.
  + Elija el tipo de transacción ingresando el número según las instrucciones dadas.

**5. Resumen**: Muestra un reporte con:

* + Cantidad de transacciones por tipo.
  + La transacción con el monto más alto.
  + La transacción con el monto más bajo.

**6. Volver**: Regresa al menú principal.

**Pruebas de funcionalidad**

**Descripción del problema**

El objetivo de esta tarea es desarrollar un sistema de procesamiento de transacciones bancarias utilizando un lenguaje de programación funcional SML. El sistema debe procesar las transacciones de forma eficiente y ser capaz de ejecutar análisis en tiempo real.

**Diseño del programa**

El diseño del programa en un solo archivo con un flujo principal basado en menús es una elección que permite una implementación simple y organizada. Cada acción que realiza el usuario se canaliza a través de menús, lo que facilita la navegación y el uso del programa, al mismo tiempo que reduce la complejidad del código. Aunque este enfoque puede no escalar bien en proyectos más grandes, es altamente efectivo para proyectos pequeños como este.

**Librerías Usadas**

1. **TextIO**:

* **TextIO.openAppend**: Abre un archivo para añadir contenido.
* **TextIO.output**: Escribe en un archivo.
* **TextIO.closeOut**: Cierra un archivo de escritura.
* **TextIO.openIn**: Abre un archivo para leer.
* **TextIO.inputAll**: Lee todo el contenido de un archivo.
* **TextIO.inputLine**: Lee una línea del estándar de entrada o de un archivo.
* **TextIO.closeIn**: Cierra un archivo de lectura.
* **TextIO.stdIn**: Representa la entrada estándar (teclado).

1. **String**:

* **String.substring**: Extrae un subcadena de una cadena dada.
* **String.tokens**: Divide una cadena en tokens basados en un delimitador.
* **String.concatWith**: Combina una lista de cadenas con un separador específico.

1. **List**:

* **List.nth**: Accede a un elemento de la lista por su índice.
* **List.tl**: Devuelve la cola de una lista (todos los elementos excepto el primero).
* **List.length**: Devuelve la longitud de una lista.
* **List.filter**: Filtra una lista basándose en una función predicado.
* **List.map**: Aplica una función a cada elemento de una lista.

1. **Real**:

* **Real.fromString**: Convierte una cadena en un número de punto flotante (**real**).

1. **Time**:

* **Time.now**: Devuelve la hora actual del sistema.

1. **Date**:

* **Date.fromTimeLocal**: Convierte un valor de tiempo en una fecha local.
* **Date.fmt**: Formatea una fecha en un formato específico.

1. **OS.Process**:

* **OS.Process.exit**: Finaliza el programa.

1. **MergeSort**

Se uso el código pueste en el siguiente repositorio: <https://github.com/amir734jj/SML-NJ-sorting-algorithms/blob/master/sml-sort.sml>

**Análisis de resultados**

|  |  |
| --- | --- |
| Creación menú principal | **✓** |
| Creación menús secundarios | **✓** |
| Registro nuevo | **✓** |
| Eliminar registros | **✓** |
| Mostrar top | **✓** |
| Actividad sospechosa | **x** |
| Transferencia por cuenta | **✓** |
| Cantidad de transacciones por tipo | **✓** |
| Resumen | **\*** |

**Objetivos no alcanzados**

**Actividad sospechosa:** No se llegó a implementar por mala organización del tiempo.

**Resumen:** Falto la parte de cuenta de origen con más transacciones y cuenta destino con mayor monto recibido. Al igual que el resto de los objetivos no se implementaron por mala organización del tiempo.

**Repositorio de Github**

<https://github.com/Mary-Paz-AN/TR2-SML.git>