Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

TEMA

Avance 03 Proyecto

Autores:

Carrera Cherrez Abel Jared
Loayza Ramírez Francis Alain
Mendez Rojas Anai Ammy
Murillo Murillo Erick Adrian

2025-2026

Docente:

CHEUNG RUIZ IRENE MEIYING

Contenido

Triggers

En el Sistema agregamos dos triggers que se ejecutan automáticamente cuando cambian los detalles de una transacción.

Al insertar un detalle:
Recalcula el valor total de la transacción.

```
CREATE TRIGGER trg_detalles_pxt_ai_recalc
AFTER INSERT ON DETALLES_PXT
FOR EACH ROW
BEGIN
    UPDATE TRANSACCION t
    JOIN (
        SELECT d.transaccionID,
               SUM(d.cantidad * (p.precio * (1 + (p.porcentajeIVA/100)))) AS total
        FROM DETALLES_PXT d
        JOIN PRODUCTO p ON p.productoID = d.productoID
        WHERE d.transaccionID = NEW.transaccionID
        GROUP BY d.transaccionID
    ) x ON x.transaccionID = t.transaccionID
    SET t.valorTotal = x.total;
END $$
```

Al eliminar un detalle:
Vuelve a calcular el total y lo ajusta, incluso con los detalles.

```
CREATE TRIGGER trg_detalles_pxt_ad_recalc
AFTER DELETE ON DETALLES_PXT
FOR EACH ROW
BEGIN
    UPDATE TRANSACCION t
    LEFT JOIN (
        SELECT d.transaccionID,
               SUM(d.cantidad * (p.precio * (1 + (p.porcentajeIVA/100)))) AS total
        FROM DETALLES_PXT d
        JOIN PRODUCTO p ON p.productoID = d.productoID
        WHERE d.transaccionID = OLD.transaccionID
        GROUP BY d.transaccionID
    ) x ON x.transaccionID = t.transaccionID
    SET t.valorTotal = COALESCE(x.total, 0)
    WHERE t.transaccionID = OLD.transaccionID;
END $$
```

Vista

vw_ventas_por_cliente_detalle

Muestra el detalle de productos vendidos por cliente (ideal para reportes de facturación).

```
CREATE VIEW vw_ventas_por_cliente_detalle AS
SELECT  c.cedula            AS clienteCedula,
        CONCAT(c.nombres,' ',c.apellidos) AS cliente,
        t.transaccionID,
        t.fecha,
        t.estado,
        p.productoID,
        p.nombre            AS producto,
        p.categoria,
        d.cantidad,
        p.precio,
        p.porcentajeIVA,
        (d.cantidad * (p.precio * (1 + (p.porcentajeIVA/100)))) AS subtotalLinea,
        t.valorTotal
FROM TRANSACCION t
JOIN CLIENTE c ON c.cedula = t.clienteID
JOIN DETALLES_PXT d ON d.transaccionID = t.transaccionID
JOIN PRODUCTO p ON p.productoID = d.productoID
;
```

vw_pedidos_courier_tracking

Seguimiento de pedidos con datos del courier y cliente.

```sql
CREATE VIEW vw_pedidos_courier_tracking AS
SELECT  pe.pedidoID,
        pe.fechaEvento,
        pe.direccionDestino,
        pe.estado,
        pe.estadoGPS,
        co.courierID,
        CONCAT(co.nombres,' ',co.apellidos) AS courier,
        t.transaccionID,
        t.fecha AS fechaTransaccion,
        t.valorTotal,
        CONCAT(cli.nombres,' ',cli.apellidos) AS cliente
FROM PEDIDO pe
JOIN COURIER co     ON co.courierID = pe.courierID
JOIN TRANSACCION t  ON t.transaccionID = pe.transaccionID
JOIN CLIENTE cli    ON cli.cedula = t.clienteID
;
```

