A plastic bag with a picture of a oval

Description automatically generated

***Proceso de Desarrollo***

El desarrollo del código para Kamaba Pay fue un proceso metódico y reflexivo que implicó varias etapas, desde la planificación inicial hasta la implementación y pruebas finales. Este proyecto se enfocó en crear una aplicación financiera que permitiera la gestión de dinero, transferencias y pagos, además de un sistema de movimientos.

1. Planificación y Diseño:

En la fase inicial, se definieron los requisitos del sistema y se estableció una estructura básica de datos. Se decidió utilizar estructuras para representar tanto a los usuarios como a los movimientos, lo que permitiría una manipulación más eficiente de los datos. También se diseñó un menú interactivo para facilitar la navegación por las distintas funcionalidades.

2. Implementación:

A medida que se avanzaba en la codificación, surgieron diversos desafíos. Uno de los errores más significativos fue al validar los diferentes campos ingresados por el usuario. Inicialmente, no se implementó correctamente dichas validaciones. Esto llevó a que se aceptaran entradas inválidas, lo que causaba problemas al intentar realizar transferencias. Después de identificar este problema, se ajustó las validaciones para asegurarnos que cada dato fuese ingresado correctamente.

3. Manejo de Archivos:

Al implementar la funcionalidad de guardar movimientos en un archivo, nos enfrentamos a otro error relacionado con la escritura de datos. Al principio, el programa sobrescribía el archivo existente en lugar de agregar nuevos movimientos. Esto se debió a no abrir el archivo en modo de adición. Tras solucionar este problema, el archivo ahora se abre correctamente en modo "append", permitiendo la conservación de datos previos.

4. Pruebas y Optimización:

Una vez implementadas las funciones principales, se realizaron pruebas para asegurarnos de que todo funcionara como se esperaba. Durante esta fase, se descubrieron que algunos cálculos de movimientos no se actualizaban correctamente debido a errores en la lógica de los ciclos. Ajusté las condiciones y las variables involucradas para garantizar que los cálculos fueran precisos.

El proceso de creación del código para Kamaba Pay fue uno de aprendizaje continuo, donde cada error proporcionó una oportunidad para mejorar y optimizar el sistema y nuestros conocimientos.

***Aplicación de Administración de Dinero:***

***“Kamaba Pay”***

Este código implementa un sistema de gestión financiera denominado “Kamaba Pay”, diseñado para proporcionar servicios de transferencia y administración de dinero de manera similar a plataformas como Mercado Pago. La aplicación permite a los usuarios registrarse, gestionar su saldo, realizar transferencias, y efectuar pagos. Además, incluye funcionalidades más específicas, como la generación y validación de CUIT, el uso de archivos binarios y de texto para persistencia de datos, y un sistema de gestión de movimientos financieros que pueden ser filtrados por criterios como fecha, tipo, monto, etc.

El sistema soporta diferentes tipos de usuarios, cada uno con un rol específico: administradores, quienes tienen acceso a todas las funcionalidades del sistema, y usuarios regulares, quienes pueden acceder a las opciones de transacciones y gestión de su cuenta. La estructura de datos está diseñada para garantizar la precisión y la seguridad en el manejo de información sensible, con validaciones rigurosas de datos personales y financieros como el CUIT, número de teléfono, correo electrónico y alias.

Aquí se explica cada sección y variable a fondo.

*Inclusión de Bibliotecas y Declaración de Funciones*

Las bibliotecas estándar permiten:

• stdio.h: entrada y salida de datos (por ejemplo, printf y scanf).

• stdlib.h: manejo de memoria dinámica y control de procesos (malloc, exit).

• string.h: funciones para manejar cadenas (strcpy, strlen, strcmp, etc.).

• time.h: funciones de tiempo (time, srand).

• ctype.h: funciones para manipular caracteres (isdigit, isalpha).

• unistd.h: control del sistema y operaciones de archivos.

void crear\_usuario();

void ingresar();

int user\_registry(const void \*field, char tipo);

void menu\_adm();

void ingresar\_dinero(usuario \*usa);

void transferencia(usuario \*usa);

void menu\_ing(usuario \*usser);

void pagar\_servicios(usuario \*usa);

void movimiento(char movtxt[100], char opcn, const usuario \*usa,float monto);

void filtros(const usuario \*usa);

void mdatos(usuario \*usa);

void admin(char tipo, usuario \*usuwu);

Aquí se declaran algunas funciones que se definirán más adelante en el código.

*Estructura de usuario*

typedef struct {

char cuit[23];

char nombre[52];

char num[14];

char email[50];

char CVU[23];

char alias[6];

char IVA[50];

char contra[50];

float saldo;

char rol;

} usuario;

Esta estructura representa a un usuario con los siguientes campos:

• cuit: cadena para almacenar el CUIT, de hasta 22 caracteres (más un terminador \0).

• nombre: nombre completo del usuario (máximo 51 caracteres).

• num: número de celular con código de área (13 caracteres).

• email: dirección de correo electrónico (49 caracteres).

• CVU: Clave Virtual Uniforme, usada en transferencias (22 caracteres).

• alias: nombre corto asignado al usuario (5 caracteres).

• IVA: cadena para almacenar el estado de verificación de IVA.

• contra: contraseña (hasta 49 caracteres).

• saldo: saldo actual en la cuenta (valor decimal).

• rol: rol del usuario (carácter que indica el tipo de usuario).

*Estructuras de cuentabanco y movimientos*

typedef struct {

char type;

char CBU[23];

char cuitl[23];

char clave1[4];

} cuentabanco;

Esta estructura define los datos de una cuenta bancaria:

• type: tipo de cuenta.

• CBU: número de cuenta (22 caracteres).

• cuitl: CUIT asociado a la cuenta.

• clave1: clave corta asociada.

typedef struct {

char tipo1;

float monto;

char iibb[50];

char fech[50];

char cuilt[50];

} movimientos;

Esta estructura define una transacción:

• tipo1: tipo de movimiento.

• monto: cantidad de dinero involucrada.

• iibb: impuestos y tasas asociados (Ingresos Brutos).

• fech: fecha del movimiento.

• cuilt: CUIT del usuario involucrado.

*main() e Inicialización de Archivos*

int main() {

char opc;

FILE \*archivo = fopen("user.dat", "rb");

En main(), se inicia abriendo el archivo user.dat, que almacena datos de usuarios. opc es una variable que captura la opción seleccionada por el usuario en el menú principal.

if (archivo == NULL) {

archivo = fopen("user.dat", "wb");

if (archivo == NULL) {

printf("Error al abrir el archivo de registros.\n");

exit(0);

} else {

usuario user;

strcpy(user.nombre, "Administrador");

strcpy(user.num, "0000000000000");

strcpy(user.email, "admin0@gmail.com");

strcpy(user.cuit, "00000");

strcpy(user.alias, "Ad000");

strcpy(user.contra, "contrasena");

user.rol = '0';

if (fwrite(&user, sizeof(usuario), 1, archivo) != 1) {

printf("Error al escribir los datos en el archivo\n");

}

}

fclose(archivo);

}

Si user.dat no existe, se crea y se añade un usuario “Administrador” con datos iniciales. El rol de '0' indica que es un administrador.

*Menú Principal y Selección de Opciones*

printf("\nSi no se especifica la opcion de salir, todo proceso puede ser detenido ingresando 0.\n\n");

repeat:

printf("\t\t\tBienvenido a Kamaba Pay!\n\n\t\t\tEscoja que desea hacer:\n\n\t\t\t1. Crear Usuario.\n\t\t\t2. Ingresar.\n\t\t\t3. Salir.\n");

scanf("%s", &opc);

El menú principal ofrece tres opciones: crear usuario, ingresar o salir del programa. Si el usuario ingresa 0 en algún momento, el proceso se detiene.

6*. Función crear\_usuario()*

Esta función registra un nuevo usuario en el sistema.

• Validación de Nombre: Se asegura de que el nombre solo contenga letras y espacios, que tenga una longitud entre 2 y 50 caracteres, y comience con una letra mayúscula.

• Validación de Número de Celular: Verifica que el número tenga 13 dígitos, comenzando con el código de área “549”.

• Validación de Correo Electrónico: Asegura que el correo solo contenga letras, números, y símbolos específicos (@, .), y esté en uno de los dominios permitidos (gmail, hotmail, yahoo, outlook).

• Validación de CUIT: Comprueba que el CUIT tenga una longitud de 11 dígitos y comience con un prefijo permitido (20, 27, 30). También realiza un cálculo de verificación de CUIT para garantizar su validez.

• Generación de CVU y Alias:

• Se genera un CVU único para el usuario utilizando valores aleatorios.

• El alias se genera automáticamente a partir de los primeros caracteres del nombre y CUIT.

*user\_registry - Validación de Datos Únicos*

int user\_registry(const void \*field, char tipo);

Esta función busca si el dato ingresado (como el nombre o CUIT) ya existe en el archivo de usuarios para evitar duplicados.

*Escritura en Archivo*

Finalmente, el usuario se guarda en user.dat y se actualiza el archivo ADMINvalidar\_IVA.txt para registrar los nuevos CUITs, que necesitarán ser verificados para que el usuario pueda ingresar.

Se implementa una función llamada ingresar que se encarga de autenticar a un usuario en un sistema en base a varios métodos de identificación (email, número de celular, CUIT/L o como administrador):

*Estructura General y Variables*

• Variables:

• opci: guarda la opción que el usuario elige para autenticarse.

• band, band1, error: son banderas que controlan el flujo de los bucles de entrada y validación de datos.

• uso, usr: variables de tipo usuario. uso representa los datos que el usuario introduce y usr los datos leídos desde el archivo.

• tip: una variable char que indica el tipo de dato (email, número de celular, etc.) que se está validando.

• arci: un puntero a un archivo que apunta a user.dat, el archivo donde están almacenados los datos de usuario.

• Apertura del archivo:

FILE \*arci = fopen("user.dat", "rb");

Abre el archivo user.dat en modo lectura binaria (rb). Si el archivo no se abre, muestra un mensaje de error y termina la función con return.

*Menú de Opciones de Ingreso*

Se muestra un menú de opciones al usuario para elegir el método de autenticación:

1. Ingreso con Email.

2. Ingreso con Número de Celular.

3. Ingreso con CUIT/L.

4. Salir.

*Lógica de Opciones con switch*

La función utiliza un switch para manejar cada opción:

Opción 1: Ingreso con Email

1. Solicitar Email:

• Pide al usuario que ingrese su email. Si ingresa "0", el proceso se detiene.

• El email se almacena en uso.email.

2. Validar Email:

• Llama a user\_registry(uso.email, 'e') para validar el email. Si la función retorna 1, el email no está registrado, mostrando un mensaje de error. Si la validación es exitosa, pasa al siguiente paso.

3. Solicitar Contraseña:

• Pide al usuario que ingrese su contraseña (almacenada en uso.contra). Si ingresa "0", el proceso se detiene.

• Llama a user\_registry(uso.contra, 'o') para validar la contraseña. Si falla, muestra un mensaje de error.

4. Comprobar Email y Contraseña en el Archivo:

• Con rewind(arci), el puntero del archivo se reinicia al principio.

• Se leen los datos de user.dat para verificar si existe una coincidencia exacta entre el email y la contraseña.

• Si se encuentra un usuario con email y contraseña correctos:

• Se muestra un mensaje de ingreso exitoso.

• Llama a menu\_ing(&uso), función que presumiblemente redirige al usuario autenticado a su menú principal.

• La bandera error se establece en 0, y se termina el bucle.

• Si error sigue siendo 1, muestra un mensaje de error indicando que la contraseña pertenece a otra cuenta.

5. Cierre de Archivo:

• Si se validó correctamente, el archivo se cierra con fclose(arci).

Opción 2: Ingreso con Número de Celular

Este caso sigue un flujo muy similar a la opción de email:

1. Solicita al usuario su número de celular (uso.num).

2. Si el número es válido, solicita la contraseña.

3. Compara el número y contraseña en user.dat.

4. Si la autenticación es exitosa, llama a menu\_ing(&uso). De lo contrario, muestra mensajes de error adecuados.

Opción 3: Ingreso con CUIT/L

Este caso es similar al ingreso con email y número de celular:

1. Solicita al usuario su CUIT/L (uso.cuit).

2. Si es válido, solicita la contraseña.

3. Compara el CUIT/L y contraseña en user.dat.

4. Si la autenticación es exitosa, llama a menu\_ing(&uso). De lo contrario, muestra mensajes de error adecuados.

Opción 0: Ingreso como Administrador

1. Verificación de CUIT/L y Contraseña de Administrador:

• Solicita el CUIT/L y la contraseña.

• Si estos son válidos y pertenecen a un administrador (usr.rol == '0'), llama a menu\_adm() para otorgar acceso al menú de administración.

• Si la autenticación falla, muestra mensajes de error.

Opción 4: Salir

• Esta opción permite al usuario salir del sistema, mostrando un mensaje y retornando de la función.

Default: Entrada Inválida

• Cualquier entrada inválida redirige al usuario al menú principal con el comando goto repeat.

*Menú de Usuario*

void menu\_ing(usuario \*usser) {

Define una función menu\_ing que recibe un puntero a una estructura usuario. Este puntero (\*usser) permite modificar la información del usuario directamente.

*Declaración de Variables*

• opcion: almacena la opción seleccionada por el usuario.

• pos: guarda la posición en el archivo donde se encuentra el usuario actual.

• opcn: no se usa en este código, podría ser un error o una variable sobrante.

• movtxt[100]: se declara, pero no se usa.

• usuario usa; es una copia de \*usser. La variable usa se actualiza con los cambios que el usuario realiza en cada operación.

*Apertura de Archivo*

FILE \*arch = fopen("user.dat", "rb+");

if (arch == NULL) {

printf("\nError al abrir el archivo de usuarios.\n");

return;

}

Se abre el archivo user.dat en modo lectura y escritura binaria (rb+). Si no se encuentra o no se puede abrir, se muestra un mensaje de error y se sale de la función.

*Verificación del Estado del Usuario*

if ((strcmp(usser->IVA, "En validacion")) != 0) {

Verifica si el campo IVA del usuario tiene el valor "En validacion". Si el usuario no está en proceso de validación, se le permite continuar.

*Menú de Opciones*

printf("\n\t\tSeleccione la opcion deseada:\n\t\t1. Transferencia\n\t\t2. Ingresar Dinero\n\t\t3. Pagar Servicios\n\t\t4. Consultar y Modificar Datos\n\t\t5. Consultar movimientos\n\t\t6. Salir\n\n");

fflush(stdin);

scanf("%c", &opcion);

Se muestran las opciones disponibles y se lee la opción seleccionada por el usuario.

*Switch Case: Opciones del Menú*

Cada caso representa una opción del menú, y cada uno llama a una función diferente para realizar la operación correspondiente.

