2025

Tercer Entrega Proyecto Final e-ClothesDB

CODERHOUSE – CURSO SQL – TERCER ENTREGA – ECLOTHES DB

NICOLÁS CALABRÓ – COMISIÓN: 81805

ÍNDICE

1. INTRODUCCIÓN

El presente documento corresponde a la tercera entrega del proyecto final del curso SQL, el cual forma parte de la Carrera de Desarrollo Backend que actualmente me encuentro cursando en la plataforma Coderhouse. Durante la misma he desarrollado una página web e-commerce llamada **eClothes,** la cual simula ser una tienda online de ropa, utilizando Javascript, HTML y CSS.

Mi objetivo en este curso SQL es desarrollar la base de datos para la página web, donde se puedan gestionar productos, clientes, pedidos y pagos de manera ordenada en una base de datos MySQL.

En la segunda entrega, se realizaron las siguientes modificaciones y actualizaciones al proyecto

- Creación de una nueva tabla de auditorías de cambio de precio de productos: Tabla AUDITORIA-PRECIO-PROD.
- Se adhirieron restricciones ON DELETE y ON UPDATE en las tablas.
- Inserción de datos a todas las tablas.
- Vistas, funciones, procedimientos almacenados y triggers que agregan valor y automatización.

Para la tercera entrega, se han realizado las siguientes actualizaciones:

- Corrección de los comentarios de la segunda entrega (eliminación de pequeño código en el informe)
- Incorporación del Diagrama de Ingeniería Inversa.
- Analítica de datos mediante herramientas de Business Intelligence.
- Descripción de las tecnologías utilizadas.
- Backup de la Base de Datos.

2. DIAGRAMA DE ENTIDAD-RELACIÓN

A continuación, se presenta el diagrama DER de la base de datos.

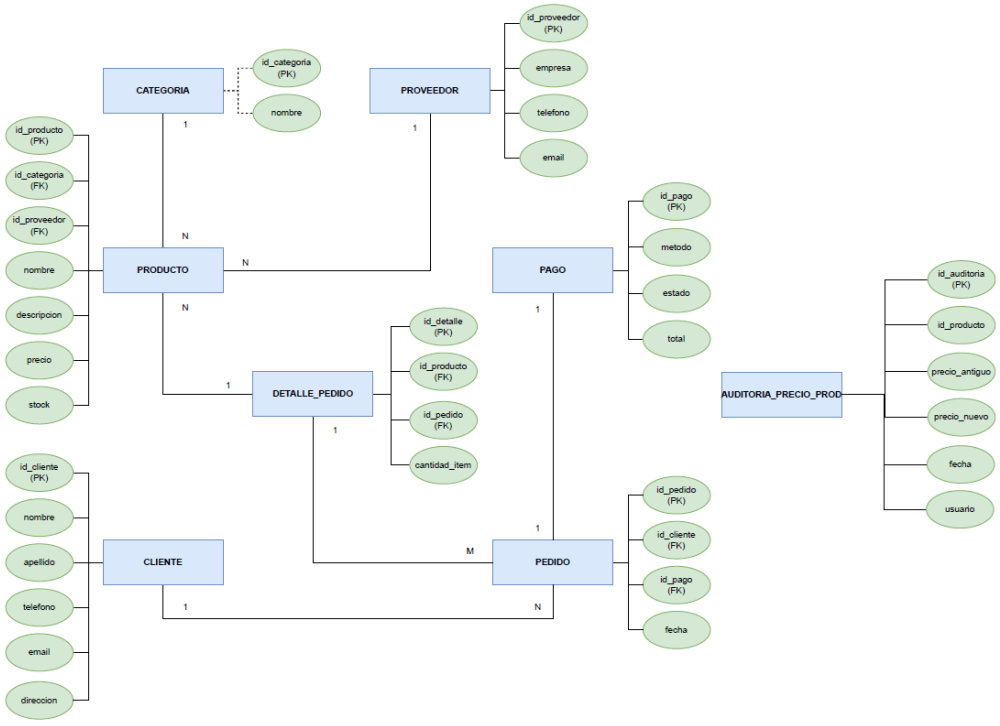

Figura 1: Diagrama Entidad Relación

En el mismo, se pueden apreciar las tablas con un recuadro celeste, los atributos con un óvalo verde y las relaciones entre las entidades mediante línea continua. Los atributos que son claves primarias y foráneas se presentan con el texto PK y FK entre paréntesis, respectivamente.

