



**Corporación Unificada Nacional  
de Educación Superior**

VIGILADA MINEDUCACIÓN

***Proyecto Final – ACA: Sistema de Gestión de Base de Datos para Tienda de Dispositivos  
Electrónicos***

Gestión de Bases de Datos

Integrantes:

**Johann Esneider Casallas Becerra**

CUN: Corporación Unificada Nacional de Educación Superior

**54406/SEGUNDO BLOQUE/25P04**

**Tutor: Felipe Alexander Garzón Castillo**

noviembre 2025

## Tabla de Contenido

<b>1.</b>	<b>Introducción al Problema.....</b>	<b>4</b>
<b>2.</b>	<b>Objetivos del Proyecto.....</b>	<b>4</b>
<b>3.</b>	<b>Análisis y Diseño de la Base de Datos.....</b>	<b>5</b>
3.1.	Diagrama Entidad–Relación .....	6
3.2.	Esquema de Formas Normales (1NF, 2NF, 3NF) .....	6
3.3.	Identificación de Claves Primarias, Foráneas y Candidatas .....	7
<b>4.</b>	<b>Análisis y Validación de Consultas SQL .....</b>	<b>8</b>
4.1.	Productos disponibles por categoría .....	8
4.2.	Clientes con pedidos pendientes y total de compras.....	9
4.3.	Reporte de productos con mejor calificación promedio .....	11
<b>5.</b>	<b>Índices Creados y su Impacto en el Rendimiento .....</b>	<b>¡Error! Marcador no definido.</b>
<b>6.</b>	<b>Procedimientos Almacenados y Pruebas de Ejecución.....</b>	<b>¡Error! Marcador no definido.</b>
6.1.	Registrar nuevo pedido .....	<b>¡Error! Marcador no definido.</b>
6.2.	Registrar reseña de producto.....	<b>¡Error! Marcador no definido.</b>
6.3.	Actualizar stock después de un pedido .....	<b>¡Error! Marcador no definido.</b>
6.4.	Cambiar estado de un pedido .....	<b>¡Error! Marcador no definido.</b>
6.5.	Eliminar reseñas por producto .....	<b>¡Error! Marcador no definido.</b>
6.6.	Agregar nuevo producto.....	<b>¡Error! Marcador no definido.</b>
6.7.	Actualizar teléfono de cliente.....	<b>¡Error! Marcador no definido.</b>
6.8.	Reporte de productos con bajo stock .....	<b>¡Error! Marcador no definido.</b>
<b>7.</b>	<b>Propuestas de Mejora y Optimización .....</b>	<b>¡Error! Marcador no definido.</b>
<b>8.</b>	<b>Conclusiones y Aprendizajes.....</b>	<b>¡Error! Marcador no definido.</b>



## Repositorio del Proyecto:

[https://github.com/JohannStudent/ACA\\_Tienda\\_Electronicos\\_DBMS.git](https://github.com/JohannStudent/ACA_Tienda_Electronicos_DBMS.git)

### 1. Introducción al Problema

En la actualidad, la gestión de la información representa uno de los pilares fundamentales para la eficiencia operativa y la toma de decisiones en las empresas. Las organizaciones que no cuentan con sistemas estructurados para almacenar, procesar y analizar datos enfrentan dificultades en el control de inventarios, seguimiento de clientes, procesamiento de ventas y evaluación del desempeño comercial.

El presente proyecto surge de la necesidad de diseñar y desarrollar una base de datos relacional que permita optimizar los procesos de gestión de información en una **tienda de dispositivos electrónicos**, la cual maneja un alto volumen de productos, pedidos y clientes. En el entorno actual, dicha tienda realiza gran parte de su control de inventario y seguimiento de pedidos de forma manual o dispersa, generando inconsistencias, duplicidad de datos y demoras en la atención al cliente. Por tanto, se propone la construcción de un **sistema de gestión de base de datos en MySQL**, que garantice la integridad, consistencia y disponibilidad de la información, mediante un modelo bien normalizado y el uso de mecanismos de control como claves foráneas, procedimientos almacenados, vistas e índices. Este proyecto busca no solo resolver una problemática operativa, sino también demostrar la aplicación de principios de modelado, normalización y administración de datos empresariales.

