

# Proyecto Final BDD

**Resumen** – Este documento presenta el análisis y diseño de una base de datos relacional orientada a la gestión de una tienda de ropa. El sistema propuesto permite administrar usuarios, clientes, productos, categorías y ventas, garantizando la integridad, consistencia y organización de la información. El diseño se fundamenta en principios de modelado de datos, normalización hasta la tercera forma normal (3FN) y cumplimiento de las propiedades ACID. Como resultado, se obtiene una estructura de base de datos coherente, escalable y alineada a necesidades reales de un entorno comercial.

## I. INTRODUCCIÓN

El manejo adecuado de la información es un aspecto clave para el correcto funcionamiento de cualquier negocio. En el caso de una tienda de ropa, la gestión de productos, clientes, ventas e inventario requiere un sistema confiable que permita registrar y consultar datos de manera eficiente.

Este proyecto se enfoca en el análisis y diseño de una base de datos relacional que sirva como base para un sistema de gestión de ventas. Se prioriza la correcta estructuración de las tablas, la definición de relaciones y la aplicación de restricciones que aseguren la integridad de los datos, sentando así las bases para una implementación sólida en un sistema gestor de bases de datos.

## II. DESARROLLO DE CONTENIDOS

Las conexiones humanas dentro de la vida universitaria son esenciales, no solo para el crecimiento intelectual, sino también para el desarrollo personal y emocional de los estudiantes. Sin embargo, las herramientas digitales actuales no están optimizadas para fomentar estas dinámicas propias del entorno académico.

A diferencia de las apps tradicionales de conexión social, esta solución se centra en facilitar encuentros basados en intereses académicos, afinidades personales y objetivos comunes, permitiendo que los estudiantes amplíen su red de contactos de manera orgánica y significativa.

### A. Objetivos y justificación

Objetivo General:

Diseñar una base de datos relacional para una tienda de ropa, aplicando principios de modelado, normalización y consistencia de datos.

Objetivos Específicos:

- Identificar las entidades, atributos y relaciones del sistema.

- Normalizar el modelo de datos hasta la tercera forma normal (3FN).
- Definir claves primarias, foráneas y restricciones de integridad.
- Garantizar el cumplimiento de las propiedades ACID en el diseño.

### B. Justificación

El diseño de esta base de datos permite aplicar conocimientos fundamentales de bases de datos relacionales en un caso práctico y realista. Además, proporciona una estructura sólida que puede ser utilizada posteriormente para el desarrollo de aplicaciones o sistemas de gestión comercial.

### C. Análisis y modelado de datos

Para el diseño de la base de datos de la tienda de ropa se realizó un análisis previo de los procesos principales del negocio, tales como la gestión de usuarios del sistema, el registro de clientes, la administración de productos y categorías, y el control de las ventas realizadas. A partir de este análisis se identificaron las entidades necesarias para representar correctamente la información y sus relaciones.

El modelo de datos se diseñó bajo el enfoque entidad–relación, priorizando la claridad, la integridad de los datos y la eliminación de redundancias.

### D. Diagrama Entidad–Relación (E-R)

El diagrama entidad–relación del sistema está compuesto por seis entidades principales: usuarios, clientes, categorías, productos, ventas y detalle\_ventas.

- Un usuario puede registrar muchas ventas, pero cada venta es registrada por un solo usuario.
- Un cliente puede realizar múltiples ventas, pero cada venta pertenece a un único cliente.
- Una venta puede incluir uno o varios productos.
- Cada producto pertenece a una sola categoría.
- Una venta se relaciona con los productos a través de la entidad detalle\_ventas, que actúa como tabla intermedia.

Este modelo permite representar de forma adecuada las relaciones uno a muchos y muchos a muchos presentes en el sistema.

#### E. Definición de entidades, atributos y relaciones

##### Usuarios:

- Representa a las personas que utilizan el sistema para gestionar la tienda.
- Almacena información de identificación, credenciales de acceso y rol.
- Permite controlar permisos y registrar qué usuario realiza cada venta.
- Un usuario puede registrar múltiples ventas.

##### Clientes:

- Almacena la información personal y de contacto de los clientes.
- Permite asociar cada venta a un cliente específico.
- Facilita el historial de compras y la gestión de clientes frecuentes.
- Un cliente puede realizar varias ventas.

