



INFORME TÉCNICO DEL PROYECTO DE BASE DE DATOS: ROBOBO

DALFREDO JESUS MELO GUADALUPE - ALUMNO CEAC FP
[NOMBRE DE LA EMPRESA]
[Dirección de la compañía]




Tabla de contenido

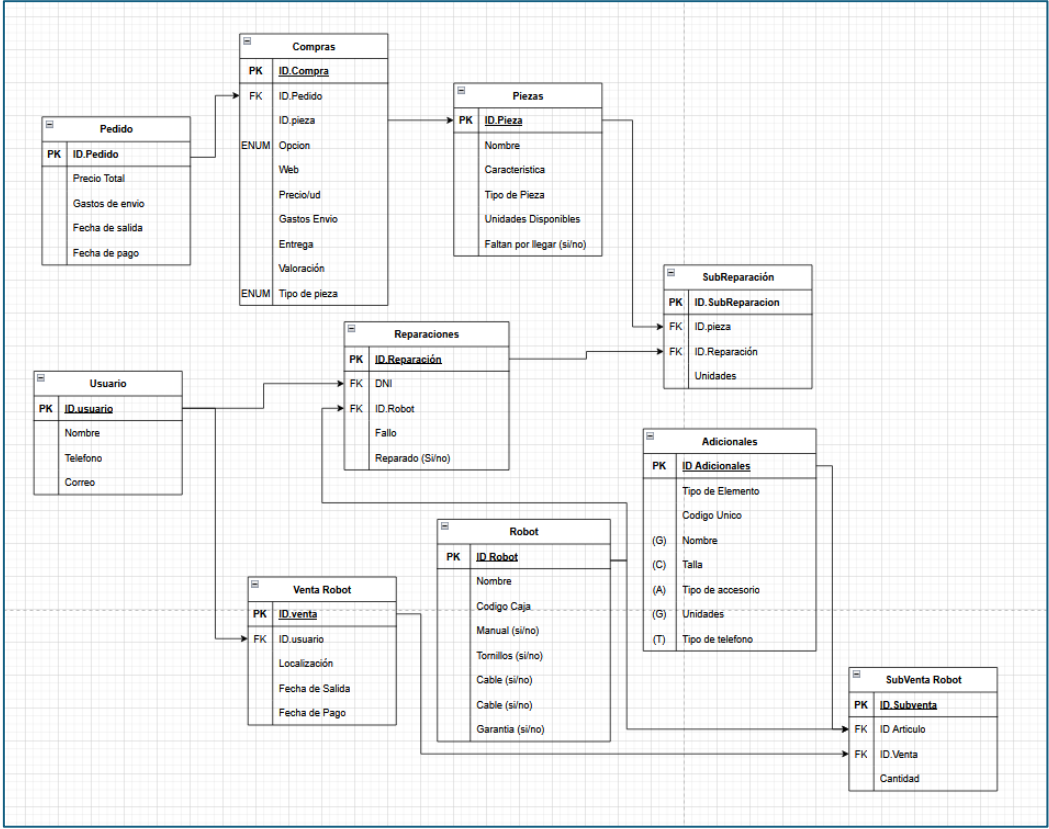
Introducción.....	2
Modelo relacional	2
Tablas	3
3.1 Pedido	3
3.2 Compras.....	3
3.3 Piezas	3
3.4 Robot.....	4
3.5 Adicionales	4
3.6 Usuarios	4
3.7 Reparaciones.....	5
3.8 SubReparacion	5
3.9 Venta_Robot	5
3.10 SubVenta_Robot	5
Automatización y Triggers de Base de Datos	6
4.1 Trigger: insertar_o_sumar_en_piezas	6
4.2 Trigger: validar_stock_reparacion	7
4.3 Trigger: restar_unidades_pieza_reparacion	7
4.4 Trigger: after_insert_subventa_robot	8
Como insertar datos	9
5.1 Registrar un nuevo pedido	9
5.2 Agregar productos al pedido	9
5.3 Actualizar información de las piezas	9
5.4 Registrar robots en inventario	10
5.5 Agregar accesorios y camisetas.....	10
5.6 Registrar un usuario (cliente)	10
5.7 Registrar una reparación	11
5.8 Decir qué piezas se usan en esa reparación	11
5.9 Registrar una venta	11
5.10 Asociar productos a esa venta	12
5.11 Actualizar la ubicación y fecha del pago	12

