import mysql.connector  **# Importa la librería mysql.connector para interactuar con MySQL**

from hashlib import sha256 **# Importa la función sha256 desde el módulo hashlib para cifrar contraseñas**

from getpass import getpass **# Importa la función getpass para ocultar la contraseña al ingresarla por consola**

import os **# Importa el módulo os para interactuar con el sistema operativo**

import time **# Importa el módulo time para agregar pausas en el programa**

from datetime import datetime  **# Importa la clase datetime desde el módulo datetime para trabajar con fechas y horas**

db = None  **# Variable para almacenar la conexión a la base de datos (inicialmente se establece como None)**

cursor = None **# Variable para almacenar el cursor de la base de datos (inicialmente se establece como None)**

sesion = None **# Variable para almacenar la sesión (inicialmente se establece como None)**

opc\_selected = None  **# Variable para almacenar la opción seleccionada (inicialmente se establece como None)**

timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S") **# Variable para almacenar** la fecha y hora actual en formato de cadena ("YYYY-MM-DD HH:MM:SS")

**# La sección anterior define algunas variables globales que se utilizarán en el código.**

class bcolors:

OK = '\033[92m' # VERDE

WARNING = '\033[93m' # AMARILLO

FAIL = '\033[91m' # ROJO

RESET = '\033[0m' # REINICIAR COLOR

**# La clase "bcolors" se utiliza para definir códigos de escape ANSI que permiten cambiar el color del texto en la consola.**

**def limpiar\_pantalla():**

**"""**

**Limpia la pantalla de la consola.**

**Utiliza el comando 'cls' en Windows y 'clear' en Unix.**

**"""**

os.system('cls' if os.name == 'nt' else 'clear')

**def inicializar\_database():**

**"""**

**Establece la conexión con la base de datos y crea el cursor.**

**"""**

try:

global db, cursor

**# Establece la conexión con la base de datos**

db = mysql.connector.connect(

host="localhost",

user="root",

password="",

database="db\_proyecto"

)

**# Crea un cursor para ejecutar consultas en la base de datos**

**cursor = db.cursor()**

**# Verifica si la conexión fue exitosa**

if db.is\_connected():

print("CONEXION EXITOSA")

print("INICANDO...")

# Pausa de 1.5 segundos antes de continuar

time.sleep(1.5)

limpiar\_pantalla()

except Exception as ex:

**# Si ocurre una excepción, muestra el mensaje de error**

print(ex)

print("POR FAVOR CONTACTE CON UN ADMINISTRADOR")

**# Pausa de 4 segundos antes de continuar**

time.sleep(4)

**# La función "inicializar\_database()" se encarga de establecer la conexión con la base de datos y crear el cursor para ejecutar consultas.**

**def cerrar\_conex\_db():**

**"""**

**Cierra la conexión con la base de datos y el cursor.**

**"""**

if cursor:

cursor.close() **# Cierra el cursor**

if db:

db.close() **# Cierra la conexión a la base de datos**

**# La función "cerrar\_conex\_db()" se encarga de cerrar la conexión con la base de datos y el cursor.**

**def register():**

**"""**

**Registra un nuevo usuario en la base de datos.**

**Solicita al usuario un nombre de usuario y contraseña, verifica que las contraseñas coincidan**

**y que el nombre de usuario no esté repetido. Luego, cifra la contraseña y la guarda en la base de datos.**

**"""**

while True:

limpiar\_pantalla()

print("---------------------------------------")

print("| Registro |")

print("---------------------------------------")

username = input("Nombre de usuario: ")

password = input("Contraseña: ")

password\_verify = input("Vuelve a introducir la contraseña: ")

if password == password\_verify:

print("Las contraseñas coinciden")

time.sleep(1)

break

else:

print("Las contraseñas no coinciden, vuelve a intentarlo")

time.sleep(1)

limpiar\_pantalla()