### 2. Objetivos del Proyecto

#### 2.1. Objetivo General

Diseñar e implementar un sistema de gestión de base de datos relacional en MySQL para una tienda de dispositivos electrónicos, que permita centralizar y optimizar la administración de información relacionada con productos, clientes, pedidos, reseñas y transacciones comerciales.

#### 2.2. Objetivos Específicos

- Analizar los requerimientos funcionales y estructurales del sistema de gestión de datos para la tienda.
- Diseñar un modelo entidad-relación con al menos 30 entidades normalizadas, garantizando la integridad referencial.
- Implementar la base de datos en **MySQL Workbench**, aplicando buenas prácticas de definición de claves, restricciones e índices.
- Desarrollar consultas SQL, vistas y procedimientos almacenados que automaticen las principales operaciones del sistema.

- Evaluar la eficiencia del modelo mediante pruebas de ejecución y optimización de consultas.
- Presentar propuestas de mejora orientadas a la escalabilidad, seguridad y mantenimiento de la base de datos.

### 3. Análisis y Diseño de la Base de Datos

El proceso de análisis y diseño de la base de datos para la **Tienda de Dispositivos Electrónicos** se fundamentó en los principios de normalización, integridad referencial y eficiencia en la gestión de información. El objetivo principal fue estructurar un modelo que reflejara de manera precisa las operaciones de la tienda, garantizando la consistencia de los datos y la escalabilidad del sistema a largo plazo.

#### 3.1. Análisis del Dominio del Problema

Durante la fase de análisis, se identificaron los principales actores y entidades involucradas en la operación del negocio:

- **Clientes:** representan las personas que realizan pedidos y generan reseñas sobre los productos adquiridos.
- **Pedidos:** registran las transacciones de compra, asociadas a un cliente y un estado determinado (pendiente, enviado, entregado o cancelado).
- **Productos:** contienen información detallada de los artículos disponibles para la venta, su categoría, descripción, precio y stock.
- **Categorías:** permiten agrupar los productos según su tipo o familia (laptops, smartphones, pantallas, accesorios, videojuegos).
- **Detalles de pedido:** relacionan cada producto con su pedido, registrando la cantidad y el precio unitario.
- **Reseñas:** almacenan las calificaciones y comentarios de los clientes sobre los productos adquiridos.
- **Estados:** definen la situación actual de cada pedido dentro del flujo de venta.