3. DIAGRAMA DE INGENIERIA INVERSA

A continuación, se presenta el diagrama de ingeniería inversa, extraído del gestor de base de datos MySQL Workbench.

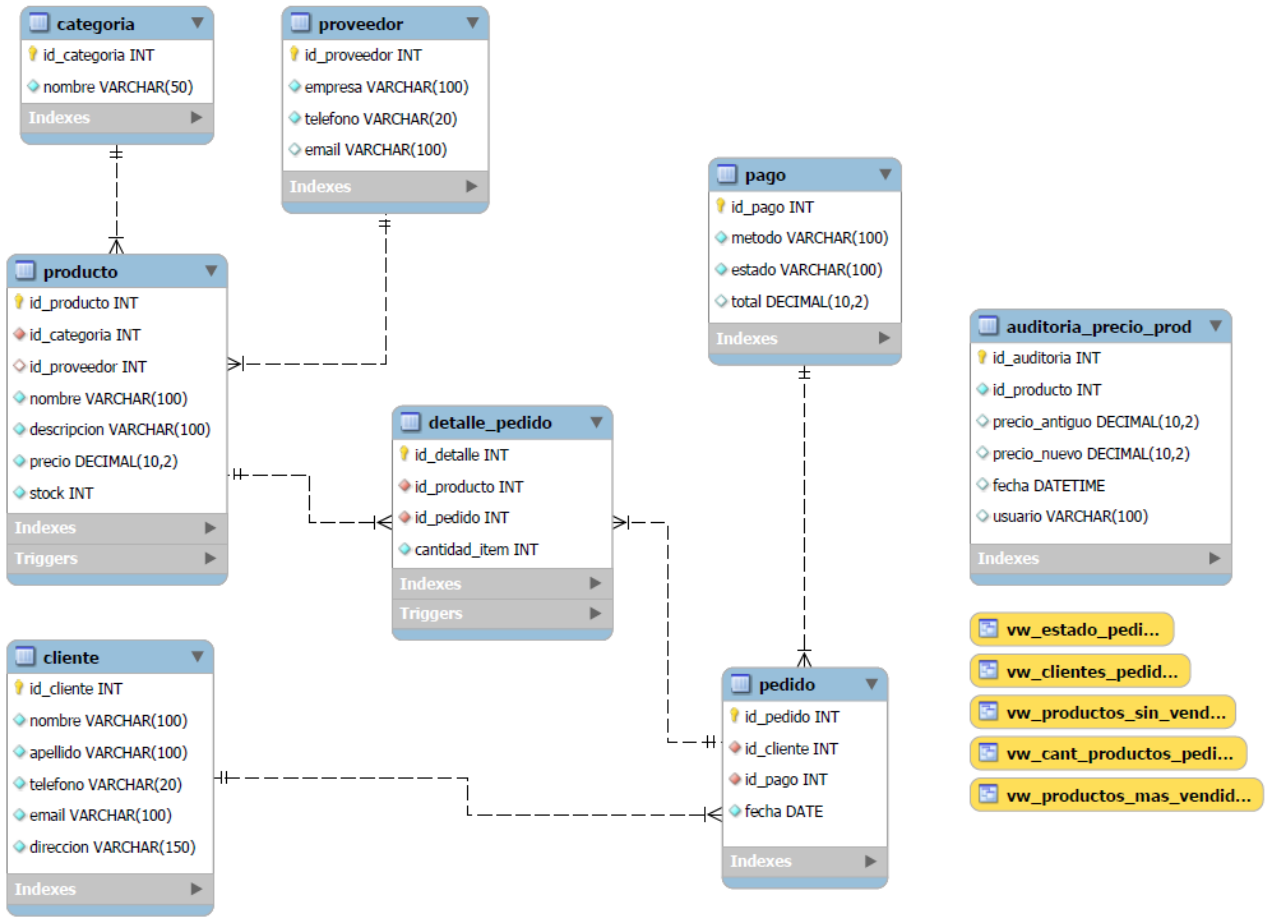

Figura 2: Diagrama Ingeniería Inversa

4. DESCRIPCIÓN DEL ESQUEMA, ENTIDADES Y RELACIONES

El esquema creado se denomina **eClothesDB**, y está compuesto por 8 tablas o entidades.

4.1 Tabla Categoría

Agrupa los productos según su tipo, por ejemplo: remeras, pantalones, calzado, etc.

CAMPO	TIPO DE DATO	PK	AI	FK	NULL	UNIQUE	DESCRIPCIÓN
id_categoria	INT	X	X				Identificador único
nombre	VARCHAR(50)						Nombre de la categoría

Tabla 1: Descripción Tabla Categoría

4.2 Tabla Proveedor

Almacena la información de las empresas proveedoras de productos.

CAMPO	TIPO DE DATO	PK	AI	FK	NULL	UNIQUE	DESCRIPCIÓN
id_proveedor	INT	X	X				Identificador único
empresa	VARCHAR(100)						Nombre de la empresa
telefono	VARCHAR(20)						Teléfono de la empresa
email	VARCHAR(100)					X	Dirección de correo de la empresa

Tabla 2: Descripción Tabla Proveedor

4.3 Tabla Producto

Catálogo de productos disponibles para la venta.

CAMPO	TIPO DE DATO	PK	AI	FK	NULL	UNIQUE	DESCRIPCIÓN
id_producto	INT	X	X				Identificador único
id_categoria	INT			X			FK a Categoría
id_proveedor	INT			X	X		FK a Proveedor
nombre	VARCHAR(100)						Nombre del producto
descripción	VARCHAR(100)						Descripción del producto