1. Transferencia

transferencia(&usa);

if (usser->saldo != usa.saldo) {

printf("\nSu saldo anterior era %.2f. Su nuevo saldo es %.2f\n", usser->saldo, usa.saldo);

rewind(arch);

while (fread(usser, sizeof(usuario), 1, arch) == 1) {

if (strcmp(usser->cuit, usa.cuit) == 0) {

pos = ftell(arch) - sizeof(usuario);

fseek(arch, pos, SEEK\_SET);

usser->saldo = usa.saldo;

if (fwrite(usser, sizeof(usuario), 1, arch) != 1) {

printf("\nError al escribir los datos en el archivo\n");

}

break;

}

}

}

fclose(arch);

sleep(2);

system("cls");

goto repeat;

1. Se llama a la función transferencia(&usa), que modifica el saldo del usuario (usa).

2. Si el saldo cambió, se actualiza el archivo:

• Se vuelve al inicio del archivo con rewind(arch).

• Lee cada usuario en el archivo hasta encontrar el que coincide con el cuit del usuario actual.

• Usa fseek para posicionarse en el lugar correcto y escribe el saldo actualizado con fwrite.

2. Ingresar Dinero

Este caso es similar al anterior, pero llama a ingresar\_dinero(&usa), que permite al usuario ingresar dinero a su cuenta.

3. Pagar Servicios

Igual que los casos anteriores, llama a pagar\_servicios(&usa) para restar el monto de un servicio del saldo del usuario.

4. Modificar Datos

mdatos(&usa);

if (memcmp(&usser, &usa, sizeof(usuario)) != 0) {

printf("\nCambios realizados con exito\n");

rewind(arch);

while (fread(usser, sizeof(usuario), 1, arch) == 1) {

if (strcmp(usser->cuit, usa.cuit) == 0) {

pos = ftell(arch) - sizeof(usuario);

fseek(arch, pos, SEEK\_SET);

if (fwrite(&usa, sizeof(usuario), 1, arch) != 1) {

printf("\nError al escribir los datos en el archivo\n");

}

break;

}

}

} else {

printf("\nNo han habido cambios.\n");

}

fclose(arch);

sleep(2);

system("cls");

goto repeat;

1. mdatos(&usa) permite al usuario modificar sus datos.

2. Compara el usuario original (usser) con usa. Si hay cambios (memcmp != 0):

• Se actualiza el archivo con los nuevos datos de usa.

5. Consultar Movimientos

filtros(&usa);

goto repeat;

Llama a filtros(&usa), función que muestra los movimientos del usuario. Luego, vuelve al menú.

6. Salir

printf("Saliendo...");

sleep(1);

return;

Imprime un mensaje de salida y termina la función.

Opción No Válida

default:

system("cls");

printf("\nOpcion invalida. Escoja de las siguientes:\n\n");

goto repeat;

break;

Si el usuario selecciona una opción no válida, limpia la pantalla, muestra un mensaje de error y vuelve al menú.

*Sin Verificación de IVA*

} else {

printf("\nUsted no posee la verificacion necesaria para continuar.\nEspere o contacte a nuestro servicio al cliente: barbararoza60@gmail.com\n\n");

sleep(1);

}

}

Si el usuario no está verificado (IVA es "En validacion"), muestra un mensaje y no le permite continuar.

*Transferencias: Definición y Variables Iniciales*

void transferencia(usuario \*usa){

float monto;

long pos;

char movtxt[100];

char opcn;

float desc;

usuario uscompar;

system("cls");

repeat:

desc = 0;

1. Parámetro usuario \*usa: Es un puntero a la estructura usuario, que representa al usuario que está realizando la transferencia.

2. Variables:

• monto: Cantidad de dinero a transferir.

• pos: Almacena la posición del archivo para actualizar el saldo del receptor.

• movtxt: Texto de la descripción del movimiento.

• opcn: Tipo de operación, en este caso, ‘t’ (transferencia).

• desc: Monto deducido por impuestos.

• uscompar: Variable de tipo usuario que se utiliza para comparar y actualizar datos del receptor.

Apertura del Archivo y Menú de Transferencia

FILE \*archive = fopen("user.dat", "rb+");

if (archive == NULL){

printf("Error al abrir el archivo para comparar usuarios durante transferencia.");

return;

}

1. Apertura de archivo: Se abre user.dat en modo lectura y escritura (rb+), que contiene los datos de los usuarios.

2. Validación de apertura: Si el archivo no se abre correctamente, muestra un error y termina la función.

Opciones de Transferencia

char entrada[50];

char opi;

int encon = 1;

printf("\n\t\tSeleccione como desea transferir:\n");

printf("\n\t\t1. Con CVU o alias:\n\t\t2. Con celular o email.\n\t\t3. Salir.\n\n");

fflush(stdin);

scanf("%c", &opi);

fflush(stdin);

1. Opciones de transferencia:

• 1: Transferir mediante el CVU o alias del receptor.

• 2: Transferir mediante el celular o email del receptor.

• 3: Salir.

Caso 1: Transferencia con CVU o Alias

Si el usuario selecciona 1, se ejecuta el siguiente código:

case '1':

if (usa->saldo > 0) {

printf("\nIngrese el CVU o alias de la persona a transferir: ");

fflush(stdin);

gets(entrada);

while (fread(&uscompar, sizeof(usuario), 1, archive)) {

if ((strcmp(uscompar.CVU, entrada) == 0) || (strcmp(uscompar.alias, entrada) == 0)) {

printf("\nUsted va a transferir dinero a %s\n", uscompar.nombre);

encon = 1;

break;

} else

encon = 0;

}

if (encon != 0) {

1. Ingreso del CVU o alias: Se ingresa la identificación del receptor en entrada.

2. Búsqueda en el archivo: Se recorre el archivo buscando un usuario con CVU o alias que coincida con entrada.

3. Confirmación: Si encuentra un usuario, muestra su nombre.

*Realización de la Transferencia*

Si se encuentra el usuario receptor:

rewind(archive);

while (fread(&uscompar, sizeof(usuario), 1, archive) == 1) {

if ((strcmp(uscompar.CVU, entrada) == 0) || (strcmp(uscompar.alias, entrada) == 0)) {

pos = ftell(archive) - sizeof(usuario);

fseek(archive, pos, SEEK\_SET);

printf("\nIngrese la cantidad de dinero a transferir: ");

scanf("%f", &monto);

if (usa->saldo >= monto) {

usa->saldo -= monto;

uscompar.saldo += monto;

1. Reinicio de lectura y posición en archivo: Retrocede al inicio para actualizar los datos del receptor.

2. Cantidad de transferencia: Se solicita al usuario ingresar la cantidad de dinero a transferir y valida que usa tenga saldo suficiente.

*Descuento por IVA*

Si el receptor tiene ciertos valores en el campo IVA, se descuenta un 5% de la transferencia:

if ((strcmp(uscompar.IVA, "Sujeto Exento") == 0) || ... ) {

uscompar.saldo += (monto - (monto \* 0.05));

printf("\nSe le ha descontado un 5 porciento a su transferencia debido a la tasa IIBB del otro usuario.\n");

desc = monto \* 0.05;

}

1. Condición de IVA: Si el receptor tiene un tipo de IVA específico, se calcula una retención del 5%.

2. Actualización de desc: Guarda el monto descontado.

*Guardado de Datos*

Finalmente, se actualiza el archivo con los nuevos saldos y se registra la transferencia.

fseek(archive, pos, SEEK\_SET);

if (fwrite(&uscompar, sizeof(usuario), 1, archive) != 1) {

printf("\nError al escribir los datos en el archivo\n");

} else {

opcn = 't';

if (desc > 0) {

snprintf(movtxt, sizeof(movtxt), "Transferencia recibida de %.2f de %s a %s, con una retencion de %.2f", monto, usa->cuit, uscompar.cuit, desc);

movimiento(movtxt, opcn, &uscompar, monto);

uscompar = \*usa;

snprintf(movtxt, sizeof(movtxt), "Transferencia de %.2f a %s ce %s, con una retencion de %.2f", monto, uscompar.cuit, usa->cuit, desc);

movimiento(movtxt, opcn, &uscompar, monto);

} else {

snprintf(movtxt, sizeof(movtxt), "Transferencia recibida de %.2f de %s a %s", monto, usa->cuit, uscompar.cuit);

movimiento(movtxt, opcn, &uscompar, monto);

snprintf(movtxt, sizeof(movtxt), "Transferencia de %.2f a %s ce %s", monto, uscompar.cuit, usa->cuit);

uscompar = \*usa;

movimiento(movtxt, opcn, &uscompar, monto);

}

printf("\nSu transferencia fue exitosa.\n");

}

}

1. Registro de movimiento: Utiliza movimiento para crear un registro en el historial de movimientos de ambos usuarios.

2. Confirmación al usuario: Se muestra que la transferencia fue exitosa.

*Casos Restantes del Menú Principal*

El código repite la misma estructura para:

• Transferencia por Celular o Correo: Realiza la transferencia usando el número de teléfono o correo del receptor.

• Salir: Finaliza la función.

*Ingresar Dinero*

La función ingresar\_dinero() permite a un usuario ingresar dinero a su cuenta a través de una cuenta bancaria vinculada. Además, permite vincular una nueva cuenta si es necesario. A continuación, se describe cada parte de la función:

1. Declaración de Variables y Configuración Inicial:

• La función recibe un puntero a un usuario (usa).

• Se definen varias variables, como opcio para la opción de menú seleccionada, contc como contador de cuentas vinculadas, clave para almacenar la clave de la cuenta, cbu para el código CBU, tipo para el tipo de cuenta (ahorro o corriente), monto para el monto de dinero a ingresar, y movtxt para guardar una descripción del movimiento.

• La función intenta abrir el archivo cuentabanco.dat en modo lectura binaria para acceder a las cuentas bancarias del usuario. Si el archivo no se abre correctamente, muestra un error y sale de la función.

2. Menú de Opciones:

• El programa presenta un menú donde el usuario puede seleccionar una de las siguientes opciones:

• 1: Ingresar dinero a una cuenta vinculada.

• 2: Vincular una nueva cuenta.

• 3: Salir de la función.

3. Opción 1 - Ingresar Dinero con Cuenta Vinculada:

• La función muestra las cuentas vinculadas del usuario leyendo el archivo cuentabanco.dat y buscando coincidencias entre el CUIT de la cuenta y el CUIT del usuario.

• Si el usuario tiene cuentas vinculadas, se le solicita seleccionar una de ellas ingresando el número correspondiente.

• Luego, el usuario ingresa el monto que desea transferir a su cuenta. El saldo del usuario (usa->saldo) se actualiza sumando el monto ingresado.

• Se crea una descripción del movimiento (movtxt) usando snprintf y se llama a la función movimiento() para registrar la transacción.

4. Opción 2 - Vincular una Nueva Cuenta:

• El programa solicita al usuario ingresar el tipo de cuenta (1 para ahorro, 2 para corriente) y el CBU de la cuenta a vincular.

• Luego, pide la clave de la cuenta. Si el CBU y la clave cumplen con los requisitos, el programa busca en cuentabanco.dat para ver si la cuenta ya existe.

• Si la cuenta existe y la clave es correcta, se vincula al usuario. Si no existe, se agrega una nueva entrada en el archivo cuentabanco.dat con los datos de la cuenta y el CUIT del usuario.

• Si la clave ingresada es demasiado larga (más de 3 caracteres), muestra un mensaje de error y solicita una clave válida.

5. Opción 3 - Salir:

• Esta opción permite al usuario salir de la función.

6. Control de Errores:

• Si el usuario ingresa una opción de menú no válida, la función reinicia el menú para solicitar una opción válida.

*Pagar servicios*

La función pagar\_servicios() permite a un usuario pagar diferentes tipos de servicios, aplicando cargos adicionales si corresponden. A continuación, se detalla su funcionamiento:

1. Configuración Inicial:

• La función recibe un puntero usuario \*usa.

• Define variables como op para seleccionar el servicio, opcn para indicar el tipo de operación, movtxt para almacenar la descripción del movimiento, deuda (sin usar en este código), y monto para el monto a pagar.

• Se crea una copia del usuario (movstru) para registrar el movimiento.

2. Menú de Servicios:

• El programa muestra un menú con las opciones de pago disponibles:

• 1: Expensas Médicas

• 2: Servicio Telefónico

• 3: Expensas del Hogar

• 4: Salir del menú de servicios

• El usuario selecciona una opción ingresando un valor que se almacena en op.

3. Pago de Servicios:

• Dependiendo del valor de op, la función ejecuta diferentes casos:

• Caso 1: Expensas Médicas:

• Informa al usuario que debe consultar el monto exacto en la página del hospital.

• Pide al usuario ingresar el monto a pagar (monto).

• Si el saldo del usuario (usa->saldo) es suficiente para cubrir el monto, se procede con el pago:

• Se deduce el monto del saldo del usuario.

• Si el usuario está categorizado bajo ciertos tipos de IVA (Sujeto Exento, Monotributista Social, Responsable Monotributo o Responsable Inscripto), se le añade un recargo del 5% por Ingresos Brutos (IIBB), siempre y cuando el saldo del usuario pueda cubrir el monto adicional.

• La operación se registra mediante la función movimiento() con una descripción detallada en movtxt, indicando el monto, el CUIT y el recargo aplicado.

• Si el saldo no es suficiente para cubrir el monto ingresado, informa de saldo insuficiente y regresa al menú.

• Caso 2: Servicio Telefónico:

• Similar al caso anterior, le indica al usuario consultar el monto en la página de su proveedor.

• Se verifica el saldo del usuario para cubrir el monto; si es suficiente, se deduce el monto y, si corresponde, se aplica el recargo del 5% por IIBB.

• La operación se registra en movimiento() con una descripción específica, detallando el tipo de servicio y el monto total con el recargo si se aplica.

• Si el saldo es insuficiente, se informa al usuario y vuelve al menú.

• Caso 3: Expensas del Hogar:

• El proceso es similar a los anteriores. Indica al usuario que consulte la página de Consorcio Abierto para obtener el monto exacto.

• Pide el monto a pagar y verifica si el saldo es suficiente.

• En caso afirmativo, se deduce el monto del saldo y, si el usuario pertenece a una categoría con recargo, se deduce el 5% adicional.

• Se llama a movimiento() para registrar la operación con detalles específicos.

• Caso 4: Salir:

• Permite al usuario salir de la función y volver al menú principal.

4. Control de Errores:

• Si el usuario ingresa una opción inválida, la función muestra un mensaje de error y regresa al menú de selección (goto repeat).

*Modificación de Datos*

Se implementa la función mdatos, que permite al usuario ver y modificar ciertos datos de su perfil (nombre, correo electrónico, número de celular, contraseña y alias). Esta función muestra un menú interactivo y permite al usuario elegir qué campo modificar, asegurándose de que los datos introducidos cumplan con ciertas restricciones y validaciones. A continuación, se detalla cada sección del código.

1. Definición de la función y variables iniciales:

void mdatos(usuario \*usa) {

char opciowo;

usuario user;

char tipo;

system("cls");

int band = 1;

int i = 0;

int contnomb = 0;

int gm = 0;

int pos, punto = 0, pos1 = 0;

char gmail[50];

• opciowo: Guarda las opciones elegidas en los menús.

• user: Estructura usuario auxiliar para recibir los datos a modificar.