##### Categorías:

- Clasifica los productos disponibles en la tienda.
- Evita la repetición del nombre de la categoría en cada producto.
- Permite una organización clara del inventario.
- Una categoría puede agrupar varios productos.

##### Productos:

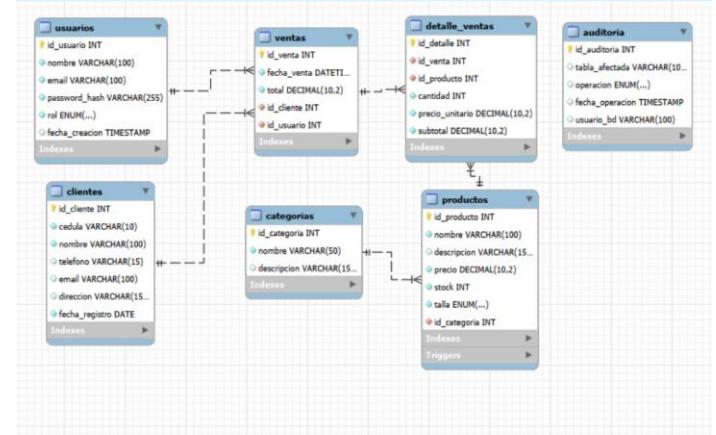
- Almacena la información de las prendas comercializadas.
- Incluye datos como precio, stock, talla y categoría.
- Permite validar disponibilidad antes de registrar una venta.
- Un producto puede aparecer en múltiples ventas.

##### Ventas:

- Registra cada transacción realizada en la tienda.
- Contiene información del cliente, usuario y total de la venta.
- Actúa como el encabezado de la transacción.
- Una venta puede incluir varios productos.

##### Detalle\_ventas:

- Registra los productos vendidos en cada venta.
- Almacena cantidad, precio unitario y subtotal.
- Permite manejar la relación entre ventas y productos.
- Conserva el histórico de precios al momento de la venta.



#### F. Normalización hasta la Tercera Forma Normal (3FN)

##### Para      Primera      Forma      Normal      (1FN):

- Todas las tablas tienen claves primarias.
- Los atributos contienen valores atómicos.
- No existen campos multivaluados ni repetidos.

##### Segunda Forma Normal (2FN):

- Todos los atributos dependen completamente de la clave primaria.
- La información de los productos vendidos se separa en detalle\_ventas.

##### Tercera Forma Normal (3FN):

- No existen dependencias transitivas.
- La información de clientes, usuarios y categorías se almacena en tablas independientes.
- Cada atributo depende únicamente de la clave primaria.

#### G. Justificación del diseño

- La separación entre ventas y detalle\_ventas permite registrar múltiples productos por venta.
- La tabla categorías evita redundancia de información en productos.
- La tabla usuarios permite control de accesos y trazabilidad de ventas.
- El diseño mejora la integridad, organización y escalabilidad del sistema.

#### H. Aplicación de ACID en el proyecto

##### Atomicidad:

- Cada venta se maneja como una única transacción lógica dentro del sistema.
- El registro de la venta incluye: creación de la venta, registro de los productos vendidos y actualización del stock.
- Si alguna de estas operaciones falla (por ejemplo, falta de stock o error al guardar un detalle), ninguna se confirma.
- Esto evita ventas incompletas o inconsistentes dentro de la base de datos.

## Consistencia:

- La base de datos mantiene reglas que garantizan información válida antes y después de cada transacción.
- Las claves foráneas aseguran que no existan ventas sin cliente o sin usuario responsable.
- Las restricciones impiden valores inválidos como precios negativos o cantidades menores a cero.
- Cada transacción lleva al sistema de un estado válido a otro estado válido.

## Aislamiento:

- El sistema permite que múltiples usuarios realicen ventas al mismo tiempo sin afectar los datos entre sí.
- Una venta en proceso no puede modificar ni ser modificada por otra transacción simultánea.
- Esto evita problemas como descontar stock dos veces incorrectamente.
- El aislamiento garantiza que cada transacción se ejecute como si fuera la única en el sistema.

## Durabilidad:

- Una vez que la venta es confirmada, la información queda almacenada de forma permanente.
- Los datos de ventas, clientes y productos no se pierden ante apagones o fallos del sistema.
- El sistema gestor de base de datos se encarga de preservar la información confirmada.
- Esto asegura confiabilidad y persistencia de los datos a largo plazo.

