

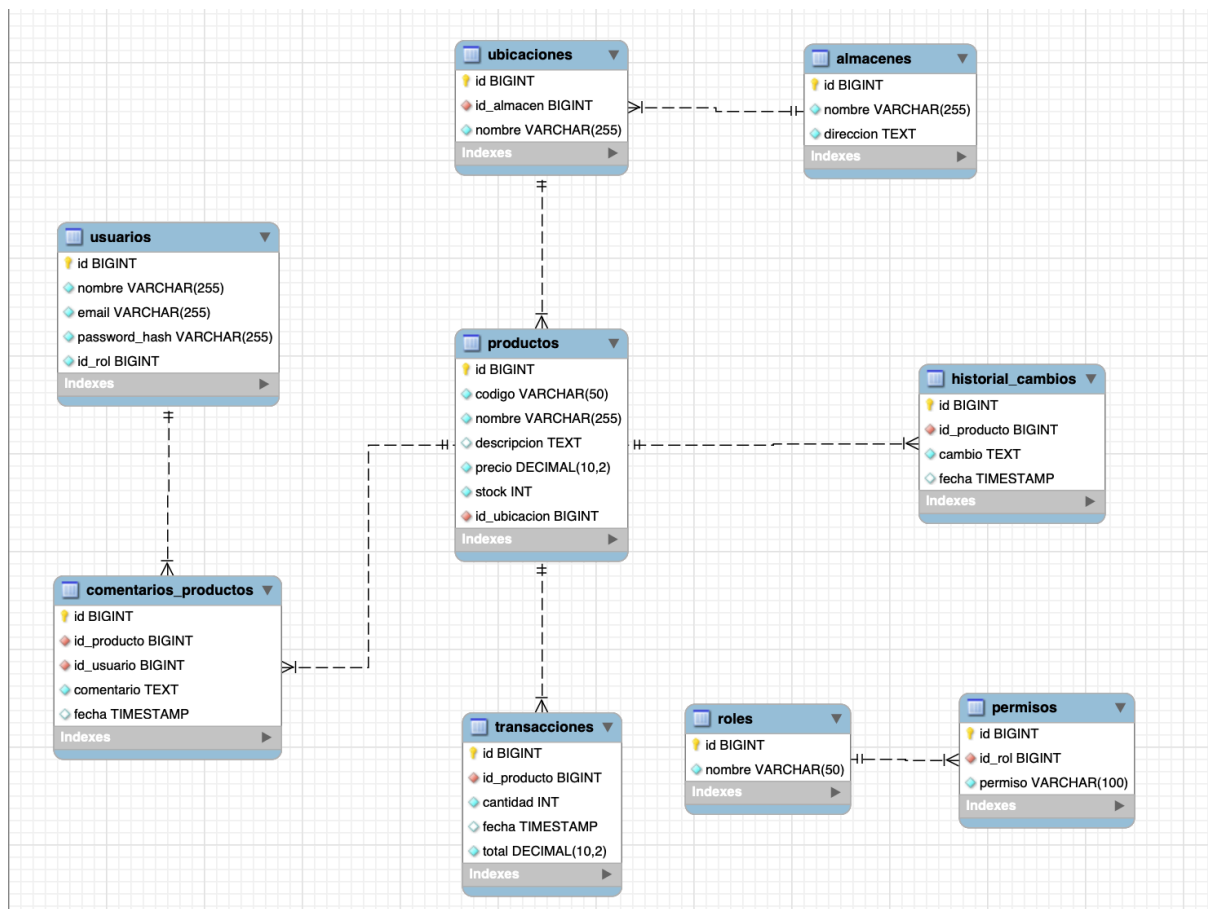
## Bases de Datos III

### Proyecto Final Bases de Datos

Entrega # 1: 17 de Marzo, 2025

#### 1. Toma de Requerimientos

a. Diagramas E-R y diseño lógico del sistema.



#### 2. Diseño de la base de datos

a. Creación de tablas, relaciones, funciones, procedimientos y/o triggers en MySQL.