• Variables de control como band, i, contnomb, gm, pos, punto, pos1, y gmail son usadas para validar entradas y almacenar datos temporales.

2. Panel de datos inicial:

printf("\n\t\t---Panel de Datos---\n");

printf("Sus datos son los siguientes:\n");

printf("Nombre: %s\n", usa->nombre);

printf("Correo: %s\n", usa->email);

printf("Celular: %s\n", usa->num);

printf("CUIT/L: %s\n", usa->cuit);

printf("Contrasena: %s\n", usa->contra);

printf("Alias: %s\n", usa->alias);

printf("CVU: %s\n", usa->CVU);

printf("Condicion fiscal: %s\n", usa->IVA);

• Se muestran los datos actuales del usuario usando el puntero usa.

3. Menú principal:

printf("\nEscoja que desea hacer:\n");

printf("\n\t\t1. Modificar Datos.\n\t\t2. Salir.\n");

fflush(stdin);

scanf("%c", &opciowo);

• Aquí, el usuario puede optar por modificar datos (1) o salir (2).

4. Opciones de modificación de datos:

switch(opciowo) {

case '1':

system("cls");

printf("\n\t\t---Panel de de Datos---\n");

printf("\nEscoja el dato a modificar:\n");

printf("\n\t\t1. Nombre\n\t\t2. Correo\n\t\t3. Numero\n\t\t4. Contrasena\n\t\t5. Alias.\n\t\t6. Salir\n");

fflush(stdin);

scanf("%c", &opciowo);

• Dependiendo de la opción seleccionada (1 al 5), el programa permitirá modificar un campo específico del perfil. Si se elige 6, se sale del panel de datos.

5. Validación y modificación de cada campo:

• Nombre (case '1'):

printf("\nIngrese su nombre completo: ");

• Solicita un nombre completo, verifica que solo contenga letras y espacios, y que la longitud esté entre 2 y 50 caracteres. El primer carácter debe ser una letra mayúscula.

• Correo electrónico (case '2'):

printf("\nIngrese su correo electronico: ");

• Valida que el correo tenga formato correcto: solo letras, números, @, y un dominio específico (como gmail, yahoo, hotmail, outlook). También verifica que termine en .com.

• Número de celular (case '3'):

printf("\nIngrese su numero completo de celular. Agregando el codigo de area: ");

• Verifica que el número tenga 13 dígitos y que comience con el prefijo 549.

• Contraseña (case '4'):

printf("\nCree una contrasena (Solo letras y numeros, al menos 8 caracteres de longitud.):\n");

• Requiere una contraseña de entre 8 y 49 caracteres, compuesta solo por letras y números.

• Alias (case '5'):

printf("\nCree un alias (Solo letras y numeros).\n");

• El alias debe tener exactamente 5 caracteres y solo puede contener letras y números.

6. Manejo de opciones y errores:

default:

system("cls");

printf("\nOpcion invalida. Volviendo al panel de datos. \n");

break;

• En el caso de una opción inválida, muestra un mensaje de error y vuelve al menú principal.

7. Retorno al menú inicial y salida:

sleep(2);

system("cls");

goto repeat;

• El goto repeat; se usa para regresar al menú de opciones principal después de cada modificación o si el usuario elige una opción incorrecta.

*Movimientos y Filtros*

Se implmentan dos funciones: movimiento y filtros. La primera registra un movimiento en archivos de texto y binarios, mientras que la segunda permite buscar movimientos específicos en el archivo de texto.

void movimiento(char movtxt[100], char opcn, const usuario \*usa, float monto)

Esta función registra un movimiento (transferencia, ingreso o pago) en un archivo de texto específico para cada usuario y en un archivo binario compartido. Los pasos son los siguientes:

1. Inicialización de Variables:

• movimientos movim: estructura que contiene los datos del movimiento.

• archname y move: arreglos de caracteres para almacenar el nombre del archivo de texto del usuario y el movimiento formateado respectivamente.

• fechtim: cadena para almacenar la fecha y hora actuales.

• tim y tm\_info: variables para capturar el tiempo actual y formatearlo.

2. Obtener Fecha y Hora Actual:

• Se usa strftime para formatear la fecha y hora en la variable fechtim en formato dd/mm/aaaa hh:mm:ss.

3. Abrir el Archivo de Texto:

• archname se construye con el CUIT del usuario (usa->cuit) y la extensión .txt.

• Si el archivo no puede abrirse en modo de anexado ("a"), se muestra un mensaje de error.

4. Registrar el Movimiento en el Archivo de Texto:

• move almacena la fecha, hora y descripción del movimiento (movtxt).

• Si move tiene más de 25 caracteres, se escribe en el archivo de texto, y si ocurre un error, se notifica al usuario.

• Se cierra el archivo de texto.

5. Registrar el Movimiento en el Archivo Binario:

• El archivo binario movimientos.dat se abre en modo de anexado binario ("ab").

• Los datos del movimiento (tipo, fecha, CUIT, monto y condición fiscal) se almacenan en la estructura movim.

• Si el archivo binario no puede abrirse, se muestra un mensaje de error.

• El movimiento se escribe en el archivo binario, y si ocurre un error, se notifica al usuario.

• Se cierra el archivo binario.

6. Tipos de Movimiento (opcn):

• Dependiendo del valor de opcn, que indica el tipo de movimiento ('t' para transferencia, 'i' para ingreso, 'p' para pago), se ejecuta la misma lógica de registro en archivos.

void filtros(const usuario \*usa)

Esta función permite buscar movimientos específicos en el archivo de texto del usuario.

1. Inicialización de Variables:

• string y temp: cadenas para la búsqueda y lectura de cada línea del archivo, respectivamente.

• archnom: almacena el nombre del archivo de texto basado en el CUIT del usuario (usa->cuit).

• enc: indicador de si se encuentra un registro correspondiente a la búsqueda.

2. Abrir el Archivo de Texto:

• Se abre el archivo de texto del usuario en modo lectura ("r").

• Si no existe, se muestra un mensaje indicando que no hay movimientos.

3. Buscar el Movimiento:

• Solicita al usuario un término de búsqueda que puede ser una fecha, tipo de movimiento o “0” para mostrar todos.

• La búsqueda no distingue mayúsculas de minúsculas, por lo que convierte ambas cadenas (string y temp) a minúsculas usando tolower.

• Lee línea por línea el archivo, buscando el término de búsqueda en cada línea.

• Si encuentra coincidencias, imprime la línea; de lo contrario, muestra un mensaje indicando que no se encontraron registros.

4. Resultado de la Búsqueda:

• Si enc es 0 (no se encontró coincidencia), muestra un mensaje informando al usuario.

*Administrador*

Función menu\_adm()

1. Declaración de variables:

• cuit[23]: un arreglo de caracteres para almacenar el CUIT (Código Único de Identificación Tributaria) de un usuario.

• usuario usuwu: una estructura que probablemente contiene información sobre el usuario.

• char opciwi: variable para almacenar la opción seleccionada por el administrador.

• char tipo: tipo de operación que se va a realizar (modificación, adjudicación, etc.).

• int pos: posición del usuario en el archivo.

• int error: indicador de error (inicializado a 0).

2. Limpiar la pantalla y abrir el archivo:

• system("cls"): limpia la consola.

• Se intenta abrir un archivo binario user.dat para lectura y escritura. Si el archivo no se puede abrir, se muestra un mensaje de error.

3. Mostrar menú:

• Se imprime un menú con opciones para modificar usuarios, adjudicar roles, consultar datos de usuario y salir.

4. Leer opción del usuario:

• Se utiliza scanf para capturar la opción seleccionada.

5. Manejo de opciones:

• Cada opción del menú está gestionada por un switch que dirige a diferentes bloques de código:

• Opción 1 (Modificar Usuarios):

• Solicita el CUIT del usuario a modificar.

• Utiliza fread para leer el archivo de usuarios y compara el CUIT.

• Si se encuentra el usuario, llama a admin(tipo, &usuwu) con el tipo m para modificar, y escribe de nuevo los datos en el archivo.

• Si no se encuentra el usuario, imprime un mensaje y regresa al menú.

• Opción 2 (Adjudicar Roles):

• Similar a la opción anterior, pero establece el tipo como a y permite adjudicar roles al usuario.

• Opción 3 (Consultar Datos de Usuario):

• Permite al administrador consultar la información de un usuario sin modificarla.

• Opción 4 (Salir):

• Imprime un mensaje de despedida y retorna de la función.

6. Manejo de errores:

• Si la opción ingresada no es válida, imprime un mensaje de error y vuelve a mostrar el menú.

Función admin(char tipo, usuario \*usuwu)

Esta función maneja las operaciones de modificación, adjudicación de roles y consulta de datos de un usuario.

1. Declaración de variables:

• Variables similares a las de menu\_adm, pero enfocadas en el proceso de administración del usuario específico.

• FILE \*ARC para abrir el archivo de movimientos.

2. Abrir archivo de movimientos:

• Intenta abrir movimientos.dat para lectura. Si no se puede abrir, se imprime un mensaje de error.

3. Manejo según el tipo de operación:

• El switch determina la operación según el tipo:

• Modificar usuario:

• Muestra un submenú para elegir qué atributo modificar (nombre, correo, número, contraseña, alias, CUIT, condición fiscal).

• Cada caso tiene su propia lógica para validar y modificar los datos. Utiliza estructuras de control como bucles while y condiciones if para asegurar que los datos sean válidos antes de modificar.

• Validaciones:

• Por ejemplo, al cambiar el nombre, se asegura que sea alfabético y cumpla con los criterios de longitud.

• Al cambiar el correo, se verifica que siga un formato válido con una arroba y un dominio.

• Para el número, verifica que sea un número válido y que cumpla con la longitud requerida, además de validar el código de área.

• Contrasena y alias:

• Se asegura que la contraseña sea alfanumérica y tenga al menos 8 caracteres.

• El alias debe ser exactamente de 5 caracteres alfanuméricos.

4. Lógica de registro:

• user\_registry() es llamado para validar si el nuevo dato (nombre, correo, etc.) ya existe en el sistema. Si ya existe, imprime un mensaje y no realiza el cambio.

5. Salida:

• En cada operación, si el usuario ingresa “0”, se detiene el proceso y se imprime un mensaje.

*Conclusión sobre el Sistema de Administración Financiera*

El sistema desarrollado para Kamaba Pay representa una solución para la gestión de operaciones financieras, ofreciendo funcionalidades similares a plataformas reconocidas como Mercado Pago. A lo largo del código, se han implementado diversas características que permiten la administración eficiente de usuarios y movimientos, garantizando una experiencia intuitiva y segura.

1. Estructura de Datos: Se ha utilizado una estructura clara y organizada para representar tanto a los usuarios como a los movimientos. Esto facilita la manipulación de datos y la expansión del sistema en el futuro.

2. Interacción del Usuario: La implementación de un menú interactivo permite al usuario navegar fácilmente por las distintas funcionalidades del sistema, lo que mejora la usabilidad.

3. Validaciones: Se han incorporado funciones de validación que aseguran que los datos ingresados, como los números de celular y CUIT, cumplen con los formatos requeridos. Esto es esencial para mantener la integridad de los datos y evitar errores en las transacciones.

4. Manejo de Archivos: El sistema utiliza archivos binarios para almacenar movimientos, lo que optimiza el rendimiento en la gestión de grandes volúmenes de datos. Las funciones para guardar y cargar movimientos aseguran que los datos se mantengan de forma persistente entre sesiones.

5. Escalabilidad y Flexibilidad: El diseño modular permite la fácil incorporación de nuevas funcionalidades y tipos de movimientos en el futuro, asegurando que el sistema pueda adaptarse a las necesidades cambiantes de los usuarios.

***Anexo***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <ctype.h>

#include <unistd.h>

void crear\_usuario();

void ingresar();

int user\_registry(const void \*field, char tipo);

void menu\_adm();

typedef struct{

char cuit[23];

char nombre[52];

char num[14];

char email[50];

char CVU[23];

char alias[6];

char IVA[50];

char contra[50];

float saldo;

char rol;

}usuario;

void ingresar\_dinero(usuario \*usa);

void transferencia(usuario \*usa);

void menu\_ing(usuario \*usser);

void pagar\_servicios(usuario \*usa);

void movimiento(char movtxt[100], char opcn, const usuario \*usa,float monto);

void filtros(const usuario \*usa);

void mdatos(usuario \*usa);

void admin(char tipo, usuario \*usuwu);

typedef struct{

char type;

char CBU[23];

char cuitl[23];

char clave1[4];

}cuentabanco;

typedef struct{

char tipo1;

float monto;

char iibb[50];

char fech[50];

char cuilt[50];

}movimientos;

int main(){

char opc;

FILE \*archivo=fopen("user.dat","rb" );

if(archivo==NULL){

archivo=fopen("user.dat", "wb");

if (archivo==NULL){

printf("Error al abrir el archivo de registros.\n");

exit(0);

}else{

usuario user;

strcpy(user.nombre, "Administrador");

strcpy(user.num, "0000000000000");

strcpy(user.email, "admin0@gmail.com");

strcpy(user.cuit, "00000");

strcpy(user.alias, "Ad000");

strcpy(user.contra, "contrasena");

user.rol='0';

if (fwrite(&user, sizeof(usuario), 1, archivo) != 1) {

printf("Error al escribir los datos en el archivo\n");

}

}

fclose(archivo);

}

fclose(archivo);

FILE \*archiv1=fopen("cuentabanco.dat","rb" );

if(archiv1==NULL){

archiv1=fopen("cuentabanco.dat", "wb");

if (archiv1==NULL){

printf("Error al abrir el archivo de registros.\n");

exit(0);

}else{

cuentabanco cuent;

strcpy(cuent.CBU, "0000000000000000000000");

strcpy(cuent.clave1, "000");

strcpy(cuent.cuitl, "0000000000000000000000");

cuent.type='0';

if (fwrite(&cuent, sizeof(cuentabanco), 1, archiv1) != 1) {

printf("Error al escribir los datos en el archivo\n");

}

}

fclose(archiv1);

}

fclose(archiv1);

printf("\nSi no se especifica la opcion de salir, todo proceso puede ser detenido ingresando 0.\n\n");

repeat:

printf("\t\t\tBienvenido a Kamaba Pay!\n\n\t\t\tEscoja que desea hacer:\n\n\t\t\t1. Crear Usuario.\n\t\t\t2. Ingresar.\n\t\t\t3. Salir.\n");

scanf("%s", &opc);

switch (opc){

case '1':

crear\_usuario();

goto repeat;

break;

case '2':

ingresar();

goto repeat;

break;

case '3':

printf("Usted ha salido.");

break;

default:

system("cls");

printf("\nOpcion invalida. Escoja de las siguientes:\n\n");

goto repeat;