precio	DECIMAL(10,2)						Valor de venta
stock	INT						Cantidad en stock

Tabla 3: Descripción Tabla Producto

4.4 Tabla Cliente

Registra a los clientes que compran los productos en la página.

CAMPO	TIPO DE DATO	PK	AI	FK	NULL	UNIQUE	DESCRIPCIÓN
id_cliente	INT	X	X				Identificador único
nombre	VARCHAR(100)						Nombre del cliente
apellido	VARCHAR(100)						Apellido del cliente
teléfono	VARCHAR(20)						Teléfono de contacto
email	VARCHAR(100)					X	Dirección de correo
dirección	VARCHAR(150)						Dirección de despacho

Tabla 4: Descripción Tabla Cliente

4.5 Tabla Pago

Representa la información del pago asociado a un pedido.

CAMPO	TIPO DE DATO	PK	AI	FK	NULL	UNIQUE	DESCRIPCIÓN
id_pago	INT	X	X				Identificador único
método	VARCHAR(100)						Método de pago: efectivo, transferencia, tarjeta, etc
estado	VARCHAR(100)						Estado del pago: pagado, pendiente, rechazado, etc
total	DECIMAL(10,2)						Total a pagar

Tabla 5: Descripción Tabla Pago

4.6 Tabla Pedido

Almacena los pedidos realizados por los clientes.

CAMPO	TIPO DE DATO	PK	AI	FK	NULL	UNIQUE	DESCRIPCIÓN
id_pedido	INT	X	X				Identificador único
id_cliente	INT			X			FK a Cliente
id_pago	INT			X			FK a Pago
fecha	DATE						Fecha de pedido

Tabla 6: Descripción Tabla Pedido

4.7 Tabla Detalle-Pedido

Tabla de hechos. Contiene la información detallada de los productos dentro de cada pedido.

