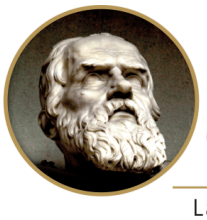


Galileo UNIVERSIDAD
La Revolución en la Educación

Instituto Von Neumann

TÉCNICO EN DESARROLLO DE SOFTWARE
Catedrático Ing. Mynor René Ruiz Guerra
Base de Datos III

Sistemas de Gestión de Inventarios

Presentado por:

Angelo Estrada - 05189095
Aleks Coronado - 24000465

Sistema de Gestión de Inventarios

1. Introducción

En este documento describimos el diseño, configuración e implementación inicial del sistema de gestión de inventarios. Este sistema permite administrar productos, ubicaciones, almacenes, transacciones y usuarios, garantizando la seguridad y optimización en la gestión de datos.

2. Diseño de la Base de Datos

2.1 Modelo Relacional

Se diseñó un modelo relacional en MySQL que incluye las siguientes entidades:

Productos: Contiene información detallada sobre los productos.

Almacenes y Ubicaciones: Representan la estructura física donde se almacenan los productos.

Transacciones: Registra las operaciones de entrada y salida de productos.

Usuarios y roles: Permite la gestión de accesos y permisos para los usuarios.

Historial de Productos: Registra los cambios en los productos mediante el seguimiento del stock de los productos.

2.1.2 Diagrama E-R (Texto)

Entidades:

Productos:

- `id_producto` (Clave Primaria, INT, AUTO_INCREMENT)
- `codigo_producto` (VARCHAR) [Restricción de unicidad para evitar duplicados]
- `nombre` (VARCHAR)
- `descripcion` (TEXT)
- `precio_unitario` (DECIMAL)
- `stock_disponible` (INT)

Almacenes:

- `id_almacen` (Clave Primaria, INT, AUTO_INCREMENT)
- `nombre` (VARCHAR)
- `direccion` (VARCHAR)

Ubicaciones:

- `id_ubicacion` (Clave Primaria, INT, AUTO_INCREMENT)
- `id_almacen` (Clave Foránea, INT)
- `nombre` (VARCHAR)

Productos_Ubicaciones: ***(Tabla de unión para la relación muchos a muchos)***

- `id_producto` (Clave Foránea, INT)
- `id_ubicacion` (Clave Foránea, INT)

Transacciones:

- `id_transaccion` (Clave Primaria, INT, AUTO_INCREMENT)
- `id_producto` (Clave Foránea, INT)
- `cantidad` (INT)
- `tipo` (VARCHAR) (Venta, Entrada, etc.)
- `fecha` (DATETIME) [Agregar índice en `id_producto`]

Roles:

- `id_rol` (Clave Primaria, INT, AUTO_INCREMENT)
- `nombre_rol` (VARCHAR)

Usuarios:

- `id_usuario` (Clave Primaria, INT, AUTO_INCREMENT)
- `nombre_usuario` (VARCHAR)
- `contrasena` (VARCHAR)
- `id_rol` (Clave Foránea, INT)