}

}

void crear\_usuario(){

int band=1;

int i=1;

int pos;

int gm=0;

char tipo;

char CODVER[23];

int DIGVERI;

char cvu1[23];

int errorcvu=0;

int contnomb=0;

int punto=0;

int pos1;

char gmail[50];

char alias[51];

usuario user;

system("cls");

printf("\n\t --- Panel de Registro de Nuevo Usuario ---\n");

printf("\nHola, nuevo usuario! Por favor siga las instrucciones para crear una cuenta.\n");

printf("\nIngrese su nombre completo: ");

while(band!=0){

fflush(stdin);

fgets(user.nombre, sizeof(user.nombre), stdin);

band=1;

i=0;

size\_t len=strlen(user.nombre);

if(len>0&&user.nombre[len-1]=='\n') {

user.nombre[len-1]='\0';

}

if (strcmp(user.nombre, "0") == 0) {

printf("\nProceso detenido por el usuario.\n");

return;

}

while(user.nombre[i] != '\0' && band != 2){

if(!((user.nombre[i] >= 'A' && user.nombre[i] <= 'Z') || (user.nombre[i] >= 'a' && user.nombre[i] <= 'z') || user.nombre[i] == ' ')){

band=2;

}

i++;

}

if((user.nombre[0]!=' ')&&(user.nombre[1]!=' ')&&(((user.nombre[0]>= 'A'))&&((user.nombre[0]<='Z')))){

if(band!=2){

if (strlen(user.nombre)>=2){

if (strlen(user.nombre)<=50){

tipo='n';

if ((contnomb=(user\_registry(user.nombre, tipo)))==1){

band=0;

}else

band=0;

}else

printf("\nEl nombre es muy largo.\n");

}else

printf("\nEl nombre es muy corto.\n");

}else

printf("\nIngrese un nombre valido. (Solo letras)\n");

}else

printf("\nNombre invalido, el primer caracter debe ser una letra mayuscula seguido de un caracter\n");

}

band=1;

printf("\nIngrese su numero completo de celular. Agregando el codigo de area: ");

while (band!=0){

fflush(stdin);

gets(user.num);

fflush(stdin);

band=1;

i=0;

if (strcmp(user.num, "0") == 0) {

printf("\nProceso detenido por el usuario.\n");

return;

}

while(user.num[i] != '\0' && band != 2){

if(!((user.num[i] >= '0' && user.num[i] <= '9'))){

band=2;

}

i++;

}

if(band!=2){

if (strlen(user.num)==13){

if ((user.num[0]=='5')&&(user.num[1]=='4')&&(user.num[2]=='9')){

tipo='c';

if (user\_registry(user.num, tipo)==1){

band=0;

}else

printf("\nNumero ya existente.\n");

}else

printf("\nIngrese un codigo de area valido.(549)\n");

}else

printf("\nIngrese un numero valido. (De 13 numeros).\n");

}else

printf("\nIngrese un numero valido (Sin letras).\n");

}

band=1;

printf("\nIngrese su correo electronico: ");

while(band!=0){

fflush(stdin);

gets(user.email);

fflush(stdin);

band=1;

i=0;

gm=0;

punto=0;

if (strcmp(user.email, "0") == 0) {

printf("\nProceso detenido por el usuario.\n");

return;

}

while(user.email[i] != '\0' && band != 2){

if(!((user.email[i] >= 'A' && user.email[i] <= 'Z') || (user.email[i] >= 'a' && user.email[i] <= 'z') || (user.email[i] >= '0' && user.email[i] <= '9')||(user.email[i] == '@')||(user.email[i]=='.'))){

band=2;

}

if(user.email[i]=='@'){

gm++;

pos=i;

}

if((user.email[i]=='.')&&(user.email[i]>pos)){

if(user.email[i]=='.'){

punto++;

pos1=i;

}

}

i++;

}

if (punto==1){

i=0;

pos=pos+1;

fflush(stdin);

strcpy(gmail, " ");

fflush(stdin);

while(user.email[pos+i]!='.'&&user.email[pos+i]!='@'){

fflush(stdin);

gmail[i]=user.email[pos+i];

fflush(stdin);

i++;

}

}else

punto=0;

if(user.email[0]!='@'){

if (band!=2){

if(gm==1){

if (punto!=0){

if (strlen(user.email)<=49){

if((strcmp(gmail, "gmail") == 0)||(strcmp(gmail, "hotmail") == 0)||(strcmp(gmail, "yahoo") == 0)||(strcmp(gmail, "outlook") == 0)){

tipo='e';

if((user.email[pos1+1]=='c')&&(user.email[pos1+2]=='o')&&(user.email[pos1+3]=='m')&&(user.email[pos1+4]=='\0')){

if (user\_registry(user.email, tipo)==1){

band=0;

}else{

printf("\nEmail ya existente.\n");

}

}else{

band=1;

printf("\nDireccion invalida. Asegurarse que los puntos y dominio esten correctos.\n");

}

}else

printf("\nDireccion invalida. Solo gmail, yahoo, outlook y hotmail.\n");

}else

printf("\nDireccion invalida. Excede la longitud permitida.\n");

}else

printf("\nDireccion invalida. Revisar los puntos\n");

}else

printf("\nDireccion invalida. Revise la cantidad de arrobas.\n");

}else

printf("\nDireccion invalida. Solo letras y numeros.\n");

}else

printf("\nDireccion invalida. El primer caracter debe ser una letra\n");

}

band=1;

printf("\nIngrese su CUIT/L: ");

while (band!=0){

fflush(stdin);

gets(user.cuit);

fflush(stdin);

band=1;

i=0;

if (strcmp(user.cuit, "0") == 0) {

printf("\nProceso detenido por el usuario.\n");

return;

}

while(user.cuit[i] != '\0' && band != 2){

if(!((user.cuit[i] >= '0' && user.cuit[i] <= '9'))){

band=2;

}

i++;

}

if(band!=2){

if (strlen(user.cuit)==11){

if (((user.cuit[0]=='2')&&(user.cuit[1]=='7'))||((user.cuit[0]=='2')&&(user.cuit[1]=='0'))||((user.cuit[0]=='3')&&(user.cuit[1]=='0'))){

if((11-((((user.cuit[0]-'0')\*5)+((user.cuit[1]-'0')\*4)+((user.cuit[2]-'0')\*3)+((user.cuit[3]-'0')\*2)+((user.cuit[4]-'0')\*7)+((user.cuit[5]-'0')\*6)+((user.cuit[6]-'0')\*5)+((user.cuit[7]-'0')\*4)+((user.cuit[8]-'0')\*3)+((user.cuit[9]-'0')\*2))%11))==(user.cuit[10]-'0')){

tipo='t';

if (user\_registry(user.cuit, tipo)==1){

band=0;

}else

printf("\nCUIT/L ya existente.\n");

}else

printf("\nCUIT/L invalido. No verifica.\n");

}else

printf("\nPrimeros digitos de CUIT/L invalidos.\n");

}else

printf("\nLa longitud del CUIT/L es invalida.\n");

}else

printf("\nIngrese un CUIT/L valido. (Solo numeros)\n");

}

band=1;

while(band!=0){

for(int i=0; i<=5; i++){

user.CVU[i]='0';

}

user.CVU[6]='3';

user.CVU[7]='1';

srand(time(NULL));

for(i=8; i<=20; i++){

user.CVU[i]='0'+rand()%10;

}

for (i=0; i<=20; i++){

CODVER[i]='1';

CODVER[i+1]='3';

CODVER[i+2]='7';

i=i+2;

}

for (i=0; i<=20; i++){

DIGVERI=(CODVER[i]+((user.CVU[i]-'0')\*(CODVER[i]-'0')));

}

DIGVERI=DIGVERI%10;

if(DIGVERI==0){

user.CVU[21]='0';

}else

user.CVU[21]=DIGVERI+'0';

user.CVU[22] = '\0';

FILE \*archiv=fopen("user.dat","rb" );

if(archiv==NULL){

printf("Error al abrir los registros para validacion. Cerrando el programa.");

return;

}

usuario usi;

while (fread(&usi, sizeof(usuario), 1, archiv)){

if (strcmp(usi.CVU, user.CVU) == 0){

fclose(archiv);

errorcvu=1;

}

}

if (errorcvu!=1){

band=0;

}

}

user.alias[0]=user.nombre[0];

user.alias[1]=user.nombre[1];

user.alias[2]=user.cuit[4];

user.alias[3]=user.cuit[5];

user.alias[4]=contnomb + '0';

user.alias[5] = '\0';

tipo='a';

fflush(stdin);

band=1;

printf("\nCree una contrasena (Solo letras y numeros, al menos 8 caracteres de longitud.):\n");

fflush(stdin);

while (band!=0){

i=0;

fflush(stdin);

fgets(user.contra, sizeof(user.contra), stdin);

fflush(stdin);

size\_t len=strlen(user.contra);

if(len>0&&user.contra[len-1]=='\n') {

user.contra[len-1]='\0';

}

if (strcmp(user.contra, "0") == 0) {

printf("\nProceso detenido por el usuario.\n");

return;

}

while(user.contra[i] !='\0' && band != 2){

if(!((user.contra[i] >= 'A' && user.contra[i] <= 'Z') || (user.contra[i] >= 'a' && user.contra[i] <= 'z') || (user.contra[i] >= '0' && user.contra[i] <= '9'))){

band=2;

}else

i++;

}

if(band!=2){

if(strlen(user.contra)<=49){

if(strlen(user.contra)>=8){

band=0;

}else

printf("\nContrasena invalida. Demasiado corta.\n");

}else

printf("\nContrasena invalida. Demasiado larga.\n");

}else

printf("\nContrasena invalida. Solo se permite el uso de numeros y letras.\n");

}

user.rol='1';

if (user\_registry(user.alias, tipo)==1){

printf("\nSu alias es %s y su CVU es %s\n", user.alias, user.CVU);

}else{

alias[5]=user\_registry(user.alias, tipo);

printf("\nSu alias es %s y su CVU es %s\n", user.alias, user.CVU);

}

strcpy(user.IVA, "En validacion");

FILE \*arc=fopen("user.dat", "a+b");

if(arc==NULL){

printf("Error al abrir el archivo de usuarios para registar los datos.\n");

return;

}

if (fwrite(&user, sizeof(usuario), 1, arc) != 1) {

printf("Error al escribir los datos en el archivo\n");

return;

}

fclose(arc);

FILE \*arcuit=fopen("ADMINvalidar\_IVA.txt", "a+b");

if(arcuit==NULL){

printf("Error al abrir archivo IVA. Cerrando el programa.");

return;

}else

fprintf(arcuit, "! Nuevo CUIT a verificar: %s\n", user.cuit);

fclose(arcuit);

printf("\nCuenta creada con exito!\n\n Te bienvenimos, %s.\nIngrese 0 para volver al menu principal, y cualquier otro numero para salir del programa.\n", user.nombre);

scanf("%d", &band);

if (band==0){

system("cls");

}else{

printf("Saliendo del programa... Usted ha salido.");

return;

}

}

int user\_registry(const void \*field, char tipo){

int contnom=1;

int contalias=1;

FILE \*archivo=fopen("user.dat","rb" );

if(archivo==NULL){

printf("Error al abrir los registros para validacion. Cerrando el programa.");

return 0;

}

usuario usu;

while (fread(&usu, sizeof(usuario), 1, archivo)){

switch (tipo){

case 'n':{

char \*nombre=(char \*)field;

if (strcmp(usu.nombre, nombre) == 0){

contnom++;

}

break;

}

case 'c':{

char \*num=(char \*)field;

if (strcmp(usu.num, num) == 0){

fclose(archivo);

return 0;

}

break;

}

case 'e':{

char \*email=(char \*)field;

if (strcmp(usu.email, email) == 0){

fclose(archivo);

return 0;

}

break;

}

case 't':{

char \*cuit=(char \*)field;

if (strcmp(usu.cuit, cuit) == 0){

fclose(archivo);

return 0;

}

break;

}

case 'a':{

char \*alias=(char \*)field;

if (strcmp(usu.alias, alias) == 0){

fclose(archivo);

contalias++;

}

break;

}

case 'o':{

char \*contr=(char \*)field;

if (strcmp(usu.contra, contr) == 0){

fclose(archivo);

return 0;

}

break;

}

}

}

if(contalias>0){

fclose (archivo);

return contalias;

}

if (contnom>0){

fclose(archivo);

return contnom;

}else

return 1;

}

void ingresar(){

system("cls");

repeat:

char opci;

int band=1;

int band1=1;

int error=1;

usuario uso;

usuario usr;

char tip;

FILE \*arci = fopen("user.dat", "rb");

if (arci==NULL) {

printf("Error al abrir el archivo.\n");

return;

}

printf("\n\t\t---Panel de Ingreso de Usuario---\t\t\n\n");

printf("\t\t\t¿Como desea ingresar?!\n\n\t\t\t1. Utilizando email. \n\t\t\t2. Utilizando numero de celular.\n\t\t\t3. Utilizando CUIT/L.\n\t\t\t4. Salir\n");

fflush(stdin);

scanf("%s", &opci);

switch (opci){

case '1':

printf("\nIngrese su email.\n");

while(band!=0){

fflush(stdin);

gets(uso.email);

fflush(stdin);

if (strcmp(uso.email, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

tip='e';

if(user\_registry(uso.email, tip)!=1){

printf("\nIngrese su contraseña\n");

while(band1!=0){

fflush(stdin);

gets(uso.contra);

fflush(stdin);

if (strcmp(uso.contra, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

tip='o';

if(user\_registry(uso.contra, tip)!=1){

rewind(arci);

error=1;

while (fread(&usr, sizeof(usuario), 1, arci)){

if ((strcmp(usr.email, uso.email) == 0)&&(strcmp(usr.contra, uso.contra) == 0)){

uso=usr;

printf("\nIngresando...Espere.\n");

sleep(1);

system("cls");

menu\_ing(&uso);

error=0;

break;

}

}

if (error==1){

printf("\nContraseña invalida. Pertenece a otra cuenta.\n");

}else{

fclose(arci);

band1=0;

}

}else

printf("\nContrasena incorrecta.\n");

}

if (band1==0){

band=0;

}

}else

printf("\nEl correo ingresado no esta vinculado a una cuenta.\n");

}

break;

case '2':

printf("\nIngrese su numero.\n");

while(band!=0){

fflush(stdin);

gets(uso.num);

fflush(stdin);

if (strcmp(uso.num, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

tip='c';

if(user\_registry(uso.num, tip)!=1){

printf("\nIngrese su contraseña\n");

while(band1!=0){

fflush(stdin);

gets(uso.contra);

fflush(stdin);

if (strcmp(uso.contra, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

tip='o';

if(user\_registry(uso.contra, tip)!=1){

rewind(arci);

error=1;

while (fread(&usr, sizeof(usuario), 1, arci)){

if ((strcmp(usr.num, uso.num) == 0)&&(strcmp(usr.contra, uso.contra) == 0)){

uso=usr;

printf("\nIngresando...Espere.\n");

sleep(1);

system("cls");

menu\_ing(&uso);

error=0;

break;

}

}

if (error==1){

printf("\nContraseña invalida. Pertenece a otra cuenta.\n");

}else{

fclose(arci);

band1=0;

}

}else

printf("\nContrasena incorrecta.\n");

}

if (band1==0){

band=0;

}

}else

printf("\nInvalido. Ingrese su numero nuevamente.\n");

