# Proyecto final - Curso SQL

## Documento del proyecto final “tienda de componentes de pc”

Alumno: Gonzalo Naim Alonso Picco

Comisión: 43420

Profesor: Gabriel Almiñana

Tutor: Santiago Angel

# Índice

# Introducción

En este documento, describiré la creación de una base de datos relacional enfocada en la gestión de componentes de computadoras. Exploraremos desde la estructuración de tablas para usuarios, productos y categorías hasta la implementación de relaciones y restricciones.

Esta base de datos busca resolver desafíos específicos en el contexto de componentes de PC. El documento ofrecerá un resumen completo del proceso, desde la problemática hasta las soluciones implementadas, reflejando la aplicación de conceptos aprendidos en el curso.

# Objetivo

El objetivo principal de esta base de datos es brindar una plataforma eficiente y organizada para gestionar los detalles esenciales de los componentes de PC, como procesadores, tarjetas gráficas, memorias RAM, gabinetes, etc. A través de esta base de datos, se pretende lograr una administración simplificada de los datos, desde la información del usuario hasta los detalles de los productos y las transacciones de compra.

# Situación problemática

La falta de una base de datos estructurada puede llevar a errores en la administración de inventario, la pérdida de datos de transacciones y la incapacidad para ofrecer a los usuarios recomendaciones personalizadas y opciones relevantes. Esta situación problemática subraya la necesidad de una solución que centralice y organice la información relacionada con los componentes de PC, permitiendo una toma de decisiones más informada, una gestión efectiva de inventario y una experiencia de compra mejorada para los usuarios.

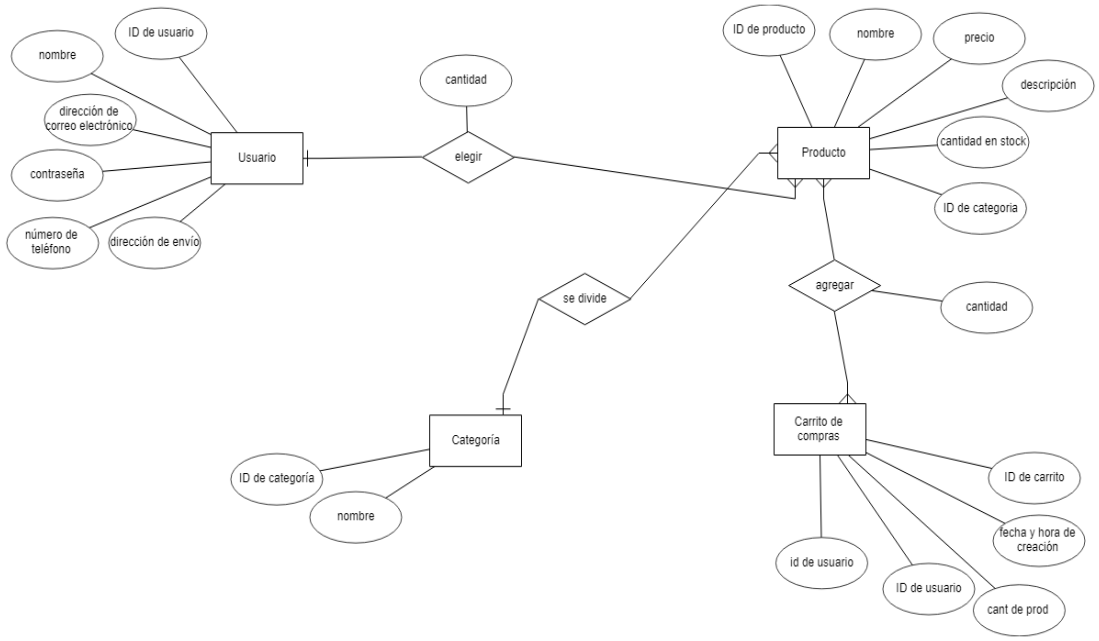
En resumen, la base de datos de componentes de PC busca abordar esta situación problemática al proporcionar una herramienta eficiente y estructurada para la gestión integral de los detalles de los productos, las transacciones de compra y las preferencias del usuario en el contexto de la tecnología de la información.