vw_productos_artesano_estado

Catálogo de productos por artesano con estado de revisión y número de fotos.

```sql
CREATE VIEW vw_productos_artesano_estado AS
SELECT  a.ruc AS artesanoRUC,
        CONCAT(a.nombres,' ',a.apellidos) AS artesano,
        p.productoID,
        p.nombre,
        p.categoria,
        p.precio,
        p.stock,
        p.estadoRevision,
        COUNT(f.fotoID) AS fotos
FROM ARTESANO a
JOIN PRODUCTO p ON p.artesanoID = a.ruc
LEFT JOIN FOTO f ON f.productoID = p.productoID
GROUP BY a.ruc, a.nombres, a.apellidos, p.productoID, p.nombre, p.categoria, p.precio, p.stock, p.estadoRevision
;
```

vw_devoluciones_detalle

Registro detallado de devoluciones, incluyendo motivo, soporte y cliente.

```sql
CREATE VIEW vw_devoluciones_detalle AS
SELECT  d.devolucionID,
        d.fechaInicio,
        d.fechaFin,
        d.tipo,
        d.motivo,
        d.comentario,
        p.productoID,
        p.nombre AS producto,
        t.transaccionID,
        t.fecha AS fechaTransaccion,
        CONCAT(c.nombres,' ',c.apellidos) AS cliente,
        s.soporteID,
        s.correoElectronico AS soporteCorreo
FROM DEVOLUCION d
LEFT JOIN PRODUCTO p    ON p.productoID = d.productoID
LEFT JOIN TRANSACCION t ON t.transaccionID = d.transaccionID
LEFT JOIN CLIENTE c     ON c.cedula = d.clienteID
LEFT JOIN SOPORTE s     ON s.soporteID = d.soporteID
;
```

Stored Procedures

Gestión de Transacciones y detalles

sp_transaccion_crear

Crea una transacción vacía.

```sql
CREATE PROCEDURE sp_transaccion_crear(IN p_clienteID CHAR(10), OUT p_transaccionID INT)
BEGIN
    DECLARE v_exists INT;
    START TRANSACTION;
    SELECT COUNT(*) INTO v_exists FROM CLIENTE WHERE cedula = p_clienteID;
    IF v_exists = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cliente no existe';
    END IF;

    INSERT INTO TRANSACCION(valorTotal, estado, fecha, clienteID)
    VALUES (0, 'PENDIENTE', CURDATE(), p_clienteID);

    SET p_transaccionID = LAST_INSERT_ID();
    COMMIT;
END $$
```

sp_detalle_agregar

Agrega productos, valida stock y descuenta inventario.

```
CREATE PROCEDURE sp_detalle_agregar(IN p_transaccionID INT, IN p_productoID INT, IN p_cantidad INT)
BEGIN
    DECLARE v_stock INT;
    DECLARE v_exists INT;

    IF p_cantidad <= 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La cantidad debe ser mayor a 0';
    END IF;

    START TRANSACTION;
    SELECT COUNT(*) INTO v_exists FROM TRANSACCION WHERE transaccionID = p_transaccionID;
    IF v_exists = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Transacción no existe';
    END IF;

    SELECT stock INTO v_stock FROM PRODUCTO WHERE productoID = p_productoID FOR UPDATE;
    IF v_stock IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Producto no existe';
    END IF;
    IF v_stock < p_cantidad THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Stock insuficiente';
    END IF;

    INSERT INTO DETALLES_PXT(productoID, transaccionID, cantidad, subtotal)
    SELECT p_productoID, p_transaccionID, p_cantidad, (p.precio * (1 + (p.porcentajeIVA/100))) * p_cantidad
    FROM PRODUCTO p WHERE p.productoID = p_productoID;

    UPDATE PRODUCTO SET stock = stock - p_cantidad WHERE productoID = p_productoID;

    COMMIT;
END $$
```