Introducción

Este informe documenta la estructura y funcionalidad de la base de datos *ROBOBO*, diseñada para gestionar las operaciones de compras, ventas, inventario y reparaciones de productos robóticos y sus componentes adicionales. El sistema tiene como objetivo ofrecer trazabilidad, automatización de procesos y control de stock.

Modelo relacional

A continuación se va a presentar el ultimo diseño del modelo relacional que se uso como base para la elaboración de la base de datos



Tablas

3.1 Pedido

- Función: Almacena información de los pedidos realizados.
- Campos principales:
 - ID_Pedido (PK),
 - Precio_Total,
 - Gastos_envio,
 - Fecha_de_Salida,
 - Fecha_de_pago.

3.2 Compras

- Función: Registra las piezas adquiridas para cada pedido.
- Campos principales:
 - ID_Compra (PK),
 - ID_Pieza,
 - Opcion,
 - Web,
 - Unidades,
 - Precio_ud,
 - Gastos_Envio,
 - Entrega,
 - Valoración,
 - tipo_de_pieza.

3.3 Piezas

- Función: Mantiene el inventario de piezas disponibles.
- Campos principales:
 - ID_Pieza (PK),
 - Nombre,
 - Caracteristica,
 - Tipo_de_Pieza,
 - Unidades_Disponibles.

3.4 Robot

- Función: Catálogo de robots ensamblados para venta.
- Campos principales:
 - ID_Robot (PK),
 - Nombre,
 - Codigo_Caja,
 - Manual, Tornillos, Cable, Garantía (todos tipo ENUM 'SI'/'NO'),
 - Vendido.

3.5 Adicionales

- Función: camisetas y accesorios no robóticos.
- Campos principales:
 - ID_Adicionales (PK),
 - Tipo_Elemento,
 - Nombre,
 - Talla,
 - Unidades.

3.6 Usuarios

- Función: Contiene información personal de los clientes.
- Campos principales:
 - DNI (PK),
 - Nombre,
 - Apellidos,
 - Teléfono,
 - Correo.

3.7 Reparaciones

- Función: Registra reparaciones solicitadas por usuarios.
- Campos principales:
 - ID_Reparacion (PK),
 - ID_Robot,
 - DNI,
 - Fallo,
 - Reparado.

3.8 SubReparacion

- Función: Detalla qué y cuantas piezas se utilizaron en cada reparación.
- Campos principales:
 - ID_SubReparacion (PK),
 - ID_Pieza,
 - Unidades,
 - ID_Reparacion.

3.9 Venta_Robot

- Función: Encabezado de ventas realizadas a usuarios.
- Campos principales:
 - ID_venta (PK),
 - DNI,
 - Localización,
 - Fecha_de_Salida,
 - Fecha_de_pago.

3.10 SubVenta_Robot

- **Función:** Registra los productos incluidos en cada venta.
- **Campos principales:**
 - ID_Subventa (PK),
 - ID_Articulo,
 - ID_venta,
 - Cantidad.

