**Análisis Técnico:**

**Sistema CRM en Consola**

**Hecho por**: Valentina Bailón Cano I **Fecha**: 20 de junio de 2025 I

# 1. Enfoque y Estrategia de Desarrollo

El desarrollo del sistema CRM (Customer Relationship Management) se abordó con el objetivo de construir una herramienta funcional por consola, sencilla de utilizar y alineada con los requisitos del enunciado.

La solución se estructuró mediante funciones modulares y un flujo claro a través de un menú interactivo que permite registrar, buscar y consultar usuarios, emitir facturas y generar reportes financieros.  
  
Desde el inicio, se consideró esencial que el sistema mantuviera la información entre sesiones, por lo que se implementó persistencia opcional utilizando archivos ‘.json’.  
Esto permite al usuario conservar todos los datos introducidos incluso si el programa se cierra, lo cual simula el comportamiento de una base de datos ligera en entornos reales.

# 2. Justificación de Tipos de Datos y Estructuras Utilizadas

* Para los usuarios se emplea un diccionario (‘dict’) en el que la clave es el email y el valor es otro diccionario con la información personal del usuario.  
  Esta estructura permite búsquedas instantáneas y asegura que no se dupliquen emails (que deben ser únicos).
* Para las facturas se utiliza una lista (‘list’) de diccionarios. Cada elemento representa una factura con su número, fecha, descripción, estado, monto y cliente.
* Se utilizan variables ‘contador\_usuarios’ y ‘contador\_facturas’ para generar identificadores únicos con prefijos ‘USR’ y ‘FAC’, respectivamente.
* Tipos de datos concretos:

- ‘str’: nombre, apellidos, email, dirección, estado, fecha, número de factura.

- ‘float’: para el monto total de la factura.

- ‘datetime’: para registrar fecha y hora de emisión de facturas y fecha de registro del usuario.

- ‘bool’ y ‘None’ no se utilizaron explícitamente ya que no eran necesarios para el caso.

# 3. Validaciones Implementadas

* Validación del formato del email utilizando expresiones regulares.
* Comprobación de que los campos obligatorios (nombre, apellidos, email) no estén vacíos.
* Comprobación de que el email no esté duplicado.
* Validación de que el monto de la factura sea un número positivo.
* Validación del estado de la factura (solo se permiten tres opciones válidas).
* Comprobación de que un usuario exista antes de permitir la creación de una factura asociada.

# 4. Implementación de Persistencia

Aunque opcional según el enunciado, se decidió implementar la persistencia para dotar al sistema de mayor robustez. Para ello, se utilizaron archivos ‘.json’ como medio de almacenamiento.  
Al iniciar el programa, se leen los archivos ‘usuarios.json’ y ‘facturas.json’ si existen. En caso contrario, se inicializan como estructuras vacías.

Cada vez que se registra un nuevo usuario o se crea una factura, se actualizan automáticamente estos archivos. Esta estrategia permite mantener todos los datos intactos incluso después de cerrar el programa, ofreciendo una experiencia de usuario realista y profesional.

# 5. Flujo del Programa

* Al ejecutarse, el programa carga los datos desde archivos y muestra un menú principal.
* Cada opción del menú llama a una función correspondiente:  
   1. Registrar nuevo usuario.  
   2. Buscar usuario por email o por nombre.  
   3. Crear factura (solo si el usuario existe).  
   4. Listar todos los usuarios registrados.  
   5. Mostrar facturas asociadas a un usuario.  
   6. Generar resumen financiero detallado (por usuario y general).  
   7. Salir del sistema (los datos ya están guardados automáticamente).

# 6. Revisión de Requisitos del Enunciado

✔️ Registro de usuarios con campos obligatorios y opcionales, más fecha automática.

✔️ Creación de facturas con número único, fecha automática, descripción, monto y estado.

✔️ Validaciones completas: email válido y único, campos requeridos, formatos correctos.

✔️ Búsqueda de usuarios por nombre o email.

✔️ Listado completo de usuarios y facturas asociadas a cada uno.

✔️ Resumen financiero con cálculo total, pagado y pendiente, tanto por usuario como global.

✔️ Persistencia opcional implementada con archivos ‘. json’.

✔️ Código funcional, organizado en funciones, fácil de mantener, bien comentado.

# 7. Conclusión

* Este sistema CRM cumple de forma íntegra con los objetivos del proyecto propuesto. Ofrece una solución funcional y reutilizable, con capacidad de mantener los datos en el tiempo y preparada para ser ampliada (por ejemplo, incorporando interfaces gráficas o integración con bases de datos reales).
* Se ha priorizado la claridad del código, la modularidad y el cumplimiento de buenas prácticas de desarrollo. Además, todos los requisitos del enunciado (obligatorios y opcionales) han sido satisfechos.