**ANÁLISIS DE DESARROLLO – PRÁCTICA FINAL DE TIPOLOGÍA DE DATOS**

**Sistema CRM para Gimnasio por Consola + MySQL**

**1. Solución adoptada**

El objetivo del proyecto fue desarrollar un sistema CRM profesional para la gestión de clientes y facturación de un gimnasio, utilizando Python para la lógica y MySQL para la persistencia de datos.  
Se optó por una arquitectura de **tres capas** para organizar el código de forma clara y escalable:

* **Capa de Interfaz:** Gestiona el menú y la interacción con el usuario por consola (main.py).
* **Capa de Lógica de Negocio:** Contiene las validaciones, reglas de funcionamiento y orquesta las operaciones principales (logica.py).
* **Capa de Acceso a Datos:** Encapsula todas las funciones de conexión y consulta a la base de datos MySQL, incluyendo inserción, búsqueda y eliminación de clientes y facturas (bd.py).

La temática elegida fue un **CRM para un gimnasio**, lo que facilitó dar contexto realista a los servicios y facturas (ej. “membresía mensual”, “clases de spinning”, etc.).

También se implementó una funcionalidad adicional para que cada factura tenga un **ID personalizado** con formato mes\_año\_numero (ejemplo: enero\_2025\_3), facilitando la trazabilidad y organización de las facturas.

**2. Justificación de tipos de datos y estructuras utilizadas**

|  |  |  |
| --- | --- | --- |
| **Campo / Elemento** | **Tipo de dato** | **Justificación** |
| nombre, apellidos | VARCHAR / str | Texto corto, necesario para identificar al cliente. |
| email | VARCHAR / str | Único para cada cliente, usado como clave lógica. |
| teléfono, dirección | VARCHAR / str | Datos opcionales, pero útiles para contacto. |
| fecha\_registro | DATETIME | Asignado automáticamente al crear el cliente. |
| id\_cliente, id\_factura | INT / int | Clave primaria autogenerada, facilita relaciones y búsquedas. |
| monto | DECIMAL(10,2) / float | Manejo preciso de dinero con decimales. |
| estado de factura | ENUM / str | Controlado con 3 opciones: Pagada, Pendiente, Cancelada. |
| codigo\_factura | VARCHAR / str | Identificador personalizado tipo enero\_2025\_1. |

**Estructuras en Python:**

* **Listas y tuplas:** Para recorrer y mostrar resultados de consultas (por ejemplo, fetchall()).
* **Funciones:** Para modularizar cada parte del sistema y facilitar el mantenimiento.
* **Diccionarios:** Para mapear opciones de estado de facturas y menús.
* **Validaciones:** Para asegurar la entrada correcta del usuario y evitar duplicados.

**3. Flujo principal del sistema *(resumen)***

1. Mostrar menú principal.
2. Registrar cliente (verifica que no exista por email).
3. Buscar cliente por email o nombre.
4. Mostrar todos los clientes.
5. Crear factura (requiere cliente existente y monto válido).
6. Mostrar todas las facturas de un cliente.
7. Ver resumen financiero por cliente.
8. Finalizar programa.