-- Crear la base de datos

```
CREATE DATABASE IF NOT EXISTS gestion_inventarios;  
USE gestion_inventarios;
```

-- Tabla de almacenes

```
CREATE TABLE almacenes (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(255) NOT NULL,  
  direccion TEXT NOT NULL  
);
```

-- Tabla de ubicaciones dentro de almacenes

```
CREATE TABLE ubicaciones (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  id_almacen BIGINT NOT NULL,  
  nombre VARCHAR(255) NOT NULL,  
  FOREIGN KEY (id_almacen) REFERENCES almacenes(id) ON DELETE CASCADE  
);
```

-- Tabla de productos

```
CREATE TABLE productos (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  codigo VARCHAR(50) UNIQUE NOT NULL,  
  nombre VARCHAR(255) NOT NULL,  
  descripcion TEXT,  
  precio DECIMAL(10,2) NOT NULL,  
  stock INT NOT NULL,  
  id_ubicacion BIGINT NOT NULL,  
  FOREIGN KEY (id_ubicacion) REFERENCES ubicaciones(id) ON DELETE CASCADE  
);
```

-- Tabla de transacciones (ventas simuladas)

```
CREATE TABLE transacciones (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  id_producto BIGINT NOT NULL,  
  cantidad INT NOT NULL,  
  fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  total DECIMAL(10,2) NOT NULL,  
  FOREIGN KEY (id_producto) REFERENCES productos(id) ON DELETE CASCADE  
);
```

-- Tabla de usuarios

```
CREATE TABLE usuarios (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(255) NOT NULL,  
  email VARCHAR(255) UNIQUE NOT NULL,  
  password_hash VARCHAR(255) NOT NULL,  
  id_rol BIGINT NOT NULL  
);
```

-- Tabla de roles

```
CREATE TABLE roles (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(50) NOT NULL UNIQUE  
);
```

-- Tabla de permisos

```
CREATE TABLE permisos (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  id_rol BIGINT NOT NULL,  
  permiso VARCHAR(100) NOT NULL,  
  FOREIGN KEY (id_rol) REFERENCES roles(id) ON DELETE CASCADE  
);
```

-- Tabla de historial de cambios de productos

```
CREATE TABLE historial_cambios (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  id_producto BIGINT NOT NULL,  
  cambio TEXT NOT NULL,  
  fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (id_producto) REFERENCES productos(id) ON DELETE CASCADE  
);
```

-- Tabla de comentarios sobre productos

```
CREATE TABLE comentarios_productos (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  id_producto BIGINT NOT NULL,  
  id_usuario BIGINT NOT NULL,  
  comentario TEXT NOT NULL,  
  fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (id_producto) REFERENCES productos(id) ON DELETE CASCADE,  
  FOREIGN KEY (id_usuario) REFERENCES usuarios(id) ON DELETE CASCADE  
);
```

-- Insertar roles por defecto

```
INSERT INTO roles (nombre) VALUES ('Administrador'), ('Operador');
```

-- Procedimiento para agregar productos

```
DELIMITER $$  
CREATE PROCEDURE agregar_producto (  
  IN p_codigo VARCHAR(50),  
  IN p_nombre VARCHAR(255),  
  IN p_descripcion TEXT,  
  IN p_precio DECIMAL(10,2),  
  IN p_stock INT,  
  IN p_id_ubicacion BIGINT  
)
```

```
BEGIN
    INSERT INTO productos (codigo, nombre, descripcion, precio, stock, id_ubicacion)
    VALUES (p_codigo, p_nombre, p_descripcion, p_precio, p_stock, p_id_ubicacion);
END$$
DELIMITER ;
```

-- Procedimiento para actualizar el stock de un producto

```
DELIMITER $$
CREATE PROCEDURE actualizar_stock (
    IN p_id_producto BIGINT,
    IN p_nuevo_stock INT
)
BEGIN
    UPDATE productos SET stock = p_nuevo_stock WHERE id = p_id_producto;
END$$
DELIMITER ;
```

-- Procedimiento para registrar una venta

```
DELIMITER $$
CREATE PROCEDURE registrar_venta (
    IN p_id_producto BIGINT,
    IN p_cantidad INT
)
BEGIN
    DECLARE v_precio DECIMAL(10,2);
    SELECT precio INTO v_precio FROM productos WHERE id = p_id_producto;
    INSERT INTO transacciones (id_producto, cantidad, total)
    VALUES (p_id_producto, p_cantidad, v_precio * p_cantidad);
    UPDATE productos SET stock = stock - p_cantidad WHERE id = p_id_producto;
END$$
DELIMITER ;
```

-- Trigger para evitar stock negativo

```
DELIMITER $$
CREATE TRIGGER before_venta_insert
BEFORE INSERT ON transacciones
FOR EACH ROW
BEGIN
    DECLARE v_stock INT;
    SELECT stock INTO v_stock FROM productos WHERE id = NEW.id_producto;
    IF v_stock < NEW.cantidad THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Stock insuficiente';
    END IF;
END$$
DELIMITER ;
```

