

**Universidad Tecmilenio**

**Proyecto Final**

**Alumnos:**

Axcel Francisco Salazar Alejo 7003106

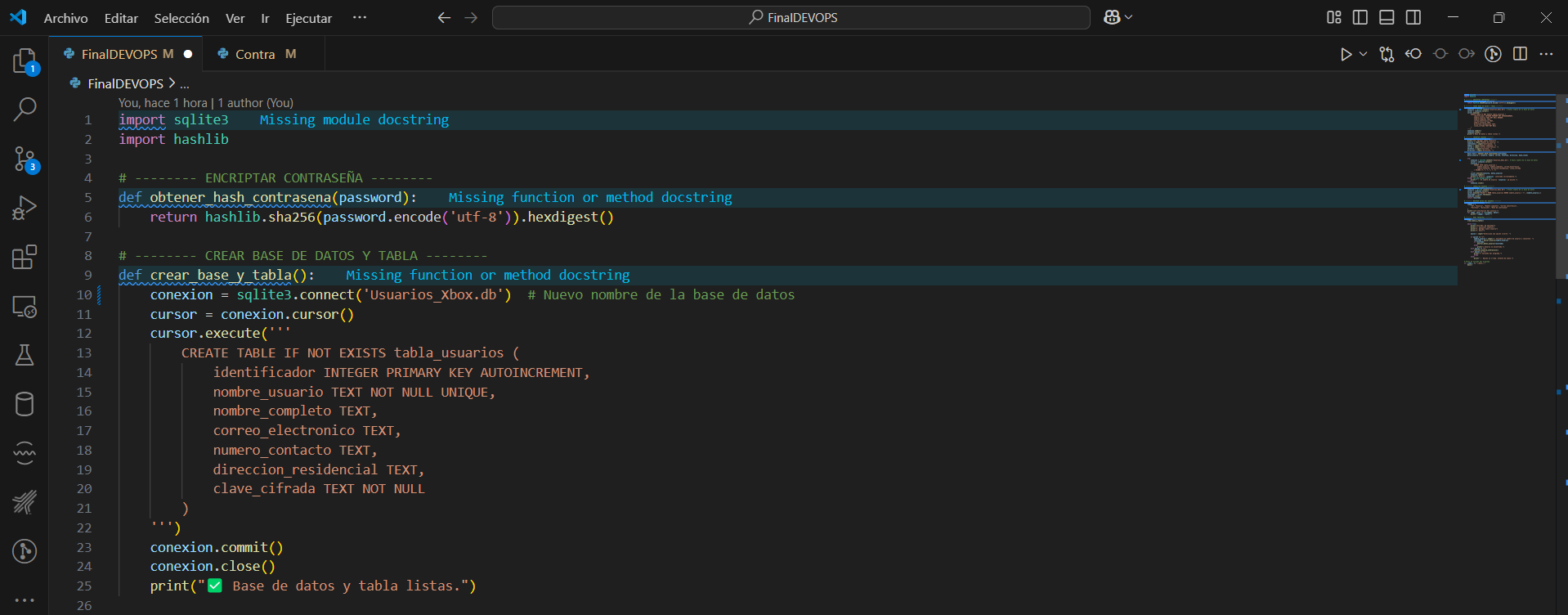
**Profesor:**

Alejandro Muraira Vargas

**Materia:** Fundamentos de DevOps

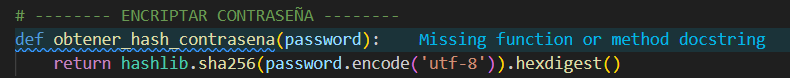
**CREACIÓN DE LA BASE DE DATOS Y TABLA**