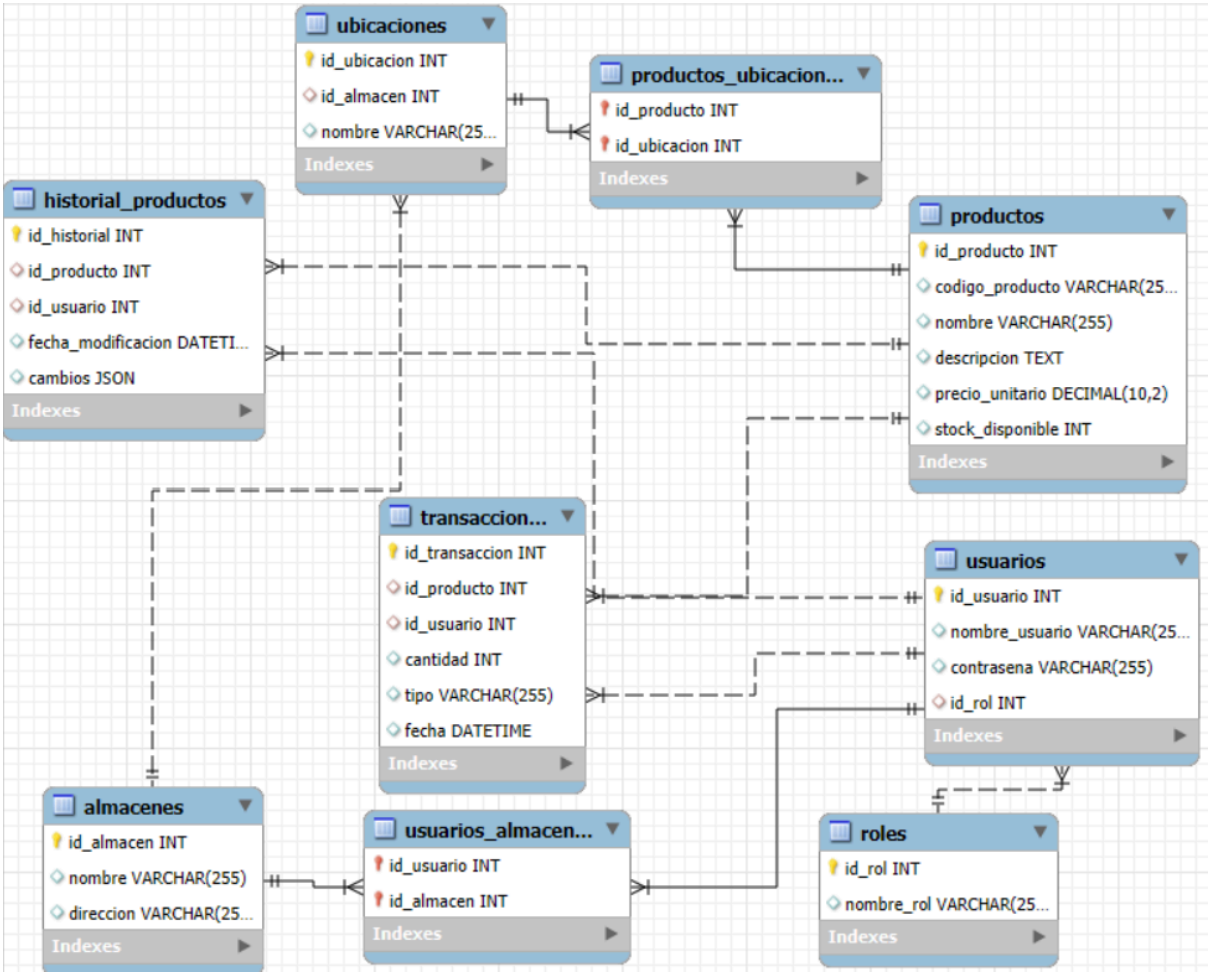
Historial_Productos:

- `id_historial` (Clave Primaria, INT, AUTO_INCREMENT)
- `id_producto` (Clave Foránea, INT)
- `fecha_modificacion` (DATETIME) [Valor por defecto: `CURRENT_TIMESTAMP`]
- `cambios` (JSON)

Relaciones:

Un Producto puede estar en muchas Ubicaciones (muchos a muchos).
Un Almacén tiene muchas Ubicaciones (uno a muchos).
Una Ubicación tiene muchos Productos (muchos a muchos).
Un Producto tiene muchas Transacciones (uno a muchos).
Un Usuario tiene un Rol (uno a uno o uno a muchos).
Un Rol tiene muchos Usuarios (uno a muchos).
Un Producto tiene mucho Historial (uno a muchos).

2.1.2 Diagrama E-R (Imágen)



2.2 Implementación en MySQL

Se creó la base de datos Control_Inventario, definiendo sus tablas, relaciones, índices y restricciones.