### I. Capturas de la implementación en el SGBD

Se utilizó Workbench para la implementación del SGBD.

Comando para crear y utilizar la base de datos seleccionada:

```
CREATE DATABASE tienda_ropa;
USE tienda_ropa;
```

Definición de las tablas, claves foráneas y las restricciones necesarias para obtener un proyecto completo. En total se crearon seis tablas diferentes:

```
create table usuarios (
    id_usuario int auto_increment primary key,
    nombre varchar(100) not null,
    email varchar(100) not null unique,
    password_hash varchar(255) not null,
    rol enum('admin','vendedor') not null,
    fecha_creacion timestamp default current_timestamp
);

create table clientes (
    id_cliente int auto_increment primary key,
    cedula varchar(10) not null unique,
    nombre varchar(100) not null,
    telefono varchar(15),
    email varchar(100),
    direccion varchar(150),
    fecha_registro date not null
);
```

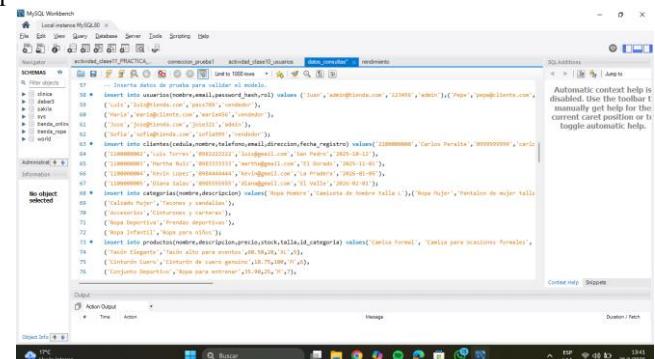
```
create table categorias (
    id_categoria int auto_increment primary key,
    nombre varchar(50) not null unique,
    descripcion varchar(150)
);

create table productos (
    id_producto int auto_increment primary key,
    nombre varchar(100) not null,
    descripcion varchar(150),
    precio decimal(10,2) not null check (precio > 0),
    stock int not null check (stock >= 0),
    talla enum('xs','s','m','l','xl') not null,
    id_categoria int not null,
    foreign key (id_categoria) references categorias(id_categoria)
);

create table ventas (
    id_venta int auto_increment primary key,
    fecha_venta datetime not null,
    total decimal(10,2) not null,
    id_cliente int not null,
    id_usuario int not null,
    foreign key (id_cliente) references clientes(id_cliente),
    foreign key (id_usuario) references usuarios(id_usuario)
);

create table detalle_ventas (
    id_detalle int auto_increment primary key,
    id_venta int not null,
    id_producto int not null,
    cantidad int not null check (cantidad > 0),
    precio_unitario decimal(10,2) not null,
    subtotal decimal(10,2) not null,
    foreign key (id_venta) references ventas(id_venta),
    foreign key (id_producto) references productos(id_producto)
);
```

Inserción de los datos de prueba. Se agregó un total de 8 datos por sección:



### J. Ejemplos de consultas y resultados

Diseño de al menos 10 consultas:

Selecciones simples y con condiciones:

```
-- simple
SELECT nombre, email FROM usuarios;
SELECT nombre, descripcion FROM categorias;
-- condición
SELECT * FROM productos WHERE stock > 20;
select * from clientes where nombre like '%A%';
```

Resultado:

```

101 •   SELECT nombre, email FROM usuarios;
102 •   SELECT nombre, descripcion FROM categorias;
103 -- condición

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

	nombre	descripcion
1	Ropa Hombre	Camiseta de hombre talla L
2	Ropa Mujer	Pantalon de mujer talla 12
3	Ropa Niña	Disfraz mickey mouse
4	Calzado Hombre	Zapatos casuales para hombre
5	Calzado Mujer	Tacones y sandalias

```

104 •   SELECT * FROM productos WHERE stock > 20;
105 •   select * from clientes where nombre like '%%%';
106 -- Join

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

	id_cliente	cedula	nombre	telefono	email	direccion	fecha_registro
1	2100000000	Carlos Peralta	0999999999	carlos@gmail.com	Nueva Loja	2025-12-01	
2	1712000000	Pablo Arturo	0999999999	pabito@gmail.com	Real Audiencia	2026-02-12	
3	1030000000	Martina Popus	0987654321	popus@gmail.com	Los cedros	2025-11-30	
4	1100000001	Andrea Molina	0981111111	andrea@gmail.com	Centro	2025-10-10	
5	1100000003	Martha Ruiz	0983333333	martha@gmail.com	El Dorado	2025-11-01	