**# Consulta si el nombre de usuario ya existe en la base de datos**

sql = "SELECT nombre\_usuario FROM usuarios WHERE nombre\_usuario = %s"

cursor.execute(sql, (username,))

result = cursor.fetchone()

if result is None:

print("Registrando...")

time.sleep(3)

rol = 'user'

**# Se cifra la contraseña**

salt = "s3cr3t5alt"

hashed\_password = sha256((password + salt).encode()).hexdigest()

**# La contraseña se guarda en la base de datos**

sql = "INSERT INTO usuarios (nombre\_usuario, contraseña, rol) VALUES (%s, %s, %s)"

values = (username, hashed\_password, rol)

cursor.execute(sql, values)

db.commit()

**# Indica que el registro fue exitoso**

limpiar\_pantalla()

print("Usuario registrado correctamente.")

time.sleep(1)

**# Registrar en el historial**

registrar\_historial(f"Registro de usuario", username)

break

else:

print("El usuario ya existe, vuelve a intentarlo")

time.sleep(1)

**# La función "register()" se encarga de solicitar al usuario un nombre de usuario y contraseña para registrarlos en la base de datos.**

**# Verifica que las contraseñas coincidan y que el nombre de usuario no esté repetido.**

**# Luego, cifra la contraseña y la guarda en la base de datos junto con el nombre de usuario y un rol predeterminado.**

**# También muestra mensajes en la consola para informar al usuario sobre el proceso de registro.**

**# Aquí puedes continuar documentando el resto del código según tus necesidades.**

**# Función que compara el usuario y contraseña con las ingresadas a la bd para iniciar sesión**

**def login():**

**"""**

**Realiza el proceso de inicio de sesión comparando el nombre de usuario y contraseña ingresados**

**con los datos almacenados en la base de datos.**

**Returns:**

**bool: True si el inicio de sesión es exitoso, False si falla.**

**"""**

global role, username, password  **# Se definen las variables globales role, username y password**

limpiar\_pantalla() # Llama a la función "limpiar\_pantalla()" para limpiar la pantalla de la consola

print("---------------------------------------")

print("| Acceso |") **# Imprime el encabezado de inicio de sesión**

print("---------------------------------------")

username = input("Nombre de usuario: ") **# Solicita al usuario que ingrese su nombre de usuario**

password = getpass("Contraseña: ") **# Solicita al usuario que ingrese su contraseña de manera segura (sin mostrarla en pantalla)**

sql = "SELECT contraseña, rol FROM usuarios WHERE nombre\_usuario = %s" # Consulta SQL para obtener la contraseña y el rol asociados al nombre de usuario ingresado

cursor.execute(sql, (username,)) **# Ejecuta la consulta SQL con el nombre de usuario proporcionado**

result = cursor.fetchone() **# Obtiene el resultado de la consulta**

if result is None: **# Si no se encontró ningún resultado en la consulta**

print("Nombre de usuario incorrecto. Vuelve a intentarlo")  **# Imprime un mensaje de error**

time.sleep(1)  **# Espera 1 segundo antes de continuar**

else:

hashed\_password = result[0] **# Obtiene la contraseña almacenada en el resultado de la consulta**

role = result[1] **# Obtiene el rol almacenado en el resultado de la consulta**

salt = "s3cr3t5alt" **# Define una sal (salt) para fortalecer la contraseña**

entered\_password = sha256((password + salt).encode()).hexdigest()  **# Aplica el hash a la contraseña ingresada por el usuario**

if hashed\_password == entered\_password:  **# Si la contraseña ingresada coincide con la almacenada en la base de datos**

limpiar\_pantalla() **# Llama a la función "limpiar\_pantalla()" para limpiar la pantalla de la consola**

if role == "admin": **# Si el rol es "admin"**

print("¡Bienvenido administrador!") **# Imprime un mensaje de bienvenida para el administrador**

menu\_home()  **# Llama a la función "menu\_home()" para mostrar el menú principal**

else:

print(f"Bienvenido @{username} inicio de sesión exitoso") **# Imprime un mensaje de bienvenida para el usuario con su nombre de usuario**

