CODER HOUSE

Documentación del Proyecto Punto LAN

ÍNDICE:

Introducción:

El presente proyecto tiene como objetivo desarrollar una base de datos relacional para gestionar el sistema de ventas de una tienda online de productos tecnológicos, llamada Punto LAN . Esta base de datos permitirá gestionar los productos, las ventas, los usuarios (clientes, vendedores y dueños), las garantías de los productos, los métodos de pago, proveedores y las plataformas de venta. El modelo de negocio incluye tanto ventas directas a través de la página web como ventas a través de plataformas de redes sociales.

Objetivo:

Objetivo: Desarrollar una base de datos relacional que permita gestionar de manera eficiente

Desarrollar una base de datos relacionales que permita gestionar de manera eficiente las operaciones de venta de productos tecnológicos en Punto LAN. La base de datos debe permitir almacenar información detallada sobre productos, ventas, clientes, vendedores, métodos de pago, proveedores, y más, mejorando así la administración de los procesos logísticos, comerciales y analíticos. También se busca que la base de datos sea escalable y flexible, permitiendo la integración futura con nuevas funcionalidades.

Situación Problemática:

Punto LAN es un emprendimiento que ha experimentado un crecimiento en sus operaciones, con ventas tanto en su tienda online como en redes sociales. La falta de una base de datos centralizada ha hecho difícil la gestión de la información, el seguimiento de ventas, el manejo de inventarios y la fidelización de clientes. La solución propuesta es crear una base de datos relacional que centralice toda la información y optimice los procesos, permitiendo una toma de decisiones más eficiente.

Modelo de Negocio:

Punto LAN se dedica a la venta de productos tecnológicos y accesorios a través de su tienda online y de plataformas sociales como Instagram y Facebook. Los clientes pueden comprar productos directamente desde la tienda online o mediante redes sociales, eligiendo su método de pago preferido. El negocio depende de vendedores, que gestionan las ventas y ayudan en la atención al cliente. Además, Punto LAN compra sus productos a través de proveedores mayoristas y ofrece garantías a los productos vendidos. Las ventas se realizan a través de diversos métodos de pago como tarjeta de crédito y PayPal.