CAMPO	TIPO DE DATO	PK	AI	FK	NULL	UNIQUE	DESCRIPCIÓN
id_detalle	INT	X	X				Identificador único
id_producto	INT			X			FK a Producto
id_pedido	INT			X			FK a Pedido
cantidad_iItem	INT						Cantidad de ítems del producto

Tabla 7: Descripción Tabla Detalle-Pedido

4.8 Tabla Auditoria-Precio-Prod

Tabla de auditoría que refleja la actualización de precios en los productos.

CAMPO	TIPO DE DATO	PK	AI	FK	NULL	UNIQUE	DESCRIPCIÓN
id_auditoria	INT	X	X				Identificador único
id_producto	INT						Identificador del producto
precio_antiguo	DECIMAL(10,2)						Valor del precio antiguo
precio_nuevo	DECIMAL(10,2)						Valor del precio nuevo
fecha	DATETIME						Fecha de actualización de precio
usuario	VARCHAR(100)						Usuario que realizo la actualización

Tabla 8: Descripción Tabla Auditoria-Precio-Prod

4.9 Relación entre tablas

Las tablas se relacionan de la siguiente manera:

- Categoría – Producto: 1 a N. Varios productos pueden ser de la misma categoría.
- Proveedor - Producto: 1 a N. Un proveedor puede proveer varios productos.
- Cliente - Pedido: 1 a N. Un cliente puede hacer muchas ordenes de productos.
- Pago - Pedido: 1 a 1. Un pago corresponde a un solo pedido.
- Pedido - Producto: M a N. Un pedido puede tener varios productos y un producto puede estar en varios pedidos. Por ello he creado la tabla intermedia Detalle_Pedido.
- Auditoria-Precio-Prod: No tiene relación con ninguna tabla. Simplemente lleva un registro de los cambios de precios en los productos cuando los mismos se actualizan.

5. VISTAS

A continuación, se detallan las vistas creadas y sus funcionalidades.

5.1 Pedidos de cada cliente: vw_clientes_pedidos

Muestra el nombre completo de los clientes junto a sus pedidos realizados, ordenados por número de pedido.

5.2 Productos sin vender: vw_productos_sin_vender

Lista los productos que no se han vendido.

5.3 Estado de los pedidos: vw_estado_pedido

Detalla el estado y método de pago de cada pedido.

5.4 Cantidad de productos por pedido: vw_cant_productos_pedido

Indica la cantidad de productos que tiene cada pedido.

5.5 Productos más vendidos: vw_productos_mas_vendidos

Muestra los productos más vendidos

6. FUNCIONES

6.1 Calcular precio de venta de un producto: f_calcular_precio_venta

Calcula el precio de venta al público, en función del porcentaje de ganancia deseado.

La función recibe como parámetros el costo de un producto y el porcentaje de ganancia que queremos obtener en la venta de dicho producto, y devuelve el precio de venta. Por ejemplo, si un producto nos cuesta $50.000 y queremos obtener el 10% de ganancia en la venta, el producto debería ser vendido a $55.000

6.2 Calcular el precio total de un pedido: f_calcular_total_pedido

Calcula el valor de la venta total de un pedido.

La función recibe como parámetro el número de pedido y calcula el total del mismo sumando los precios y cantidades de los productos asociados. Por ejemplo, si un cierto pedido tiene asociado:

- 1x Remera estampada urbana = $5.200
- 1x Falda negra = $34.000
- 2x Falda de cuero sintético = $70.000

La función devuelve $109.200

7. STORED PROCEDURES

Se incorporaron dos procedimientos almacenados que automatizan tareas críticas de actualización dentro del sistema. Estos procedimientos permiten mantener la integridad y consistencia de los datos en las operaciones relacionadas con el control de stock y el cálculo de totales de los pedidos.