}

break;

case '3':

printf("\nIngrese su CUIT/L.\n");

while(band!=0){

fflush(stdin);

gets(uso.cuit);

fflush(stdin);

if (strcmp(uso.cuit, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

tip='t';

if(user\_registry(uso.cuit, tip)!=1){

printf("\nIngrese su contraseña\n");

while(band1!=0){

fflush(stdin);

gets(uso.contra);

fflush(stdin);

if (strcmp(uso.contra, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

tip='o';

if(user\_registry(uso.contra, tip)!=1){

rewind(arci);

error=1;

while (fread(&usr, sizeof(usuario), 1, arci)){

if ((strcmp(usr.cuit, uso.cuit) == 0)&&(strcmp(usr.contra, uso.contra) == 0)){

uso=usr;

printf("\nIngresando...Espere.\n");

sleep(1);

system("cls");

menu\_ing(&uso);

error=0;

break;

}

}

if (error==1){

printf("\nContraseña invalida. Pertenece a otra cuenta.\n");

}else{

fclose(arci);

band1=0;

}

}else

printf("\nContrasena incorrecta.\n");

}

if (band1==0){

band=0;

}

}else

printf("\nInvalido. Ingrese su CUIT/L nuevamente.\n");

}

break;

case '0':

printf("\nIngrese el CUIT/L.\n");

fflush(stdin);

gets(uso.cuit);

fflush(stdin);

if (strcmp(uso.cuit, "0") == 0) {

printf("\nProceso detenido por el Administrador.\n\n");

break;

}

tip='t';

if(user\_registry(uso.cuit, tip)!=1){

printf("\nIngrese su contraseña\n");

fflush(stdin);

gets(uso.contra);

fflush(stdin);

if (strcmp(uso.cuit, "0") == 0) {

printf("\nProceso detenido por el Administrador.\n\n");

break;

}

tip='o';

if(user\_registry(uso.contra, tip)!=1){

rewind(arci);

error=1;

while (fread(&usr, sizeof(usuario), 1, arci)){

if ((strcmp(usr.cuit, uso.cuit) == 0)&&(strcmp(usr.contra, uso.contra) == 0)){

if(usr.rol=='0'){

printf("\nBienvenido, Administrador. Ingresando...Espere.\n");

sleep(1);

system("cls");

menu\_adm();

error=0;

break;

}else{

error=1;

printf("\nNo posee los permisos para acceder.\n");

sleep(2);

system("cls");

}

}

}

if (error==1){

printf("\nContraseña invalida. Volviendo al menu...\n");

sleep(2);

system("cls");

return;

}else{

fclose(arci);

sleep(1);

system("cls");

return;

}

}else{

printf("\nContrasena incorrecta. Volviendo al menu...\n");

sleep(1);

system("cls");

return;

}

}else

printf("\nEl correo ingresado no esta vinculado a una cuenta.\n");

break;

case '4':

printf("Saliendo...");

sleep(1);

return;

default:

system("cls");

printf("\nOpcion invalida. Escoja de las siguientes:\n\n");

goto repeat;

break;

}

}

void menu\_ing(usuario \*usser){

char opcion;

long pos;

char opcn;

usuario usa;

usa=\*usser;

repeat:

FILE \*arch=fopen("user.dat", "rb+");

if(arch==NULL){

printf("\nError al abrir el archivo de usuarios.\n");

return;

}

if((strcmp(usser->IVA, "En validacion"))!=0){

usser->saldo=usa.saldo;

printf("\nHola, %s!\n\n", usser->nombre);

printf("\nSaldo: %f\n", usa.saldo);

printf("\n\t\tSeleccione la opcion deseada:\n\t\t1. Transferencia\n\t\t2. Ingresar Dinero\n\t\t3. Pagar Servicios\n\t\t4. Consultar y Modificar Datos\n\t\t5. Consultar movimientos\n\t\t6. Salir\n\n");

fflush(stdin);

scanf("%c", &opcion);

switch(opcion){

case '1':

transferencia(&usa);

if (usser->saldo != usa.saldo) {

printf("\nSu saldo anterior era %.2f. Su nuevo saldo es %.2f\n", usser->saldo, usa.saldo);

rewind(arch);

while (fread(usser, sizeof(usuario), 1, arch)==1) {

if (strcmp(usser->cuit, usa.cuit)==0) {

pos=ftell(arch)-sizeof(usuario);

fseek(arch, pos, SEEK\_SET);

usser->saldo=usa.saldo;

if (fwrite(usser, sizeof(usuario), 1, arch)!= 1) {

printf("\nError al escribir los datos en el archivo\n");

}

break;

}

}

}

fclose(arch);

sleep(2);

system("cls");

goto repeat;

break;

case '2':

ingresar\_dinero(&usa);

if (usser->saldo != usa.saldo) {

printf("\nSu saldo anterior era %.2f. Su nuevo saldo es %.2f\n", usser->saldo, usa.saldo);

rewind(arch);

while (fread(usser, sizeof(usuario), 1, arch) == 1) {

if (strcmp(usser->cuit, usa.cuit) == 0) {

pos = ftell(arch) - sizeof(usuario);

fseek(arch, pos, SEEK\_SET);

usser->saldo = usa.saldo;

if (fwrite(usser, sizeof(usuario), 1, arch) != 1) {

printf("\nError al escribir los datos en el archivo\n");

}

break;

}

}

}

fclose(arch);

sleep(2);

system("cls");

goto repeat;

break;

case '3':

pagar\_servicios(&usa);

if (usser->saldo != usa.saldo) {

printf("\nSu saldo anterior era %.2f. Su nuevo saldo es %.2f\n", usser->saldo, usa.saldo);

rewind(arch);

while (fread(usser, sizeof(usuario), 1, arch) == 1) {

if (strcmp(usser->cuit, usa.cuit) == 0) {

pos = ftell(arch) - sizeof(usuario);

fseek(arch, pos, SEEK\_SET);

usser->saldo = usa.saldo;

if (fwrite(usser, sizeof(usuario), 1, arch) != 1) {

printf("\nError al escribir los datos en el archivo\n");

}

break;

}

}

}

fclose(arch);

sleep(2);

system("cls");

goto repeat;

break;

case '4':

mdatos(&usa);

if (memcmp(&usser, &usa, sizeof(usuario)) != 0){

printf("\nCambios realizados con exito\n");

rewind(arch);

while (fread(usser, sizeof(usuario), 1, arch) == 1) {

if (strcmp(usser->cuit, usa.cuit) == 0) {

pos=ftell(arch)-sizeof(usuario);

usser=&usa;

fseek(arch, pos, SEEK\_SET);

if (fwrite(usser, sizeof(usuario), 1, arch) != 1) {

printf("\nError al escribir los datos en el archivo\n");

}

break;

}

}

}else

printf("\nNo han habido cambios.\n");

fclose(arch);

sleep(2);

system("cls");

goto repeat;

break;

case '5':

filtros(&usa);

goto repeat;

break;

case '6':

printf("Saliendo...");

sleep(1);

return;

default:

system("cls");

printf("\nOpcion invalida. Escoja de las siguientes:\n\n");

goto repeat;

break;

}

}else{

printf("\nUsted no posee la verificacion necesaria para continuar.\nEspere o contacte a nuestro servicio al cliente: barbararoza60@gmail.com\n\n");

sleep(1);

}

}

void transferencia(usuario \*usa){

float monto;

long pos;

char movtxt[100];

char opcn;

float desc;

usuario uscompar;

system("cls");

repeat:

desc=0;

FILE \*archive=fopen("user.dat", "rb+");

if (archive==NULL){

printf("Error al abrir el archivo para comparar usuarios durante transferencia.");

return;

}

char entrada[50];

char opi;

int encon=1;

printf("\n\t\tSeleccione como desea transferir:\n");

printf("\n\t\t1. Con CVU o alias:\n\t\t2. Con celular o email.\n\t\t3. Salir.\n\n");

fflush(stdin);

scanf("%c", &opi);

fflush(stdin);

switch(opi){

case '1':

if(usa->saldo>0){

printf("\nIngrese el CVU o alias de la persona a transferir: ");

fflush(stdin);

gets(entrada);

fflush(stdin);

while (fread(&uscompar, sizeof(usuario), 1, archive)){

if ((strcmp(uscompar.CVU, entrada)==0)||(strcmp(uscompar.alias, entrada)==0)){

printf("\nUsted va a transferir dinero a %s\n", uscompar.nombre);

encon=1;

break;

}else

encon=0;

}

if(encon!=0){

rewind(archive);

while (fread(&uscompar, sizeof(usuario), 1, archive)==1){

if ((strcmp(uscompar.CVU, entrada)==0)||(strcmp(uscompar.alias, entrada)==0)){

pos=ftell(archive)-sizeof(usuario);

fseek(archive, pos, SEEK\_SET);

printf("\nIngrese la cantidad de dinero a transferir: ");

scanf("%f", &monto);

if (usa->saldo>=monto){

usa->saldo-=monto;

uscompar.saldo+= monto;

if((strcmp(uscompar.IVA, "Sujeto Exento")==0)||(strcmp(uscompar.IVA, "Monotributista Social")==0)||(strcmp(uscompar.IVA, "Responsable Monotributo")==0)||(strcmp(uscompar.IVA, "Responsable Inscripto")==0)){

uscompar.saldo+=(monto-(monto\*0.05));

printf("\nSe le ha descontado un 5 porciento a su transferencia debido a la tasa IIBB del otro usuario.\n");

desc=monto\*0.05;

}

printf("\nUsted esta transferiendo %.2f\n a %s...\n", monto, uscompar.nombre);

fseek(archive, pos, SEEK\_SET);

if (fwrite(&uscompar, sizeof(usuario), 1, archive) != 1){

printf("\nError al escribir los datos en el archivo\n");

}else{

opcn='t';

if(desc>0){

snprintf(movtxt, sizeof(movtxt), "Transferencia recibida de %.2f de %s a %s, con una retencion de %.2f", monto, usa->cuit, uscompar.cuit, desc);

movimiento(movtxt, opcn, &uscompar, monto);

uscompar=\*usa;

snprintf(movtxt, sizeof(movtxt), "Transferencia de %.2f a %s ce %s, con una retencion de %.2f", monto, uscompar.cuit, usa->cuit, desc);

movimiento(movtxt, opcn, &uscompar, monto);

}else{

snprintf(movtxt, sizeof(movtxt), "Transferencia recibida de %.2f de %s a %s", monto, usa->cuit, uscompar.cuit);

movimiento(movtxt, opcn, &uscompar, monto);

snprintf(movtxt, sizeof(movtxt), "Transferencia de %.2f a %s ce %s", monto, uscompar.cuit, usa->cuit);

uscompar=\*usa;

movimiento(movtxt, opcn, &uscompar, monto);

}

printf("\nSu transferencia fue exitosa.\n");

}

}else{

printf("\nEl saldo de la cuenta es insuficiente.\n");

}

break;

}

}

}else

printf("\nNo se ha encontrado al usuario. Volviendo al menú principal...\n");

} else {

printf("\nSaldo insuficiente para transferencias. Volviendo al menu principal...\n");

}

fclose(archive);

sleep(2);

break;

case '2':

if(usa->saldo>0){

printf("\nIngrese el celular o correo de la persona a transferir: ");

fflush(stdin);

gets(entrada);

fflush(stdin);

while (fread(&uscompar, sizeof(usuario), 1, archive)){

if((strcmp(uscompar.email, entrada)==0)||(strcmp(uscompar.num, entrada)==0)){

printf("\nUsted va a transferir dinero a %s\n", uscompar.nombre);

encon=1;

break;

}else

encon=0;

}

if(encon!=0){

rewind(archive);

while (fread(&uscompar, sizeof(usuario), 1, archive)==1){

if ((strcmp(uscompar.email, entrada)==0)||(strcmp(uscompar.num, entrada)==0)){

pos=ftell(archive)-sizeof(usuario);

fseek(archive, pos, SEEK\_SET);

printf("\nIngrese la cantidad de dinero a transferir: ");

scanf("%f", &monto);

if (usa->saldo>=monto){

usa->saldo-=monto;

uscompar.saldo += monto;

if((strcmp(uscompar.IVA, "Sujeto Exento")==0)||(strcmp(uscompar.IVA, "Monotributista Social")==0)||(strcmp(uscompar.IVA, "Responsable Monotributo")==0)||(strcmp(uscompar.IVA, "Responsable Inscripto")==0)){

uscompar.saldo+=(monto-(monto\*0.05));

printf("\nSe le ha descontado un 5 porciento a su transferencia debido a la tasa IIBB del otro usuario.\n");

desc=monto\*0.05;

}

printf("\nUsted esta transferiendo %.2f a %s...\n", monto, uscompar.nombre);

fseek(archive, pos, SEEK\_SET);

if (fwrite(&uscompar, sizeof(usuario), 1, archive) != 1){

printf("\nError al escribir los datos en el archivo\n");

}else{

opcn='t';

if(desc>0){

snprintf(movtxt, sizeof(movtxt), "Transferencia recibida de %.2f de %s a %s, con una retencion de %.2f", monto, usa->cuit, uscompar.cuit, desc);

movimiento(movtxt, opcn, &uscompar, monto);

uscompar=\*usa;

snprintf(movtxt, sizeof(movtxt), "Transferencia de %.2f a %s ce %s, con una retencion de %.2f", monto, uscompar.cuit, usa->cuit, desc);

movimiento(movtxt, opcn, &uscompar, monto);

}else{

snprintf(movtxt, sizeof(movtxt), "Transferencia recibida de %.2f de %s a %s", monto, usa->cuit, uscompar.cuit);

movimiento(movtxt, opcn, &uscompar, monto);

snprintf(movtxt, sizeof(movtxt), "Transferencia de %.2f a %s ce %s", monto, uscompar.cuit, usa->cuit);

uscompar=\*usa;

movimiento(movtxt, opcn, &uscompar, monto);

}

printf("\nSu transferencia fue exitosa.\n");

}

}else{

printf("\nEl saldo de la cuenta es insuficiente.\n");

}

break;

}

}

}else

printf("\nNo se ha encontrado al usuario. Volviendo al menú principal...\n");

} else

printf("\nSaldo insuficiente para transferencias. Volviendo al menu principal...\n");

fclose(archive);

sleep(2);

break;

case '3':

printf("Saliendo...");

sleep(1);

return;

break;

default:

system("cls");

printf("\nOpcion invalida. Escoja de las siguientes:\n\n");

goto repeat;

break;

}

return;

}

void ingresar\_dinero(usuario \*usa){

char opcio;

int contc;

char clave[50];

char cbu[23];

char tipo;

float monto;

char opcn;

char movtxt[100];

system("cls");

repeat:

usuario movstr=\*usa;

cuentabanco cuenta;

FILE \*arch=fopen("cuentabanco.dat", "rb");

if(arch==NULL){

printf("\nError al abrir el archivo de cuenta de banco.\n");

return;