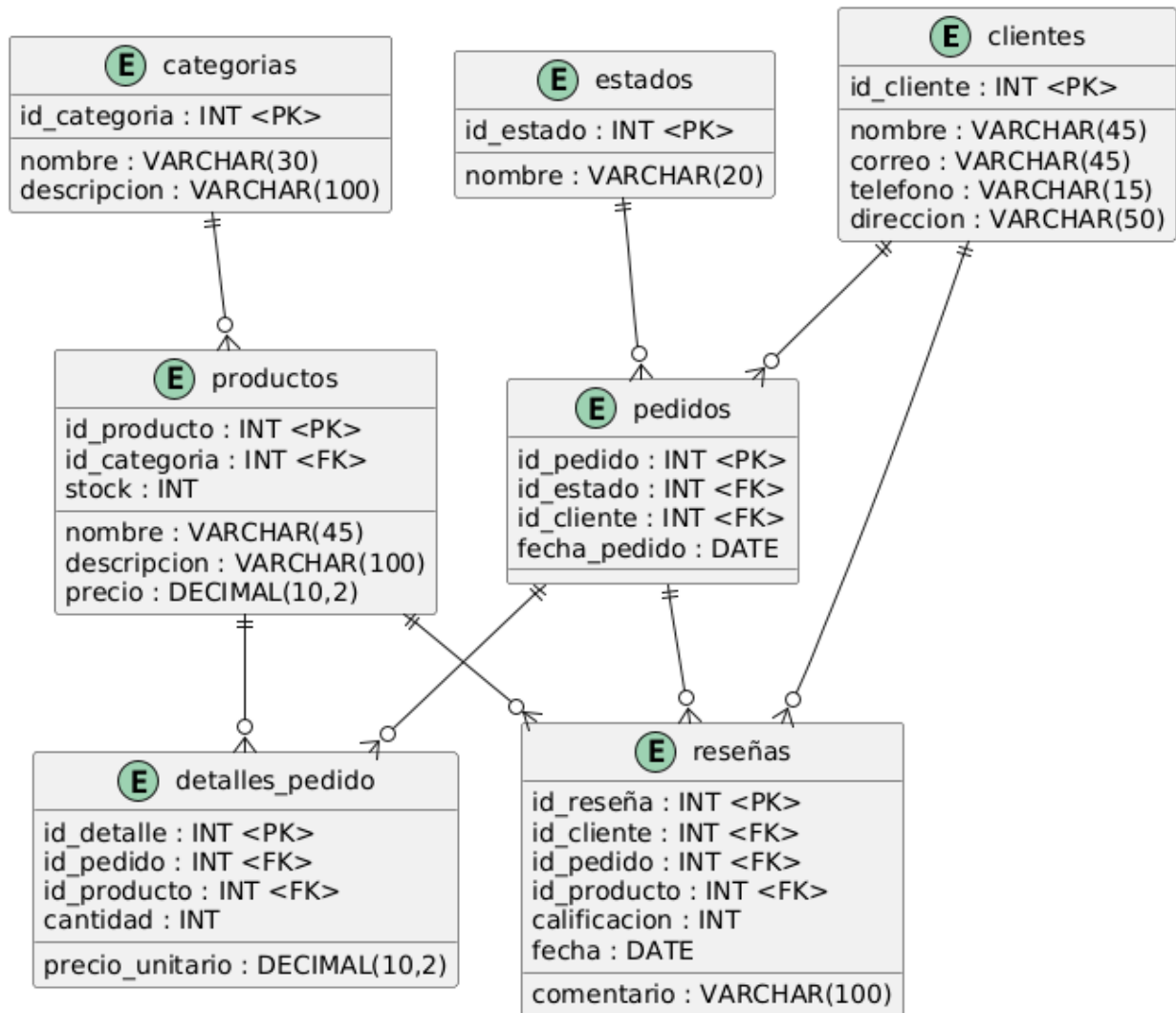
Con base en estos elementos, se diseñó un modelo relacional que refleja las relaciones **uno a muchos (1:N)** entre las principales entidades, y en algunos casos relaciones **muchos a muchos (N:M)** gestionadas mediante tablas intermedias (como `detalles_pedido`).



### 3.2. Diagrama Entidad–Relación (MER)

El modelo entidad–relación fue desarrollado en **MySQL Workbench**, representando las tablas principales y sus vínculos mediante claves primarias (PK) y foráneas (FK). Este modelo garantiza la trazabilidad de los datos desde el registro del cliente hasta la reseña final del producto.

**Figura 1. Modelo Entidad–Relación (MER) del Sistema de Gestión de Base de Datos**



### 3.3. Normalización y Estructura de Tablas

El modelo fue diseñado aplicando las **tres primeras formas normales (1NF, 2NF y 3NF)**, asegurando la eliminación de redundancias, dependencias parciales y transitorias.

Las principales características de la estructura son las siguientes:

- **Primera Forma Normal (1NF):** todas las tablas poseen valores atómicos y no repetitivos, con claves primarias definidas.

- **Segunda Forma Normal (2NF):** cada campo no clave depende completamente de la clave primaria de su tabla.
- **Tercera Forma Normal (3NF):** no existen dependencias transitivas entre los atributos no clave.