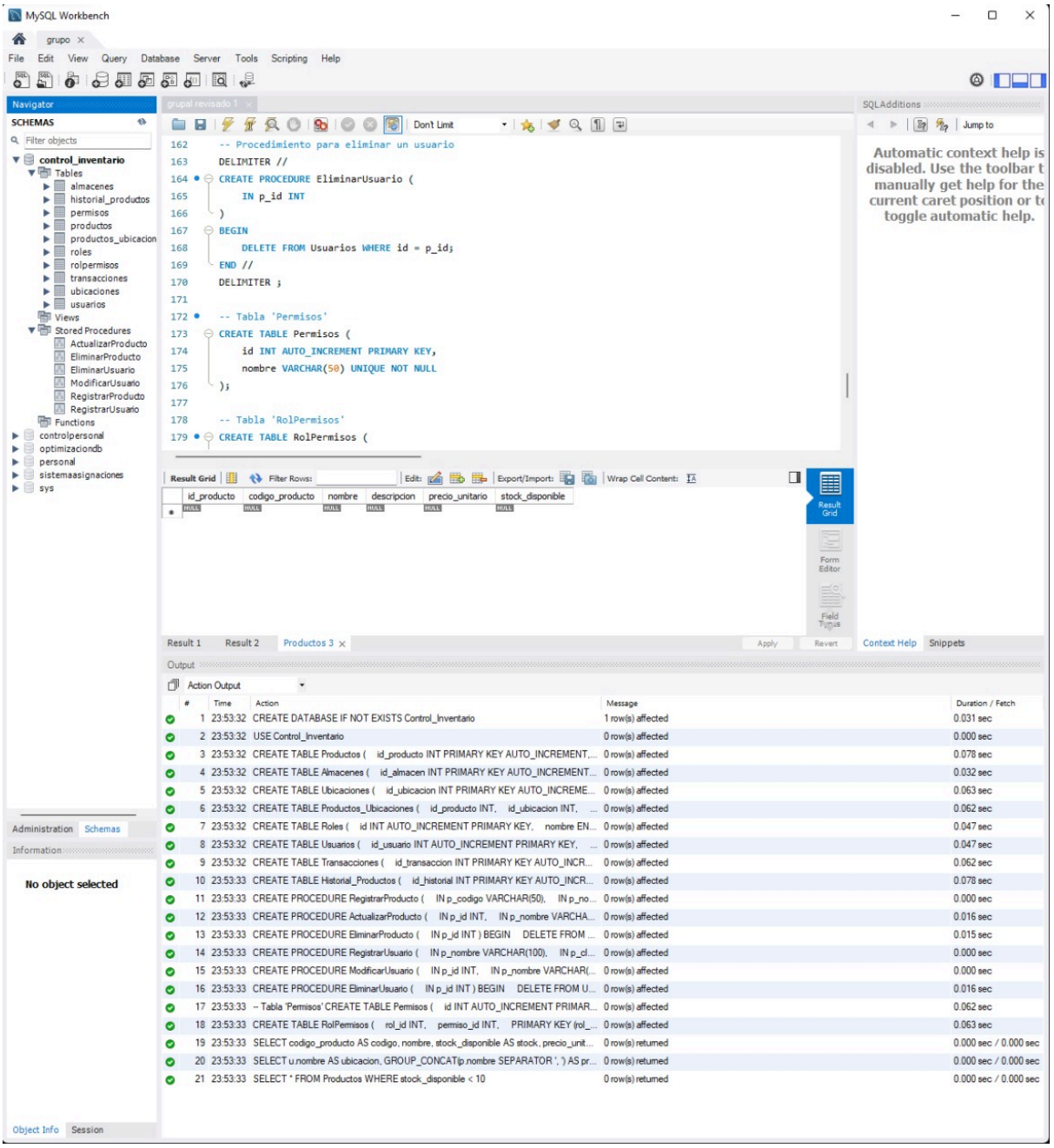
```
-- Crear base de datos 'Control_Inventario'

CREATE DATABASE IF NOT EXISTS Control_Inventario;

USE Control_Inventario;



-- Tabla 'Productos'

CREATE TABLE Productos (

    id_producto INT PRIMARY KEY AUTO_INCREMENT,

    codigo_producto VARCHAR(255) UNIQUE,

    nombre VARCHAR(255),

    descripcion TEXT,

    precio_unitario DECIMAL(10, 2),

    stock_disponible INT

);



-- Tabla 'Almacenes'

CREATE TABLE Almacenes (

    id_almacen INT PRIMARY KEY AUTO_INCREMENT,

    nombre VARCHAR(255),

    direccion VARCHAR(255)

);



-- Tabla 'Ubicaciones'
```

```
CREATE TABLE Ubicaciones (

    id_ubicacion INT PRIMARY KEY AUTO_INCREMENT,

    id_almacen INT,

    nombre VARCHAR(255),

    FOREIGN KEY (id_almacen) REFERENCES Almacenes(id_almacen)

);



-- Tabla 'Productos_Ubicaciones'

CREATE TABLE Productos_Ubicaciones (

    id_producto INT,

    id_ubicacion INT,

    PRIMARY KEY (id_producto, id_ubicacion),

    FOREIGN KEY (id_producto) REFERENCES Productos(id_producto),

    FOREIGN KEY (id_ubicacion) REFERENCES Ubicaciones(id_ubicacion)

);



-- Crear roles y usuarios

CREATE TABLE Roles (

    id INT AUTO_INCREMENT PRIMARY KEY,

    nombre ENUM('Administrador', 'Operador') UNIQUE NOT NULL

);



-- Tabla 'Usuarios'

CREATE TABLE Usuarios (

    id_usuario INT AUTO_INCREMENT PRIMARY KEY,
```

```
    nombre VARCHAR(100) NOT NULL,
    rol ENUM('Administrador', 'Operador') NOT NULL,
    clave_hash VARCHAR(255) NOT NULL,
    rol_id INT NOT NULL,