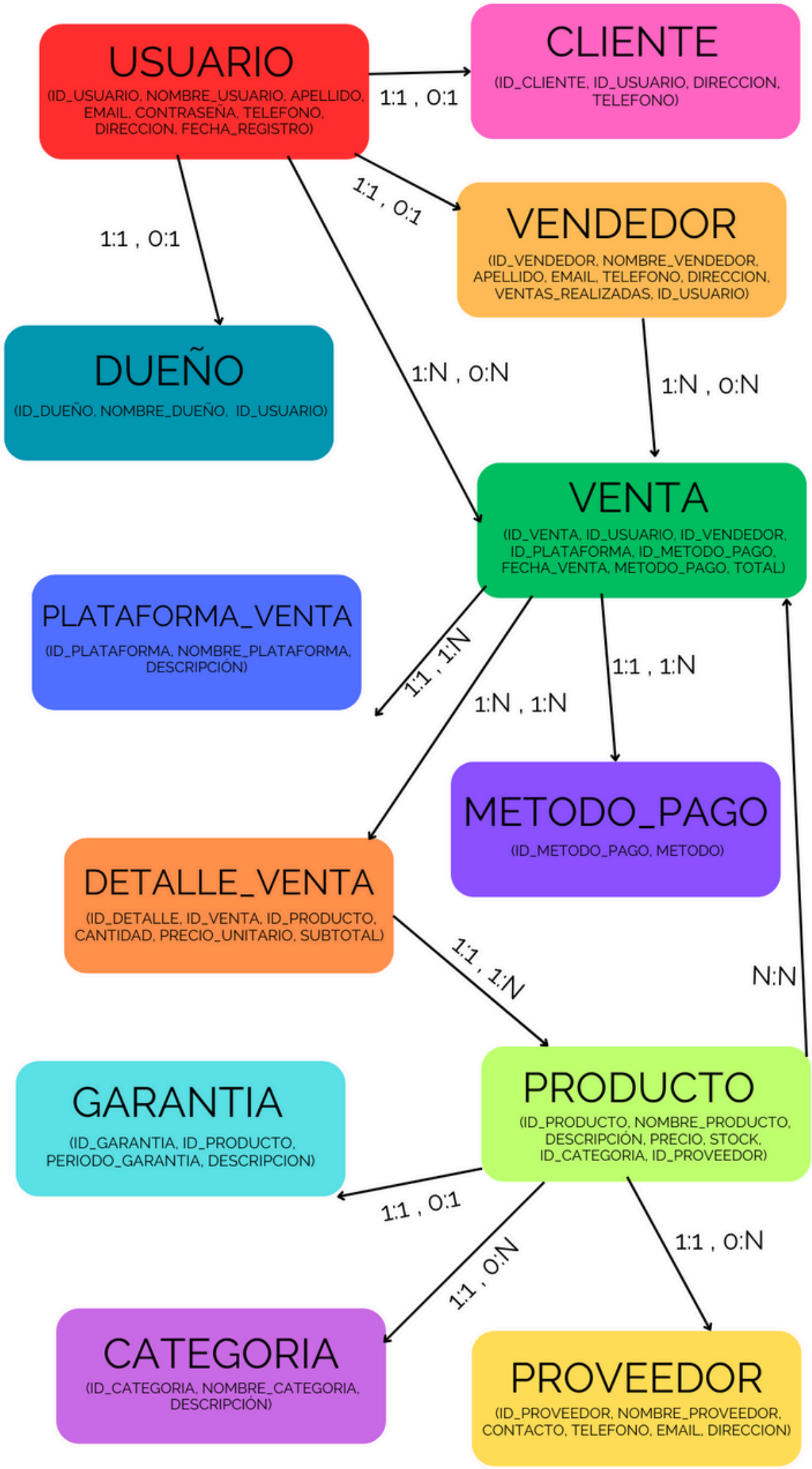
USUARIO
(ID_USUARIO, NOMBRE_USUARIO, APELLIDO, EMAIL, CONTRASEÑA, TELEFONO, DIRECCION, FECHA_REGISTRO)
CLIENTE
(ID_CLIENTE, ID_USUARIO, DIRECCION, TELEFONO)
1:1 , 0:1
VENDEDOR
(ID_VENDEDOR, NOMBRE_VENDEDOR, APELLIDO, EMAIL, TELEFONO, DIRECCION, VENTAS_REALIZADAS, ID_USUARIO)
1:1 , 0:1
1:1 , 0:1
DUEÑO
(ID_DUEÑO, NOMBRE_DUEÑO, ID_USUARIO)
1:N , 0:N
1:N , 0:N
VENTA
(ID_VENTA, ID_USUARIO, ID_VENDEDOR, ID_PLATAFORMA, ID_METODO_PAGO, FECHA_VENTA, METODO_PAGO, TOTAL)
PLATAFORMA_VENTA
(ID_PLATAFORMA, NOMBRE_PLATAFORMA, DESCRIPCIÓN)
1:1 , 1:N
1:N , 1:N
1:1 , 1:N
METODO_PAGO
(ID_METODO_PAGO, METODO)
DETALLE_VENTA
(ID_DETALLE, ID_VENTA, ID_PRODUCTO, CANTIDAD, PRECIO_UNITARIO, SUBTOTAL)
1:1 , 1:N
N:N
GARANTIA
(ID_GARANTIA, ID_PRODUCTO, PERIODO_GARANTIA, DESCRIPCION)
PRODUCTO
(ID_PRODUCTO, NOMBRE_PRODUCTO, DESCRIPCIÓN, PRECIO, STOCK, ID_CATEGORIA, ID_PROVEEDOR)
1:1 , 0:1
1:1 , 0:N
1:1 , 0:N
CATEGORIA
(ID_CATEGORIA, NOMBRE_CATEGORIA, DESCRIPCIÓN)
PROVEEDOR
(ID_PROVEEDOR, NOMBRE_PROVEEDOR, CONTACTO, TELEFONO, EMAIL, DIRECCION)

Diagrama ER (Entidad-Relación)

Relaciones y Cardinalidades

1. Usuario - Vendedor: (1,1) - (0,1)
   - Un usuario puede ser un vendedor o no.
   - Un vendedor está asociado a un solo usuario.
2. Usuario - Dueño: (1,1) - (0,1)
   - Un usuario puede ser un dueño o no.
   - Un dueño está asociado a un solo usuario.
3. Usuario - Cliente: (1,1) - (0,1)
   - Un usuario puede ser un cliente o no.
   - Un cliente está asociado a un solo usuario.
4. Usuario - Venta: (1,n) - (0,n)
   - Un usuario puede realizar muchas ventas.
   - Una venta pertenece a un solo usuario.
5. Vendedor - Venta: (1,n) - (0,n)
   - Un vendedor puede realizar múltiples ventas.
   - Una venta está asociada a un solo vendedor.
6. Venta - Método de Pago: (1,1) - (1,n)
   - Una venta tiene un solo método de pago.
   - Un método de pago puede estar en muchas ventas.
7. Venta - Plataforma de Venta: (1,1) - (1,n)
   - Una venta se realiza en una sola plataforma.
   - Una plataforma puede registrar múltiples ventas.
8. Venta - Detalle Venta: (1,n) - (1,n)
   - Una venta puede incluir múltiples productos (detalle_venta).
   - Cada detalle de venta está asociado a una única venta.
9. Detalle Venta - Producto: (1,1) - (1,n)
   - Un detalle de venta hace referencia a un solo producto.
   - Un producto puede aparecer en múltiples detalles de venta.
10. Producto - Categoría: (1,1) - (0,n)
    - Un producto pertenece a una sola categoría.
    - Una categoría puede incluir múltiples productos.
11. Producto - Proveedor: (1,1) - (0,n)
    - Un producto es suministrado por un solo proveedor.
    - Un proveedor puede suministrar múltiples productos.
12. Producto - Garantía: (1,1) - (0,1)
    - Un producto puede tener o no garantía.
    - Una garantía pertenece a un solo producto.