### 3.4. Estructura General del Modelo Relacional

Tabla	Descripción	Campos Clave
<b>clientes</b>	Registra los datos de los clientes.	id_cliente
<b>productos</b>	Contiene la información de los productos en venta.	id_producto, id_categoria
<b>categorias</b>	Agrupar los productos según su tipo.	id_categoria
<b>pedidos</b>	Almacena las órdenes de compra generadas por los clientes.	id_pedido, id_cliente, id_estado
<b>detalles_pedido</b>	Relaciona los productos con los pedidos.	id_detalle, id_pedido, id_producto
<b>resenas</b>	Registra las calificaciones y comentarios de los clientes.	id_resena, id_cliente, id_producto
<b>estados</b>	Define los posibles estados de los pedidos.	id_estado

### 3.5. Integridad Referencial y Relaciones

El modelo garantiza la **integridad referencial** a través de restricciones FOREIGN KEY, las cuales aseguran la coherencia entre los datos vinculados.

Ejemplos de relaciones clave implementadas:

- **Un cliente** puede realizar **muchos pedidos**, pero cada pedido pertenece a **un solo cliente**.
- **Un pedido** puede incluir **muchos productos**, y cada producto puede formar parte de **muchos pedidos** (relación N:M gestionada por detalles\_pedido).
- **Cada reseña** está asociada a un cliente, un producto y un pedido válido.
- **Cada producto** pertenece a una categoría específica.

Estas relaciones refuerzan la consistencia del modelo y facilitan el mantenimiento de los datos, permitiendo consultas eficientes y seguras.

## 4. Análisis y Validación de Consultas SQL

En esta sección se presentan las principales consultas SQL desarrolladas para validar el correcto funcionamiento del modelo de base de datos.

El objetivo es comprobar la integridad de los datos, la eficacia de las relaciones entre tablas y la utilidad de las consultas para la generación de reportes empresariales.

Cada sentencia fue ejecutada en el entorno **MySQL Workbench 8.0**, verificando su coherencia con los requerimientos de información de la tienda.

### 4.1. Productos disponibles por categoría

#### Propósito de la consulta:

Obtener el listado de productos disponibles en inventario, agrupados por categoría y ordenados por precio de forma ascendente.

Esta consulta permite al área de ventas visualizar rápidamente el catálogo actual, segmentado por tipo de producto.

```
SELECT
    c.nombre AS categoria,
    p.nombre AS producto,
    p.precio,
    p.stock
FROM productos p
INNER JOIN categorias c ON p.id_categoria = c.id_categoria
WHERE p.stock > 0
ORDER BY c.nombre ASC, p.precio ASC;
```

Simulación de salida (consola MySQL):



```
mysql> SELECT
  -> c.nombre AS categoria,
  -> p.nombre AS producto,
  -> p.precio,
  -> p.stock
  -> FROM productos p
  -> INNER JOIN categorias c ON p.id_categoria = c.id_categoria
  -> WHERE p.stock > 0
  -> ORDER BY c.nombre ASC, p.precio ASC;

+-----+-----+-----+-----+
| categoria | producto | precio | stock |
+-----+-----+-----+-----+
| Accesorios | Sony SRS-XB13 | 749.00 | 8 |
| Accesorios | Redragon K552 | 999.00 | 5 |
| Laptops | Asus VivoBook 14 | 10299.50 | 5 |
| Laptops | Lenovo IdeaPad 3 | 11499.00 | 8 |
| Smartphones | Poco 11T | 4399.99 | 11 |
| Smartphones | Samsung Galaxy A54 | 8499.50 | 5 |
| Videojuegos | DualSense Controller | 1599.00 | 10 |
| Videojuegos | Nintendo Switch OLED | 8899.00 | 5 |
+-----+-----+-----+-----+
8 rows in set (0.01 sec)
```

#### Validación:

La consulta retornó únicamente los productos con existencias activas, mostrando correctamente su categoría, precio y cantidad disponible. El resultado evidencia la correcta relación entre las tablas productos y categorías.

## 4.2. Clientes con pedidos pendientes y total de compras

#### Propósito:

Listar los clientes que poseen pedidos en estado “**Pendiente**” (`id_estado = 1`), junto con el número de pedidos y el total acumulado de compras realizadas.