-- Trigger para registrar eliminación de productos en historial

```
DELIMITER $$  
CREATE TRIGGER after_producto_delete  
AFTER DELETE ON productos  
FOR EACH ROW  
BEGIN  
    INSERT INTO historial_cambios (id_producto, cambio)  
    VALUES (OLD.id, 'Producto eliminado');  
END$$  
DELIMITER ;
```

-- Función para calcular el valor total del inventario

```
DELIMITER $$  
CREATE FUNCTION calcular_valor_total_inventario()  
RETURNS DECIMAL(10,2)  
DETERMINISTIC  
BEGIN  
    DECLARE total DECIMAL(10,2);  
    SELECT SUM(precio * stock) INTO total FROM productos;  
    RETURN total;  
END$$  
DELIMITER ;
```

-- Procedimiento para agregar usuarios

```
DELIMITER $$  
CREATE PROCEDURE agregar_usuario (  
    IN p_nombre VARCHAR(255),  
    IN p_email VARCHAR(255),  
    IN p_password_hash VARCHAR(255),  
    IN p_id_rol BIGINT  
)  
BEGIN  
    INSERT INTO usuarios (nombre, email, password_hash, id_rol)  
    VALUES (p_nombre, p_email, p_password_hash, p_id_rol);  
END$$  
DELIMITER ;
```

-- Procedimiento para actualizar el rol de un usuario

```
DELIMITER $$  
CREATE PROCEDURE actualizar_rol_usuario (  
    IN p_id_usuario BIGINT,  
    IN p_nuevo_id_rol BIGINT  
)  
BEGIN  
    UPDATE usuarios SET id_rol = p_nuevo_id_rol WHERE id = p_id_usuario;  
END$$  
DELIMITER ;
```

**-- Procedimiento para eliminar un usuario**

```
DELIMITER $$  
CREATE PROCEDURE eliminar_usuario (  
    IN p_id_usuario BIGINT  
)  
BEGIN  
    DELETE FROM usuarios WHERE id = p_id_usuario;  
END$$  
DELIMITER ;
```

**-- Procedimiento para listar usuarios y sus roles**

```
DELIMITER $$  
CREATE PROCEDURE listar_usuarios_roles()  
BEGIN  
    SELECT u.id, u.nombre, u.email, r.nombre AS rol FROM usuarios u  
    JOIN roles r ON u.id_rol = r.id;  
END$$  
DELIMITER ;
```

**-- Procedimiento para asignar un permiso a un rol**

```
DELIMITER $$  
CREATE PROCEDURE asignar_permiso (  
    IN p_id_rol BIGINT,  
    IN p_permiso VARCHAR(100)  
)  
BEGIN  
    INSERT INTO permisos (id_rol, permiso) VALUES (p_id_rol, p_permiso);  
END$$  
DELIMITER ;
```

**Reportes Generales**

**-- Reporte de inventario general**

```
DELIMITER $$  
CREATE PROCEDURE reporte_inventario_general()  
BEGIN  
    SELECT p.codigo, p.nombre, p.stock, p.precio, (p.stock * p.precio) AS valor_total  
    FROM productos p;  
END$$  
DELIMITER ;
```

**-- Reporte de productos por ubicación**

```
DELIMITER $$  
CREATE PROCEDURE reporte_productos_por_ubicacion()  
BEGIN  
    SELECT a.nombre AS almacen, u.nombre AS ubicacion, p.nombre AS producto, p.stock  
    FROM productos p
```

```
JOIN ubicaciones u ON p.id_ubicacion = u.id  
JOIN almacenes a ON u.id_almacen = a.id  
ORDER BY a.nombre, u.nombre;  
END$$  
DELIMITER ;
```

#### -- Reporte de ventas simuladas

```
DELIMITER $$  
CREATE PROCEDURE reporte_ventas_simuladas()  
BEGIN  
    SELECT t.id, p.nombre AS producto, t.cantidad, t.fecha, t.total  
    FROM transacciones t  
    JOIN productos p ON t.id_producto = p.id  
    ORDER BY t.fecha DESC;  
END$$  
DELIMITER ;
```

#### -- Reporte de productos con bajo stock

```
DELIMITER $$  
CREATE PROCEDURE reporte_productos_bajo_stock(IN nivel_minimo INT)  
BEGIN  
    SELECT p.codigo, p.nombre, p.stock  
    FROM productos p  
    WHERE p.stock < nivel_minimo  
    ORDER BY p.stock ASC;  
END$$  
DELIMITER ;
```