7.1 Actualizar el stock de productos: sp_actualizar_stock_productos

El procedimiento tiene como finalidad modificar el valor de stock disponible de un producto determinado, validando que el resultado nunca sea negativo.

El procedimiento recibe dos parámetros de entrada:

- Identificador del producto a actualizar.
- Cantidad a modificar, que puede ser positiva (para incrementar el stock) o negativa (para disminuirlo en caso de venta).

Internamente, el procedimiento obtiene el stock actual desde la tabla Producto, calcula el nuevo valor y lo actualiza. Además, verifica que el nuevo stock no sea menor que cero. En caso de que esta situación ocurra, el sistema genera un mensaje de error controlado evitando inconsistencias en los datos.
Por ejemplo, si un determinado producto posee un stock actual de 75 unidades, al llamar al procedimiento con una cantidad de 10 unidades, el stock pasará a 85 unidades. En cambio, al llamarlo con una cantidad negativa de 10 unidades, el stock se reducirá a 65 unidades.

7.2 Actualizar el total de un pedido: sp_actualizar_total_pedido

Este procedimiento actualiza el total a pagar de un pedido en particular.

El procedimiento recibe un único parámetro de entrada, el número identificador del pedido a actualizar.

Durante su ejecución, el procedimiento llama a la función descripta en el apartado 6.2, la cual obtiene el monto total del pedido sumando los precios unitarios de los productos por la cantidad de ítems adquiridos. Luego, almacena el resultado en el campo total de la tabla Pago, asociada al pedido correspondiente.
Por ejemplo, para el ejemplo descripto en el apartado 6.2, este procedimiento escribirá $109.200 en el campo total de la tabla Pago para el numero de pago correspondiente.

8. TRIGGERS

Se implementaron tres triggers con el objetivo de automatizar procesos críticos de actualización y control de integridad dentro de la base de datos. Cada uno de ellos se ejecuta de forma automática ante determinados eventos de inserción o modificación, garantizando que los datos se mantengan consistentes sin necesidad de intervención manual.

8.1 Actualización automática del stock al agregar una venta

- Trigger: tr_actualizar_stock_after_ins
- Evento: AFTER INSERT ON Detalle_Pedido

Este trigger se activa automáticamente después de que se inserta un nuevo registro en la tabla Detalle_Pedido, es decir, cuando se agrega un producto a un pedido. Su función es actualizar el stock del producto vendido, reduciendo la cantidad disponible en función del número de unidades incluidas en la venta. Para ello, el trigger invoca al procedimiento almacenado del apartado 7.1, enviando como parámetros el identificador del producto y la cantidad vendida (multiplicada por -1, ya que se descuenta del stock actual).

Por ejemplo, si un producto tiene un stock de 80 unidades, al agregar 4 unidades del mismo a un nuevo pedido el sistema descuenta automáticamente estas unidades del stock disponible del producto, quedado 76.

8.2 Actualización automática del total de una venta

- Trigger: tr_calcular_total_after_ins
- Evento: AFTER INSERT ON Detalle_Pedido

Este trigger se ejecuta inmediatamente después de insertar una nueva entrada en la tabla Detalle_Pedido. Su finalidad es recalcular el monto total del pedido afectado, actualizando dicho valor en la tabla Pago asociada. Para ello, el trigger llama al procedimiento almacenado del apartado 7.2, enviando como parámetro el identificador del pedido. De esta manera, cada vez que se agregan productos a un pedido, el total asociado se actualiza automáticamente, evitando errores o desfasajes entre los registros de ventas y pagos.