menu\_home()  **# Llama a la función "menu\_home()" para mostrar el menú principal**

registrar\_historial(f"Inicio de sesión", username) **# Registra el inicio de sesión en el historial**

return True  **# Retorna True para indicar un inicio de sesión exitoso**

else:

print("Contraseña incorrecta.")  **# Imprime un mensaje de error de contraseña incorrecta**

time.sleep(1.5) **# Espera 1.5 segundos antes de continuar**

return False  **# Retorna False para indicar un inicio de sesión fallido**

**# Código para cambio de contraseña**

**def change\_password():**

**"""**

**Permite al usuario cambiar su contraseña actual ingresando su nombre de usuario, contraseña actual y la nueva contraseña.**

**"""**

limpiar\_pantalla()  **# Llama a la función "limpiar\_pantalla()" para limpiar la pantalla de la consola**

print("---------------------------------------")

print("| Cambiar contraseña |")

print("---------------------------------------")

username = input("Nombre de usuario: ") **# Solicita al usuario que ingrese el nombre de usuario.**

current\_password = getpass("Contraseña actual: ") **# Solicita al usuario que ingrese la contraseña actual sin mostrarla en pantalla.**

new\_password = getpass("Nueva contraseña: ") **# Solicita al usuario que ingrese la nueva contraseña sin mostrarla en pantalla.**

**# Consulta la base de datos para obtener la contraseña almacenada asociada al nombre de usuario ingresado.**

sql = "SELECT contraseña FROM usuarios WHERE nombre\_usuario = %s"

cursor.execute(sql, (username,))

result = cursor.fetchone()

if result is None:

print("Nombre de usuario incorrecto.")  **# Si no se encuentra un resultado, el nombre de usuario ingresado es incorrecto.**

else:

hashed\_password = result[0] **# La contraseña almacenada en la base de datos**.

salt = "s3cr3t5alt" # Una sal (salt) que se agrega a la contraseña antes de generar su hash.

entered\_password = sha256((current\_password + salt).encode()).hexdigest()  **# Genera el hash de la contraseña actual ingresada por el usuario.**

if hashed\_password == entered\_password:

**# Si el hash de la contraseña ingresada coincide con el hash almacenado en la base de datos, procede a actualizar la contraseña.**

new\_hashed\_password = sha256((new\_password + salt).encode()).hexdigest() # Genera el hash de la nueva contraseña.

update\_sql = "UPDATE usuarios SET contraseña = %s WHERE nombre\_usuario = %s"

cursor.execute(update\_sql, (new\_hashed\_password, username))

db.commit()

limpiar\_pantalla()  **# Llama a la función "limpiar\_pantalla()" para limpiar la pantalla de la consola**

print("Contraseña actualizada correctamente.")

**# Registrar en el historial**

registrar\_historial(f"Cambio de contraseña", username)  **# Registra el evento de cambio de contraseña en el historial.**

else:

limpiar\_pantalla()  **# Llama a la función "limpiar\_pantalla()" para limpiar la pantalla de la consola**

print("Contraseña incorrecta.")  **# Si los hashes no coinciden, la contraseña actual ingresada es incorrecta.**

**# Código por si no se introduce un carácter válido**

**def opcion\_incorrecta():**

**"""**

**Maneja la situación en la que se selecciona una opción incorrecta en el menú.**

**Imprime un mensaje de error indicando que la opción seleccionada es incorrecta y**

**motiva al usuario a intentarlo nuevamente.**

**"""**

print("Error: Opción Incorrecta")  **# Imprime un mensaje de error indicando que la opción seleccionada es incorrecta.**

print("La opción seleccionada no es un número o no es una opción disponible") **# Imprime un mensaje que indica que la opción no es un número válido o no está disponible.**

print("VUELVE A INTENTARLO")  **# Imprime un mensaje instando al usuario a intentarlo nuevamente.**