## 6 Joins entre múltiples tablas:

El primer JOIN se centra en el total de ventas por cliente:

```
-- 1.total ventas cliente
Select v.id_venta, c.nombre, v.total
From ventas v
JOIN clientes c ON v.id_cliente = c.id_cliente;
```

Resultado:

```

108 •   Select v.id_venta, c.nombre, v.total
109 From ventas v
110 JOIN clientes c ON v.id_cliente = c.id_cliente;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

	id_venta	nombre	total
1	Carlos Peralta	80.00	
2	Pablo Arturo	45.00	
3	Martina Popus	66.08	
4	Andrea Molina	55.00	
5	Luis Torres	121.00	

El segundo JOIN se realizó para seleccionar los datos de las ventas realizadas, mostrando el identificador de la venta (v.id\_venta), el nombre del vendedor (u.nombre) y la fecha de la venta:

```
-- 2.ventas vendedor
Select v.id_venta, u.nombre AS vendedor, v.fecha_venta
FROM ventas v
JOIN usuarios u ON v.id_usuario = u.id_usuario;
```

Resultado:

```

111 -- 2.ventas vendedor
112 •   Select v.id_venta, u.nombre AS vendedor, v.fecha_venta
113 From ventas v
114 JOIN usuarios u ON v.id_usuario = u.id_usuario;
115 -- 3. Cliente, fecha y total
116 •   SELECT c.nombre, v.fecha_venta, v.total

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

	vendedor	fecha_venta
1	Pepe	2025-01-15 10:30:00
2	Pepe	2026-01-25 20:01:55
3	Lolita	2025-12-01 09:46:00
4	Ana	2025-11-15 11:20:00
5	Luis	2025-11-20 16:45:00

El tercer JOIN selecciona información sobre las ventas realizadas por los clientes:

```
-- 3. Cliente, fecha y total
SELECT c.nombre, v.fecha_venta, v.total
FROM clientes c
JOIN ventas v ON c.id_cliente = v.id_cliente;
```

Resultado:

```

115 -- 3. Cliente, fecha y total
116 •   SELECT c.nombre, v.fecha_venta, v.total
117 From clientes c
118 JOIN ventas v ON c.id_cliente = v.id_cliente;
119 -- 4.por categoria
120 •   Select p.nombre, c.nombre AS categoria

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

	nombre	fecha_venta	total
1	Carlos Peralta	2025-01-15 10:30:00	80.00
2	Pablo Arturo	2026-01-25 20:01:55	45.00
3	Martina Popus	2025-12-01 09:46:00	66.08
4	Andrea Molina	2025-11-15 11:20:00	55.00
5	Luis Torres	2025-11-20 16:45:00	121.00

El cuarto y quinto JOIN; el cuarto selecciona el nombre de los productos junto con su categoría correspondiente y el quinto es la información sobre las ventas realizadas por un vendedor específico:

```
-- 4.por categoria
Select p.nombre, c.nombre AS categoria
From productos p
JOIN categorias c ON p.id_categoria = c.id_categoria;
-- 5.ventas por ID vendedor
Select v.id_venta, c.nombre AS cliente
From ventas v
JOIN clientes c ON v.id_cliente = c.id_cliente
Where v.id_usuario = 2;
```

Resultados:

```

119 -- 4.por categoria
120 •   Select p.nombre, c.nombre AS categoria
121 From productos p
122 JOIN categorias c ON p.id_categoria = c.id_categoria;
123 -- 5.ventas por ID vendedor
124 •   Select v.id_venta, c.nombre AS cliente

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

	nombre	categoria
1	Camisa Formal	Ropa Hombre
2	Vestido de gala	Ropa Mujer
3	Terno de baño	Ropa Deportiva
4	Zapato Casual	Calzado Mujer
5	Tacón Elegante	Calzado Mujer

Result Grid | Filter Rows: | Export: | Wrap Cell Content

	id_venta	diente
1	Carlos Peralta	
2	Pablo Arturo	

El ultimo JOIN trata de obtener el subtotal multiplicando precio por cantidad:

```
-- 6. Multiplicación con JOIN
Select dv.cantidad, p.nombre, precio_unitario * cantidad AS SUBTOTAL
From detalle_ventas dv
JOIN productos p ON dv.id_producto = p.id_producto;
```