8.3 Auditoría de cambios en los precios de los productos

- Trigger: tr_registrar_precio_after_upd
- Evento: AFTER UPDATE ON Producto

Este trigger tiene como propósito registrar automáticamente las modificaciones de precios en los productos dentro de la tabla Auditoria_Precio_Prod. De esta forma, el sistema mantiene un historial de los cambios realizados sobre los precios, lo que permite realizar controles y trazabilidad sobre las actualizaciones efectuadas.
El trigger compara el valor anterior con el nuevo, y solo genera un registro si el precio realmente ha cambiado. Por ejemplo, si un producto tiene un precio de $70000, y se actualiza su precio a $75000, en la tabla de auditaría se registrará el cambio de precio junto con la fecha y el usuario que realizo la actualización.

9. ANALITICA DE DATOS

En esta etapa se incorporó el uso de herramientas de visualización (Power BI) para analizar la información contenida en la base de datos, con el fin de extraer valor a partir de los datos registrados en las distintas tablas, para de esta manera generar una visión más clara del desempeño de las ventas y del comportamiento de los clientes.
Se generaron diferentes paneles de análisis utilizando la información contenida en la base de datos en MySQL, transformándola en fuente de información dinámica dentro de Power BI. A continuación, se detallan los análisis realizados.

9.1 Productos más vendidos

Se elaboró un gráfico de barras que muestra los productos con mayor volumen de ventas. El mismo fue construido con la ayuda de la vista vw_productos_mas_vendidos, detallada en el apartado 5.5, y la tabla categoría.

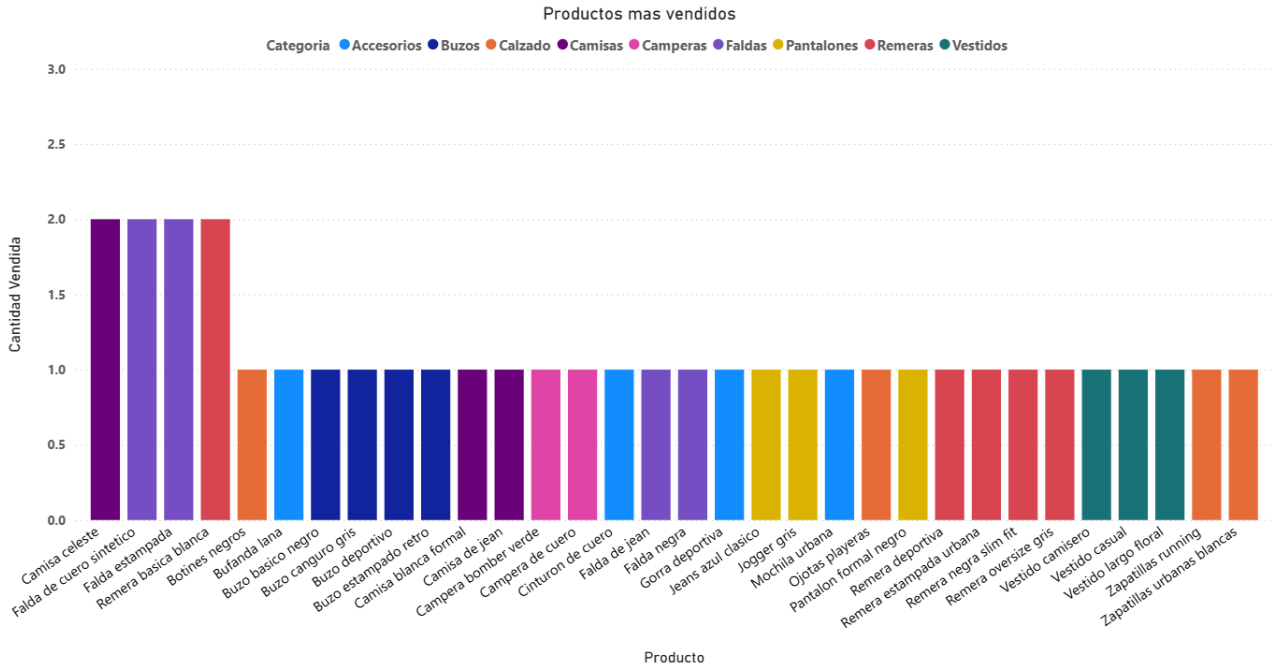

Figura 3: Productos más vendidos

En él se puede apreciar:

- Se identifican los productos con mayor rotación, es decir, aquellos que presentan una demanda sostenida y alta frecuencia de compra.
- Los productos de la categoría “Remeras” y “Faldas” aparecen entre los más vendidos, lo cual sugiere una fuerte preferencia por prendas básicas y de moda casual.
- La visualización permite detectar también productos con ventas bajas o nulas, que podrían ser objeto de revisión en las estrategias de promoción o reposición.

