**Desarrollo de Análisis:**

En este documento se explican los inconvenientes que se ha tenido, junto a sus soluciones, las explicaciones de los tipos de datos y el diagrama de flujo del ***CRM***.

1. **Solución:**

En mi caso, he presentado algunos inconvenientes en la generación del ***CRM***. Estas dificultades han sido:

* Conectar la *BBDD* a partir de *MySQL*.
* Inserción de nuevos usuarios

En el primer caso, la causa fue que se intentó inicialmente establecer la conexión entre la base de datos y el script clientes.py utilizando MySQL. Sin embargo, debido a dificultades técnicas con dicha conexión, se optó por migrar a SQLite, lo cual permitió simplificar la gestión de la base de datos y evitar los problemas presentados.

Se decidió debido a que SQLite es mejor para conectar con Python en proyectos pequeños o en desarrollo ya que no requiere instalar ni configurar un servidor debido a que funciona con un solo archivo de base de datos. Además, Python incluye soporte nativo para SQLite, facilitando su uso sin necesidad de controladores adicionales. Por otro lado, MySQL es más complejo y está pensado para aplicaciones grandes con muchos usuarios y datos, lo que puede ser innecesario para proyectos simples.

En el segundo caso, se presentó uno de los principales desafíos debido a la falta de generación automática del campo id\_cliente al insertar nuevos usuarios. Para solventar esta situación, se modificó la tabla usuarios incorporando la columna id\_cliente como INTEGER PRIMARY KEY AUTOINCREMENT, garantizando así la creación automática y única del identificador para cada registro, lo que permitió una gestión eficiente y coherente de los datos almacenados.

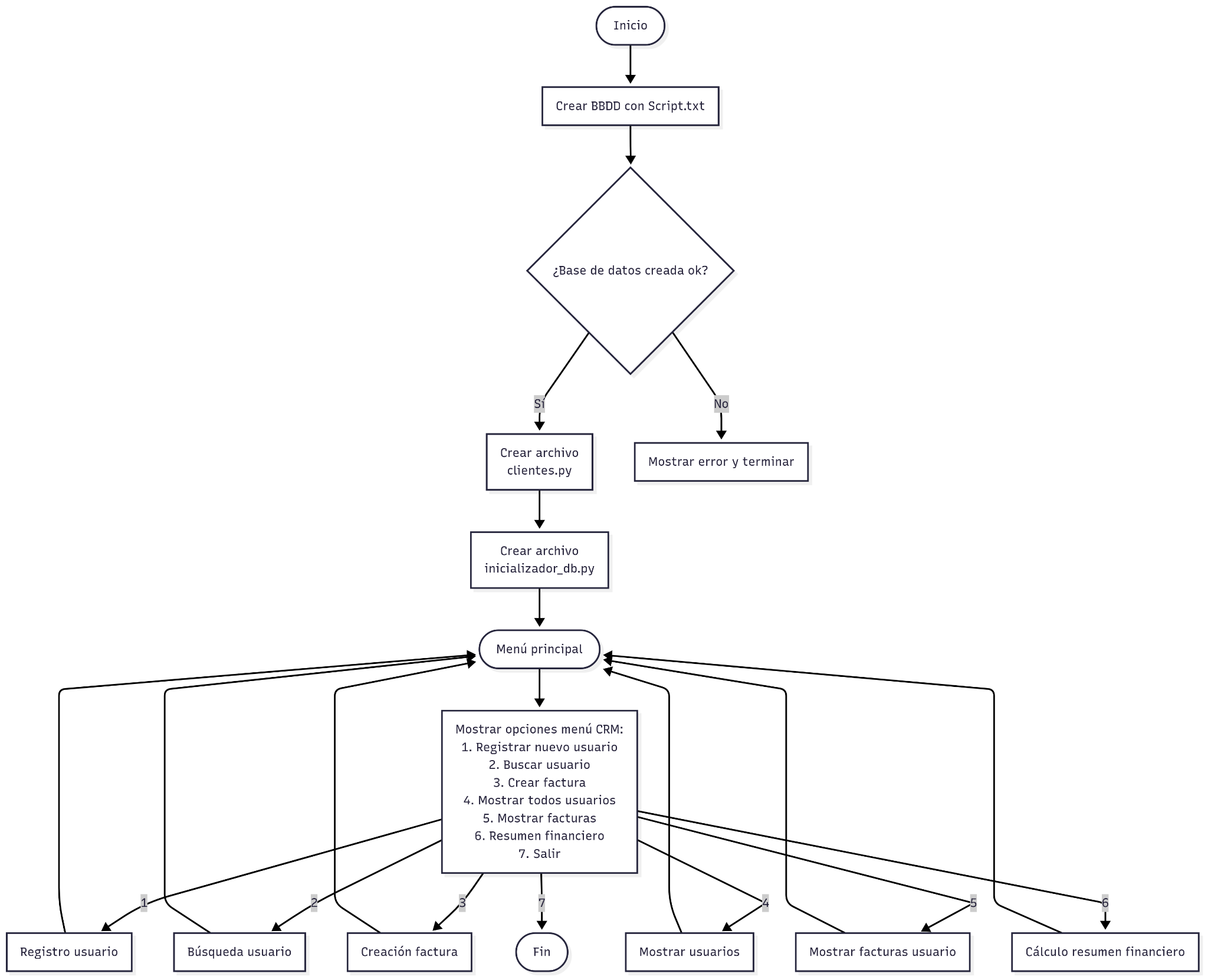
1. **Justificación de tipos de datos y estructuras utilizadas:**