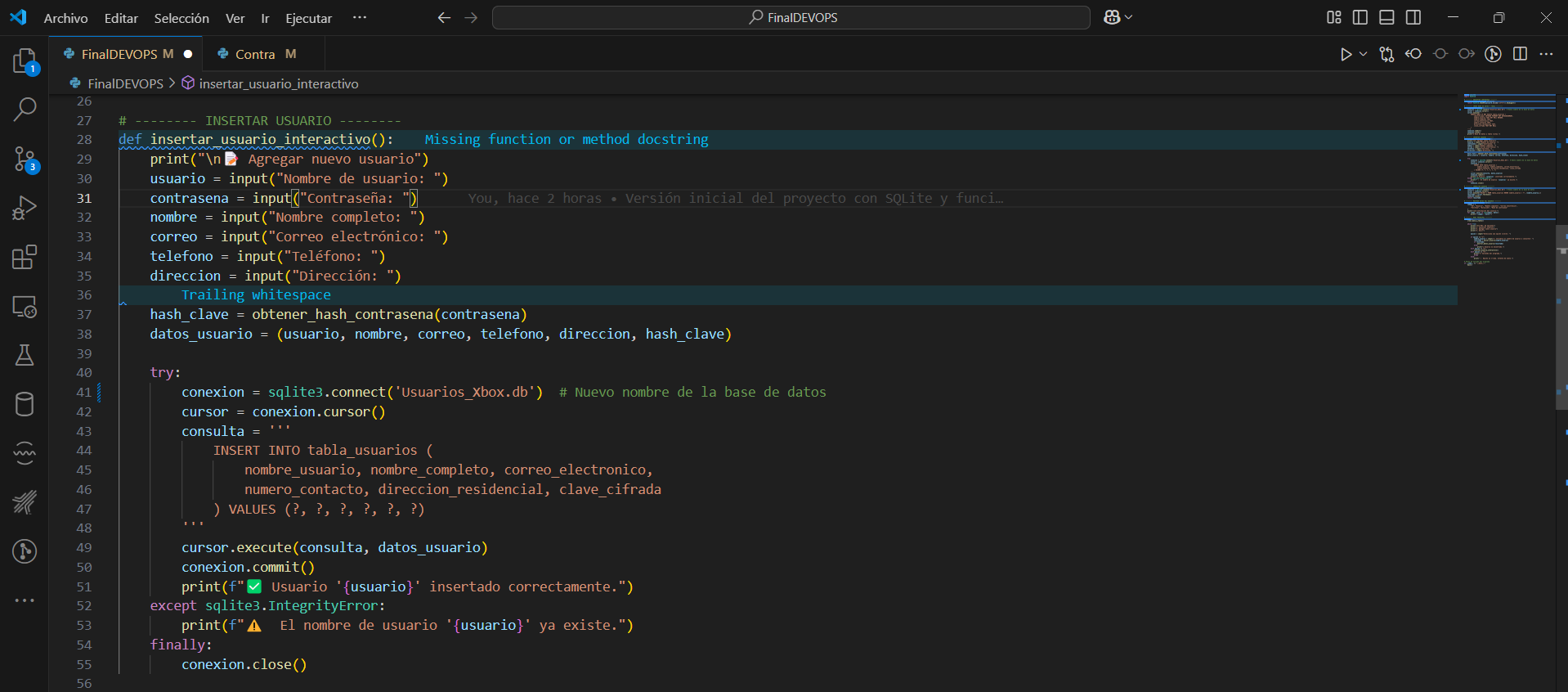
En esta etapa se desarrolló un script en Python que crea una base de datos utilizando SQLite y define una tabla para almacenar información de usuarios. La tabla incluye campos como identificador, nombre de usuario, nombre completo, correo electrónico, número de contacto, dirección residencial y una clave cifrada. El script verifica si la tabla ya existe antes de crearla para evitar errores. Al ejecutar el programa, se muestra un mensaje confirmando la creación de la base de datos **Usuarios\_Xbox.db** y su tabla.



**CIFRADO DE CONTRASEÑAS E INSERCIÓN DE DATOS**

Se implementó una función de cifrado con el algoritmo SHA-256 para proteger las contraseñas de los usuarios antes de almacenarlas. Esto permite que las contraseñas no se guarden en texto plano, incrementando así la seguridad del sistema. A continuación, se insertaron los datos de un usuario, incluyendo su nombre completo, correo, número telefónico, dirección y contraseña cifrada. Se validó si el nombre de usuario ya existía, y se mostró un mensaje apropiado si había duplicados.