Automatización y Triggers de Base de Datos

4.1 Trigger: insertar_o_sumar_en_piezas

- Evento: AFTER INSERT ON Compras
- Propósito:
Automatiza la gestión de inventario cuando se registra una compra de piezas.
- Acción ejecutada:
 - Si la pieza no existe, se inserta.
 - Si ya existe, se actualiza sumando las unidades compradas.

```
CREATE TRIGGER insertar_o_sumar_en_piezas
AFTER INSERT ON Compras
FOR EACH ROW
BEGIN
    INSERT INTO Piezas (
        ID_Pieza,
        Tipo_de_Pieza,
        Nombre,
        Caracteristica,
        Unidades_Disponibles
    ) VALUES (
        NEW.ID_Pieza,
        NEW.tipo_de_pieza,
        NULL,
        NULL,
        NEW.Unidades
    )
    ON DUPLICATE KEY UPDATE
        Unidades_Disponibles = Unidades_Disponibles + NEW.Unidades;
END$$

DELIMITER ;
```

4.2 Trigger: validar_stock_reparacion

- Evento: BEFORE INSERT ON SubReparacion
- Propósito:
Validar que haya stock suficiente antes de registrar una pieza usada en reparación.
- Acción ejecutada:
Lanza un error si las unidades solicitadas superan las disponibles.

```
CREATE TRIGGER validar_stock_reparacion
BEFORE INSERT ON SubReparacion
FOR EACH ROW
BEGIN
    DECLARE disponibles INT;

    SELECT Unidades_Disponibles INTO disponibles
    FROM Piezas
    WHERE ID_Pieza = NEW.ID_Pieza;

    IF disponibles < NEW.Unidades THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No hay suficientes unidades en stock';
    END IF;
END$$
```

4.3 Trigger: restar_unidades_pieza_reparacion

- Evento: AFTER INSERT ON SubReparacion
- Propósito:
Disminuir automáticamente el inventario de piezas utilizadas en reparaciones.
- Acción ejecutada:
Resta las unidades indicadas del campo Unidades_Disponibles.

```
CREATE TRIGGER restar_unidades_pieza_reparacion
AFTER INSERT ON SubReparacion
FOR EACH ROW
BEGIN
    UPDATE Piezas
    SET Unidades_Disponibles = Unidades_Disponibles - NEW.Unidades
    WHERE ID_Pieza = NEW.ID_Pieza;
END$$
```


4.4 Trigger: after_insert_subventa_robot

- Evento: AFTER INSERT ON SubVenta_Robot
- Propósito:
Automatizar la gestión de productos vendidos (robots o adicionales).
- Acción ejecutada:
 - Si el artículo es un robot → se marca como vendido.
 - Si es un adicional → se descuentan las unidades.

```
CREATE TRIGGER after_insert_subventa_robot
AFTER INSERT ON SubVenta_Robot
FOR EACH ROW
BEGIN
    -- Si el ID_Articulo está en la tabla Robot: actualizar Vendido
    IF EXISTS (SELECT 1 FROM Robot WHERE ID_Robot = NEW.ID_Articulo) THEN
        UPDATE Robot
        SET Vendido = 'SI'
        WHERE ID_Robot = NEW.ID_Articulo;

    -- Si el ID_Articulo está en la tabla Adicionales: disminuir Unidades
    ELSEIF EXISTS (SELECT 1 FROM Adicionales WHERE ID_Adicionales = NEW.ID_Articulo) THEN
        UPDATE Adicionales
        SET Unidades = Unidades - NEW.Cantidad
        WHERE ID_Adicionales = NEW.ID_Articulo;
    END IF;
END;
```

Como insertar datos

5.1 Registrar un nuevo pedido

```
INSERT INTO Pedido (Precio_Total, Gastos_envio) VALUES (0, 0);  
SET @id_pedido := LAST_INSERT_ID();
```

¿Qué es esto?

- Crea un pedido nuevo (como un carrito de compras).
- Guarda el ID de ese pedido en una variable (@id_pedido) para poder usarlo en los pasos siguientes.

5.2 Agregar productos al pedido

```
INSERT INTO compras()  
values ('1',@id_pedido, 'RB0201','1','https://de.screwex.com/es/shop/detail/stp/STP22A0220080S.html',  
'37','0.102','0','2025-06-14','4','1');
```

¿Qué está pasando aquí?

- Se agregan productos (piezas) a ese pedido.
- Cada producto incluye:
 - ID de pieza (por ejemplo 'RB0201')
 - Cantidad, precio, página web de compra, etc.
- Al insertar, el sistema automáticamente suma esas piezas al inventario (esto lo hace el sistema por detrás con un trigger).