# Modelo de negocio

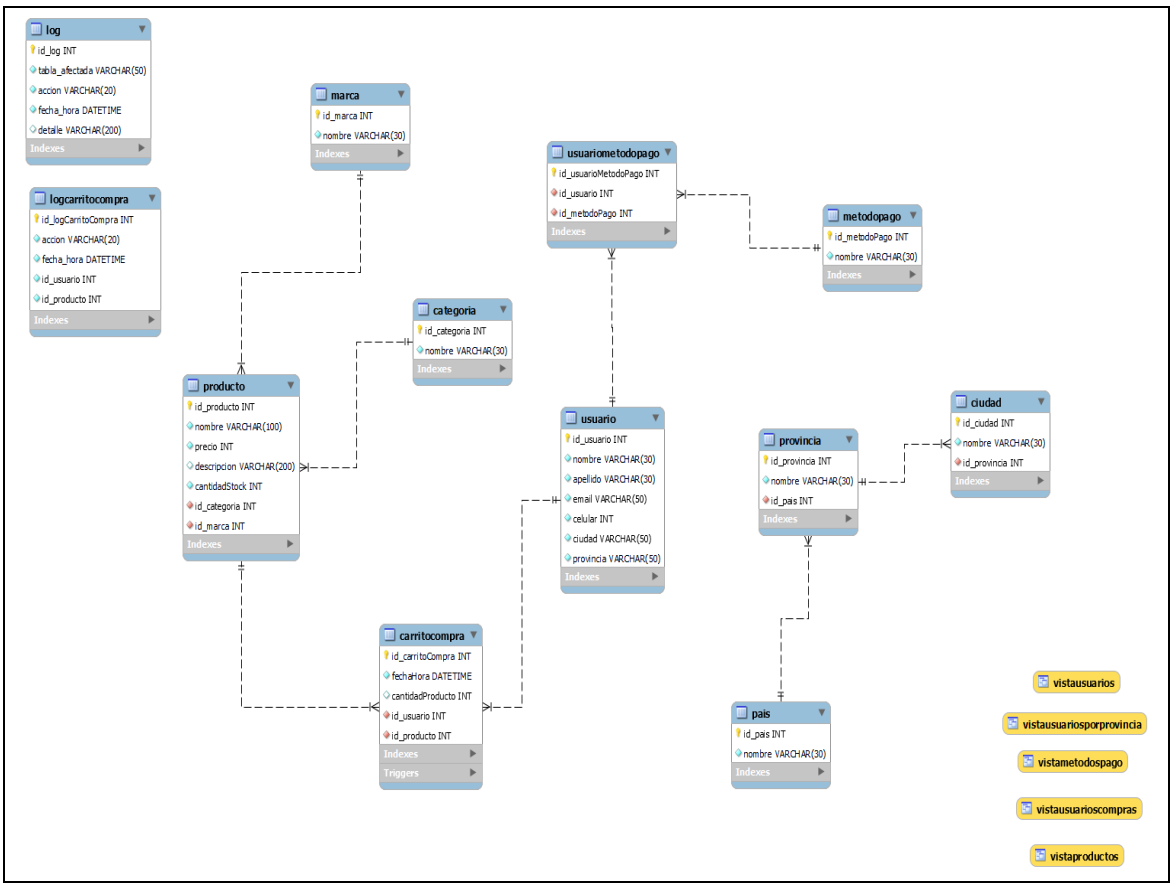
La empresa se dedica a la venta y gestión de componentes de computadoras. Ofrece una amplia gama de productos, incluyendo procesadores, tarjetas gráficas, memorias RAM, gabinetes y placas madre, entre otros. El enfoque principal es brindar a los usuarios la posibilidad de armar sus propias computadoras personalizadas con componentes de alta calidad y rendimiento.