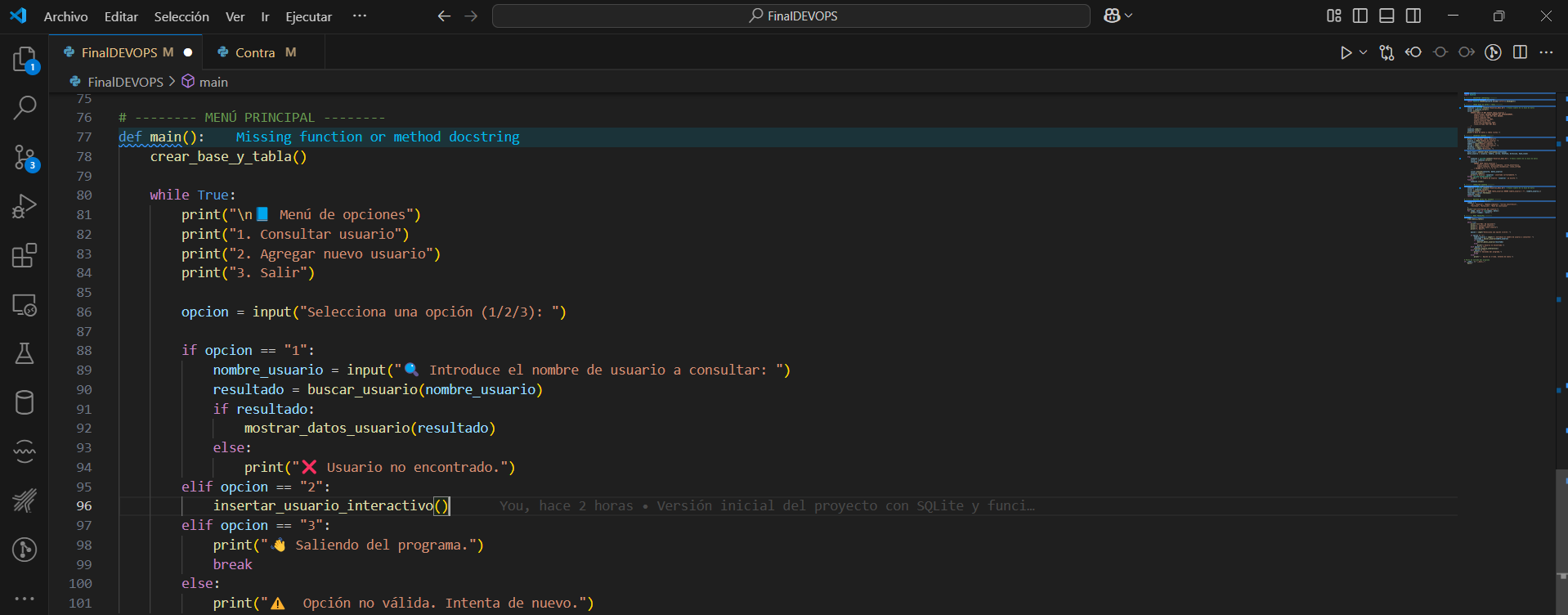


**CONSULTA Y VISUALIZACIÓN DE USUARIOS**

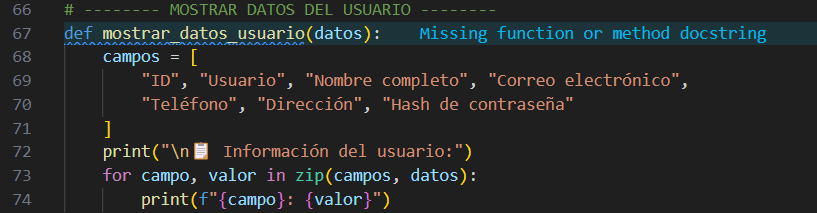
Se agregó un menú interactivo que permite al usuario:

1. Consultar los datos de un usuario ingresando su nombre de usuario.
2. Insertar nuevos usuarios al sistema con su información personal y contraseña cifrada.
3. Salir del programa.

Si se encuentra el usuario consultado, el sistema muestra todos los datos almacenados. También se mejoró el script anterior que mostraba los hashes de las contraseñas para que ahora muestre los nombres de usuario registrados en la base de datos **Usuarios\_Xbox.db**, reforzando el enfoque en la administración de usuarios.

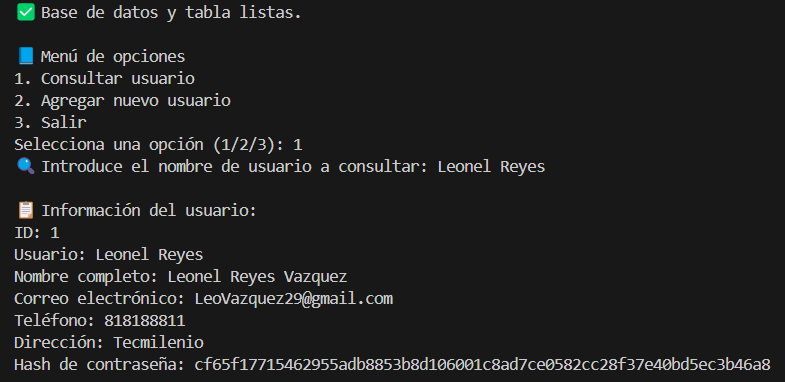


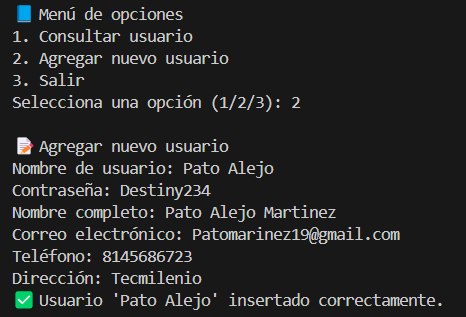
**Código adicional para mostrar todos los usuarios registrados**

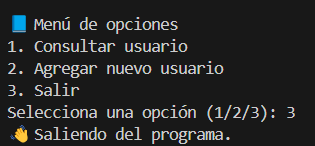


**Resultados**

* **Creación de la base de datos y tabla:** Al iniciar el programa, se genera correctamente la base de datos Usuarios\_Xbox.db y la tabla tabla\_usuarios, confirmándose con un mensaje en pantalla.
* **Registro e inserción de usuarios:** El sistema permite agregar nuevos usuarios solicitando sus datos personales y contraseña. Se valida si el nombre de usuario ya existe, mostrando un mensaje de éxito o advertencia según corresponda.
* **Consulta y visualización de usuarios:** Desde el menú interactivo, se puede consultar información detallada de un usuario existente, incluyendo su nombre, correo, teléfono, dirección y la contraseña cifrada (hash).