Resultado:

```
128 -- 6. Multiplicación con JOIN
129 • Select dv.cantidad, p.nombre,precio_unitario* cantidad AS SUBTOTAL
130 From detalle_ventas dv
131 JOIN productos p ON dv.id_producto = p.id_producto;
132
133 -- Funciones de agregación (SUM, COUNT, AVG)
```

Result Grid   Filter Rows: Export: Wrap Cell Content:		
cantidad	nombre	SUBTOTAL
3	Chaqueta Infantil	85.20
4	Vestido de gala	183.60
3	Terno de baño	75.48
1	Camisa Formal	35.00
2	Vestido de qala	91.80

## K. Funciones

La consulta suma el total de compras para cada cliente, mostrando el nombre del cliente y el total de sus compras; La segunda función, cuenta la cantidad de productos disponibles en la tabla productos; y, la última función, calcula el precio promedio de cada producto, mostrando el nombre del producto:

```
-- Funciones de agregación (SUM, COUNT, AVG)
Select c.nombre, sum(v.total) AS total_compras
From ventas v
JOIN clientes c ON v.id_cliente = c.id_cliente
group by c.nombre;

Select nombre, count(*) AS total_productos
From productos
group by nombre;

select nombre, avg(precio) AS promedio_precios
from productos
group by nombre;
```

Resultados:

Función de suma:

```
133 -- Funciones de agregación (SUM, COUNT, AVG)
134 • Select c.nombre, sum(v.total) AS total_compras
135 From ventas v
136 JOIN clientes c ON v.id_cliente = c.id_cliente
137 group by c.nombre;
138
139 • Select nombre, count(*) AS total_productos
```

Result Grid   Filter Rows: Export: Wrap Cell Content:	
nombre	total_compras
Carlos Peralta	80.00
Pablo Arturo	45.00
Martina Popus	66.08
Andrea Molina	55.00
Lilia Torres	121.00

Función count():

```
139 • Select nombre, count(*) AS total_productos
140 From productos
141 group by nombre;
142
143 • select nombre, avg(precio) AS promedio_prec
```

Result Grid   Filter Rows: Export: Wrap Cell Content:	
nombre	total_productos
Camisa Formal	1
Vestido de gala	1
Terno de baño	1
Zapato Casual	1
Tacón Elegante	1

Función avg():

```
143 • select nombre, avg(precio) AS promedio_precios
144 from productos
145 group by nombre;
```

Result Grid   Filter Rows: Export: Wrap Cell Content:	
nombre	promedio_precios
Camisa Formal	35.000000
Vestido de gala	45.900000
Terno de baño	25.160000
Zapato Casual	55.000000
Tacón Elegante	60.500000

L. Seguridad, administración, backups y auditoría del sistema

Para garantizar el correcto funcionamiento del sistema de gestión de la tienda de ropa, es fundamental implementar mecanismos de seguridad, control y respaldo de la información. Estas medidas permiten proteger los datos almacenados, controlar los accesos al sistema, registrar los cambios realizados y asegurar la recuperación de la información ante posibles fallos.

La aplicación de estas prácticas fortalece la confiabilidad del sistema y complementa el diseño de la base de datos propuesto.

## Administración y seguridad

### Creación de usuarios:

La base de datos contempla la creación de usuarios del sistema, los cuales representan a las personas que interactúan con la información de la tienda. Cada usuario posee credenciales de acceso que permiten identificarlo de manera única dentro del sistema.

### Roles y permisos:

Para un control adecuado de los accesos, se asignan por roles a grupos específicos según las responsabilidades de cada usuario. Mediante la asignación de permisos se controla qué operaciones pueden realizarse sobre las tablas del sistema, como inserciones, consultas, actualizaciones o eliminaciones de datos.

### **Confidencialidad e integridad:**

La confidencialidad de la información se garantiza restringiendo el acceso a los datos según el rol del usuario. La integridad se mantiene mediante el uso de claves primarias, claves foráneas y restricciones, asegurando que los datos almacenados sean correctos, coherentes y consistentes.