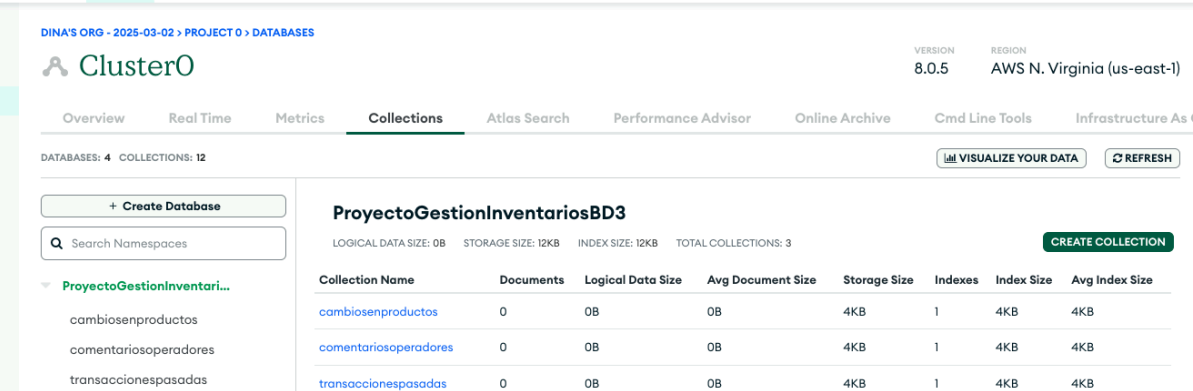
#### -- Reporte de usuarios y sus roles

```
DELIMITER $$  
CREATE PROCEDURE reporte_usuarios_rols()  
BEGIN  
    SELECT u.id, u.nombre, u.email, r.nombre AS rol  
    FROM usuarios u  
    JOIN roles r ON u.id_rol = r.id;  
END$$  
DELIMITER ;
```

## b. Configuración de bases de datos en MongoDB.

Se crea en MongoDB Atlas la base de datos llamada **ProyectoGestionInventariosBD3**. En ella se pretende llevar un registro histórico de

los cambios en los productos, de los comentarios realizados por los operadores, y de las transacciones pasadas. Para ello se agregan estas 3 colecciones.



The screenshot displays the MongoDB Atlas 'Collections' tab for a database named 'ProyectoGestionInventariosBD3'. The interface shows a sidebar with a search bar and a list of collections. The main area contains a table with the following data:

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
cambiosenproductos	0	0B	0B	4KB	1	4KB	4KB
comentariosoperadores	0	0B	0B	4KB	1	4KB	4KB
transaccionespasadas	0	0B	0B	4KB	1	4KB	4KB

## Explicacion del diseño y configuracion:

### 1. Diseño de la Base de Datos

La base de datos gestion\_inventarios está diseñada para administrar almacenes, productos, transacciones, usuarios y roles. Se estructura en distintas tablas con relaciones bien definidas.

#### Tablas y Relaciones

##### Almacenes y Ubicaciones

**Almacenes:** Contiene información sobre los almacenes.

ubicaciones: Define ubicaciones dentro de cada almacén y tiene una relación con almacenes.

Productos

**Productos:** Contiene información sobre cada producto, incluyendo precio, stock y ubicación.

Relación: Cada producto pertenece a una ubicación (id\_ubicacion).

Transacciones (Ventas Simuladas)

**Transacciones:** Registra cada venta, asociada a un producto (id\_producto).

Usuarios, Roles y Permisos

**Usuarios:** Almacena los datos de los usuarios, incluyendo el rol asignado.

roles: Define los distintos roles en el sistema (Administrador, Operador, etc.).

permisos: Relaciona roles con permisos específicos.

Historial y Comentarios

**historial\_cambios:** Registra modificaciones o eliminaciones de productos.

comentarios\_productos: Permite a los usuarios dejar comentarios sobre los productos.

### 2. Creación de Tablas



Cada tabla tiene claves primarias (PRIMARY KEY), y en algunas se establecen relaciones mediante claves foráneas (FOREIGN KEY) con la opción ON DELETE CASCADE para eliminar datos relacionados automáticamente.

Ejemplo:

I

Copy

Edit

```
CREATE TABLE productos (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  codigo VARCHAR(50) UNIQUE NOT NULL,  
  nombre VARCHAR(255) NOT NULL,  
  descripcion TEXT,  
  precio DECIMAL(10,2) NOT NULL,  
  stock INT NOT NULL,  
  id_ubicacion BIGINT NOT NULL,  
  FOREIGN KEY (id_ubicacion) REFERENCES ubicaciones(id) ON DELETE CASCADE  
);
```

Aquí:

id es la clave primaria.

codigo es único para cada producto.

id\_ubicacion es clave foránea vinculada a ubicaciones(id).