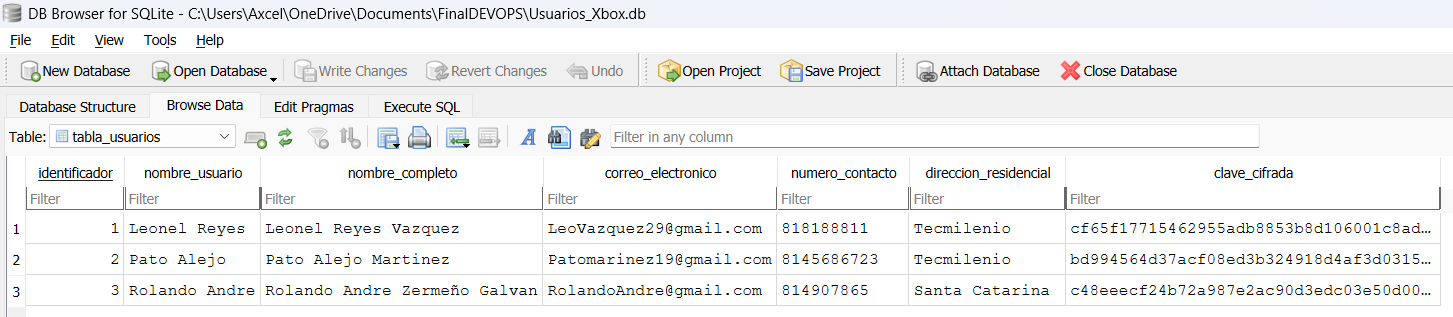


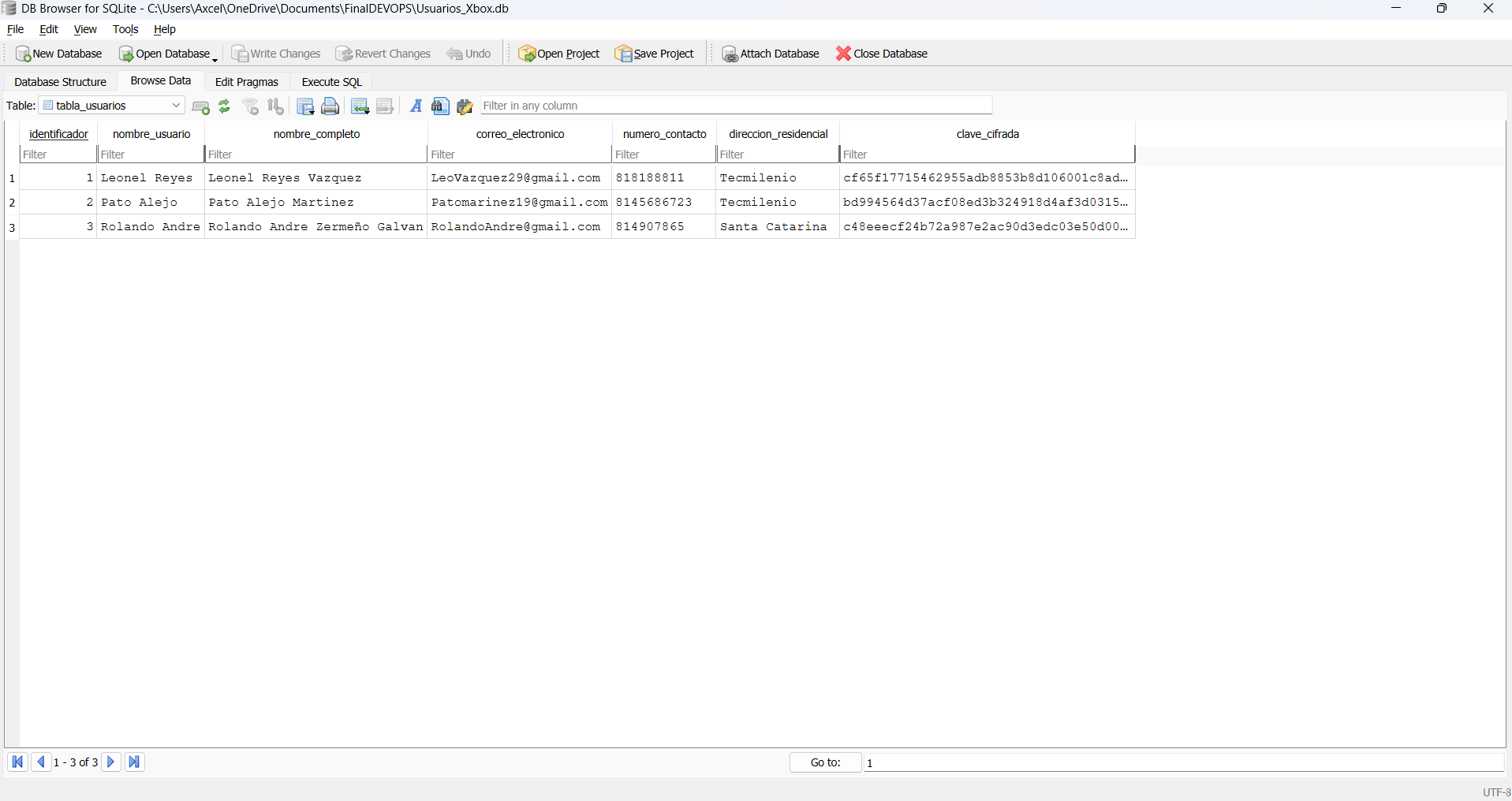




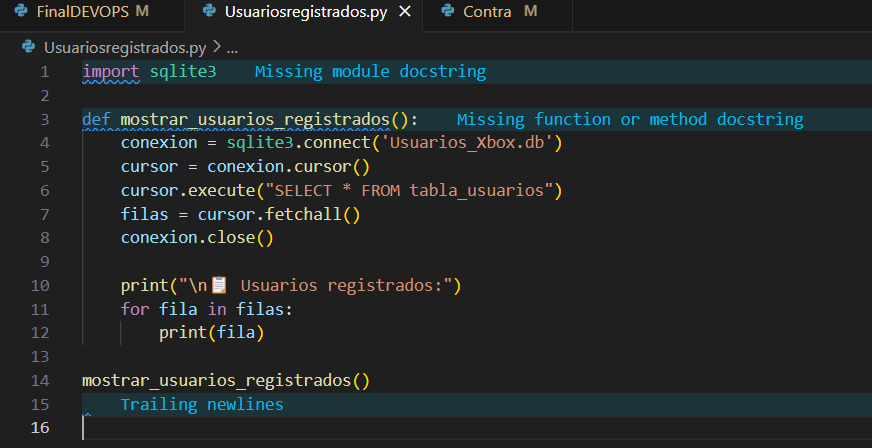
**Base de Datos**

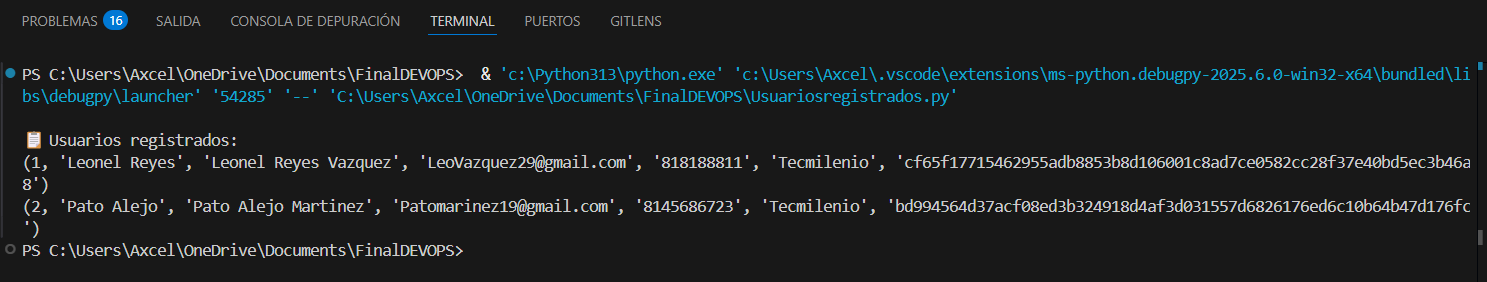
Aquí lo que muestro es la Base de Datos y como aparecen los registros de los usuarios que ya registramos y los usuarios que se van agregando después.





Aquí otra manera de checar la base de datos atreves de un código como se muestra en la captura de abajo.