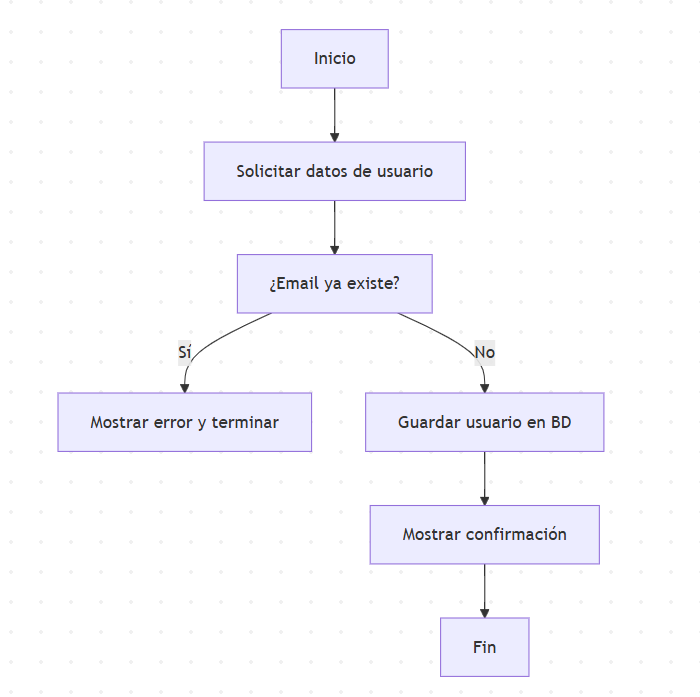
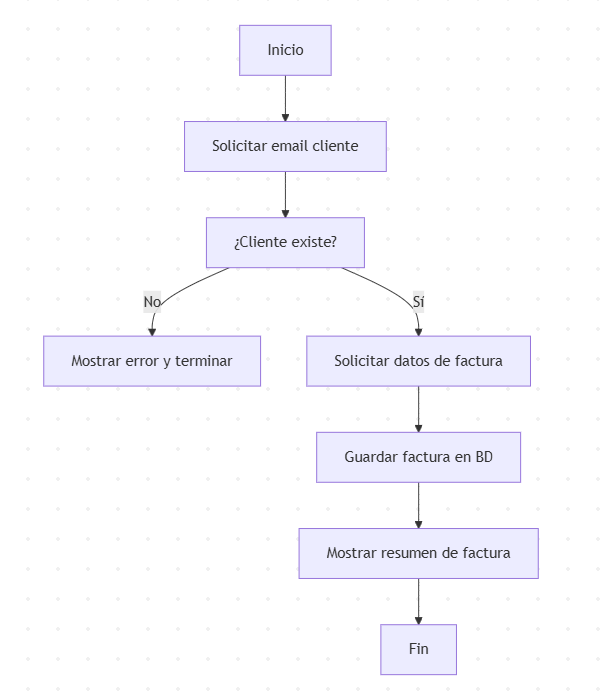
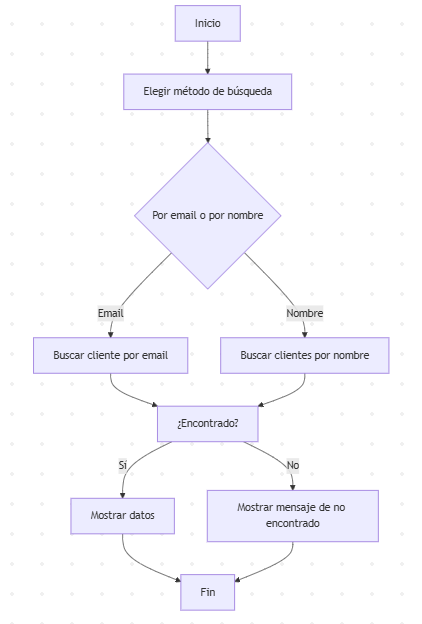
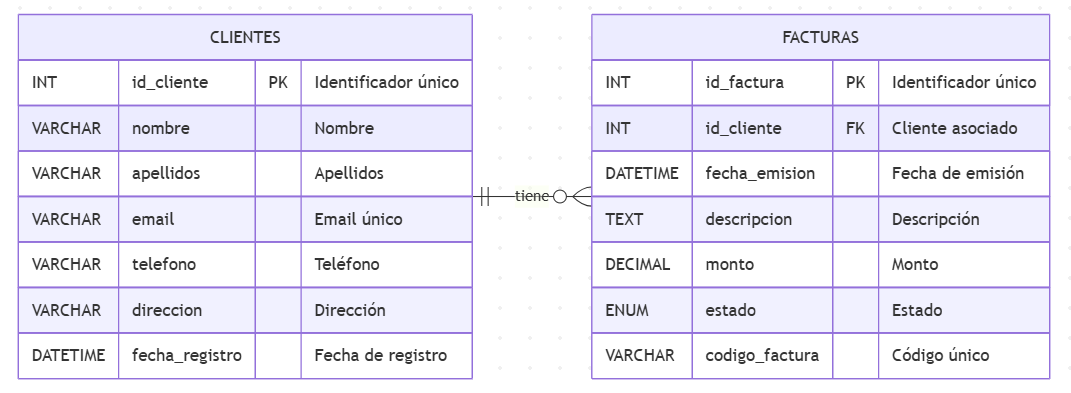
**4. Desafíos encontrados y cómo los resolvimos**

* **Conexión segura con MySQL:** Se utilizó el paquete mysql-connector-python y variables de entorno (con .env) para proteger las credenciales.
* **Evitar emails duplicados:** Se validó la existencia de un cliente antes de insertarlo, mostrando un mensaje si ya existe.
* **Eliminación en cascada:** Al eliminar un cliente, primero se eliminan sus facturas asociadas para mantener la integridad referencial.
* **Generar IDs personalizados para facturas:** Se creó una función que cuenta las facturas del mes actual y forma el código con el patrón mes\_año\_numero, usando datetime y COUNT() en SQL.
* **Separación en capas:** Aunque al principio parecía más complejo, la división en tres capas (interfaz, lógica, datos) facilitó la organización, el mantenimiento y la escalabilidad del sistema.

**5. Funcionalidades implementadas**

* Registrar cliente nuevo.
* Buscar cliente por email o nombre.
* Mostrar lista completa de clientes.
* Crear factura con código personalizado.
* Ver facturas asociadas a un cliente.
* Ver resumen financiero por cliente.
* Menú interactivo en consola (opciones 1 a 8).

**6. Diagramas de flujo (resumen visual)**

* Registro de usuario:
* Creación de factura:
* Búsqueda de cliente:
* Diagrama ER:

**6. Extras (opcional)**

* Se incluyó un script SQL (crear\_base\_datos.sql) con datos de ejemplo para probar todas las funcionalidades sin necesidad de introducirlos manualmente.
* El sistema utiliza un archivo .env para la configuración segura de la base de datos (no se sube a git).
* El código está documentado y preparado para ampliaciones futuras.

**7. Conclusión**

Este proyecto permitió aplicar conceptos de tipología de datos, validación de entradas, estructura lógica del código y conexión a una base de datos relacional.  
El uso de Python y MySQL resultó ser una combinación eficiente para desarrollar un sistema completo por consola, reforzando la importancia de la organización en capas y la seguridad en el manejo de datos sensibles.

**8. Archivos entregados**

* main.py – Menú de opciones
* logica.py – Reglas y validaciones
* bd.py – Acceso a MySQL y lógica de facturas
* crear\_base\_datos.sql – Script para base de datos y datos de ejemplo
* requirements.txt – Dependencias del proyecto
* .env – Variables de entorno (no incluido en el repositorio)
* Este documento Word (analisis\_crm\_gimnasio.docx)