**# Menú Principal**

**def menu\_principal():**

**"""**

**Muestra el menú principal con opciones para registro, inicio de sesión, cambio de contraseña y salir.**

**"""**

limpiar\_pantalla() # Llama a la función "limpiar\_pantalla()" para limpiar la pantalla de la consola

print("---------------------------------------")

print("| MENU PRINCIPAL |")

print("|-------------------------------------|")

print("| 1.- Registro |") # Opción 1: Registro

print("| 2.- Acceso |") # Opción 2: Inicio de sesión

print("| 3.- Cambiar contraseña |") # Opción 3: Cambiar contraseña

print("| 4.- Salir |") # Opción 4: Salir

print("|-------------------------------------|")

print("|Consult Date:", timestamp, " |") # Fecha de consulta

print("---------------------------------------")

**#Funcion para el proceso de seleccion en los menus**

**def seleccion():**

**"""**

**Esta función permite al usuario seleccionar una opción del menú.**

**Solicita al usuario que ingrese la opción seleccionada y valida que sea un número entero.**

**Si el usuario ingresa un valor no válido, se llama a la función 'opcion\_incorrecta',**

**se realiza una pausa de 0.5 segundos y se limpi**a la pantalla de la consola.

**Devuelve el valor de la opción seleccionada.**

**"""**

global opc\_selected  **# Declarando la variable opc\_selected como global**

while True: # Bucle infinito

try:

opc\_selected = int(input("Escribe la opción a seleccionar: "))  **# Solicitando al usuario que ingrese la opción seleccionada**

return opc\_selected  **# Devolviendo el valor de la opción seleccionada**

except ValueError: **# Capturando una excepción si el usuario ingresa un valor no válido**

opcion\_incorrecta()  **# Llamando a una función llamada opcion\_incorrecta**

time.sleep(0.5)  **# Haciendo una pausa de 0.5 segundos**

limpiar\_pantalla()  **# Llama a la función "limpiar\_pantalla()" para limpiar la pantalla de la consola**

**def mostrar\_historial():**

**"""**

**Esta función muestra el historial guardado en la base de datos.**

**Realiza una consulta SQL para obtener el historial del usuario específico.**

**Imprime el historial en la consola, mostrando el ID de referencia, la acción realizada,**

**la fecha y hora de la acción.**

**Si no hay registros en el historial, se imprime un mensaje indicando que no hay registros.**

**"""**

sql = "SELECT \* FROM historial WHERE usuario = %s ORDER BY fecha\_hora DESC" **# Consulta SQL para obtener el historial del usuario específico**

cursor.execute(sql, (username,)) **# Ejecutar la consulta SQL con el valor del nombre de usuario proporcionado**

result = cursor.fetchall() **# Obtener todos los resultados de la consulta**

if result: **# Si hay resultados**

limpiar\_pantalla() **# Llamar a la función para limpiar la pantalla**

print("--------------------------------------------")

print("| Historial |") **# Imprimir encabezado del historial**

print("--------------------------------------------")

for row in result:  **# Iterar sobre cada fila de resultados**

print("ID de referencia:", row[0]) **# Imprimir el ID de referencia de la acción**

print(bcolors.OK + "Acción:", row[1] + bcolors.RESET)  **# Imprimir la acción resaltada en color verde**

print("Fecha y hora:", row[2]) **# Imprimir la fecha y hora de la acción**

print("--------------------------------------------")

else:

print("No hay registros en el historial.")  **# Imprimir mensaje si no hay registros en el historial**