9.2 Estado de los pedidos

A partir de la información proporcionado por la vista vw_estado_pedido, detalla en el apartado 5.3, se construyó un gráfico de barras agrupadas que relaciona el método de pago con el estado del pedido.

En él se puede apreciar:

- Los métodos más utilizados son “Transferencia bancaria” y “Tarjeta de débito”, lo cual demuestra la preferencia del cliente por medios electrónicos.
- La mayoría de los pedidos figuran como “Pagados”, lo que indica una alta tasa de concreción de ventas.
- Se registran algunos pedidos con estado “Pendiente” o “Cancelado”, principalmente asociados a transferencias bancarias, lo que podría sugerir demoras o errores en la confirmación de pagos.

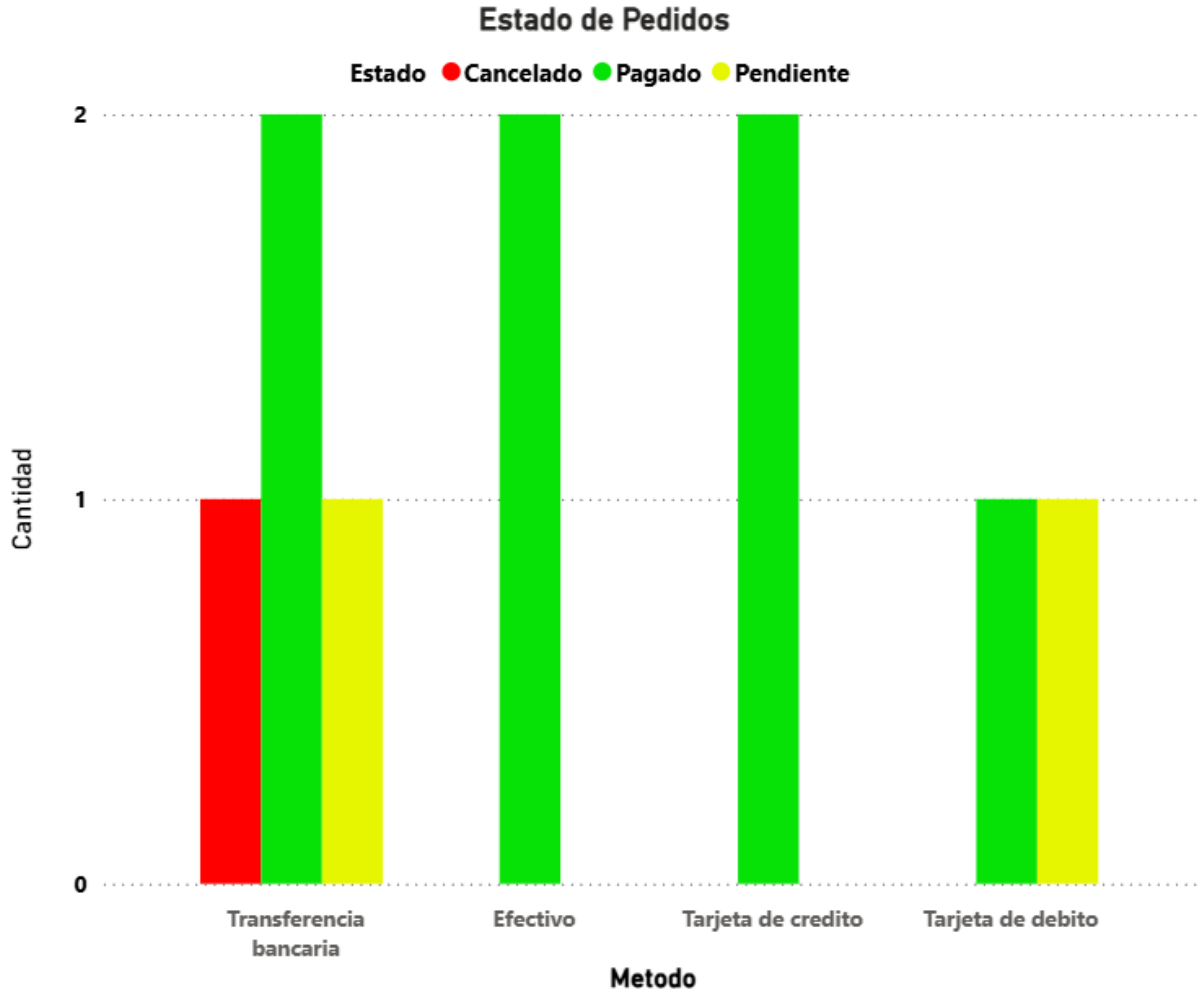

Figura 4: Estado de los pedidos

10. TECNOLOGIAS UTILIZADAS

Se integraron diversas herramientas y tecnologías del ecosistema de desarrollo backend y de análisis de datos, combinando la potencia de MySQL como sistema gestor de bases de datos con las capacidades visuales de Power BI para la exploración y comprensión de la información.

A continuación, se detallan las principales tecnologías empleadas:

- MySQL 8.0: Sistema de gestión de bases de datos relacional (RDBMS) utilizado para la creación del esquema, las tablas, relaciones, vistas, funciones, procedimientos almacenados y triggers.
- MySQL Workbench 8.0 CE: Entorno visual para el diseño, modelado y administración de la base de datos.
- Power BI Desktop 2025: Herramienta de Microsoft para la creación de reportes y dashboards. Utilizada para la analítica de datos a partir de las vistas.
- Git y GitHub: Sistema de control de versiones distribuido utilizado para gestionar los cambios del proyecto y almacenar los archivos del mismo en un repositorio remoto. La dirección del mismo es la siguiente: ***https://github.com/nicolascalabro/E-ClothesDB.git***