}

int band=1;

int enc=0;

int i=0;

contc=0;

printf("\nSeleccione que desea hacer:\n");

printf("\n\t\t1. Ingresar dinero con cuenta vinculada.\n\t\t2. Agregar nueva cuenta\n\t\t3. Salir\n");

fflush(stdin);

scanf("%c", &opcio);

fflush(stdin);

rewind(arch);

switch(opcio){

case '1':

rewind(arch);

printf("\nElija una de sus cuentas para transferir:\n");

while (fread(&cuenta, sizeof(cuentabanco), 1, arch)){

if (strcmp(cuenta.cuitl, usa->cuit) == 0){

contc++;

printf("\n%d. %s\n", contc, cuenta.CBU);

}

}

if (contc>0){

while(band!=0){

scanf("%c", &opcio);

fflush(stdin);

if (opcio >= '1' && opcio <= contc + '0') {

band=0;

}else

printf("\nOpcion invalida. Elija de las existentes.\n");

}

rewind(arch);

contc=0;

while (fread(&cuenta, sizeof(cuentabanco), 1, arch)){

if (strcmp(cuenta.cuitl, usa->cuit) == 0){

contc++;

if(contc==opcio-'0'){

strcpy(cbu, cuenta.CBU);

printf("\nUsted ha elegido el siguiente CBU: %s\n", cuenta.CBU);

}

}

}

fclose(arch);

printf("\nPor favor ingrese el monto a transferir a su cuenta: ");

scanf("%f", &monto);

usa->saldo=(usa->saldo+monto);

printf("\nUsted ha transferido $%.2f\n", monto);

opcn='i';

snprintf(movtxt, sizeof(movtxt), "Ingreso de %.2f a %s desde %s", monto, cuenta.CBU, usa->CVU);

movimiento(movtxt, opcn, &movstr, monto);

}

if (contc==0){

printf("\nUsted no posee ninguna cuenta vinculada. Agregue una para volver a intentar.\n");

}

contc=0;

goto repeat;

break;

case '2':

system("cls");

printf("\n\t\tPanel de Vinculacion de Cuenta Bancaria---\n");

printf("\nIngrese el tipo de cuenta\n1. Ahorro\n2. Corriente\n");

while(band!=0){

fflush(stdin);

scanf("%c", &tipo);

fflush(stdin);

if((tipo=='1')||(tipo=='2')){

band=0;

}else

printf("\nTipo invalido, elija de las opciones dadas.\n");

}

band=1;

printf("\nIngrese su CBU\n");

while(band!=0){

fflush(stdin);

gets(cbu);

fflush(stdin);

if (strcmp(cbu, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

i=0;

band=1;

while(cbu[i] != '\0' && band != 2){

if(!((cbu[i] >= '0' && cbu[i] <= '9'))){

band=2;

}

i++;

}

if(band!=2){

if(strlen(cbu)==22){

printf("\nIngrese la clave de su cuenta.\n");

fflush(stdin);

gets(clave);

fflush(stdin);

if (strcmp(clave, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

if(strlen(clave)<4){

while (fread(&cuenta, sizeof(cuentabanco), 1, arch)){

if (strcmp(cuenta.CBU, cbu) == 0){

enc++;

if(strcmp(cuenta.clave1, clave) == 0){

strcpy(cuenta.cuitl, usa->cuit);

cuenta.type=tipo;

fclose(arch);

fflush(stdin);

arch=fopen("cuentabanco.dat", "ab");

if (fwrite(&cuenta, sizeof(cuentabanco), 1, arch) != 1) {

printf("\nError al escribir los datos en el archivo\n");

band=0;

}else

printf("\nCuenta vinculada con exito, %d\n", enc);

fclose(arch);

band=0;

}else

printf("\nLa clave es incorrecta.\n");

}else

enc=0;

}

if(enc==0){

strcpy(cuenta.CBU, cbu);

strcpy(cuenta.clave1, clave);

strcpy(cuenta.cuitl, usa->cuit);

cuenta.type=tipo;

fclose(arch);

arch=fopen("cuentabanco.dat", "ab");

if (fwrite(&cuenta, sizeof(cuentabanco), 1, arch) != 1) {

printf("\nError al escribir los datos en el archivo\n");

band=0;

}else{

printf("\nCuenta vinculada con exito por primera vez\n");

fclose(arch);

band=0;

}

}

}else{

band=1;

printf("\nClave muy larga. Solo 3 caracteres.\n");

}

}else{

band=1;

printf("\nInvalido. Vuelva a revisar el CBU o la contrasena.\n");

}

}else{

band=1;

printf("\nInvalido. Ingrese el CBU nuevamente.\n");

}

}

break;

case '3':

printf("\nSaliendo...\n");

sleep(1);

system("cls");

return;

break;

default:

system("cls");

printf("\nError. Ingrese una de las siguientes opciones.\n");

goto repeat;

break;

}

}

void pagar\_servicios(usuario \*usa){

char op;

char opcn;

char movtxt[100];

int deuda;

float monto;

usuario movstru=\*usa;

system("cls");

repeat:

printf("\n\t\t---Panel de Servicios---\n");

printf("\nEscoja el servicio a pagar:\n");

printf("\n\t\t1. Expensas Medicas.\n\t\t2. Servicio Telefonico.\n\t\t3. Expensas del hogar.\n\t\t4.Salir\n");

scanf("%c", &op);

switch(op){

case '1':

printf("\nPara consultar el monto a pagar de sus expensas medicas, consulte la pagina del Hospital Madariaga,\nal que usted esta asociado de acuerdo a sus registros fiscales.\n");

printf("\nIngrese el monto a pagar: ");

scanf("%f", &monto);

if(usa->saldo>=monto){

usa->saldo-=monto;

if((strcmp(usa->IVA, "Sujeto Exento")==0)||(strcmp(usa->IVA, "Monotributista Social")==0)||(strcmp(usa->IVA, "Responsable Monotributo")==0)||(strcmp(usa->IVA, "Responsable Inscripto")==0)){

if(usa->saldo>=(monto+(monto\*0.05))){

usa->saldo-=(monto+(monto\*0.05));

printf("\nSe le ha aumentado un 5 porciento a su pago debido a la tasa IIBB.\n");

opcn='p';

snprintf(movtxt, sizeof(movtxt), "Pago de servicio medico de %.2f de %s, con un aumento de %.2f", monto, usa->cuit, monto\*0.05);

movimiento(movtxt, opcn, &movstru, monto);

break;

}

}

opcn='p';

snprintf(movtxt, sizeof(movtxt), "Pago de servicio de %.2f de %s", monto, usa->cuit );

movimiento(movtxt, opcn, &movstru, monto);

}else

printf("Saldo insuficiente. Volviendo al menu principal...");

sleep(1);

system("cls");

break;

case '2':

printf("\nPara consultar el monto a pagar de su servicio telefonico, consulte la pagina de Claro,\nal que usted esta asociado de acuerdo a sus registros telefonicos.\n");

printf("\nIngrese el monto a pagar: ");

scanf("%f", &monto);

if(usa->saldo>=monto){

usa->saldo-=monto;

if((strcmp(usa->IVA, "Sujeto Exento")==0)||(strcmp(usa->IVA, "Monotributista Social")==0)||(strcmp(usa->IVA, "Responsable Monotributo")==0)||(strcmp(usa->IVA, "Responsable Inscripto")==0)){

if(usa->saldo>=(monto+(monto\*0.05))){

usa->saldo-=(monto+(monto\*0.05));

printf("\nSe le ha aumentado un 5 porciento a su pago debido a la tasa IIBB.\n");

opcn='p';

snprintf(movtxt, sizeof(movtxt), "Pago de servicio telefonico de %.2f de %s, con un aumento de %.2f", monto, usa->cuit, monto\*0.05);

movimiento(movtxt, opcn, &movstru, monto);

break;

}

}

opcn='p';

snprintf(movtxt, sizeof(movtxt), "Pago de servicio de %.2f de %s", monto, usa->cuit );

movimiento(movtxt, opcn, &movstru, monto);

}else

printf("Saldo insuficiente. Volviendo al menu principal...");

sleep(1);

system("cls");

break;

case '3':

printf("\nPara consultar el monto a pagar de sus expensas del hogar, consulte la pagina de Consorcio Abierto,\ncadena de departamentos en el que usted reside de acuerdo a sus registros fiscales.\n");

printf("\nIngrese el monto a pagar: ");

scanf("%f", &monto);

if(usa->saldo>=monto){

usa->saldo-=monto;

if((strcmp(usa->IVA, "Sujeto Exento")==0)||(strcmp(usa->IVA, "Monotributista Social")==0)||(strcmp(usa->IVA, "Responsable Monotributo")==0)||(strcmp(usa->IVA, "Responsable Inscripto")==0)){

if(usa->saldo>=(monto+(monto\*0.05))){

usa->saldo-=(monto+(monto\*0.05));

printf("\nSe le ha aumentado un 5 porciento a su pago debido a la tasa IIBB.\n");

opcn='p';

snprintf(movtxt, sizeof(movtxt), "Pago de servicio al hogar de %.2f de %s, con un aumento de %.2f", monto, usa->cuit, monto\*0.05);

movimiento(movtxt, opcn, &movstru, monto);

break;

}

}

opcn='p';

snprintf(movtxt, sizeof(movtxt), "Pago de servicio de %.2f de %s", monto, usa->cuit );

movimiento(movtxt, opcn, &movstru, monto);

}else

printf("Saldo insuficiente. Volviendo al menu principal...");

sleep(2);

system("cls");

break;

case '4':

printf("\nSaliendo...\n");

sleep(2);

system("cls");

return;

break;

default:

system("cls");

printf("Invalido. Escoja una de las opciones dadas:\n");

goto repeat;

}

}

void mdatos(usuario \*usa){

char opciowo;

usuario user;

char tipo;

system("cls");

repeat:

int band=1;

int i=0;

int contnomb=0;

int gm=0;

int pos;

int punto=0;

int pos1=0;

char gmail[50];

printf("\n\t\t---Panel de Datos---\n");

printf("Sus datos son los siguientes:\n");

printf("Nombre: %s\n", usa->nombre);

printf("Correo: %s\n", usa->email);

printf("Celular: %s\n", usa->num);

printf("CUIT/L: %s\n", usa->cuit);

printf("Contrasena: %s\n", usa->contra);

printf("Alias: %s\n", usa->alias);

printf("CVU: %s\n", usa->CVU);

printf("Condicion fiscal: %s\n", usa->IVA);

printf("\nEscoja que desea hacer:\n");

printf("\n\t\t1. Modificar Datos.\n\t\t2. Salir.\n");

fflush(stdin);

scanf("%c", &opciowo);

switch(opciowo){

case '1':

system("cls");

printf("\n\t\t---Panel de de Datos---\n");

printf("\nEscoja el dato a modificar:\n");

printf("\n\t\t1. Nombre\n\t\t2. Correo\n\t\t3. Numero\n\t\t4. Contrasena\n\t\t5. Alias.\n\t\t6. Salir\n");

fflush(stdin);

scanf("%c", &opciowo);

switch(opciowo){

case '1':

printf("\nIngrese su nombre completo: ");

while(band!=0){

fflush(stdin);

fgets(user.nombre, sizeof(user.nombre), stdin);

band=1;

i=0;

size\_t len=strlen(user.nombre);

if(len>0&&user.nombre[len-1]=='\n') {

user.nombre[len-1]='\0';

}

if (strcmp(user.nombre, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

while(user.nombre[i] != '\0' && band != 2){

if(!((user.nombre[i] >= 'A' && user.nombre[i] <= 'Z') || (user.nombre[i] >= 'a' && user.nombre[i] <= 'z') || user.nombre[i] == ' ')){

band=2;

}

i++;

}

if((user.nombre[0]!=' ')&&(user.nombre[1]!=' ')&&(((user.nombre[0]>= 'A'))&&((user.nombre[0]<='Z')))){

if(band!=2){

if (strlen(user.nombre)>=2){

if (strlen(user.nombre)<=50){

tipo='n';

if ((contnomb=(user\_registry(user.nombre, tipo)))==1){

band=0;

printf("\nNombre cambiado con exito de %s a %s\n", usa->nombre, user.nombre);

strcpy(usa->nombre, user.nombre);

}else

band=0;

}else

printf("\nEl nombre es muy largo.\n");

}else

printf("\nEl nombre es muy corto.\n");

}else

printf("\nIngrese un nombre valido. (Solo letras)\n");

}else

printf("\nNombre invalido, el primer caracter debe ser una letra mayuscula seguido de un caracter\n");

}

band=1;

break;

case '2':

printf("\nIngrese su correo electronico: ");

while(band!=0){

fflush(stdin);

gets(user.email);

fflush(stdin);

band=1;

i=0;

gm=0;

punto=0;

if (strcmp(user.email, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

while(user.email[i] != '\0' && band != 2){

if(!((user.email[i] >= 'A' && user.email[i] <= 'Z') || (user.email[i] >= 'a' && user.email[i] <= 'z') || (user.email[i] >= '0' && user.email[i] <= '9')||(user.email[i] == '@')||(user.email[i]=='.'))){

band=2;

}

if(user.email[i]=='@'){

gm++;

pos=i;

}

if((user.email[i]=='.')&&(user.email[i]>pos)){

if(user.email[i]=='.'){

punto++;

pos1=i;

}

}

i++;

}

if (punto==1){

i=0;

pos=pos+1;

fflush(stdin);

strcpy(gmail, " ");

fflush(stdin);

while(user.email[pos+i]!='.'&&user.email[pos+i]!='@'){

fflush(stdin);

gmail[i]=user.email[pos+i];

fflush(stdin);

i++;

}

}else

punto=0;

if(user.email[0]!='@'){

if (band!=2){

if(gm==1){

if (punto!=0){

if (strlen(user.email)<=49){

if((strcmp(gmail, "gmail") == 0)||(strcmp(gmail, "hotmail") == 0)||(strcmp(gmail, "yahoo") == 0)||(strcmp(gmail, "outlook") == 0)){

tipo='e';

if((user.email[pos1+1]=='c')&&(user.email[pos1+2]=='o')&&(user.email[pos1+3]=='m')&&(user.email[pos1+4]=='\0')){

if (user\_registry(user.email, tipo)==1){

printf("\nCorreo cambiado con exito de %s a %s\n", usa->email, user.email);

strcpy(usa->email, user.email);

band=0;

}else{

printf("\nEmail ya existente.\n");

}

}else{

band=1;

printf("\nDireccion invalida. Asegurarse que los puntos y dominio esten correctos.\n");

}

}else

printf("\nDireccion inivalida. Solo gmail, yahoo, outlook y hotmail.\n");

}else

printf("\nDireccion inivalida. Excede la longitud permitida.\n");

}else

printf("\nDireccion inivalida. Revisar los puntos\n");

}else

printf("\nDireccion inivalida. Revise la cantidad de arrobas.\n");

}else

printf("\nDireccion inivalida. Solo letras y numeros.\n");

}else

printf("\nDireccion inivalida. El primer caracter debe ser una letra\n");

}

band=1;

break;

case '3':

printf("\nIngrese su numero completo de celular. Agregando el codigo de area: ");

while (band!=0){

fflush(stdin);

gets(user.num);

fflush(stdin);

if (strcmp(user.num, "0") == 0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

band=1;

i=0;

while(user.num[i] != '\0' && band != 2){

if(!((user.num[i] >= '0' && user.num[i] <= '9'))){

band=2;