Para el diseño de la base de datos se seleccionaron tipos de datos específicos para cada campo, con el fin de optimizar el almacenamiento, la integridad y la eficiencia en las consultas. En la tabla usuarios, el campo id\_cliente se definió como INTEGER PRIMARY KEY AUTOINCREMENT para garantizar un identificador único y autogenerado que facilite la relación con otras tablas. Los campos nombre y apellidos se almacenan como VARCHAR(20) para contener la mayoría de los nombres comunes, asegurando un uso eficiente del espacio. El campo email se definió como VARCHAR(100) para cubrir direcciones de correo electrónico completas y con un índice único que evita duplicados. Para telefono, se optó por VARCHAR(20) debido a que puede incluir códigos internacionales, espacios o símbolos, mientras que direccion utiliza TEXT para permitir direcciones de longitud variable y potencialmente extensa. La fecha de registro se guarda como DATETIME con un valor por defecto de la fecha y hora actual, permitiendo un seguimiento temporal preciso.

En la tabla facturas, el numero\_factura también se definió como INTEGER PRIMARY KEY AUTOINCREMENT para la generación automática y única de cada factura. El campo id\_cliente es de tipo INT para mantener la relación con la tabla usuarios. La fecha de emisión se registró como DATETIME para documentar con exactitud el momento en que se generó la factura. La descripción es un VARCHAR(100), suficiente para un texto conciso que explique el motivo de la factura. El campo monto se almacenó como DECIMAL(10, 2) para permitir valores monetarios con dos decimales, asegurando precisión en cálculos financieros y restricciones que evitan valores negativos. Finalmente, el estado de la factura se definió como TEXT con una restricción CHECK que limita los valores a ‘Pendiente’, ‘Pagada’ o ‘Cancelada’, asegurando la integridad de los datos.

En cuanto a las estructuras de datos en el código, se utilizaron listas y diccionarios para manejar colecciones y asociaciones de datos de forma dinámica y eficiente. Las listas son adecuadas para almacenar conjuntos ordenados de registros, mientras que los diccionarios facilitan el acceso rápido a datos asociados mediante claves (por ejemplo, datos de un usuario con su id\_cliente). Esta combinación permite una gestión clara y flexible de la información durante la ejecución del programa, facilitando operaciones como búsqueda, actualización y eliminación de registros.

1. **Diagrama de flujo:**

****