sp_detalle_eliminar

Elimina productos del detalle y restaura stock.

```
CREATE PROCEDURE sp_detalle_eliminar(IN p_transaccionID INT, IN p_productoID INT)
BEGIN
    DECLARE v_cantidad INT;
    START TRANSACTION;
    SELECT cantidad INTO v_cantidad FROM DETALLES_PXT
    WHERE transaccionID = p_transaccionID AND productoID = p_productoID
    FOR UPDATE;

    IF v_cantidad IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Detalle no existe';
    END IF;

    DELETE FROM DETALLES_PXT
    WHERE transaccionID = p_transaccionID AND productoID = p_productoID;

    UPDATE PRODUCTO SET stock = stock + v_cantidad WHERE productoID = p_productoID;
    COMMIT;
END $$
```

CRUD de productos con validaciones

sp_producto_insert

Inserta nuevos productos en la tabla PRODUCTO

```
CREATE PROCEDURE sp_producto_insert(
    IN p_nombre VARCHAR(30),
    IN p_descripcion CHAR(60),
    IN p_categoria VARCHAR(50),
    IN p_precio FLOAT,
    IN p_porcentajeIVA FLOAT,
    IN p_stock INT,
    IN p_artesanoID VARCHAR(13),
    IN p_estadoRevision VARCHAR(20),
    OUT p_productoID INT
)
BEGIN
    IF p_precio < 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Precio inválido'; END IF;
    IF p_porcentajeIVA < 0 OR p_porcentajeIVA > 100 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='IVA inválido'; END IF;
    IF p_stock < 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Stock inválido'; END IF;

    START TRANSACTION;
    INSERT INTO PRODUCTO(nombre, descripcion, categoria, precio, porcentajeIVA, stock, artesanoID, estadoRevision)
    VALUES (p_nombre, p_descripcion, p_categoria, p_precio, p_porcentajeIVA, p_stock, p_artesanoID, COALESCE(p_estadoRevision,'PENDIENTE'));
    SET p_productoID = LAST_INSERT_ID();
    COMMIT;
END $$
```

sp_producto_update

Actualiza productos de la tabla PRODUCTO

```
CREATE PROCEDURE sp_producto_update(
    IN p_productoID INT,
    IN p_nombre VARCHAR(30),
    IN p_descripcion CHAR(60),
    IN p_categoria VARCHAR(50),
    IN p_precio FLOAT,
    IN p_porcentajeIVA FLOAT,
    IN p_stock INT,
    IN p_estadoRevision VARCHAR(20)
)
BEGIN
    IF p_precio < 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Precio inválido'; END IF;
    IF p_porcentajeIVA < 0 OR p_porcentajeIVA > 100 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='IVA inválido'; END IF;
    IF p_stock < 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Stock inválido'; END IF;

    START TRANSACTION;
    UPDATE PRODUCTO
    SET nombre=p_nombre,
        descripcion=p_descripcion,
        categoria=p_categoria,
        precio=p_precio,
        porcentajeIVA=p_porcentajeIVA,
        stock=p_stock,
        estadoRevision=p_estadoRevision
    WHERE productoID = p_productoID;
    IF ROW_COUNT() = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Producto no existe';
    END IF;
    COMMIT;
END $$
```

sp_producto_delete

Elimina productos de la tabla PRODUCTO

```
CREATE PROCEDURE sp_producto_delete(IN p_productoID INT)
BEGIN
    START TRANSACTION;
    DELETE FROM PRODUCTO WHERE productoID = p_productoID;
    IF ROW_COUNT() = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Producto no existe';
    END IF;
    COMMIT;
END $$
```