    FOREIGN KEY (rol_id) REFERENCES Roles(id) ON DELETE CASCADE
);

-- Tabla 'Transacciones'
CREATE TABLE Transacciones (
    id_transaccion INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT,
    id_usuario INT, -- Relación con Usuarios
    cantidad INT,
    tipo VARCHAR(255),
    fecha DATETIME,
    FOREIGN KEY (id_producto) REFERENCES Productos(id_producto),
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario), -- Nueva
relación
    INDEX (id_producto)
);

-- Tabla 'Historial_Productos'
CREATE TABLE Historial_Productos (
    id_historial INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT,
```

```
    id_usuario INT, -- Relación con Usuarios
    fecha_modificacion DATETIME DEFAULT CURRENT_TIMESTAMP,
    cambios JSON,
    FOREIGN KEY (id_producto) REFERENCES Productos(id_producto),
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario) -- Nueva relación
);

-- Tabla 'Permisos'
CREATE TABLE Permisos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) UNIQUE NOT NULL
);

-- Tabla 'RolPermisos'
CREATE TABLE RolPermisos (
    rol_id INT,
    permiso_id INT,
    PRIMARY KEY (rol_id, permiso_id),
    FOREIGN KEY (rol_id) REFERENCES Roles(id) ON DELETE CASCADE,
    FOREIGN KEY (permiso_id) REFERENCES Permisos(id) ON DELETE CASCADE
);

-- Ejemplo de asignación de permisos --
```

```
INSERT INTO Permisos (nombre) VALUES

('Gestionar Usuarios'),

 ('Ver Reportes');

-- INSERT INTO RolPermisos (rol_id, permiso_id)

 VALUES (1, 1), (1, 2), (2, 2); -- Admin tiene todos, Operador solo
reportes
```