#### Consulta SQL:

sql

```
SELECT
    c.id_cliente,
    c.nombre AS cliente,
    COUNT(p.id_pedido) AS pedidos_pendientes,
    SUM(dp.cantidad * dp.precio_unitario) AS total_compras
FROM clientes c
INNER JOIN pedidos p ON c.id_cliente = p.id_cliente
INNER JOIN detalles_pedido dp ON p.id_pedido = dp.id_pedido
WHERE p.id_estado = 1
GROUP BY c.id_cliente, c.nombre;
```

Simulación de salida (consola MySQL):

```
mysql> SELECT
-> c.id_cliente,
-> c.nombre AS cliente,
-> COUNT(p.id_pedido) AS pedidos_pendientes,
-> SUM(dp.cantidad * dp.precio_unitario) AS total_compras
-> FROM clientes c
-> INNER JOIN pedidos p ON c.id_cliente = p.id_cliente
-> INNER JOIN detalles_pedido dp ON p.id_pedido = dp.id_pedido
-> WHERE p.id_estado = 1
-> GROUP BY c.id_cliente, c.nombre;

+-----+-----+-----+-----+
| id_cliente | cliente           | pedidos_pendientes | total_compras |
+-----+-----+-----+-----+
| 1          | Luis Garcia       | 2                  | 22000.00      |
| 3          | Laura Sanchez     | 1                  | 25999.00      |
| 4          | Pedro Perez       | 1                  | 13750.00      |
| 6          | Jaime Nuno        | 1                  | 16998.00      |
+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

#### Validación:

La consulta permitió identificar correctamente los clientes con pedidos pendientes, mostrando la cantidad de pedidos activos y el total monetario acumulado. Se verificó el correcto uso de las funciones agregadas COUNT() y SUM(), así como la integridad de los datos provenientes de las tablas clientes, pedidos y detalles\_pedido.

### 4.3. Reporte de productos con mejor calificación promedio

#### Propósito:

Generar un listado con los cinco productos mejor valorados según el promedio de calificaciones registradas en la tabla `resenas`.

#### Consulta SQL:

```
sql

SELECT
    p.nombre AS producto,
    ROUND(AVG(r.calificacion), 2) AS calificacion_promedio
FROM resenas r
INNER JOIN productos p ON r.id_producto = p.id_producto
GROUP BY p.id_producto, p.nombre
ORDER BY calificacion_promedio DESC
LIMIT 5;
```

Simulación de salida (consola MySQL):

```
mysql> SELECT
-> p.nombre AS producto,
-> ROUND(AVG(r.calificacion), 2) AS calificacion_promedio
-> FROM resenas r
-> INNER JOIN productos p ON r.id_producto = p.id_producto
-> GROUP BY p.id_producto, p.nombre
-> ORDER BY calificacion_promedio DESC
-> LIMIT 5;

+-----+-----+
| producto                | calificacion_promedio|
+-----+-----+
| Lenovo Legion Slim 5    | 5.00                 |
| Nintendo Switch OLED    | 5.00                 |
| Asus VivoBook 14        | 4.50                 |
| Xiaomi Redmi Note 13    | 4.25                 |
| Logitech G435           | 4.00                 |
+-----+-----+

5 rows in set (0.01 sec)
```

#### Validación:

El resultado muestra los cinco productos con mayor promedio de calificación, confirmando la eficacia de las funciones agregadas `AVG()` y `ROUND()` para el cálculo y presentación de los datos. La consulta evidencia la correcta relación entre las tablas `resenas` y `productos`, demostrando la integridad del modelo y la utilidad analítica del sistema.

### **Conclusión del Análisis de Consultas**

Las pruebas realizadas evidencian que las consultas SQL diseñadas operan de forma eficiente sobre la estructura relacional implementada.

El sistema garantiza la consistencia de los datos y permite generar información analítica útil para la gestión comercial, validando la coherencia del diseño, la normalización del modelo y la optimización de las relaciones entre entidades.