Gestión de pedidos

sp_pedido_crear

Crea pedidos vinculados a courier y cliente.

```
CREATE PROCEDURE sp_pedido_crear(
    IN p_transaccionID INT,
    IN p_courierID INT,
    IN p_direccionDestino VARCHAR(100),
    IN p_fechaEvento DATE,
    OUT p_pedidoID INT
)
BEGIN
    DECLARE v_exists INT;
    START TRANSACTION;
    SELECT COUNT(*) INTO v_exists FROM TRANSACCION WHERE transaccionID = p_transaccionID;
    IF v_exists = 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Transacción inválida'; END IF;

    SELECT COUNT(*) INTO v_exists FROM COURIER WHERE courierID = p_courierID;
    IF v_exists = 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Courier inválido'; END IF;

    INSERT INTO PEDIDO(fechaEvento, direccionDestino, estado, estadoGPS, transaccionID, courierID)
    VALUES (p_fechaEvento, p_direccionDestino, 'CREADO', 'PENDIENTE', p_transaccionID, p_courierID);

    SET p_pedidoID = LAST_INSERT_ID();
    COMMIT;
END $$
```

Indices

Se añadieron 5 indices nuevos para acelerar búsquedas frecuentes:

- Búsquedas por artesano (**idx_producto_artesanoID**).
- Catálogos por categoría (**idx_producto_categoria**).
- Consultas de transacciones por cliente y fecha (**idx_transaccion_cliente_fecha**).
- Seguimiento de pedidos por courier (**idx_pedido_courierID**).
- Devoluciones por producto (**idx_devolucion_productoID**).

```
CREATE INDEX IF NOT EXISTS idx_producto_artesanoID ON PRODUCTO(artesanoID) $$
CREATE INDEX IF NOT EXISTS idx_producto_categoria ON PRODUCTO(categoria) $$
CREATE INDEX IF NOT EXISTS idx_transaccion_cliente_fecha ON TRANSACCION(clienteID, fecha) $$
CREATE INDEX IF NOT EXISTS idx_pedido_courierID ON PEDIDO(courierID) $$
CREATE INDEX IF NOT EXISTS idx_devolucion_productoID ON DEVOLUCION(productoID) $$
```

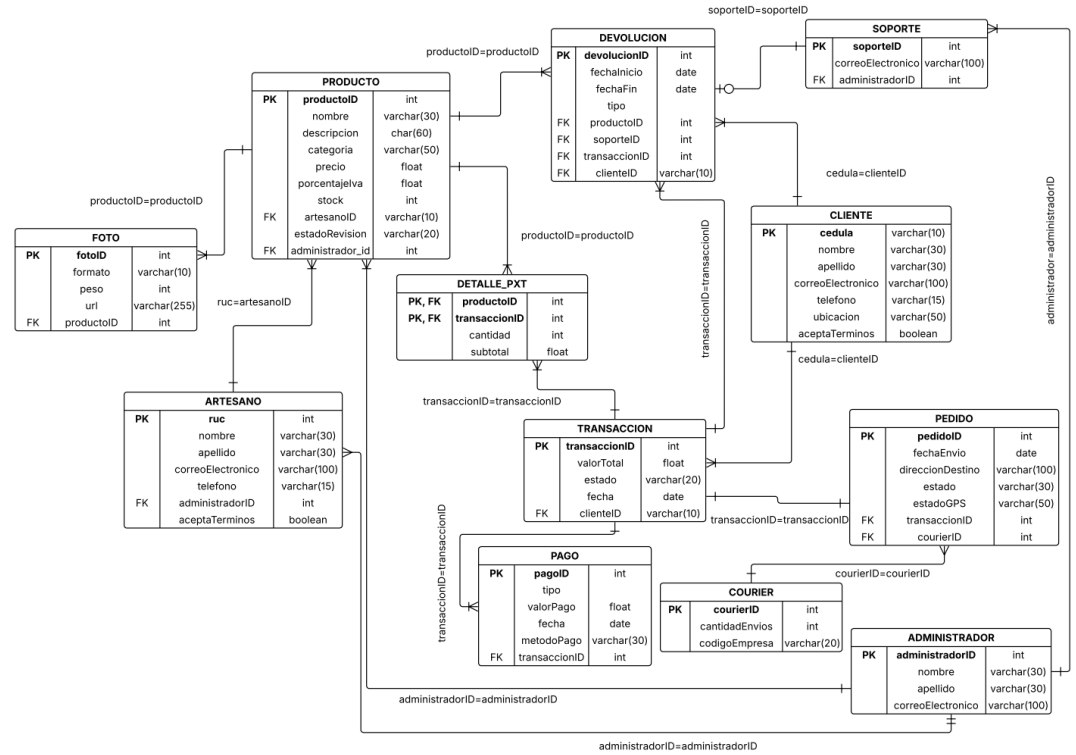
Usuarios y Permisos

USUARIO	PERMISOS
app_writer	Ejecutar SP de transacciones, detalles y pedidos.