}

i++;

}

if(band!=2){

if (strlen(user.num)==13){

if ((user.num[0]=='5')&&(user.num[1]=='4')&&(user.num[2]=='9')){

tipo='c';

if (user\_registry(user.num, tipo)==1){

printf("\nNumero cambiado con exito de %s a %s\n", usa->num, user.num);

strcpy(usa->num, user.num);

band=0;

}else

printf("\nNumero ya existente.\n");

}else

printf("\nIngrese un codigo de area valido.(549)\n");

}else

printf("\nIngrese un numero valido. (De 13 numeros).\n");

}else

printf("\nIngrese un numero valido (Sin letras).\n");

}

band=1;

break;

case '4':

printf("\nCree una contrasena (Solo letras y numeros, al menos 8 caracteres de longitud.):\n");

fflush(stdin);

while (band!=0){

i=0;

fflush(stdin);

fgets(user.contra, sizeof(user.contra), stdin);

fflush(stdin);

size\_t len=strlen(user.contra);

if(len>0&&user.contra[len-1]=='\n') {

user.contra[len-1]='\0';

}

if (strcmp(user.contra, "0")==0){

printf("\nProceso detenido por el usuario.\n\n");

break;

}

while(user.contra[i] !='\0' && band != 2){

if(!((user.contra[i] >= 'A' && user.contra[i] <= 'Z') || (user.contra[i] >= 'a' && user.contra[i] <= 'z') || (user.contra[i] >= '0' && user.contra[i] <= '9'))){

band=2;

}else

i++;

}

if(band!=2){

if(strlen(user.contra)<=49){

if(strlen(user.contra)>=8){

printf("\nContrasena cambiada con exito de %s a %s\n", usa->contra, user.contra);

strcpy(usa->contra, user.contra);

band=0;

}else

printf("\nContrasena invalida. Demasiado corta.\n");

}else

printf("\nContrasena invalida. Demasiado larga.\n");

}else

printf("\nContrasena invalida. Solo se permite el uso de numeros y letras.\n");

}

band=1;

break;

case '5':

printf("\nCree un alias (Solo letras y numeros).\n");

fflush(stdin);

while (band!=0){

i=0;

band=1;

fflush(stdin);

gets(user.alias);

if (strcmp(user.alias, "0")==0) {

printf("\nProceso detenido por el usuario.\n\n");

break;

}

while(user.alias[i] !='\0' && band != 2){

if(!((user.alias[i] >= 'A' && user.alias[i] <= 'Z') || (user.alias[i] >= 'a' && user.alias[i] <= 'z') || (user.alias[i] >= '0' && user.alias[i] <= '9'))){

band=2;

}else

i++;

}

if(band==1){

if((strlen(user.alias)==5)){

printf("\nAlias cambiado con exito de %s a %s\n", usa->alias, user.alias);

strcpy(usa->alias, user.alias);

band=0;

}else

printf("\nAlias invalido. Longitud incorrecta (5 caracteres).\n");

}else

printf("\nAlias invalido. Solo se permite el uso de numeros y letras.\n");

}

band=1;

break;

case '6':

printf("\nSaliendo...\n");

sleep(2);

system("cls");

return;

break;

default:

system("cls");

printf("\nOpcion invalida. Volviendo al panel de datos. \n");

break;

}

sleep(2);

system("cls");

goto repeat;

break;

case '2':

printf("\nSaliendo...\n");

sleep(2);

system("cls");

return;

break;

default:

system("cls");

printf("\nError. Ingrese una de las siguientes opciones.\n");

goto repeat;

break;

}

}

void movimiento(char movtxt[100], char opcn, const usuario \*usa, float monto){

movimientos movim;

char archname[50];

char move[200];

time\_t tim;

struct tm \*tm\_info;

char fechtim[20];

time(&tim);

tm\_info=localtime(&tim);

strftime(fechtim, sizeof(fechtim), "%d/%m/%Y %H:%M:%S", tm\_info);

snprintf(archname, sizeof(archname), "%s.txt", usa->cuit);

FILE \*archivito=fopen(archname, "a");

if(archivito==NULL){

printf("\nError al abrir el archivo de movimientos.\n");

}

switch(opcn){

case 't':

snprintf(move, sizeof(move), "%s %s", fechtim, movtxt);

if(strlen(move)>25){

if (fprintf(archivito, "%s\n", move)>0){

printf("\n");

}else

printf("\nSu transferencia no se ha cargado en el sistema.\n");

fclose(archivito);

}

archivito=fopen("movimientos.dat", "ab");

movim.tipo1=opcn;

strcpy(movim.fech, fechtim);

strcpy(movim.cuilt, usa->cuit);

movim.monto=monto;

strcpy(movim.iibb, usa->IVA);

if (archivito == NULL) {

printf("\nError al abrir el archivo binario de movimientos.\n");

return;

}

if (strlen(move) > 25) {

if ((fwrite(&movim, sizeof(movimientos), 1, archivito)!=1)) {

printf("\nSu transferencia no se ha cargado en el sistema.\n");

}

}

fclose(archivito);

sleep(1);

return;

break;

case 'i':

snprintf(move, sizeof(move), "%s %s", fechtim, movtxt);

if(strlen(move)>25){

if (fprintf(archivito, "%s\n", move)>0){

printf("\n");

}else

printf("\nSu transferencia no se ha cargado en el sistema.\n");

fclose(archivito);

}

archivito=fopen("movimientos.dat", "ab");

movim.tipo1=opcn;

strcpy(movim.fech, fechtim);

strcpy(movim.cuilt, usa->cuit);

movim.monto=monto;

strcpy(movim.iibb, usa->IVA);

if (archivito == NULL) {

printf("\nError al abrir el archivo binario de movimientos.\n");

return;

}

if (strlen(move) > 25) {

if ((fwrite(&movim, sizeof(movimientos), 1, archivito)!=1)) {

printf("\nSu transferencia no se ha cargado en el sistema.\n");

}

}

fclose(archivito);

sleep(1);

return;

break;

case 'p':

snprintf(move, sizeof(move), "%s %s", fechtim, movtxt);

if(strlen(move)>25){

if (fprintf(archivito, "%s\n", move)>0){

printf("\n");

}else

printf("\nSu transferencia no se ha cargado en el sistema.\n");

fclose(archivito);

}

archivito=fopen("movimientos.dat", "ab");

movim.tipo1=opcn;

strcpy(movim.fech, fechtim);

strcpy(movim.cuilt, usa->cuit);

movim.monto=monto;

strcpy(movim.iibb, usa->IVA);

if (archivito == NULL) {

printf("\nError al abrir el archivo binario de movimientos.\n");

return;

}

if (strlen(move) > 25) {

if ((fwrite(&movim, sizeof(movimientos), 1, archivito)!=1)) {

printf("\nSu transferencia no se ha cargado en el sistema.\n");

}

}

fclose(archivito);

sleep(1);

return;

break;

}

}

void filtros(const usuario \*usa){

system("cls");

char string[200];

char temp[200];

char archnom[100];

int enc=0;

snprintf(archnom, sizeof(archnom), "%s.txt", usa->cuit);

FILE \*archivote=fopen(archnom, "r");

if (archivote==NULL) {

printf("Error al abrir el archivo %s debido a que aun no existen movimientos en esta cuenta.\n", archnom);

return;

}

printf("\n\t\t---Panel de Filtros---\n");

printf("\nIngrese el movimiento que desea buscar.\nSea por fecha(xx/xx/xxxx), tipo de movimiento(transferencia, ingreso, pago), etc.\n 0 para mostrar todos.\n");

fflush(stdin);

fgets(string, sizeof(string), stdin);

string[strcspn(string, "\n")]='\0';

for (int i=0; string[i]; i++){

string[i]=tolower((unsigned char)string[i]);

}

while (fgets(temp, sizeof(temp), archivote)){

for (int i=0; temp[i];i++){

temp[i]=tolower((unsigned char)temp[i]);

}

if((strstr(temp, string)!=NULL)){

printf("%s\n", temp);

enc=1;

}

}

fclose(archivote);

if(enc==0){

printf("\nNo se encontraron registros para lo ingresado. Volviendo al menu principal...\n");

}

sleep(1);

}

void menu\_adm(){

char cuit[23];

usuario usuwu;

char opciwi;

char tipo;

int pos;

int error=0;

system("cls");

repeat:

FILE \*archivinho=fopen("user.dat", "rb+");

if(archivinho==NULL){

printf("\nError al abrir el archivo de usuarios durante el panel de administrador.\n");

}

printf("\n\t\t---Panel de Administrador---\n");

printf("\nSeleccione que desea hacer:\n");

printf("Bienvenido, Administrador.\n");

printf("\n\t\t1. Modificacion de Usuarios\n\t\t2. Adjudicar Roles\n\t\t3. Datos de Usuario y Resumenes de Actividades \n\t\t4. Salir\n");

fflush(stdin);

scanf("%c", &opciwi);

switch(opciwi){

case '1':

error=0;

printf("\nIngrese el CUIT/L del usuario a modificar: ");

fflush(stdin);

scanf("%s", &cuit);

while ((fread(&usuwu, sizeof(usuario), 1, archivinho))==1){

if (strcmp(usuwu.cuit, cuit) == 0) {

pos = ftell(archivinho) - sizeof(usuario);

fseek(archivinho, pos, SEEK\_SET);

tipo='m';

admin(tipo, &usuwu);

if (fwrite(&usuwu, sizeof(usuario), 1, archivinho) != 1) {

printf("\nError al escribir los datos en el archivo\n");

}

error=1;

break;

}

}

if(error==0){

printf("\nUsuario no encontrado. Volviendo al menu principal...\n");

}

fclose(archivinho);

sleep(1);

goto repeat;

break;

case '2':

error=0;

printf("\nIngrese el CUIT/L del usuario a adjudicar rol: ");

fflush(stdin);

scanf("%s", &cuit);

while ((fread(&usuwu, sizeof(usuario), 1, archivinho))==1){

if (strcmp(usuwu.cuit, cuit) == 0) {

pos=ftell(archivinho)-sizeof(usuario);

fseek(archivinho, pos, SEEK\_SET);

tipo='a';

admin(tipo, &usuwu);

if (fwrite(&usuwu, sizeof(usuario), 1, archivinho) != 1) {

printf("\nError al escribir los datos en el archivo\n");

}

error=1;

break;

}

}

if(error==0){

printf("\nUsuario no encontrado. Volviendo al menu principal...\n");

}

fclose(archivinho);

sleep(1);

goto repeat;

break;

case '3':

error=0;

printf("\nIngrese el CUIT/L del usuario a consultar: ");

fflush(stdin);

scanf("%s", &cuit);

while ((fread(&usuwu, sizeof(usuario), 1, archivinho))==1){

if (strcmp(usuwu.cuit, cuit) == 0) {

pos=ftell(archivinho)-sizeof(usuario);

fseek(archivinho, pos, SEEK\_SET);

tipo='r';

admin(tipo, &usuwu);

error=1;

break;

}

}

if(error==0){

printf("\nUsuario no encontrado. Volviendo al menu principal...\n");

}

fclose(archivinho);

sleep(1);

goto repeat;

break;

case '4':

printf("\nSaliendo...\n");

sleep(2);

system("cls");

return;

break;

default:

system("cls");

printf("\nError. Ingrese una de las siguientes opciones.\n");

goto repeat;

break;

}

printf("dea");

}

void admin(char tipo, usuario \*usuwu){

usuario uss;

char opcion;

movimientos movi;

system("cls");

repeat:

FILE \*ARC=fopen("movimientos.dat", "rb");

int error=0;

int movim=0;

float toti=0;

float totp=0;

float tott=0;

int band=1;

int i=0;

int gm=0;

int punto=0;

int pos;

int pos1;

int condicion;

char gmail[50];

int rol;

FILE \*archivinho=fopen("user.dat", "rb");

switch(tipo){

case 'm':

printf("\n---Modificar usuario---\n");

printf("Ingrese lo que desea modificar:\n ");

printf("\n\t\t1. Nombre\n\t\t2. Correo\n\t\t3. Numero\n\t\t4. Contrasena\n\t\t5. Alias\n\t\t6. CUIT\n\t\t7. Condicion Fiscal\n\t\t8. Salir\n");

fflush(stdin);

scanf("%c", &opcion);

switch(opcion){

case '1':

printf("\nIngrese el nuevo nombre: ");

while(band!=0){

fflush(stdin);

fgets(uss.nombre, sizeof(uss.nombre), stdin);

band=1;

i=0;

size\_t len=strlen(uss.nombre);

if(len>0&&uss.nombre[len-1]=='\n') {

uss.nombre[len-1]='\0';

}

if (strcmp(uss.nombre, "0") == 0) {

printf("\nProceso detenido por el administrador.\n");

break;

}

while(uss.nombre[i] != '\0' && band != 2){

if(!((uss.nombre[i] >= 'A' && uss.nombre[i] <= 'Z') || (uss.nombre[i] >= 'a' && uss.nombre[i] <= 'z') || uss.nombre[i] == ' ')){

band=2;

}

i++;

}

if((uss.nombre[0]!=' ')&&(uss.nombre[1]!=' ')&&(((uss.nombre[0]>= 'A'))&&((uss.nombre[0]<='Z')))){

if(band!=2){

if (strlen(uss.nombre)>=2){

if (strlen(uss.nombre)<=50){

tipo='n';

if (((user\_registry(uss.nombre, tipo)))==1){

band=0;

printf("\nNombre cambiado con exito de %s a %s\n", usuwu->nombre, uss.nombre);

strcpy(usuwu->nombre, uss.nombre);

}else

band=0;

}else

printf("\nEl nombre es muy largo.\n");

}else

printf("\nEl nombre es muy corto.\n");

}else

printf("\nIngrese un nombre valido. (Solo letras)\n");

}else

printf("\nNombre invalido, el primer caracter debe ser una letra mayuscula seguido de un caracter\n");

}

band=1;

break;

case '2':

printf("Ingrese el nuevo correo: \n");

while(band!=0){

fflush(stdin);

gets(uss.email);

fflush(stdin);

if (strcmp(uss.email, "0") == 0) {

printf("\nProceso detenido por el administrador.\n");

break;

}

band=1;

i=0;

gm=0;

punto=0;

while(uss.email[i] != '\0' && band != 2){

if(!((uss.email[i] >= 'A' && uss.email[i] <= 'Z') || (uss.email[i] >= 'a' && uss.email[i] <= 'z') || (uss.email[i] >= '0' && uss.email[i] <= '9')||(uss.email[i] == '@')||(uss.email[i]=='.'))){

band=2;

}

if(uss.email[i]=='@'){

gm++;

pos=i;

}

if((uss.email[i]=='.')&&(uss.email[i]>pos)){

if(uss.email[i]=='.'){

punto++;

pos1=i;

}

}

i++;

}

if (punto==1){

i=0;

pos=pos+1;

fflush(stdin);

strcpy(gmail, " ");

fflush(stdin);

while(uss.email[pos+i]!='.'&&uss.email[pos+i]!='@'){

fflush(stdin);

gmail[i]=uss.email[pos+i];

fflush(stdin);

i++;