3. Implementación de Funcionalidades

3.1 Procedimientos Almacenados

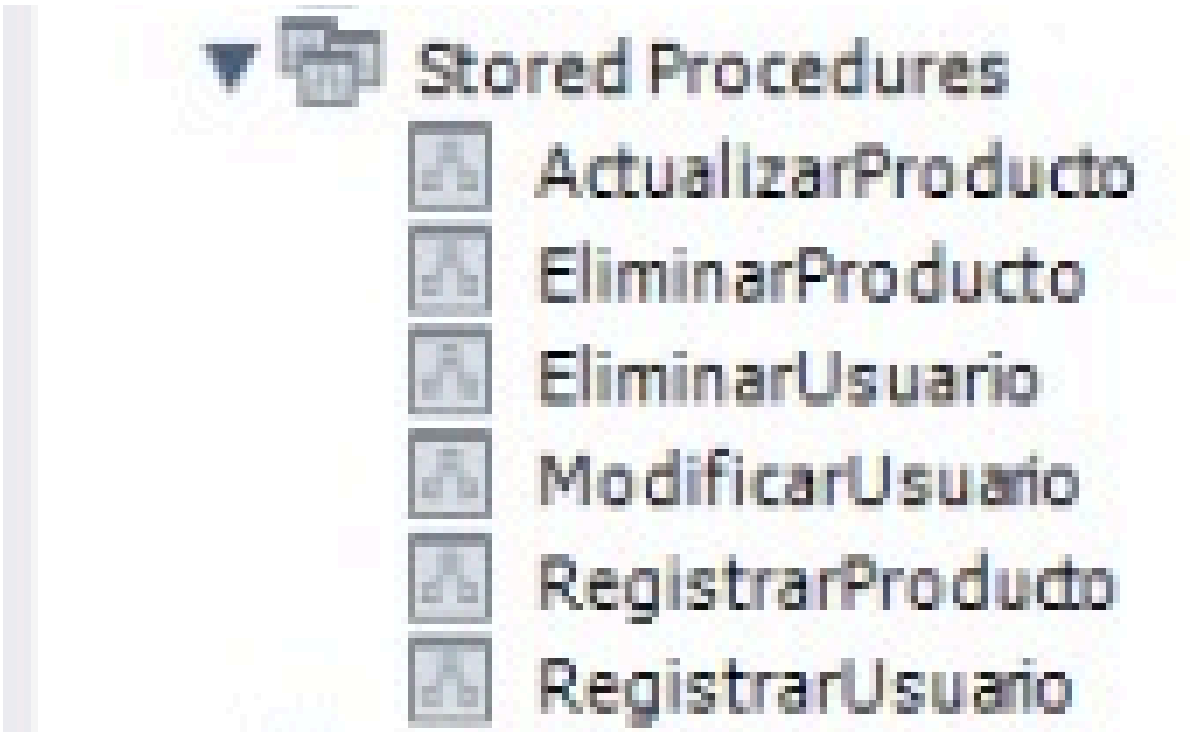
Se implementaron procedimientos almacenados para las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de productos y usuarios.

```
-- Procedimiento para registrar un producto
DELIMITER //

CREATE PROCEDURE RegistrarProducto (

    IN p_codigo VARCHAR(50),

    IN p_nombre VARCHAR(100),

    IN p_descripcion TEXT,

    IN p_precio DECIMAL(10, 2),

    IN p_stock INT,

    IN p_ubicacion VARCHAR(100)

)

BEGIN

    INSERT INTO Productos (codigo, nombre, descripcion, precio, stock,
ubicacion)

    VALUES (p_codigo, p_nombre, p_descripcion, p_precio, p_stock,
p_ubicacion);

END //

DELIMITER ;



-- Procedimiento para actualizar un producto

DELIMITER //

CREATE PROCEDURE ActualizarProducto (
```

```
    IN p_id INT,
    IN p_nombre VARCHAR(100),
    IN p_descripcion TEXT,
    IN p_precio DECIMAL(10, 2),
    IN p_stock INT,
    IN p_ubicacion VARCHAR(100)
)
BEGIN
    UPDATE Productos
    SET nombre = p_nombre, descripcion = p_descripcion, precio =
p_precio, stock = p_stock, ubicacion = p_ubicacion
    WHERE id_producto = p_id;
END //
DELIMITER ;

-- Procedimiento para eliminar un producto
DELIMITER //
CREATE PROCEDURE EliminarProducto (
    IN p_id INT
)
BEGIN
    DELETE FROM Productos WHERE id_producto = p_id;
END //
DELIMITER ;
```

```sql
-- Procedimiento para registrar un usuario

DELIMITER //

CREATE PROCEDURE RegistrarUsuario (

    IN p_nombre VARCHAR(100),

    IN p_clave_hash VARCHAR(255),

    IN p_rol_nombre ENUM('Administrador', 'Operador')

)

BEGIN

    DECLARE v_rol_id INT;



    -- Obtener el ID del rol

    SELECT id INTO v_rol_id FROM Roles WHERE nombre = p_rol_nombre;



    -- Insertar el usuario

    INSERT INTO Usuarios (nombre, clave_hash, rol_id)

    VALUES (p_nombre, p_clave_hash, v_rol_id);

END //

DELIMITER ;



-- Procedimiento para modificar un usuario

DELIMITER //

CREATE PROCEDURE ModificarUsuario (

    IN p_id INT,

    IN p_nombre VARCHAR(100),

    IN p_clave_hash VARCHAR(255),
```

```
    IN p_rol_nombre ENUM('Administrador', 'Operador')
)
BEGIN
    DECLARE v_rol_id INT;

    -- Obtener el ID del rol
    SELECT id INTO v_rol_id FROM Roles WHERE nombre = p_rol_nombre;

    -- Actualizar el usuario
    UPDATE Usuarios
    SET nombre = p_nombre, clave_hash = p_clave_hash, rol_id = v_rol_id
    WHERE id = p_id;
END //
DELIMITER ;

-- Procedimiento para eliminar un usuario
DELIMITER //
CREATE PROCEDURE EliminarUsuario (
    IN p_id INT
)
BEGIN
    DELETE FROM Usuarios WHERE id = p_id;
END //
DELIMITER ;
```



3.2 Consultas Y Reportes

Ejemplo de consulta de inventario general:

```
SELECT codigo_producto AS codigo, nombre, stock_disponible AS stock,
precio_unitario AS precio,
```

4. Configuración en MongoDB

Se utilizaron colecciones de MongoDB para almacenar datos no relacionales y flexibles, complementando la base de datos MySQL. Las colecciones son:

- **almacenes**: para la información de los almacenes y sus ubicaciones.
- **comentarrios_productos**: para los comentarios sobre los productos.
- **historial_productos**: para el registro de los cambios en los productos.
- **productos**: para la información detallada de los productos.
- **inventario_roles**: para la información de los roles de los usuarios.
- **transacciones**: para el registro de las transacciones de los productos.
- **usuarios**: para la información de los usuarios del sistema.