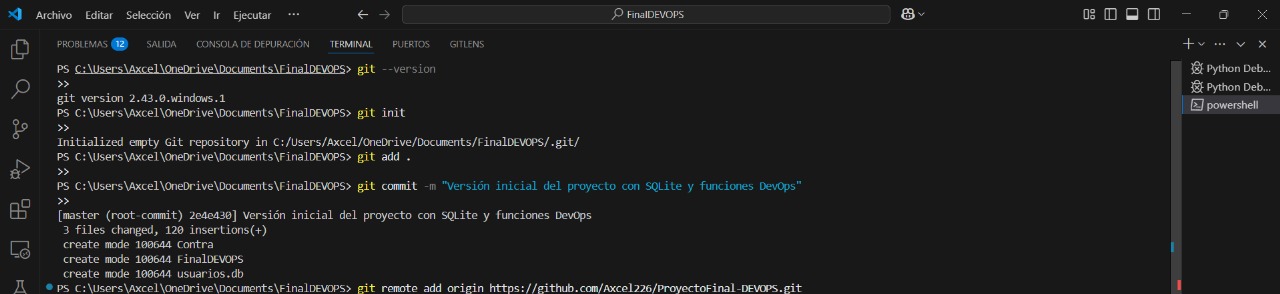


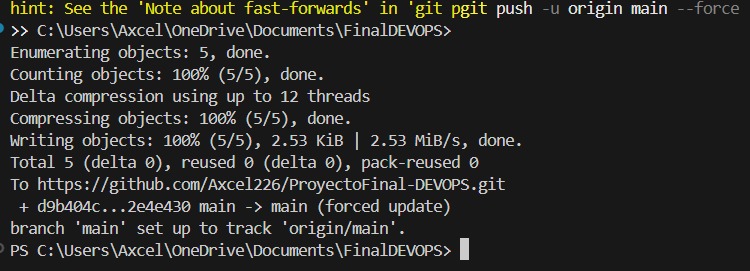


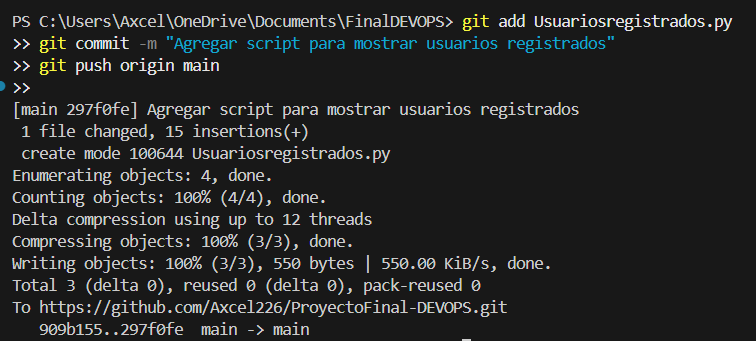
**INTEGRACIÓN CON GIT Y GITHUB**

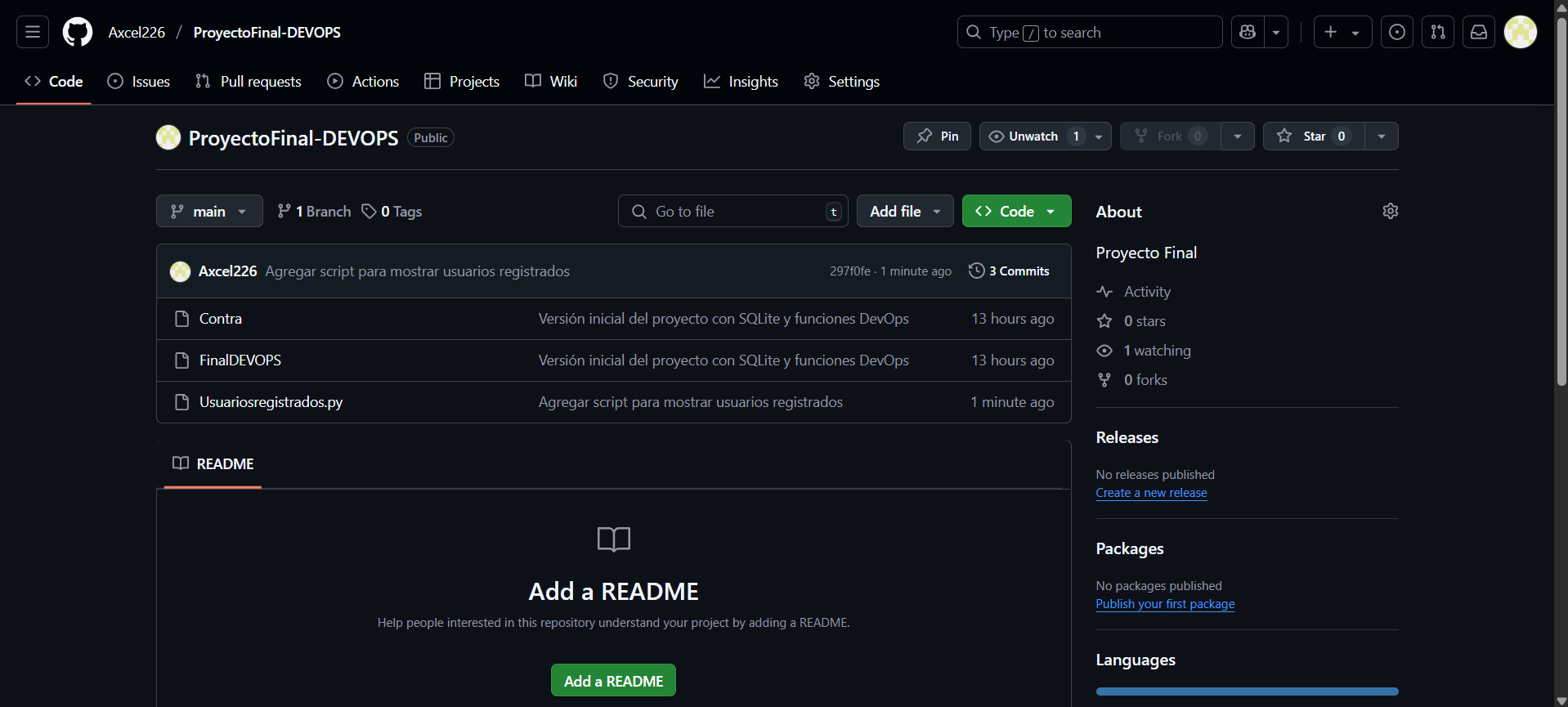
Como parte del enfoque DevOps del proyecto, se utilizó Git para el control de versiones del código y GitHub como repositorio remoto. Aquí detallo los pasos que hice:

1. **Inicialización del repositorio local:**  
   Primero, se inicializó un nuevo repositorio Git en el directorio local del proyecto para comenzar a hacer un seguimiento de los cambios del código.
2. **Vinculación con el repositorio remoto:**  
   A continuación, se vinculó el repositorio local con un repositorio remoto en GitHub, de manera que el código pudiera ser gestionado de manera centralizada y accesible desde diferentes ubicaciones.
3. **Añadir archivos y hacer commit:**  
   Se añadieron todos los archivos del proyecto al repositorio local y se realizó un primer commit con un mensaje que describía la versión inicial del proyecto, incluyendo la implementación de SQLite y las funciones relacionadas con DevOps. Entre los archivos añadidos se encuentra el script Usuariosregistrados.py, encargado de mostrar todos los usuarios almacenados en la base de datos.
4. **Push al repositorio remoto:**  
   Tras realizar el commit, se intentó realizar un "push" para subir el código al repositorio remoto en GitHub. Sin embargo, Git devolvió un error debido a que el repositorio remoto ya contenía archivos previos.
5. **Solución del error usando fuerza:**  
   Para resolver el conflicto y sobrescribir los archivos existentes en el repositorio remoto, se utilizó un "push" con la opción de fuerza, permitiendo que la versión local del proyecto reemplazara la versión remota. Este método se usó de manera controlada para evitar pérdidas de datos.