- HTML, CSS y JavaScript: Tecnologías empleadas en el desarrollo del frontend del e-commerce eClothes, que sirvió como contexto de aplicación de la base de datos.

Estas herramientas, en conjunto, permitieron desarrollar un entorno completo de trabajo que abarca desde la creación del modelo lógico de datos hasta la visualización analítica de la información.

11. CONCLUSIONES

El desarrollo de la base de datos eClothesDB permitió consolidar un modelo relacional sólido y escalable para el sistema e-commerce que me encuentro desarrollando. A lo largo del proyecto se cumplieron los principales objetivos propuestos: diseño del esquema, establecimiento de relaciones entre entidades, implementación de mecanismos de automatización mediante stored procedures y triggers, y finalmente, la integración con herramientas de análisis de datos.
Los resultados de la analítica con Power BI demuestran el potencial de la base de datos como fuente de información estratégica. Las vistas implementadas permitieron obtener métricas claves del negocio, como los productos más vendidos, el comportamiento de los métodos de pago y el estado general de las ventas.

Entre las principales conclusiones se destacan:

- La correcta estructuración de las relaciones entre tablas garantiza la integridad referencial y facilita consultas complejas.
- La automatización mediante procedimientos y disparadores reduce el riesgo de errores manuales y mantiene actualizados los datos de stock y ventas.
- La integración con herramientas de Business Intelligence evidencia el valor de los datos en la toma de decisiones.
- El proyecto constituye una base sólida para futuros desarrollos, como la incorporación de dashboards interactivos, segmentación de clientes o predicción de ventas.