**1. Párrafo de solución**

El proyecto CRM Evolve fue desarrollado como una aplicación de consola en Python utilizando una base de datos SQLite para la gestión de usuarios, facturas y resúmenes financieros. La estrategia principal fue organizar el código en módulos separados dentro de la carpeta src/, manteniendo una estructura clara y modular. Se creó un script principal con menú interactivo y otros scripts auxiliares para insertar datos de ejemplo. Se implementaron tests unitarios en la carpeta test/ usando pytest.

Uno de los principales desafíos fue asegurar la integridad de los datos y manejar adecuadamente errores comunes, como claves duplicadas o campos faltantes. Además, fue importante garantizar que la base de datos se inicializara correctamente antes de ejecutar cualquier script que dependiera de ella.

Además, se creó un repositorio en GitHub (<https://github.com/Samuel-Cantarero/crm-evolve>) con tres ramas: main, develop y feature. La rama main se reservó para versiones estables, la rama develop para la integración de funcionalidades y pruebas, y las ramas feature para el desarrollo de nuevas funcionalidades específicas. Esto permitió aplicar buenas prácticas de control de versiones, tal como se aprendió en los módulos anteriores. También se añadieron carpetas organizadas según las buenas prácticas: la carpeta src/ contiene el código fuente, database/ aloja la base de datos y el esquema SQL, y test/ está dedicada a pruebas automatizadas con pytest. Además, se incluyó un archivo README.md con instrucciones de uso y un requirements.txt con las dependencias, facilitando la instalación y ejecución del proyecto.

**2. Justificación de tipos de datos**

Se seleccionaron tipos de datos apropiados en función de la naturaleza de cada campo y con criterios de integridad y automatización:

* first\_name, last\_name, email, phone, address: tipo **TEXT**, ya que son cadenas de caracteres.
* first\_name, last\_name y email se definen con NOT NULL porque son datos esenciales para identificar a un usuario.
* email incluye la restricción UNIQUE para asegurar que ningún correo se registre más de una vez, evitando duplicidades.
* registration\_date y issue\_date usan tipo **TEXT** con formato ISO 8601, junto a DEFAULT (datetime('now')), lo que permite que la fecha se genere automáticamente al insertar el registro.
* amount: tipo **REAL**, para representar importes decimales de forma precisa.
* status: tipo **TEXT**, con valores definidos por la lógica del sistema ('Pending', 'Paid', 'Cancelled').
* id en ambas tablas (users, invoices) es INTEGER PRIMARY KEY AUTOINCREMENT, garantizando una clave única que se incrementa automáticamente.
* user\_id en invoices es FOREIGN KEY REFERENCES users(id), asegurando la integridad referencial: una factura siempre debe pertenecer a un usuario existente.

Estas decisiones permiten mantener la coherencia de la base de datos, evitar errores comunes y facilitar operaciones automáticas como la asignación de fechas o claves únicas.

**3. Tipos de datos utilizados por campo**

| **Campo** | **Tipo de dato (SQLite)** | **Descripción** |
| --- | --- | --- |
| first\_name | TEXT | Nombre del usuario |
| last\_name | TEXT | Apellidos del usuario |
| email | TEXT UNIQUE | Correo electrónico, único por usuario |
| phone | TEXT | Teléfono de contacto (opcional) |
| address | TEXT | Dirección del usuario (opcional) |
| registration\_date | TEXT (ISO 8601) | Fecha de registro con valor por defecto automático |
| user\_id | INTEGER | Clave foránea que enlaza la factura con el usuario |
| issue\_date | TEXT (ISO 8601) | Fecha de emisión de la factura |
| description | TEXT | Descripción del servicio o producto facturado |
| amount | REAL | Importe monetario de la factura |
| status | TEXT | Estado de la factura: 'Pending', 'Paid' o 'Cancelled' |