**def borrar\_historial():**

**"""**

**Esta función borra todo el historial almacenado en la base de datos.**

**Solicita confirmación al usuario antes de realizar la operación.**

**Si el usuario confirma, se ejecuta una consulta SQL para borrar todos los registros de la tabla 'historial'.**

**Se confirman los cambios en la base de datos y se imprime un mensaje de confirmación.**

**Si el usuario cancela la operación, se imprime un mensaje indicando que la operación fue cancelada.**

**"""**

limpiar\_pantalla()  **# Llamar a la función para limpiar la pantalla**

confirmacion = input(bcolors.WARNING + "¿Estás seguro de que quieres borrar el historial? (s/n): " + bcolors.RESET) **# Solicitar confirmación para borrar el historial**

if confirmacion.lower() == 's':  **# Si la confirmación es 's'**

sql = "TRUNCATE TABLE historial"  **# Consulta SQL para borrar todos los registros de la tabla 'historial'**

cursor.execute(sql) **# Ejecutar la consulta SQL**

db.commit()  **# Confirmar los cambios en la base de datos**

time.sleep(1)

print("Historial borrado correctamente.")  **# Imprimir mensaje de confirmación**

else:

print("Operación cancelada.") **# Imprimir mensaje si la operación es cancelada**

time.sleep(1)

**def mostrar\_usuarios():**

**"""**

**Esta función muestra los usuarios registrados en la base de datos.**

**Realiza una consulta SQL para obtener todos los usuarios ordenados por fecha y hora.**

**Imprime los usuarios en la consola, mostrando el ID de referencia, la acción realizada,**

**la fecha y hora de la acción, y el nombre de usuario.**

**Si no hay registros de usuarios, se imprime un mensaje indicando que no hay registros.**

**"""**

limpiar\_pantalla()  **# Llamar a la función para limpiar la pantalla**

sql = "SELECT \* FROM usuarios ORDER BY fecha\_hora ASC"  **# Consulta SQL para obtener todos los usuarios ordenados por fecha y hora**

cursor.execute(sql)  **# Ejecutar la consulta SQL**

result = cursor.fetchall() **# Obtener todos los resultados de la consulta**

if result:  **# Si hay resultados**

print("------------------------------------------------")

print("| Uuarios registrados |") **# Imprimir encabezado de los usuarios** registrados

print("------------------------------------------------")

limpiar\_pantalla() **# Llamar a la función para limpiar la pantalla**

for row in result: **# Iterar sobre cada fila de resultados**

print("ID de referencia:", row[0])  **# Imprimir el ID de referencia del usuario**

print(bcolors.OK + "Acción:", row[1] + bcolors.RESET) **# Imprimir la acción resaltada en color verde**

print("Fecha y hora:", row[2]) **# Imprimir la fecha y hora de la acción**

print(bcolors.OK + "Usuario:", row[3] + bcolors.RESET) **# Imprimir el nombre de usuario resaltado en color verde**

print("---------------------------------------")

else:

print("No hay registros en el historial de usuarios.") **# Imprimir mensaje si no hay registros de usuarios**

**def eliminar\_usuario():**

**"""**

**Esta función permite eliminar a un usuario del sistema.**

**Solicita el nombre de usuario a eliminar y la contraseña de administrador.**

**Si la contraseña de administrador coincide, se solicita confirmación para eliminar al usuario.**

**Si la confirmación es 's', se ejecuta una consulta SQL para eliminar al usuario de la tabla 'usuarios',**

**se confirman los cambios en la base de datos y se imprime un mensaje de confirmación.**

**Si la confirmación es 'n' o el usuario cancela la operación, se imprime un mensaje indicando que la operación fue cancelada.**

**Si la contraseña de administrador no coincide, se imprime un mensaje indicando que la contraseña no es correcta.**

**"""**

limpiar\_pantalla() **# Llamar a la función para limpiar la pantalla**

user = input(bcolors.WARNING + "Ingresa el nombre de usuario que deseas eliminar del sistema: " + bcolors.RESET) **# Solicitar el nombre de usuario a eliminar**

verify = getpass(bcolors.WARNING + "Ingresa tu contraseña administrador: " + bcolors.RESET)  **# Solicitar la contraseña de administrador**

if verify == password:  **# Si la contraseña coincide con la contraseña de administrador**

print(bcolors.OK + "\nLas contraseñas coinciden\n" + bcolors.RESET)  **# Imprimir mensaje de confirmación**