### 3. Procedimientos Almacenados

Los procedimientos almacenados facilitan la gestión de productos, usuarios y transacciones.

Ejemplo 1: Agregar un producto

sql

Copy

Edit

DELIMITER \$\$

```
CREATE PROCEDURE agregar_producto (  
  IN p_codigo VARCHAR(50),  
  IN p_nombre VARCHAR(255),  
  IN p_descripcion TEXT,  
  IN p_precio DECIMAL(10,2),  
  IN p_stock INT,  
  IN p_id_ubicacion BIGINT  
)  
BEGIN  
  INSERT INTO productos (codigo, nombre, descripcion, precio, stock, id_ubicacion)  
  VALUES (p_codigo, p_nombre, p_descripcion, p_precio, p_stock, p_id_ubicacion);  
END$$  
DELIMITER ;
```

Objetivo: Inserta un nuevo producto en la base de datos.

Parámetros: Código, nombre, descripción, precio, stock y ubicación.

#### Ejemplo 2: Registrar una venta

sql

Copy

Edit

DELIMITER \$\$

CREATE PROCEDURE registrar\_venta (

IN p\_id\_producto BIGINT,

IN p\_cantidad INT

)

BEGIN

DECLARE v\_precio DECIMAL(10,2);

SELECT precio INTO v\_precio FROM productos WHERE id = p\_id\_producto;

INSERT INTO transacciones (id\_producto, cantidad, total)

VALUES (p\_id\_producto, p\_cantidad, v\_precio \* p\_cantidad);

UPDATE productos SET stock = stock - p\_cantidad WHERE id = p\_id\_producto;

END\$\$

DELIMITER ;

Objetivo: Registra una venta y actualiza el stock.

Lógica:

Obtiene el precio del producto.

Inserta la venta en transacciones.

Reduce el stock en productos.

#### 4. Triggers

Los triggers ejecutan acciones automáticamente ante ciertos eventos.

#### Ejemplo 1: Evitar stock negativo antes de una venta

sql

Copy

Edit

DELIMITER \$\$

CREATE TRIGGER before\_venta\_insert

BEFORE INSERT ON transacciones

FOR EACH ROW

BEGIN

DECLARE v\_stock INT;

SELECT stock INTO v\_stock FROM productos WHERE id = NEW.id\_producto;

IF v\_stock < NEW.cantidad THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Stock insuficiente';

END IF;

END\$\$

DELIMITER ;

Antes de insertar una venta, verifica que haya suficiente stock.

Si no hay stock suficiente, lanza un error.

#### Ejemplo 2: Registrar eliminación de productos en el historial

sql

Copy

Edit

DELIMITER \$\$

CREATE TRIGGER after\_producto\_delete

AFTER DELETE ON productos

FOR EACH ROW

BEGIN

INSERT INTO historial\_cambios (id\_producto, cambio)

VALUES (OLD.id, 'Producto eliminado');

END\$\$

DELIMITER ;

Después de eliminar un producto, guarda un registro en historial\_cambios.

## 5. Función para Calcular Valor Total del Inventario

Las funciones devuelven valores específicos, útiles para reportes.

Copy

Edit

DELIMITER \$\$

CREATE FUNCTION calcular\_valor\_total\_inventario()

RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE total DECIMAL(10,2);

SELECT SUM(precio \* stock) INTO total FROM productos;

RETURN total;

END\$\$

DELIMITER ;

Retorna la suma total del inventario (precio \* stock).

## 6. Reportes con Procedimientos

Se han creado reportes para consultas comunes.

### Ejemplo 1: Reporte de Inventario General

sql

Copy

Edit

DELIMITER \$\$

CREATE PROCEDURE reporte\_inventario\_general()

BEGIN

SELECT p.codigo, p.nombre, p.stock, p.precio, (p.stock \* p.precio) AS valor\_total

FROM productos p;

END\$\$

DELIMITER ;

Objetivo: Muestra código, nombre, stock, precio y valor total de cada producto.

## Ejemplo 2: Reporte de Productos con Bajo Stock

sql

Copy

Edit

DELIMITER \$\$

CREATE PROCEDURE reporte\_productos\_bajo\_stock(IN nivel\_minimo INT)

BEGIN

    SELECT p.codigo, p.nombre, p.stock

    FROM productos p

    WHERE p.stock < nivel\_minimo

    ORDER BY p.stock ASC;

END\$\$

DELIMITER ;

Objetivo: Lista productos con stock menor al valor mínimo ingresado.