INFORME EXPLICATIVO

SISTEMA DE GESTIÓN DE BASE DE DATOS

Análisis Técnico y Funcional

DESCRIPCIÓN GENERAL:
• Sistema completo de gestión de bases de datos en CSV/JSON
• Interfaz de línea de comandos interactiva
• Soporte para múltiples tablas relacionadas
• Funciones CRUD completas (Crear, Leer, Actualizar, Eliminar)

CARACTERÍSTICAS PRINCIPALES:
• Gestión automática de campos ID
• Exportación a múltiples formatos
• Validación y normalización de datos
• Interfaz intuitiva con menús jerárquicos

Fecha de generación: 2025-10-28 13:46

ARQUITECTURA DEL SISTEMA

COMPONENTES PRINCIPALES:

1. GESTIÓN DE ARCHIVOS
   • Carga automática de tablas desde archivos CSV
   • Exportación a formatos CSV y JSON
   • Manejo robusto de errores de archivos
   • Creación automática de directorios

2. GESTIÓN DE DATOS
   • Detección automática de campos ID
   • Generación inteligente de nuevos IDs
   • Normalización de valores numéricos
   • Validación y parsing de entradas

3. INTERFAZ DE USUARIO
   • Menú principal con lista de tablas
   • Submenús específicos por tabla
   • Operaciones CRUD individuales
   • Confirmaciones de guardado

FLUJO DE DATOS:

Archivos CSV → Diccionario Python → Operaciones CRUD → Archivos CSV/JSON
↑ ↑ ↑ ↑
Carga Inicial | Memoria Principal | Modificación Usuario | Persistencia Cambios

MENÚ JERÁRQUICO:

Menú Principal → Selección Tabla → Submenú CRUD → Guardado
↓ ↓ ↓ ↓
Lista tablas | Tabla específica | A/M/B/E | Confirmación

FUNCIONES HELPER Y UTILIDADES

FUNCIONES PRINCIPALES DE IDENTIFICACIÓN:

is_id_field(field_name):
• Detecta campos que son identificadores
• Reglas: empieza con 'id', contiene 'id_', termina con '_id'
• Ejemplos: 'id', 'id_cliente', 'producto_id', 'ID'

get_main_id_field(registro):
• Encuentra el campo ID principal de un registro
• Prioriza campos que empiezan con 'id_' o son exactamente 'id'

get_modifiable_fields(registro):
• Devuelve campos que pueden modificarse (excluye IDs)
• Permite operaciones seguras de modificación

GENERACIÓN Y NORMALIZACIÓN:

generate_new_id(tabla, id_field):
• Genera nuevo ID como máximo existente + 1
• Maneja conversión robusta de tipos numéricos
• Fallback a longitud actual si hay errores

normalize_id_value(value):
• Normaliza valores ID para comparación consistente
• Convierte a float→int si es posible
• Maneja valores nulos, vacíos y strings

is_numeric_field(field_name):
• Detecta campos que probablemente son numéricos
• Keywords: 'precio', 'stock', 'cantidad', 'monto', 'total'

parse_input_value(field_name, value):
• Convierte entrada de usuario al tipo apropiado
• Limpia símbolos de moneda, comas, etc.
• Convierte a int/float según el campo

GESTIÓN DE ARCHIVOS Y PERSISTENCIA

FUNCIONES DE CARGA:

cargar_tablas():
• Carga todas las tablas definidas en el mapping
• Maneja archivos faltantes creando tablas vacías
• Convierte DataFrames pandas a listas de diccionarios
• Mapping predefinido para 10 tablas diferentes

TABLAS SOPORTADAS:
• clientes, localidades, provincias, productos
• rubros, sucursales, facturaenc, facturadet
• ventas, proveedores

FUNCIONES DE GUARDADO:

guardar_todo(tablas):
• Guarda todas las tablas en archivos CSV
• Crea directorios si no existen
• Manejo robusto de errores de escritura

guardar_todo_json(tablas):
• Exporta todas las tablas a formato JSON
• Usa sanitización para tipos no serializables
• Maneja NaN/NaT/inf convirtiéndolos a null

exportar_tabla_json(tabla, nombre):
• Exporta tabla individual a JSON
• Sanitiza valores: pandas NA → None, numpy → nativos
• Convierte timestamps a formato ISO
• Encoding UTF-8 con indentación para legibilidad

CARACTERÍSTICAS DE PERSISTENCIA:
• Guardado automático tras operaciones CRUD
• Opciones: CSV, JSON, o ambos
• Confirmación de usuario antes de guardar
• Mensajes de estado claros

INTERFAZ DE USUARIO Y NAVEGACIÓN

ESTRUCTURA DE MENÚS:

MENÚ PRINCIPAL:
• Lista numerada de todas las tablas disponibles
• Muestra conteo de registros por tabla
• Opciones globales: Guardar todo (CSV/JSON), Salir
• Validación de entrada numérica

SUBMENÚ POR TABLA:
• (A)gregar: Crea nuevo registro con ID automático
• (M)odificar: Edita campos existentes por ID
• (B)orrar: Vacía campos no-ID (preserva estructura)
• (E)xportar: Exporta tabla individual a JSON
• (V)olver: Regresa al menú principal

FUNCIONALIDADES DE VISUALIZACIÓN:

mostrar_tabla(tabla, nombre):
• Muestra tabla formateada con tabulate
• Normaliza valores numéricos (elimina .0 decimal)
• Reemplaza valores vacíos/nulos por '-'
• Oculta índices automáticos, muestra solo IDs reales
• Encabezados claros con conteo de registros

EXPERIENCIA DE USUARIO:
• Mensajes descriptivos con emojis
• Validación en tiempo real
• Confirmaciones antes de acciones destructivas
• Feedback inmediato de operaciones
• Opciones de guardado flexibles

OPERACIONES CRUD DETALLADAS

AGREGAR (A):
1. Genera automáticamente nuevo ID
2. Solicita valores para campos no-ID
3. Parsea entradas según tipo de campo
4. Agrega registro a la tabla en memoria
5. Muestra tabla actualizada
6. Ofrece opciones de guardado

MODIFICAR (M):
1. Solicita ID del registro a modificar
2. Busca usando comparación normalizada
3. Muestra campos modificables con valores actuales
4. Permite modificar campo individual o todos
5. Actualiza solo campos con nuevos valores
6. Ofrece opciones de guardado

BORRAR (B):
1. Solicita ID del registro a "borrar"
2. En lugar de eliminar, vacía campos no-ID
3. Preserva campos ID y estructura del registro
4. Mantiene integridad referencial
5. Ofrece opciones de guardado

EXPORTAR (E):
1. Exporta tabla actual a JSON
2. Aplica sanitización completa
3. Mantiene estructura de datos original

CARACTERÍSTICAS DE SEGURIDAD:
• No se permiten modificaciones en campos ID
• Validación de existencia de registros
• Preservación de estructura de datos
• Confirmaciones antes de guardar

MANEJO DE ERRORES Y ROBUSTEZ

ESTRATEGIAS DE MANEJO DE ERRORES:

1. ARCHIVOS:
   • Verifica existencia de archivos antes de cargar
   • Crea tablas vacías para archivos faltantes
   • Maneja excepciones de lectura/escritura
   • Proporciona mensajes de error descriptivos

2. DATOS:
   • Conversión segura de tipos numéricos
   • Manejo de valores nulos/vacíos
   • Normalización para comparaciones
   • Parsing robusto de entradas de usuario

3. INTERFAZ:
   • Validación de entradas numéricas
   • Manejo de opciones inválidas
   • Verificación de rangos y existencia
   • Recuperación graceful de errores

4. SERIALIZACIÓN:
   • Sanitización de tipos no serializables
   • Manejo de valores especiales (NaN, inf)
   • Conversión de tipos numpy/pandas
   • Encoding UTF-8 explícito

CARACTERÍSTICAS DE ROBUSTEZ:
• El sistema nunca crashea por datos inválidos
• Mensajes de error informativos
• Estado consistente tras errores
• Capacidad de recuperación
• Preservación de datos existentes

CONCLUSIÓN Y CARACTERÍSTICAS DESTACADAS

RESUMEN DEL SISTEMA:

FORTALEZAS PRINCIPALES:
 Gestión automática e inteligente de IDs
 Interfaz intuitiva con navegación jerárquica
 Soporte múltiple de formatos (CSV/JSON)
 Operaciones CRUD completas y seguras
 Manejo robusto de errores y datos inválidos
 Normalización inteligente de valores
 Persistencia flexible con confirmaciones

INNOVACIONES TÉCNICAS:
• Detección automática de campos ID y numéricos
• Generación inteligente de IDs secuenciales
• Comparación normalizada para búsquedas
• Sanitización completa para serialización JSON
• Parsing contextual de entradas de usuario
• Preservación de estructura en operaciones de borrado

USO PRÁCTICO:
• Ideal para bases de datos pequeñas/medianas
• Perfecto para prototipado y desarrollo
• Excelente para migraciones entre formatos
• Útil para enseñanza de conceptos de BD
• Flexible para personalización y extensión

POTENCIALES MEJORAS:
• Agregar validaciones de integridad referencial
• Implementar búsquedas y filtros avanzados
• Agregar soporte para transacciones
• Implementar logging de operaciones
• Agregar interfaz web o gráfica