SISTEMA DE VENTAS EN LÍNEA

Grupo 16

LUCAS ROJAS, CELINA VEGA RAMOS, MAXIMO MARTINEZ , GIULIANO RINALDI

INTRODUCCIÓN AL PROYECTO:

El propósito del sistema es gestionar productos, clientes y órdenes de venta para una aplicación de escritorio que realiza operaciones CRUD y genera reportes. El diseño de la base de datos prioriza:

- Integridad y consistencia de los datos.
- Normalización hasta 3NF para evitar redundancias.
- Facilidad para ejecutar consultas de negocio (reportes y búsquedas).
- Simplicidad adecuada al alcance académico (facilitando tests y consultas desde Python).

Se eligió MySQL como gestor; el esquema implementado se denomina gestion_de_Ventas y contiene tres tablas principales: productos, clientes y ordenes.

1. Entidades, atributos y cardinalidades: Entidades fuertes

1. **productos**
    - Atributos: idProductos (PK), nombre, categoria, precio, stock.
    - Comentario: almacena los datos maestros de cada producto disponible.
2. **clientes**
    - Atributos: idClientes (PK), nombre, email (UNIQUE), telefono, direccion.
    - Comentario: almacena la información de los compradores.
3. **ordenes** (modelo simplificado)
    - Atributos: idOrden (PK), cliente_id (FK -> clientes.idClientes), producto_id (FK -> productos.idProductos), cantProductos, fecha.
    - Comentario: en este diseño cada fila representa la compra de una cantidad de un producto por un cliente. Es la simplificación que se implementó para la entrega.

Relaciones y cardinalidades

- Un **cliente** puede tener **0..N** órdenes. (clientes 1 — N ordenes)
- Un **producto** puede aparecer en **0..N** órdenes. (productos 1 — N ordenes)

La relación ordenes es una tabla asociativa que modela una relación N:M entre clientes y productos si se piensa en pedidos multi-producto (pero **cada fila** solo guarda un producto por orden).

2- NORMALIZACION:

1NF — Forma Normal 1

- **Regla**: todos los atributos deben ser atómicos, sin listas ni conjuntos en una misma celda.
- **Cumplimiento**: las tablas productos, clientes y ordenes contienen atributos (nombre, email, precio, stock, cantProductos, etc.). No hay columnas que almacenen listas o valores repetidos en una celda.

2NF — Forma Normal 2

- **Regla**: todos los atributos no claves deben depender completamente de la clave primaria; es relevante cuando existen claves compuestas.
- **Cumplimiento**: las tablas usan claves primarias simples (idProductos, idClientes, idOrden). Los atributos en cada tabla dependen de la clave primaria completa: p.ej. precio y stock dependen solo de idProductos.

3NF — Forma Normal 3

- **Regla**: no debe existir dependencia transitiva entre atributos no clave.
- **Cumplimiento**: no almacenamos, por ejemplo, el email del cliente dentro de productos ni duplicamos información entre tablas. En ordenes se guarda cliente_id y producto_id y no se repiten otros datos del cliente o del producto (salvo cantProductos y fecha). Si se necesitara conservar el precio histórico en el momento de la venta se introduciría precio_unitario en la tabla de detalle (o orden_items) para evitar dependencias transitorias al modificar productos.precio.

Implementación en Python:

Se detalla la parte de la aplicación desarrollada en Python que permite interactuar con la base de datos MySQL:

- **Conexión a la base de datos:**
  Uso de mysql.connector para conectar Python con MySQL. Creación

automática de la base de datos si no existe. Manejo de errores y transacciones para mantener consistencia.

- **Tablas y relaciones:**
  Tablas: *clientes*, *productos* y *ordenes*.
  Relaciones con FOREIGN KEY usando ON UPDATE CASCADE y ON DELETE RESTRICT.
  Índice creado en productos(nombre) para optimizar búsquedas.
- **CRUD de productos y clientes:**
  Alta, baja, modificación y consulta de registros con validación de datos.
  Se evita borrar registros con dependencias activas.
- **Gestión de órdenes:**
  Creación de órdenes verificando cliente, producto y stock.
  Actualización automática del stock.
  Uso de transacciones para evitar inconsistencias.
- **Consultas avanzadas y reportes:**
  Uso de INNER JOIN y LEFT JOIN en reportes.
  Funciones agregadas (SUM) y cláusulas GROUP BY, ORDER BY.
  Reportes como “productos más vendidos”.
- **Manejo de errores y transacciones:**
  Control mediante try/except con commit y rollback.
  Protección contra fallos durante operaciones críticas.

- **Interfaz CLI (Menú):**
  Menú interactivo que permite gestionar toda la base desde la consola.