Listado de Tablas:

El siguiente apartado representa de manera fiel todas las entidades incluidas en el script de creación del esquema de datos del proyecto. Por cada entidad se describen detalladamente sus campos (columnas), el tipo de dato de cada uno, así como sus claves primarias, foráneas, únicas e índices en los casos que correspondan. Esta estructura asegura la integridad de la información y permite un diseño coherente y funcional del sistema.

Tabla: `usuario`

Campo	Tipo de Dato	Restricciones
id_usuario	INT	PRIMARY KEY, AUTO_INCREMENT
nombre_usuario	VARCHAR(100)	NOT NULL
apellido	VARCHAR(50)	NOT NULL
email	VARCHAR(100)	NOT NULL, UNIQUE
contraseña	VARCHAR(100)	NOT NULL
telefono	VARCHAR(20)	NULLABLE
direccion	VARCHAR(255)	NULLABLE
fecha_registro	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Tabla: `cliente`

Campo	Tipo de Dato	Restricciones
id_cliente	INT	PRIMARY KEY, AUTO_INCREMENT
id_usuario	INT	FOREIGN KEY → usuario(id_usuario)
direccion	VARCHAR(255)	NULLABLE
telefono	VARCHAR(15)	NULLABLE

Tabla: `producto`

Campo	Tipo de Dato	Restricciones
id_producto	INT	PRIMARY KEY, AUTO_INCREMENT
nombre_producto	VARCHAR(100)	NOT NULL
descripcion	TEXT	NULLABLE
precio	DECIMAL(10,2)	NOT NULL
stock	INT	NOT NULL
id_categoria	INT	FOREIGN KEY → categoria(id_categoria) ON DELETE SET NULL
id_proveedor	INT	FOREIGN KEY → proveedor(id_proveedor) ON DELETE SET NULL

Tabla: `venta`

Campo	Tipo de Dato	Restricciones
id_venta	INT	PRIMARY KEY, AUTO_INCREMENT
id_usuario	INT	FOREIGN KEY → usuario(id_usuario)
id_vendedor	INT	FOREIGN KEY → vendedor(id_vendedor)
fecha_venta	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
total	DECIMAL(10,2)	NOT NULL

Tabla: `detalle_venta`

Campo	Tipo de Dato	Restricciones
id_detalle	INT	PRIMARY KEY, AUTO_INCREMENT
id_venta	INT	FOREIGN KEY → venta(id_venta)
id_producto	INT	FOREIGN KEY → producto(id_producto)
cantidad	INT	NOT NULL
precio_unitario	DECIMAL(10,2)	NOT NULL
subtotal	DECIMAL(10,2)	NOT NULL

Tabla: `vendedor`

Campo	Tipo de Dato	Restricciones
id_vendedor	INT	PRIMARY KEY, AUTO_INCREMENT
nombre_vendedor	VARCHAR(100)	NOT NULL
apellido	VARCHAR(50)	NOT NULL
email	VARCHAR(100)	NOT NULL, UNIQUE
telefono	VARCHAR(20)	NULLABLE
direccion	VARCHAR(255)	NULLABLE
ventas_realizadas	INT	DEFAULT 0
id_usuario	INT	FOREIGN KEY → usuario(id_usuario) ON DELETE CASCADE

Tabla: `proveedor`

Campo	Tipo de Dato	Restricciones
id_proveedor	INT	PRIMARY KEY, AUTO_INCREMENT
nombre_proveedor	VARCHAR(100)	NOT NULL
telefono	VARCHAR(20)	NULLABLE
email	VARCHAR(100)	NOT NULL, UNIQUE
direccion	VARCHAR(255)	NULLABLE

Tabla: `categoria`

Campo	Tipo de Dato	Restricciones
id_categoria	INT	PRIMARY KEY, AUTO_INCREMENT
nombre_categoria	VARCHAR(100)	NOT NULL

