UNIVERSIDAD ARGENTINA DE LA EMPRESA (UADE)

KIWILLET

Billetera virtual – Proyecto de Programación I

Carrera: Ingeniería en Informática

Materia: Programación I / Algoritmos y Estructura de Datos I

Profesor/a: Ing. María Eugenia Varando

Turno: Noche – 2° Cuatrimestre 2025

Integrantes

- Massun, Felipe / 1195389
- Scala Merani, Damián / 1139436
- Arias Castroman, Santiago / 1224594

Descripción general del proyecto:

Kiwillet busca simular el funcionamiento básico de una billetera virtual, pero ejecutada de forma local, todos los datos y registros se manejan en archivos de texto. El programa permite a un único usuario crear su cuenta, iniciar sesión, administrar sus tarjetas, realizar pagos de servicios, efectuar transferencias, ingresar dinero y consultar su saldo.

Objetivos:

Kiwillet tiene como objetivo hacer una billetera virtual que permita al usuario gestionar sus finanzas de forma simple y rápida.

Los objetivos del programa son:

1. Que el usuario pueda visualizar su saldo, registrar ingresos y egresos, y realizar pagos de servicios.
2. Mostrar las tarjetas registradas por el usuario además de poder dar el alta y baja de cada tarjeta.
3. Sistema de pagos de servicios.
4. Registrar la actividad del usuario en archivos locales.
5. Seguridad de los datos del usuario, mediante contraseñas, validaciones de ingreso y control de errores en tiempo de ejecución.
6. Mostrar informes que permitan al usuario conocer el saldo, pagos realizados y movimientos en la cuenta.

Menú:

1. **Consultar Saldo:** Muestra el monto de dinero disponible en la cuenta.
2. **Ingresar Dinero:** Permite agregar fondos a la cuenta. Se requiere que el monto sea mayor a cero ($ > 0). Esta acción actualiza el saldo y deja un registro del movimiento.
3. **Transferir:** Envía dinero a otro usuario mediante un alias (el alias debe tener entre 3 y 20 caracteres). Se verifica la disponibilidad de fondos antes de realizar el descuento y se registra la transacción.
4. **Pagar Servicio:** Facilita el pago de facturas de servicios básicos, incluyendo Agua, Gas y Luz.
5. **Gestionar Tarjetas:** Ofrece sub-opciones para la administración de tarjetas:
    - Ver las tarjetas asociadas a la cuenta.
    - Añadir una nueva tarjeta.
    - Eliminar una tarjeta existente.
6. **Cambiar Contraseña:** Proceso para establecer una nueva clave de acceso (debe tener entre 4 y 12 caracteres), previa validación de la contraseña actual.
7. **Ver Reportes:** Acceso a informes detallados sobre la actividad y los movimientos de la cuenta.
8. **Simulador Plazo Fijo:** Herramienta de proyección financiera
9. **Salir:** Finaliza la sesión del usuario en el sistema y registra el evento de cierre en el log de actividad.

Funciones implementadas en el sistema

Log(texto)

- Propósito: Registra en ArchivoLog.txt los eventos o errores del sistema con sello de tiempo ISO.
- Entrada: texto (str) → Mensaje descriptivo.
- Retorna: Escribe en archivo ArchivoLog.txt.

leerUsuario()

- Lee el archivo Usuario.txt y devuelve un diccionario con nombre, contraseña y saldo.
- Si el archivo no existe o el formato es inválido, se registra el error en el log.
- Retorna: diccionario con datos del usuario o error en log.

escribirUsuario(usuario)

- Guarda los datos del usuario en Usuario.txt.
- Entrada: diccionario con claves nombre, contraseña, saldo.
- Retorna: True si se guardó correctamente, False en caso de error.

ingresarNombre(texto, minimo, maximo)

- Solicita un nombre o alias por consola, validando longitud y no vacío.
- Retorna: string con el nombre validado.

ingresarContraseña(texto, minimo, maximo)

- Solicita contraseña validando longitud.
- Retorna: contraseña ingresada.

crearCuenta()

- Crea una nueva cuenta con nombre, contraseña y saldo inicial cero.
- Llama a escribirUsuario() y registra el evento en log.
- Retorna: True si la cuenta se creó correctamente.

verificarExistenciaUsuario()

- Comprueba si el archivo Usuario.txt existe.
- Retorna: True o False.

login()

- Solicita nombre y contraseña, los compara con los guardados en Usuario.txt.
- Retorna: True si el login es exitoso, False si las credenciales son incorrectas.

loopLogin()

- Cicla hasta que el login sea exitoso. Luego accede al menú principal (menuPrincipal()).

mostrarSaldo(usuario)

- Imprime en pantalla el saldo actual del usuario.
- Entrada: diccionario usuario.

modificarContraseña()

- Permite cambiar la contraseña actual del usuario tras verificar la existente.
- Retorna: True si se modifica correctamente.