---

# Primer Diagrama Entidad – Relación



# Diagrama Entidad – Relación (final) de MySQL Workbench



# Descripción de las tablas

| tabla | campo abreviado | nombre del campo completo | clave primaria (PK) | clave foranea (FK) | tipo de datos |
|---|---|---|---|---|---|
| **usuario** | **id_usuario** | clave primaria del usuario | PK | | INT |
| | nombre | nombre del usuario | | | VARCHAR |
| | apellido | apellido del usuario | | | VARCHAR |
| | email | email del usuario | | | VARCHAR |
| | celular | celular del usuario | | | INT |
| | ciudad | ciudad del usuario | | | VARCHAR |
| | provincia | provincia del usuario | | | VARCHAR |
| **producto** | **id_producto** | clave primaria del producto | PK | | INT |
| | nombre | nombre del producto | | | VARCHAR |
| | precio | precio del producto | | | INT |
| | descripción | descripción del producto | | | VARCHAR |
| | cant de stock | cantidad del stock | | | INT |
| | id_categoría | clave foranea de la categoría | | FK | INT |
| | id_marca | clave foranea de la marca | | FK | INT |
| **categoría** | **id_categoría** | clave primaria de la categoría | PK | | INT |
| | nombre | nombre de la categoría | | | VARCHAR |
| **carrito de compra** | **id_carritoCompra** | clave primaria de carrito de compra | PK | | INT |
| | fecha y hora | fecha y hora de la compra | | | DATETIME |
| | cant de prod | cantidad de productos | | | INT |
| | id_usuario | clave foranea del usuario | | FK | INT |
| | id_producto | clave foranea del producto | | FK | INT |
| **marca** | **id_marca** | clave primaria de la marca | PK | | INT |
| | nombre | nombre de la marca | | | VARCHAR |
| **usuario metodo de pago** | **id_usuarioMetodoPago** | clave primaria del usuario metodo de pago | PK | | INT |
| | id_usuario | clave foranea del usuario | | FK | INT |
| | id_metodoPago | clave foranea del metodo de pago | | FK | INT |
| **metodoPago** | **id_metodoPago** | clave primaria del metodo de pago | PK | | INT |
| | nombre | nombre del metodo de pago | | | VARCHAR |
| **pais** | **id_pais** | clave primaria del pais | PK | | INT |
| | nombre | nombre del pais | | | VARCHAR |
| **provincia** | **id_provincia** | clave primaria de la provincia | PK | | INT |
| | nombre | nombre de la provincia | | | VARCHAR |
| | id_pais | clave foranea del pais | | FK | INT |
| **ciudad** | **id_ciudad** | clave primaria de la ciudad | PK | | INT |
| | nombre | nombre de la ciudad | | | VARCHAR |
| | id_provincia | clave foranea de la provincia | | FK | INT |
| **log** | **id_log** | clave primaria de log | PK | | INT |
| | tabla_afectada | tabla donde se realizo un after un usuario | | | VARCHAR |
| | accion | que tipo de after se realizo | | | VARCHAR |
| | fecha_hora | hora donde se realizo un after | | | DATETIME |
| | detalle | a que usuartio se le realizo el after | | | VARCHAR |
| **logCarritoCompra** | **id_logCarritoCompra** | clave primaria de logCarritoCompra | PK | | INT |
| | accion | que tipo de after se realizo | | | VARCHAR |
| | fecha_hora | hora donde se realizo un after | | | DATETIME |
| | id_usuario | clave foranea de usuario | | FK | INT |
| | id_producto | clave foranea de producto | | FK | INT |

# Vistas

**Vista 1** - Vista de Usuarios con Ciudad y Provincia

Esta vista mostrará información detallada de los usuarios, incluyendo la ciudad y la provincia en la que se encuentran.

```
CREATE VIEW vistaUsuarios AS
SELECT u.id_usuario, u.nombre, u.apellido, u.email, u.celular, u.ciudad, u.provincia, p.nombre AS pais
FROM usuario u
JOIN provincia pr ON u.provincia = pr.nombre
JOIN pais p ON pr.id_pais = p.id_pais;
```