### **Seguridad del sistema**

#### **Prevención de inyección SQL:**

La inyección SQL representa una amenaza para los sistemas que interactúan con bases de datos. Para reducir este riesgo, el sistema está diseñado para utilizar consultas definidas desde la aplicación, evitando que los datos ingresados por el usuario sean interpretados como código SQL.

#### **Cifrado de contraseñas:**

Las contraseñas de los usuarios no se almacenan en texto plano dentro de la base de datos. En su lugar, se utiliza un proceso de cifrado mediante funciones hash, lo que protege la información sensible y evita la exposición directa de las credenciales.

#### **Control de accesos:**

El control de accesos se realiza mediante la combinación de autenticación de usuarios, roles y permisos. De esta manera, cada usuario solo puede acceder a las funcionalidades y datos permitidos según su rol dentro del sistema.

### **Backups y recuperación de la información**

#### **Backup completo:**

El backup completo consiste en realizar una copia total de la base de datos, incluyendo su estructura y los datos almacenados. Este tipo de respaldo permite restaurar el sistema en caso de fallos graves o pérdida total de información.

#### **Backup lógico o incremental:**

El backup lógico permite respaldar la información mediante archivos que contienen las sentencias necesarias para reconstruir la base de datos. Este tipo de respaldo es más ligero y facilita la realización de copias periódicas.

#### **Recuperación de la información:**

En caso de errores del sistema o pérdida de datos, los backups permiten recuperar la base de datos y restablecer su

funcionamiento normal, garantizando la continuidad del sistema y la disponibilidad de la información.

### **Pruebas del sistema**

#### **Pruebas de carga:**

Las pruebas de carga permiten evaluar el comportamiento del sistema bajo condiciones normales de uso, simulando múltiples usuarios realizando operaciones simultáneamente. Estas pruebas ayudan a verificar la estabilidad y el rendimiento de la base de datos.

#### **Pruebas de estrés:**

Las pruebas de estrés consisten en someter al sistema a condiciones extremas para identificar sus límites y detectar posibles fallos. Este tipo de pruebas permite evaluar la resistencia del sistema ante una alta demanda.

### **Auditoría del sistema**

Para llevar un control de las operaciones realizadas sobre la base de datos, se implementó un sistema de auditoría mediante el uso de triggers.

#### **Triggers de auditoría:**

Se crearon triggers para registrar las operaciones de inserción, actualización y eliminación de datos. Estos mecanismos permiten almacenar información relevante sobre cada cambio realizado, como la tabla afectada, el tipo de operación y la fecha del cambio.

La auditoría facilita la trazabilidad de las acciones realizadas y permite detectar modificaciones no autorizadas dentro del sistema.

### **Aplicación práctica de ACID**

#### **Atomicidad:**

Las operaciones críticas del sistema, como el registro de una venta, se manejan como transacciones completas. Esto garantiza que todas las acciones asociadas a la transacción se ejecuten correctamente o se reviertan en caso de error.

**Consistencia:**

Las restricciones definidas en la base de datos aseguran que cada transacción mantenga la información en un estado válido antes y después de su ejecución.

**Aislamiento:**

El sistema permite que múltiples transacciones se ejecuten simultáneamente sin interferir entre sí, evitando inconsistencias en los datos.

**Durabilidad:**

Una vez confirmada una transacción, los datos quedan almacenados de forma permanente. La durabilidad se refuerza mediante el uso de respaldos, asegurando la persistencia de la información.

**Aporte de la seguridad al sistema**

La implementación de mecanismos de seguridad, respaldo y auditoría fortalece la confiabilidad del sistema de base de datos. Estas prácticas garantizan la protección de la información, el control de accesos y la correcta gestión de los datos, complementando el diseño lógico y estructural de la base de datos propuesta.B

**III. CONCLUSIONES**

El diseño de la base de datos para una tienda de ropa se ha realizado siguiendo principios de modelado y normalización, garantizando la integridad y consistencia de los datos. Se han identificado claramente las entidades, atributos y relaciones para evitar redundancias y asegurar que todos los datos sean dependientes de sus claves primarias. Además, se han definido claves primarias y foráneas adecuadas, junto con restricciones de integridad que aseguran la validez de los datos. Finalmente, el cumplimiento de las propiedades ACID en el diseño asegura que la base de datos mantenga su integridad y fiabilidad en operaciones concurrentes y transacciones, lo que es fundamental para el correcto funcionamiento de la tienda.