5.3 Actualizar información de las piezas

```
UPDATE Piezas  
SET Nombre = 'tornillo plástico',  
    Caracteristica = 'tornillo plastico, 2,2 x 8mm'  
WHERE id_pieza = 'RB0201';
```

¿Para qué?

- Se le asigna un nombre y descripción más clara a cada pieza (esto puede venir de un formulario en el frontend).

5.4 Registrar robots en inventario

```
INSERT INTO Robot ()
values
('NS01120917GCS0004', 'ROB-WMP', 'RBEX0004', 'NO', 'NO', 'NO', 'NO', 'NO'),
('NS01120917GCS0007', 'ROB-0HG', 'RBEX0007', 'NO', 'NO', 'NO', 'NO', 'SI'),
('NS01120917GCS0008', 'ROB-WHS', 'RBEX0008', 'NO', 'NO', 'NO', 'NO', 'SI'),
```

¿Qué es esto?

- Agregas robots al inventario, cada uno con su ID, nombre, código de caja y un estado de qué tiene (manual, tornillos, etc.).
- También se indica si ya fue **vendido** o no.

5.5 Agregar accesorios y camisetas

```
INSERT INTO Adicionales(ID_Adicionales, Tipo_elemento, Nombre, talla, Unidades)
values('C1','camiseta','Camisa XL','XL','4'),
```

```
INSERT INTO Adicionales(ID_Adicionales, Tipo_Elemento, Nombre, Unidades)
values('A1','accesorio','Pusher','12'),
```

¿Qué incluye?

- Elementos adicionales como camisetas o piezas tipo LEGO.
- Pueden tener talla (si es ropa) y una cantidad.
- el primer insert es para camisetas y el segundo funciona para accesorios

5.6 Registrar un usuario (cliente)

```
INSERT INTO Usuarios()
VALUES('12345678R','Armando','Paredes','123 45 67 89','Armando@gmail.com');
```

¿Qué representa?

- Un usuario que puede comprar o solicitar reparaciones.
- Incluye DNI, nombre, apellido, teléfono, email.

5.7 Registrar una reparación

```
insert into Reparaciones()  
values ('1', 'NS01120917GCS0004', '12345678R', 'Faltan Tornillos', '1');
```

¿Qué significa?

- El usuario reporta un fallo en un robot.
- Por ejemplo: “Faltan tornillos”.

5.8 Decir qué piezas se usan en esa reparación

```
INSERT INTO Subreparacion()  
values ('1', 'RB0201', '10', '1');
```

¿Qué pasa acá?

- Se registran las piezas utilizadas en esa reparación.
- El sistema automáticamente:
 - Verifica si hay suficientes piezas disponibles.
 - Si hay, las descuenta del inventario.

5.9 Registrar una venta

```
INSERT INTO Venta_robot(DNI)  
values('12345678R');  
  
SET @id_venta := LAST_INSERT_ID();
```

¿Qué ocurre?

- Se registra que una persona compró un robot o accesorio.
- Se guarda el ID de la venta para usarlo después.

5.10 Asociar productos a esa venta

```
INSERT INTO SubVenta_Robot ()  
VALUES ('1','NS01120917GCS0004',@id_venta,'1'),  
('2','C1',@id_venta,'2'),  
('3','A1',@id_venta,'3');
```

¿Qué está pasando?

- Se añaden a la venta:
 - Robots
 - Camisetas (ej. C1)
 - Accesorios (ej. A1)
- El sistema:
 - Marca el robot como vendido.
 - Resta unidades del accesorio vendido.

5.11 Actualizar la ubicación y fecha del pago

```
UPDATE Venta_Robot  
SET Localización = 'Madrid',  
    Fecha_de_pago = CURDATE()  
WHERE ID_venta = 1;
```

¿Qué hace esto?

- Añade información de dónde se hizo la venta y en qué fecha se pagó.