**Vista 2** - Vista de Productos con Categoría y Marca

Esta vista mostrará información detallada de los productos, incluyendo la categoría y la marca a la que pertenecen.

```
CREATE VIEW vistaProductos AS
SELECT p.id_producto, p.nombre, p.precio, p.descripcion, p.cantidadStock, c.nombre AS categoria, m.nombre AS marca
FROM producto p
JOIN categoria c ON p.id_categoria = c.id_categoria
JOIN marca m ON p.id_marca = m.id_marca;
```

**Vista 3** - Vista de Usuarios y sus Compras

Esta vista mostrará los detalles de los usuarios junto con la información de las compras que han realizado.

```
CREATE VIEW vistaUsuariosCompras AS
SELECT u.id_usuario, u.nombre AS nombre_usuario, u.apellido, u.email,
       c.id_carritoCompra, c.fechaHora, c.cantidadProducto,
       p.nombre AS nombre_producto, p.precio
FROM usuario u
JOIN carritoCompra c ON u.id_usuario = c.id_usuario
JOIN producto p ON c.id_producto = p.id_producto;
```

**Vista 4** - Vista de Métodos de Pago por Usuario

Esta vista mostrará los métodos de pago utilizados por cada usuario.

```
CREATE VIEW vistaMetodosPago AS
SELECT ump.id_usuarioMetodoPago, u.nombre AS usuario, mp.nombre AS metodo_pago
FROM usuarioMetodoPago ump
JOIN usuario u ON ump.id_usuario = u.id_usuario
JOIN metodoPago mp ON ump.id_metodoPago = mp.id_metodoPago;
```

**Vista 5** - Vista de Usuarios por Provincia

Esta vista mostrará la cantidad de usuarios por provincia.

```
CREATE VIEW vistaUsuariosPorProvincia AS
SELECT pr.nombre AS provincia, COUNT(u.id_usuario) AS cantidad_usuarios
FROM usuario u
JOIN provincia pr ON u.provincia = pr.nombre
GROUP BY pr.nombre;
```

---

# Stored Procedures

**Stored Procedures 1** – mostrarProductosPorCategoria

se creó para proporcionar una forma reutilizable y eficiente de obtener una lista de productos que pertenecen a una categoría específica en la base de datos.

```
DELIMITER //

CREATE PROCEDURE mostrarProductosPorCategoria(IN nombre_categoria VARCHAR(30))
BEGIN
    SELECT p.nombre, p.precio, p.descripcion, p.cantidadStock, c.nombre AS categoria
    FROM producto p
    INNER JOIN categoria c ON p.id_categoria = c.id_categoria
    WHERE c.nombre = nombre_categoria;
END;

//

DELIMITER ;
```

**Stored Procedures 2** – mostrarDetalleCompra

fue creado con el objetivo de mostrar el detalle de una compra específica realizada en la base de datos.

```
DELIMITER //

CREATE PROCEDURE mostrarDetalleCompra(IN id_carritoCompra INT)
BEGIN
    SELECT u.nombre AS nombre_usuario, u.email, u.celular, u.ciudad, u.provincia,
    p.nombre AS nombre_producto, p.precio, cc.cantidadProducto,
           cc.fechaHora
    FROM carritoCompra cc
    INNER JOIN usuario u ON cc.id_usuario = u.id_usuario
    INNER JOIN producto p ON cc.id_producto = p.id_producto
    WHERE cc.id_carritoCompra = id_carritoCompra;
END;
//
DELIMITER ;
```

---

# Trigger

**Trigger 1** - after_insert_usuario

Creamos un Trigger AFTER INSERT en la tabla "usuario" para registrar las inserciones en el log.

```
DELIMITER //
CREATE TRIGGER after_insert_usuario
AFTER INSERT ON usuario
FOR EACH ROW
BEGIN
  INSERT INTO log (tabla_afectada, accion, fecha_hora, detalle)
  VALUES ('usuario', 'INSERT', NOW(), CONCAT('ID de usuario insertado: ', NEW.id_usuario));
END;
//
DELIMITER ;
```