time.sleep(0.5)

confirmacion = input(bcolors.FAIL + "¿Estás seguro de que quieres eliminar al usuario? (s/n): " + bcolors.RESET)  **# Solicitar confirmación para eliminar al usuario**

if confirmacion.lower() == 's':  **# Si la confirmación es 's'**

time.sleep(1)

print(bcolors.WARNING + "Eliminando usuario..." + bcolors.RESET)

sql = "DELETE FROM usuarios WHERE nombre\_usuario = %s;"  **# Consulta SQL para eliminar al usuario de la tabla 'usuarios'**

cursor.execute(sql, (user,))  **# Ejecutar la consulta SQL con el valor del nombre de usuario proporcionado**

db.commit()  **# Confirmar los cambios en la base de datos**

time.sleep(2)

print(bcolors.OK + "Usuario eliminado exitosamente" + bcolors.RESET) **# Imprimir mensaje de confirmación**

**# Registrar en el historial**

registrar\_historial(f"Eliminación de usuario", user) **# Llamar a la función para registrar la acción en el historial**

else:

print("Operación cancelada.")  **# Imprimir mensaje si la operación es cancelada**

time.sleep(1)

else:

print("La contraseña no es correcta.") **# Imprimir mensaje si la contraseña no coincide**

print("Cancelando operación...")

time.sleep(1)

import mysql.connector

from datetime import datetime

**# Función para mostrar el historial de acciones realizadas por el administrador**

**def mostrar\_historial\_admin():**

**"""**

**Muestra el historial de acciones realizadas por el administrador.**

**Consulta la base de datos para obtener el historial ordenado por fecha y hora,**

**e imprime la información en la pantalla.**

**"""**

sql = "SELECT \* FROM historial ORDER BY fecha\_hora ASC"  **# Consulta SQL para obtener todo el historial ordenado por fecha y hora**

cursor.execute(sql) **# Ejecutar la consulta SQL**

result = cursor.fetchall()  **# Obtener todos los resultados de la consulta**

if result: **# Si hay resultados**

limpiar\_pantalla() **# Llamar a la función para limpiar la pantalla**

print("---------------------------------------")

print("| Historial de Usuarios |") **# Imprimir encabezado del historial de usuarios**

print("---------------------------------------")

for row in result:  **# Iterar sobre cada fila de resultados**

print("ID de referencia:", row[0]) **# Imprimir el ID de referencia de la acción**

print(bcolors.OK + "Acción:", row[1] + bcolors.RESET) **# Imprimir la acción resaltada en color verde**

print("Fecha y hora:", row[2])  **# Imprimir la fecha y hora de la acción**

print(bcolors.OK + "Usuario:", row[3] + bcolors.RESET)  **# Imprimir el nombre de usuario resaltado en color verde**

print("---------------------------------------")

else:

print("No hay registros en el historial de usuarios.")  **# Imprimir mensaje si no hay registros en el historial de usuarios**

**# Función para mostrar el menú del administrador**

**def menu\_admin():**

**"""**

**Muestra el menú de opciones para el administrador.**

**El administrador puede elegir entre diferentes opciones, como mostrar el historial,**

**borrar el historial, eliminar un usuario o salir del programa.**

**"""**

while True:

print("""----------------------------------

| MENÚ ADMINISTRADOR |

|-------------------------------------|

| 1.- Mostrar Historial |

| 2.- Borrar Historial |

| 3.- Eliminar usuario |

| 4.- Salir |

---------------------------------------""")

seleccion()  **# Solicitar al usuario que seleccione una opción del menú**

if opc\_selected == 1:

mostrar\_historial\_admin() **# Llamar a la función para mostrar el historial de usuarios como administrador**

input("Presiona Enter para continuar...")

limpiar\_pantalla()

elif opc\_selected == 2:

borrar\_historial()  **# Llamar a la función para borrar el historial**

input("Presiona Enter para continuar...")