Link del Github

<https://github.com/Axcel226/ProyectoFinal-DEVOPS>

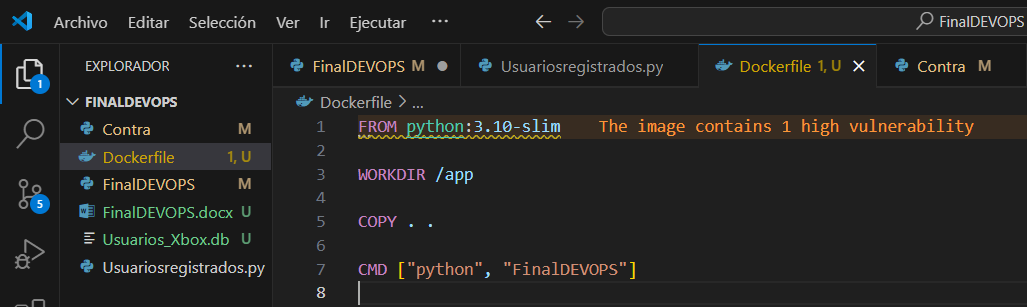
**USO DE DOCKER**

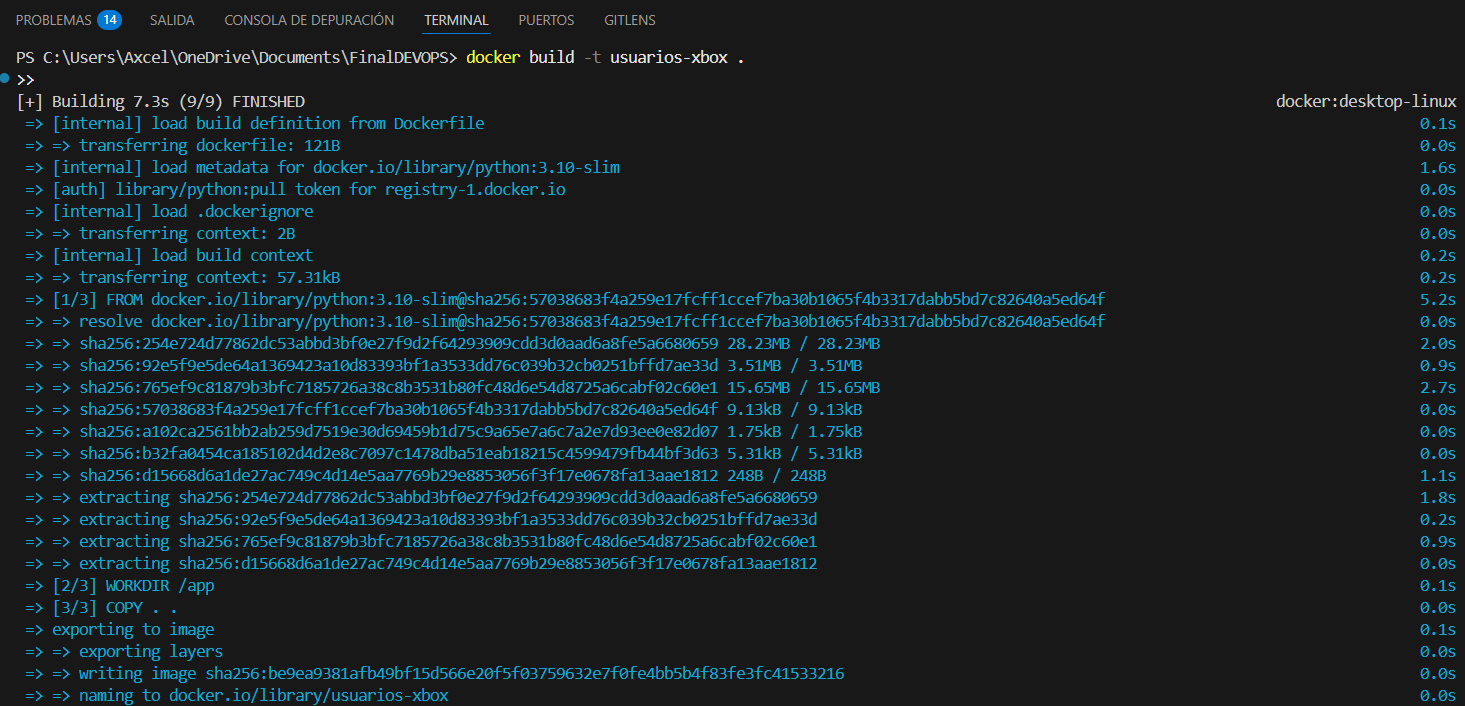
Además de usar Git y GitHub, también se utilizó Docker para facilitar la ejecución del proyecto en cualquier computadora. Esto permite que el programa funcione igual sin importar el sistema operativo, ya que todo corre dentro de un contenedor.