report_viewer	SELECT en vistas de ventas y pedidos.
admin_ops	Administrador con todos los privilegios (localhost).
soporte_user	SELECT en vista de devoluciones.
artesano_user	Ejecutar SP de productos y ver catálogo por artesano.

```
-- (Ajuste las contraseñas en su entorno)
DROP USER IF EXISTS 'app_writer'@'%' $$
CREATE USER 'app_writer'@'%' IDENTIFIED BY 'PwdApp!2025' $$

DROP USER IF EXISTS 'report_viewer'@'%' $$
CREATE USER 'report_viewer'@'%' IDENTIFIED BY 'PwdReport!2025' $$

DROP USER IF EXISTS 'admin_ops'@'localhost' $$
CREATE USER 'admin_ops'@'localhost' IDENTIFIED BY 'PwdAdmin!2025' $$

DROP USER IF EXISTS 'soporte_user'@'%' $$
CREATE USER 'soporte_user'@'%' IDENTIFIED BY 'PwdSoporte!2025' $$

DROP USER IF EXISTS 'artesano_user'@'%' $$
CREATE USER 'artesano_user'@'%' IDENTIFIED BY 'PwdArtesano!2025' $$

-- Permisos a procedimientos (al menos 1)
GRANT EXECUTE ON PROCEDURE proyectob.sp_producto_insert  TO 'artesano_user'@'%' $$
GRANT EXECUTE ON PROCEDURE proyectob.sp_producto_update  TO 'artesano_user'@'%' $$
GRANT EXECUTE ON PROCEDURE proyectob.sp_transaccion_crear TO 'app_writer'@'%' $$
GRANT EXECUTE ON PROCEDURE proyectob.sp_detalle_agregar   TO 'app_writer'@'%' $$
GRANT EXECUTE ON PROCEDURE proyectob.sp_detalle_eliminar  TO 'app_writer'@'%' $$
GRANT EXECUTE ON PROCEDURE proyectob.sp_pedido_crear      TO 'app_writer'@'%' $$

-- Permisos a vistas (al menos 2)
GRANT SELECT ON proyectob.vw_ventas_por_cliente_detalle  TO 'report_viewer'@'%' $$
GRANT SELECT ON proyectob.vw_pedidos_courier_tracking    TO 'report_viewer'@'%' $$
GRANT SELECT ON proyectob.vw_devoluciones_detalle        TO 'soporte_user'@'%' $$
GRANT SELECT ON proyectob.vw_productos_artesano_estado   TO 'artesano_user'@'%' $$

-- Permisos administrativos
GRANT ALL PRIVILEGES ON proyectob.* TO 'admin_ops'@'localhost' WITH GRANT OPTION $$
```

Diagrama de la Base de Datos



Manual de Usuario

Artesano:

- Registra sus productos con sp_producto_insert.
- Consulta su catálogo con la vista vw_productos_artesano_estado.

Aplicación (app_writer):

- Crea transacciones con sp_transaccion_crear.
- Agrega o elimina productos de la compra.
- Crea pedidos asignando couriers.

Report Viewer:

- Consulta reportes de ventas (vw_ventas_por_cliente_detalle) y pedidos (vw_pedidos_courier_tracking).

Soporte:

- Revisa devoluciones con vw_devoluciones_detalle.

Admin:

- Gestiona toda la base de datos y usuarios.

Link de Github

https://github.com/eriamuri/BaseDatosArteSanos.git