limpiar\_pantalla()

elif opc\_selected == 3:

eliminar\_usuario()  **# Llamar a la función para eliminar un usuario**

input("Presiona Enter para continuar...")

limpiar\_pantalla()

elif opc\_selected == 4:

break

else:

opcion\_incorrecta() **# Llamar a la función si se selecciona una opción incorrecta**

**# Función para mostrar el menú principal**

**def menu\_home():**

**"""**

**Muestra el menú principal para el usuario.**

**Dependiendo del rol del usuario, se muestran diferentes opciones,**

**como cambiar la contraseña, acceder al menú de administrador o ver el historial.**

**"""**

global sesion

if sesion:

while True:

print("------------------------------")

print("| ¿Qué deseas hacer? |")

print("|----------------------------|")

print("| 5.- Cambiar contraseña |")

if role == 'admin':

print("| 6.- Menu Administrador |")

else:

print("| 6.- Historial |")

print("| 7.- Salir |")

print("------------------------------")

seleccion()  **# Solicitar al usuario que seleccione una opción del menú**

if opc\_selected == 5:

change\_password() **# Llamar a la función para cambiar la contraseña**

input("Presiona Enter para continuar...")

limpiar\_pantalla()

elif opc\_selected == 6:

if role == 'admin':

limpiar\_pantalla()

menu\_admin() **# Llamar al menú del administrador**

input("Presiona Enter para continuar...")

limpiar\_pantalla()

else:

limpiar\_pantalla()

mostrar\_historial() **# Llamar a la función para mostrar el historial del usuario**

input("Presiona Enter para continuar...")

limpiar\_pantalla()

elif opc\_selected == 7:

sesion = False

break

else:

opcion\_incorrecta()  **# Llamar a la función si se selecciona una opción incorrecta**

**# Función para registrar una acción en el historia**l

**def registrar\_historial(accion, username):**

**"""**

**Registra una acción en el historial de acciones.**

**Toma la acción realizada y el nombre de usuario como parámetros,**

**obtiene la marca de tiempo actual y guarda la información en la base de datos.**

**"""**

timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  **# Obtener la marca de tiempo actual**

sql = "INSERT INTO historial (accion, fecha\_hora, usuario) VALUES (%s, %s, %s)"  **# Consulta SQL para registrar la acción en el historial**

values = (accion, timestamp, username)  **# Valores a insertar en la consulta SQL**

cursor.execute(sql, values) **# Ejecutar la consulta SQL con los valores proporcionados**

db.commit() **# Confirmar los cambios en la base de datos**

try:

inicializar\_database()  **# Llamar a la función para inicializar la base de datos**

limpiar\_pantalla()  **# Llamar a la función para limpiar la pantalla**

print("Sistema de Registro de Usuarios")

print("---------------------------------")

while True:

print("---------------------------------------")

print("| MENÚ INICIO |")

print("|--------------------------------------|")

print("| 1.- Iniciar sesión |")

print("| 2.- Registrar usuario |")

print("| 3.- Salir |")

print("----------------------------------------")

seleccion()  **# Solicitar al usuario que seleccione una opción del menú**

if opc\_selected == 1:

login() **# Llamar a la función para iniciar sesión**

if sesion:

menu\_home()  **# Llamar al menú principal si la sesión se ha iniciado correctamente**

break

else:

input("Presiona Enter para continuar...")

limpiar\_pantalla()

elif opc\_selected == 2:

register()  **# Llamar a la función para registrar un usuario**

input("Presiona Enter para continuar...")

limpiar\_pantalla()

elif opc\_selected == 3:

break

else:

opcion\_incorrecta() **# Llamar a la función si se selecciona una opción incorrecta**

cursor.close()  **# Cerrar el cursor**

db.close()  **# Cerrar la conexión a la base de datos**

print("¡Hasta luego!")

except Exception as e:

print("Se produjo un error:", e)

cursor.close()  **# Cerrar el cursor**

db.close() **# Cerrar la conexión a la base de datos**