Esto es lo que hizo el Docker en una pequeña explicaion:

* Se creó un archivo Dockerfile con los pasos para instalar Python y copiar el código.
* Se construyó la imagen con docker build -t usuarios-xbox ..
* Se ejecutó con docker run -it usuarios-xbox para probar el sistema.

Esto ayudó a aplicar buenas prácticas DevOps al tener el código y su entorno listos para funcionar en cualquier lugar.





**CONCLUSIÓN**

Este proyecto permitió integrar conocimientos de bases de datos, seguridad y herramientas DevOps. Se diseñó e implementó un sistema de administración de usuarios utilizando Python y SQLite, cuidando la seguridad mediante cifrado de contraseñas. Además, se aplicaron buenas prácticas de desarrollo mediante Git y GitHub, lo que facilitó el control de versiones y la colaboración.

Los retos incluyeron la correcta gestión del repositorio remoto y el manejo de errores comunes de Git. Estas experiencias reforzaron la importancia de las herramientas DevOps en el desarrollo moderno de software, y brindaron una comprensión práctica sobre el manejo seguro de datos y la automatización del flujo de trabajo de desarrollo.

**Codigo que se uso:**

import sqlite3

import hashlib

# -------- ENCRIPTAR CONTRASEÑA --------

def obtener\_hash\_contrasena(password):

    return hashlib.sha256(password.encode('utf-8')).hexdigest()

# -------- CREAR BASE DE DATOS Y TABLA --------

def crear\_base\_y\_tabla():

    conexion = sqlite3.connect('Usuarios\_Xbox.db')  # Nuevo nombre de la base de datos

    cursor = conexion.cursor()

    cursor.execute('''

        CREATE TABLE IF NOT EXISTS tabla\_usuarios (

            identificador INTEGER PRIMARY KEY AUTOINCREMENT,

            nombre\_usuario TEXT NOT NULL UNIQUE,

            nombre\_completo TEXT,

            correo\_electronico TEXT,

            numero\_contacto TEXT,

            direccion\_residencial TEXT,

            clave\_cifrada TEXT NOT NULL

        )

    ''')

    conexion.commit()

    conexion.close()

    print("✅ Base de datos y tabla listas.")

# -------- INSERTAR USUARIO --------

def insertar\_usuario\_interactivo():

    print("\n📝 Agregar nuevo usuario")

    usuario = input("Nombre de usuario: ")

    contrasena = input("Contraseña: ")

    nombre = input("Nombre completo: ")

    correo = input("Correo electrónico: ")

    telefono = input("Teléfono: ")

    direccion = input("Dirección: ")

    hash\_clave = obtener\_hash\_contrasena(contrasena)

    datos\_usuario = (usuario, nombre, correo, telefono, direccion, hash\_clave)

    try:

        conexion = sqlite3.connect('Usuarios\_Xbox.db')  # Nuevo nombre de la base de datos

        cursor = conexion.cursor()

        consulta = '''

            INSERT INTO tabla\_usuarios (

                nombre\_usuario, nombre\_completo, correo\_electronico,

                numero\_contacto, direccion\_residencial, clave\_cifrada

            ) VALUES (?, ?, ?, ?, ?, ?)

        '''

        cursor.execute(consulta, datos\_usuario)

        conexion.commit()

        print(f"✅ Usuario '{usuario}' insertado correctamente.")

    except sqlite3.IntegrityError:

        print(f"⚠️  El nombre de usuario '{usuario}' ya existe.")

    finally:

        conexion.close()

# -------- CONSULTAR USUARIO --------

def buscar\_usuario(nombre\_usuario):

    conexion = sqlite3.connect('Usuarios\_Xbox.db')  # Nuevo nombre de la base de datos

    cursor = conexion.cursor()

    cursor.execute("SELECT \* FROM tabla\_usuarios WHERE nombre\_usuario = ?", (nombre\_usuario,))

    resultado = cursor.fetchone()

    conexion.close()

    return resultado

# -------- MOSTRAR DATOS DEL USUARIO --------

def mostrar\_datos\_usuario(datos):

    campos = [

        "ID", "Usuario", "Nombre completo", "Correo electrónico",

        "Teléfono", "Dirección", "Hash de contraseña"

    ]

    print("\n📋 Información del usuario:")

    for campo, valor in zip(campos, datos):

        print(f"{campo}: {valor}")

# -------- MENÚ PRINCIPAL --------

def main():

    crear\_base\_y\_tabla()

    while True:

        print("\n📘 Menú de opciones")

        print("1. Consultar usuario")

        print("2. Agregar nuevo usuario")

        print("3. Salir")

        opcion = input("Selecciona una opción (1/2/3): ")

        if opcion == "1":

            nombre\_usuario = input("🔍 Introduce el nombre de usuario a consultar: ")

            resultado = buscar\_usuario(nombre\_usuario)

            if resultado:

                mostrar\_datos\_usuario(resultado)

            else:

                print("❌ Usuario no encontrado.")

        elif opcion == "2":

            insertar\_usuario\_interactivo()

        elif opcion == "3":

            print("👋 Saliendo del programa.")

            break

        else:

            print("⚠️  Opción no válida. Intenta de nuevo.")

# Punto de entrada del programa

if \_\_name\_\_ == "\_\_main\_\_":

    main()