Stored Procedures, Funciones, Vistas y Triggers:

Procedimientos Almacenados (Stored Procedures)

- sp_registrar_venta

Funcionalidad: Registra una nueva venta y su detalle en la base de datos.

```
DELIMITER //

CREATE PROCEDURE registrar_venta(
   IN id_usuario INT,
   IN id_vendedor INT,
   IN id_metodo_pago INT,
   IN total DECIMAL(10,2)
)
BEGIN
   INSERT INTO venta (id_usuario, id_vendedor, id_metodo_pago, total)
   VALUES (id_usuario, id_vendedor, id_metodo_pago, total);
END //

DELIMITER ;
```

- sp_agregarproducto

Funcionalidad: Agrega un producto.

```
CREATE PROCEDURE agregar_producto(
   IN nombre_producto VARCHAR(100),
   IN descripcion TEXT,
   IN precio DECIMAL(10,2),
   IN stock INT,
   IN id_categoria INT,
   IN id_proveedor INT
)
BEGIN
   INSERT INTO producto (nombre_producto, descripcion, precio, stock,
id_categoria, id_proveedor)
   VALUES (nombre_producto, descripcion, precio, stock, id_categoria,
id_proveedor);
END //

DELIMITER ;
```

Funciones

- fn_calcular_subtotal

Funcionalidad: Calcula el total según precio unitario y cantidad.

```
DELIMITER //

CREATE FUNCTION calcular_total_venta(p_id_venta INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
   DECLARE total DECIMAL(10,2);

   -- Calcular el total sumando los subtotales de cada producto
   SELECT SUM(subtotal)
   INTO total
   FROM detalle_venta
   WHERE id_venta = p_id_venta;

   -- Devolver el total calculado
   RETURN total;
END //

DELIMITER ;
```

- fn_obtenerstockdeunproducto

Funcionalidad: Calcula el stock de un producto.

```
DELIMITER //

CREATE FUNCTION obtener_stock_producto(p_id_producto INT)
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE stock INT;
   SELECT stock INTO stock
   FROM producto
   WHERE id_producto = p_id_producto;
   RETURN stock;
END //

DELIMITER ;
```

Vistas (Views)

- vw_vista_productos

Funcionalidad: Muestra el detalle completo del producto.

```
CREATE VIEW vista_productos AS
SELECT
   p.id_producto,
   p.nombre_producto,
   p.descripcion,
   p.precio,
   p.stock,
   c.nombre_categoria,
   pr.nombre_proveedor
FROM
   producto p
JOIN
   categoria c ON p.id_categoria = c.id_categoria
JOIN
   proveedor pr ON p.id_proveedor = pr.id_proveedor;
```

- vw_vista_ventas

Funcionalidad: Muestra el detalle completo de la venta.

```
CREATE VIEW vista_ventas AS
SELECT
   v.id_venta,
   u.nombre_usuario,
   u.email,
   ven.nombre_vendedor,
   mp.metodo,
   v.total,
   v.fecha_venta
FROM
   venta v
JOIN
   usuario u ON v.id_usuario = u.id_usuario
JOIN
   vendedor ven ON v.id_vendedor = ven.id_vendedor
JOIN
   metodo_pago mp ON v.id_metodo_pago = mp.id_metodo_pago;
```

Triggers (Desencadenantes)

- tr_actualizarstock

Funcionalidad: Actualiza el stock automáticamente al insertar un detalle de venta.

```
DELIMITER //

CREATE TRIGGER actualizar_stock_after_venta
AFTER INSERT ON detalle_venta
FOR EACH ROW
BEGIN
   UPDATE producto
   SET stock = stock - NEW.cantidad
   WHERE id_producto = NEW.id_producto;
END //

DELIMITER ;
```

- tr_verificarstock

Funcionalidad: Verifica el stock automáticamente al insertar un detalle de venta.

```
DELIMITER //

CREATE TRIGGER verificar_stock_before_venta
BEFORE INSERT ON detalle_venta
FOR EACH ROW
BEGIN
   DECLARE stock_actual INT;
   SELECT stock INTO stock_actual FROM producto WHERE id_producto =
NEW.id_producto;
   IF stock_actual < NEW.cantidad THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Stock insuficiente para el
producto';
   END IF;
END //

DELIMITER ;
```

Conclusión

Este documento detalla el esquema de la base de datos del proyecto, incluyendo el listado de tablas, tipos de datos y restricciones, además de los procedimientos almacenados, funciones, vistas y desencadenantes. La estructura ha sido diseñada para garantizar la integridad referencial y el correcto funcionamiento del sistema.