}

}else

punto=0;

if(uss.email[0]!='@'){

if (band!=2){

if(gm==1){

if (punto!=0){

if (strlen(uss.email)<=49){

if((strcmp(gmail, "gmail") == 0)||(strcmp(gmail, "hotmail") == 0)||(strcmp(gmail, "yahoo") == 0)||(strcmp(gmail, "outlook") == 0)){

tipo='e';

if((uss.email[pos1+1]=='c')&&(uss.email[pos1+2]=='o')&&(uss.email[pos1+3]=='m')&&(uss.email[pos1+4]=='\0')){

if (user\_registry(uss.email, tipo)==1){

printf("\nCorreo cambiado con exito de %s a %s\n", usuwu->email, uss.email);

strcpy(usuwu->email, uss.email);

band=0;

}else{

printf("\nEmail ya existente.\n");

}

}else{

band=1;

printf("\nDireccion invalida. Asegurarse que los puntos y dominio esten correctos.\n");

}

}else

printf("\nDireccion invalida. Solo gmail, yahoo, outlook y hotmail.\n");

}else

printf("\nDireccion invalida. Excede la longitud permitida.\n");

}else

printf("\nDireccion invalida. Revisar los puntos\n");

}else

printf("\nDireccion invalida. Revise la cantidad de arrobas.\n");

}else

printf("\nDireccion invalida. Solo letras y numeros.\n");

}else

printf("\nDireccion invalida. El primer caracter debe ser una letra\n");

}

band=1;

break;

case '3':

printf("\nIngrese el numero completo de celular. Agregando el codigo de area: ");

while (band!=0){

fflush(stdin);

gets(uss.num);

fflush(stdin);

band=1;

i=0;

if (strcmp(uss.num, "0") == 0) {

printf("\nProceso detenido por el Administrador.\n");

return;

}

while(uss.num[i] != '\0' && band != 2){

if(!((uss.num[i] >= '0' && uss.num[i] <= '9'))){

band=2;

}

i++;

}

if(band!=2){

if (strlen(uss.num)==13){

if ((uss.num[0]=='5')&&(uss.num[1]=='4')&&(uss.num[2]=='9')){

tipo='c';

if (user\_registry(uss.num, tipo)==1){

printf("\nNumero cambiado con exito de %s a %s\n", usuwu->num, uss.num);

strcpy(usuwu->alias, uss.alias);

band=0;

}else

printf("\nNumero ya existente.\n");

}else

printf("\nIngrese un codigo de area valido.(549)\n");

}else

printf("\nIngrese un numero valido. (De 13 numeros).\n");

}else

printf("\nIngrese un numero valido (Sin letras).\n");

}

band=1;

case '4':

printf("\nCree una contrasena (Solo letras y numeros, al menos 8 caracteres de longitud.):\n");

fflush(stdin);

while (band!=0){

i=0;

fflush(stdin);

fgets(uss.contra, sizeof(uss.contra), stdin);

fflush(stdin);

size\_t len=strlen(uss.contra);

if(len>0&&uss.contra[len-1]=='\n') {

uss.contra[len-1]='\0';

}

if (strcmp(uss.contra, "0") == 0) {

printf("\nProceso detenido por el Administrador.\n");

return;

}

while(uss.contra[i] !='\0' && band != 2){

if(!((uss.contra[i] >= 'A' && uss.contra[i] <= 'Z') || (uss.contra[i] >= 'a' && uss.contra[i] <= 'z') || (uss.contra[i] >= '0' && uss.contra[i] <= '9'))){

band=2;

}else

i++;

}

if(band!=2){

if(strlen(uss.contra)<=49){

if(strlen(uss.contra)>=8){

band=0;

}else

printf("\nContrasena invalida. Demasiado corta.\n");

}else

printf("\nContrasena invalida. Demasiado larga.\n");

}else

printf("\nContrasena invalida. Solo se permite el uso de numeros y letras.\n");

}

case '5':

printf("\nCree un alias (Solo letras y numeros).\n");

fflush(stdin);

while (band!=0){

i=0;

band=1;

fflush(stdin);

gets(uss.alias);

if (strcmp(uss.alias, "0")==0) {

printf("\nProceso detenido por el Administrador.\n\n");

break;

}

while(uss.alias[i] !='\0' && band != 2){

if(!((uss.alias[i] >= 'A' && uss.alias[i] <= 'Z') || (uss.alias[i] >= 'a' && uss.alias[i] <= 'z') || (uss.alias[i] >= '0' && uss.alias[i] <= '9'))){

band=2;

}else

i++;

}

if(band==1){

if((strlen(uss.alias)==5)){

printf("\nAlias cambiado con exito de %s a %s\n", usuwu->alias, uss.alias);

strcpy(usuwu->alias, uss.alias);

band=0;

}else

printf("\nAlias invalido. Longitud incorrecta (5 caracteres).\n");

}else

printf("\nAlias invalido. Solo se permite el uso de numeros y letras.\n");

}

band=1;

break;

case '6':

printf("\nIngrese el nuevo CUIT/L: ");

while (band!=0){

fflush(stdin);

gets(uss.cuit);

fflush(stdin);

band=1;

i=0;

if (strcmp(uss.cuit, "0") == 0) {

printf("\nProceso detenido por el Administrador.\n");

break;

}

while(uss.cuit[i] != '\0' && band != 2){

if(!((uss.cuit[i] >= '0' && uss.cuit[i] <= '9'))){

band=2;

}

i++;

}

if(band!=2){

if (strlen(uss.cuit)==11){

if (((uss.cuit[0]=='2')&&(uss.cuit[1]=='7'))||((uss.cuit[0]=='2')&&(uss.cuit[1]=='0'))||((uss.cuit[0]=='3')&&(uss.cuit[1]=='0'))){

if((11-((((uss.cuit[0]-'0')\*5)+((uss.cuit[1]-'0')\*4)+((uss.cuit[2]-'0')\*3)+((uss.cuit[3]-'0')\*2)+((uss.cuit[4]-'0')\*7)+((uss.cuit[5]-'0')\*6)+((uss.cuit[6]-'0')\*5)+((uss.cuit[7]-'0')\*4)+((uss.cuit[8]-'0')\*3)+((uss.cuit[9]-'0')\*2))%11))==(uss.cuit[10]-'0')){

tipo='t';

if (user\_registry(uss.cuit, tipo)==1){

printf("\nCUIT/L cambiado con exito de %s a %s\n", usuwu->cuit, uss.cuit);

strcpy(usuwu->cuit, uss.cuit);

band=0;

}else

printf("\nCUIT/L ya existente.\n");

}else

printf("\nCUIT/L invalido. No verifica.\n");

}else

printf("\nPrimeros digitos de CUIT/L invalidos.\n");

}else

printf("\nLa longitud del CUIT/L es invalida.\n");

}else

printf("\nIngrese un cuit valido. (Solo numeros)\n");

}

band=1;

break;

case '7':

tipo='t';

if (user\_registry(uss.cuit, tipo)==1){

printf("\nElija la condicion ante el IVA de la persona en base a los datos de la AFIP:\n");

printf("\n\t\t1. Sujeto Exento\n\t\t2. Consumidor Final\n\t\t3. Responsable Monotributo\n\t\t4. No categorizado\n\t\t5. Proveedor del Exterior\n\t\t6. Cliente del Exterior\n\t\t7. IVA Liberado\n\t\t8. Monotributista Social\n\t\t9. No Acutualizadon\t\t9. Responsable Inscripto\n");

while(band!=0){

fflush(stdin);

scanf("%d", &condicion);

if(condicion==0){

band=0;

printf("\nProceso terminado por el usuario.\n");

sleep(1);

break;

}

if (((condicion>=1)&&(condicion<=10))){

if(condicion==1){

strcpy(usuwu->IVA, "Sujeto Exento");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==2){

strcpy(usuwu->IVA, "Consumidor Final");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==3){

strcpy(usuwu->IVA, "Responsable Monotributo");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==4){

strcpy(usuwu->IVA, "No Categorizado");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==5){

strcpy(usuwu->IVA, "Proveedor del Exentor");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==6){

strcpy(usuwu->IVA, "Cliente del Exterior");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==7){

strcpy(usuwu->IVA, "IVA Liberado");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==8){

strcpy(usuwu->IVA, "Monotributista Social");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==9){

strcpy(usuwu->IVA, "IVA no Alcanzado");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

if(condicion==10){

strcpy(usuwu->IVA, "Responsable Inscripto");

printf("\nCondicion cambiada con exito!\n");

band=0;

}

}else

printf("\nInvalido. Elija una de las opciones dadas.\n");

}

}

break;

case '8':

printf("\nSaliendo...\n");

sleep(2);

system("cls");

return;

break;

default:

system("cls");

printf("\nError. Ingrese una de las siguientes opciones.\n");

goto repeat;

break;

}

sleep(2);

return;

break;

case 'a':

system("cls");

printf("\n\t\t---Adjudicar Rol---\n");

printf("\nIngrese el nuevo rol:\n");

printf("\n\t\t1. Administrador\n\t\t2. Usuario\n");

band=1;

while(band!=0){

scanf("%d", &rol);

if(rol==0){

printf("Proceso detenido por el Administrador.");

band=0;

break;

}

if(((rol==1)||(rol==2))){

if(rol==1){

usuwu->rol='0';

printf("\nRol cambiado con exito!\n");

band=0;

}

else{

usuwu->rol='1';

printf("\nRol cambiado con exito!\n");

band=0;

}

}else

printf("\nInvalido. Ingrese una de las opciones dadas.\n");

}

sleep(2);

return;

break;

case 'r':

movim=0;

error=0;

printf("\n\t\t---Panel de Datos de Usuario y Resumen de Cuenta---\n");

printf("Los datos del usuario seleccionado son los siguientes:\n");

printf("Nombre: %s\n", usuwu->nombre);

printf("Correo: %s\n", usuwu->email);

printf("Celular: %s\n", usuwu->num);

printf("CUIT/L: %s\n", usuwu->cuit);

printf("Contrasena: %s\n", usuwu->contra);

printf("Alias: %s\n", usuwu->alias);

printf("CVU: %s\n", usuwu->CVU);

printf("Condicion fiscal: %s\n", usuwu->IVA);

while ((fread(&movi, sizeof(movimientos), 1, ARC))==1){

if (strcmp(usuwu->cuit, movi.cuilt) == 0) {

movim++;

if(movi.tipo1=='t'){

tott=tott+movi.monto;

}

if(movi.tipo1=='i'){

toti=toti+movi.monto;

}

if(movi.tipo1=='p'){

totp=totp+movi.monto;

error=1;

}

}

}

fclose(ARC);

if(error==0){

printf("\nUsuario no encontrado. Volviendo al menu principal...\n");

break;

}

printf("\nEl usuario posee %d movimientos.\n", movim);

printf("\nEl usuario ingreso %.2f pesos.\n", toti);

printf("\nEl usuario transfirio %.2f pesos.\n", tott);

printf("\nEl usuario pago %.2f pesos.\n", totp);

sleep(2);

return;

break;

case '4':

printf("\nSaliendo...\n");

sleep(2);

system("cls");

return;

break;

default:

system("cls");

printf("\nError. Ingrese una de las siguientes opciones.\n");

goto repeat;

break;

}

fclose(ARC);

}

Imagen de archive binario de usuario:

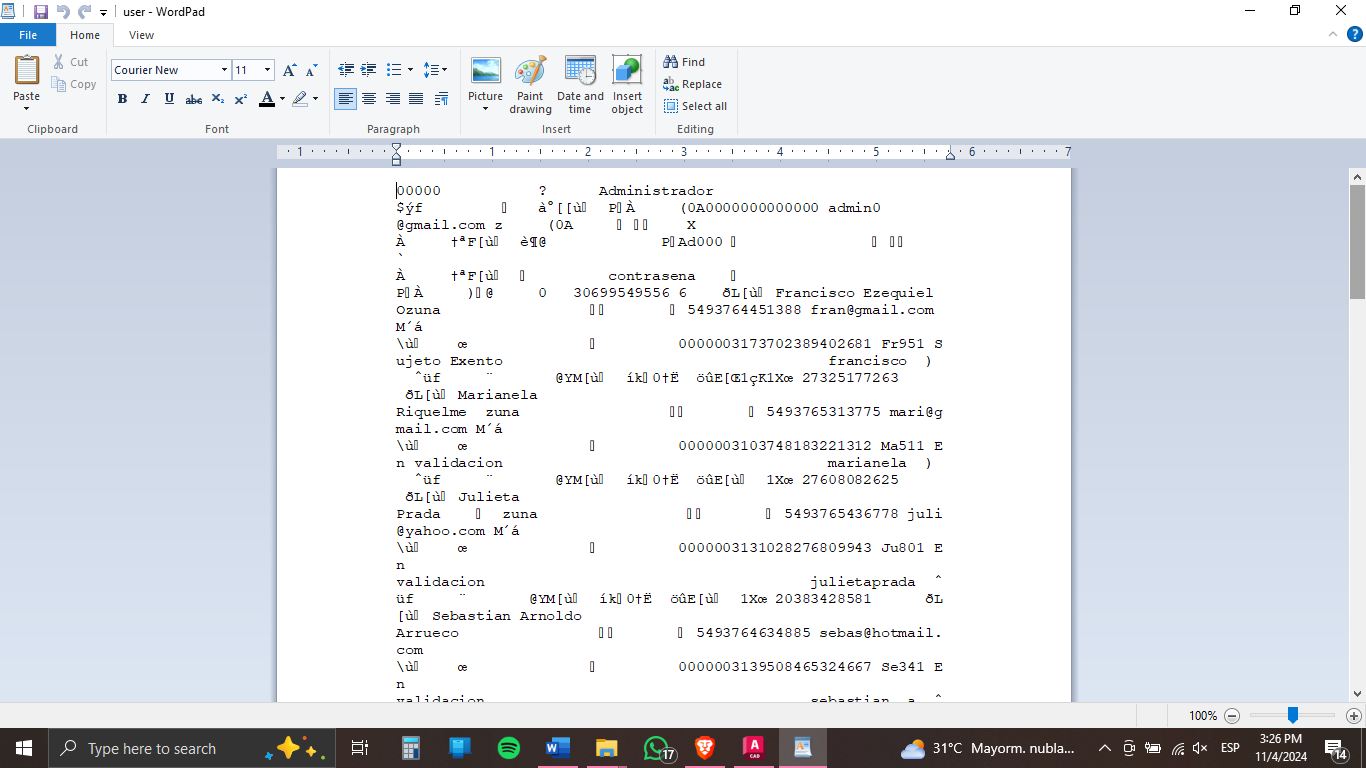


Imagen de archivo de movimientos:

A screenshot of a computer

Description automatically generated

Imagen archivo cuentabanco:

A screenshot of a computer

Description automatically generated

Imagen de archivo para validar IVA:

A screenshot of a computer

Description automatically generated

Imagen de archivo de movimientos de un usuario:

A screenshot of a computer

Description automatically generated