---

leerTarjetas()

- Carga todas las tarjetas desde Tarjetas.txt en un diccionario indexado por código.
- Retorna: diccionario de tarjetas.

escribirTarjetas(tarjetas)

- Guarda todas las tarjetas en Tarjetas.txt.
- Retorna: True si la operación fue exitosa.

tipoTarjeta()

- Permite elegir entre tipos de tarjeta: VISA, MASTERCARD o AMEX.
- Retorna: tipo de tarjeta (str).

validarCodigo(minimo, maximo, diccTarjetas)

- Pide el código de una nueva tarjeta, verificando longitud, dígitos y no duplicación.

- Retorna: código de tarjeta válido.

agregarTarjeta()

- Registra una nueva tarjeta pidiendo tipo, número, titular y vencimiento.
- Valida formato y guarda en archivo.
- Efectos secundarios: Escribe en Tarjetas.txt y ArchivoLog.txt.

verTarjetas()

- Muestra todas las tarjetas registradas, ordenadas por tipo.
- Los números se enmascaran mostrando solo los últimos 4 dígitos.
- Retorna: True si existen tarjetas, False si no hay registros.

eliminarTarjeta()

- Permite eliminar una tarjeta según su código.
- Actualiza el archivo y registra el evento.

menuTarjetas()

- Submenú con opciones para ver, agregar o eliminar tarjetas.
- Llama a las funciones anteriores.

---

pagarServicio()

- Simula el pago de servicios básicos (“Agua”, “Gas”, “Luz”).
- Genera de forma aleatoria el monto adeudado y permite un pago parcial.
- Actualiza saldo y registra movimiento.
- Retorna: True si se completa el pago.

---

registrarMovimientos(descripcion, monto, saldo_final)

- Guarda en Movimientos.txt una línea con fecha, descripción, monto y saldo resultante.

obtenerMovimientos()

- Carga todos los movimientos del archivo y los devuelve en una lista de diccionarios.

- Retorna: lista con movimientos válidos.

ingresarDinero()

- Permite acreditar fondos al saldo del usuario.
- Actualiza Usuario.txt y Movimientos.txt.
- Retorna: True si se registró correctamente.

transferir()

- Permite transferir dinero a otro alias, validando saldo y monto.
- Descuenta el monto y registra el movimiento.
- Retorna: True si se completó la transferencia.

mostrarReportes()

- Muestra los últimos 5 movimientos ordenados por fecha.
- Calcula y presenta saldo promedio, total de ingresos y egresos.
- Retorna: True si existen movimientos.

calcularPlazoFijo(saldo, meses)

- Función recursiva que calcula el crecimiento del saldo aplicando un 5% mensual.
- Retorna: saldo proyectado luego de meses iteraciones.

simularPlazoFijo()

- Solicita cantidad de meses y muestra capital estimado y ganancia proyectada.
- Retorna: True si la simulación se realizó correctamente.

menuPrincipal()

- Interfaz central del programa que coordina todas las operaciones.
- Muestra opciones numeradas del 1 al 9 y ejecuta la función correspondiente.
- Registra toda acción en ArchivoLog.txt.
- Retorna: False al salir del sistema.

iniciarAplicacion()

- Verifica la existencia de un usuario. Si no existe, crea una cuenta.
- Luego inicia el ciclo de login (loopLogin()).
- Es el punto de arranque de toda la aplicación.

main()

- Importa el módulo principal (Mudulo) y ejecuta iniciarAplicacion().
- Es el entry point del programa.

Reportes esperados:

1. Informe estadístico con:
   - Promedios de saldo.
   - Total de gastos y pagos.
   - Evolución del saldo.
2. Archivo LOG con todos los movimientos realizados.
3. Resumen mensual de movimientos y balances.

Persistencia de datos

El sistema almacena la información del usuario y las operaciones realizadas en archivos de texto plano. Cada archivo cumple un propósito específico:

Entrada/Salida
- Usuario.txt: contiene las credenciales y el saldo actual.

Entrada/Salida
- Tarjetas.txt: registra las tarjetas asociadas al usuario.

Entrada/Salida
- Movimientos.txt: guarda cada transacción con fecha, descripción y saldo resultante.

Salida
- ArchivoLog.txt: bitácora que documenta todos los eventos relevantes del sistema.

Github:

https://github.com/Lospumas25/Progra1