**Trigger 2** - after_update_usuario

Creamos un Trigger AFTER UPDATE en la tabla "usuario" para registrar las actualizaciones en el log.

```
DELIMITER //
CREATE TRIGGER after_update_usuario
AFTER UPDATE ON usuario
FOR EACH ROW
BEGIN
  INSERT INTO log (tabla_afectada, accion, fecha_hora, detalle)
  VALUES ('usuario', 'UPDATE', NOW(), CONCAT('ID de usuario actualizado: ', NEW.id_usuario));
END;
//
DELIMITER ;
```

**Trigger 3** - after_delete_usuario

Creamos un Trigger AFTER DELETE en la tabla "usuario" para registrar las eliminaciones en el log.

```
DELIMITER //
CREATE TRIGGER after_delete_usuario
AFTER DELETE ON usuario
FOR EACH ROW
BEGIN
  INSERT INTO log (tabla_afectada, accion, fecha_hora, detalle)
  VALUES ('usuario', 'DELETE', NOW(), CONCAT('ID de usuario eliminado: ', OLD.id_usuario));
END;
//
DELIMITER ;
```

**Trigger 4** - after_insert_carritoCompra

Creamos un Trigger AFTER INSERT en la tabla "carritoCompra" para registrar las inserciones en el log.

```
DELIMITER //
CREATE TRIGGER after_insert_carritoCompra
AFTER INSERT ON carritoCompra
FOR EACH ROW
BEGIN
    INSERT INTO logCarritoCompra (accion, fechaHora, id_usuario, id_producto)
    VALUES ('Inserción', NOW(), NEW.id_usuario, NEW.id_producto);
END;
//
DELIMITER ;
```

# Funciones

**Función 1** – Función para Obtener la Cantidad de Productos en una Categoría.

```
DELIMITER //
CREATE FUNCTION cantidadProductosCategoria(nombreCategoria VARCHAR(30)) RETURNS INT DETERMINISTIC
BEGIN
    DECLARE productCount INT;

    SELECT COUNT(*) INTO productCount
    FROM producto p
    JOIN categoria c ON p.id_categoria = c.id_categoria
    WHERE c.nombre = nombreCategoria;

    RETURN productCount;
END;
//
DELIMITER ;
```

**Función 2** – Función para calcular el total gastado por un usuario en sus compras.

```
DELIMITER //
CREATE FUNCTION calcular_total_gastado(idUsuario INT)
RETURNS DECIMAL(10, 2) DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10, 2);

    SELECT SUM(p.precio * cc.cantidadProducto) INTO total
    FROM carritoCompra cc
    JOIN producto p ON cc.id_producto = p.id_producto
    WHERE cc.id_usuario = idUsuario;

    RETURN total;
END //
DELIMITER ;
```

# Scripts

- Script de creación de cada objeto de la base de datos **SCRIPT**
- Scripts de inserción de datos **SCRIPT**
- Archivo de la base de datos completa **GITHUB**

# Futuras líneas

- Optimización de consultas, a medida la base de datos crece, es importante optimizar las consultas para que sigan siendo rápidas y eficientes.
- expandir el mercado a nivel internacional, esto puede ser importante para aumentar las ventas. eso con lleva a hacer nuevas tablas con los distintos países y sus ciudades.

---

# Herramientas utilizadas

Microsoft Excel: utilizado para la descripción detallada de las tablas.

Microsoft Word: utilizado para crear el documento donde se describe la base de datos.

Paint 3D: utilizado para modificar algunas imágenes.


Herramienta Recortes: utilizado para recortar partes del código.

Google Docs: utilizado para copiar el código del Workbench y luego usarlo como links de los scripts.


MySQL Workbench: utilizado para diseñar, gestionar y hacer mantenimiento de la base de datos.