MANUAL BASE DE DATOS INVENTORY

ABEL AUDINO PANTOJA RODRIGUEZ

Presentado a Brayan Arcos

INSTITUTO TECNOLOGICO DEL PUTUMAYO
DESARROLLO DE BASE DE DATOS
MOCOA-PUTUMAYO
2024

Tabla de contenido

INTRODUCCIÓN

Inventory3 es una robusta solución de gestión de inventario diseñada para optimizar las operaciones de cualquier local pequeño. Con esta herramienta integral permitirá un control preciso de los stocks, desde la recepción de mercancías y la generación de órdenes de compra hasta la emisión de facturas y el seguimiento del flujo de pagos. Con Inventory3, las organizaciones pueden garantizar la disponibilidad de productos, garantiza la reducción de costos operativos y mejorar la toma de decisiones basadas en datos precisos y actualizados. Este manual será una guía a través de la estructura interna del sistema, sus funcionalidades clave y las mejores prácticas para aprovechar al máximo todas sus capacidades.

Ha sido diseñado para adaptarse a las necesidades específicas de cada usuario, Inventory3 ofrece una lógica clara y organizada que facilita la búsqueda, actualización y análisis de datos. Desde la localización precisa de productos en el almacén hasta la generación de informes detallados, Inventory3 proporciona una visión completa y actualizada del estado de su inventario.

DESCRIPCIÓN GENERAL

Modelo Relacional

La base de datos está compuesta por 22 tablas interconectadas, organizadas en módulos clave:

Usuarios y Roles: Gestión de permisos y autenticación.
Productos y Categorías: Control del inventario y precios.
Órdenes de Compra: Manejo de compras y detalles de órdenes.
Facturación y Pagos: Emisión de facturas y registro de pagos.
Ubicación: Almacenamiento y asociación de productos con ubicaciones específicas.

Visión General del Sistema: Inventory3 está diseñada para gestionar información de productos, usuarios, roles, pedidos y pagos. Es útil para aplicaciones de gestión de inventarios y facturación.

Arquitectura General: La base de datos contiene tablas que organizan la información en módulos como usuarios, productos, pedidos, pagos, y roles. Utiliza claves foráneas para relacionar datos entre tablas.

Requisitos de Software y Hardware

Servidor de Base de Datos: requiere MySQL (versión recomendada 5.7 o superior), un servidor compatible, y espacio suficiente en disco para manejar el almacenamiento de inventario y datos de transacciones.
Cliente SQL: MySQL Workbench, DBeaver, o cualquier herramienta compatible.

Conocimientos Previos
Conceptos básicos de bases de datos relacionales.
Familiaridad con SQL (Structured Query Language).

Audiencia

Está dirigido a desarrolladores, administradores de bases de datos, y usuarios técnicos que necesiten gestionar la información contenida en Inventory3.

Características

- Relaciones bien definidas: Uso de claves foráneas para garantizar la integridad de los datos.
- Normalización: Tablas estructuradas para minimizar redundancias.
- Extensibilidad: Diseño adaptable a nuevas funciones o módulos. Por ejemplo, agregar nuevas tablas o relaciones para funcionalidades adicionales sin alterar significativamente el esquema existente.
- Escalabilidad: Para la base de datos Inventory3 es relevante porque se espera manejar un gran volumen de datos o un número creciente de usuarios y transacciones. Describiré las formas de abordar la escalabilidad para este caso:

- Escalabilidad Vertical:
  - Optimización del Hardware: Migrar la base de datos a un servidor con mayor capacidad (CPU, memoria RAM, almacenamiento rápido como SSD).
  - Optimización de Consultas: Asegurarse de que las consultas están bien diseñadas para minimizar el uso de recursos. Por ejemplo:
    - Utilizar índices en columnas frecuentemente consultadas, como userName o idProduct.
    - Evitar consultas con SELECT * y limitar las columnas solicitadas.
- Escalabilidad Horizontal
  - Fragmentación (Sharding): Dividir los datos en múltiples servidores según algún criterio, como rango de valores (id) o tipo de datos. Por ejemplo:
    - Almacenar los productos y las órdenes de compra en servidores diferentes.
  - Replicación de Base de Datos: Configurar una réplica de la base de datos para que las consultas de lectura se ejecuten en servidores secundarios, mientras que las operaciones de escritura se realizan en el principal.

ESTRUCTURA DE LA BASE DE DATOS

A continuación, se describen las tablas principales y sus relaciones:

Tabla **addresses**
Descripción: Almacena la información de direcciones asociadas a personas y ubicaciones.

Columnas:
id: Identificador único de la dirección (clave primaria).
street: Calle de la dirección (no nulo).
city: Ciudad de la dirección (no nulo).
state: Estado o región de la dirección.
zipCode: Código postal.
createdAt, updatedAt: Tiempos de creación y actualización de la entrada.

Tabla **identificationType**
Descripción: Define los tipos de identificación para personas (por ejemplo, “CC”, “CE”).

Columnas:
id: Identificador único del tipo de identificación (clave primaria).
type: Tipo de identificación (p.ej., “CC”, “CE”).
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **roles**
Descripción: Define los roles que pueden tener los usuarios (p.ej., administrador, cliente).

Columnas:
id: Identificador único del rol (clave primaria).
roleName: Nombre del rol.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **users**
Descripción: Almacena la información de los usuarios registrados.

Columnas:
id: Identificador único del usuario (clave primaria).
userName: Nombre del usuario (no nulo).
password: Contraseña (no nulo).
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **userRoles**
Descripción: Relaciona usuarios con roles, permitiendo asignar múltiples roles a cada usuario.

Columnas:
id: Identificador único de la relación (clave primaria).
idUser: Referencia al identificador en users.
idRole: Referencia al identificador en roles.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **people**
Descripción: Almacena la información de personas, asociadas a usuarios y direcciones.

Columnas:
id: Identificador único de la persona (clave primaria).
idUser: Referencia al usuario en users.
idType: Referencia al tipo de identificación en identificationType.
identificationNumber: Número de identificación de la persona.
firstName, middleName, lastName: Nombre y apellidos de la persona.
email, phone: Información de contacto.
idAddress: Referencia a la dirección en addresses.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **orderStatus**

Descripción: Define los posibles estados de las órdenes de compra (p.ej., pendiente, completado).

Columnas:
id: Identificador único del estado (clave primaria).
name: Nombre del estado.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **productCategories**
Descripción: Almacena las categorías de productos.

Columnas:
id: Identificador único de la categoría (clave primaria).
categoryName: Nombre de la categoría.
description: Descripción de la categoría.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **products**
Descripción: Almacena la información de productos disponibles en el inventario.

Columnas:
id: Identificador único del producto (clave primaria).
name: Nombre del producto.
description: Descripción detallada del producto.
price: Precio actual del producto.
idCategory: Referencia a la categoría en productCategories.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **purchaseOrders**
Descripción: Registra las órdenes de compra generadas por la empresa.

Columnas:
id: Identificador único de la orden de compra (clave primaria).
idSupplier: Referencia al proveedor en users.
idStatus: Estado de la orden (referencia en orderStatus).
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **orderDetails**
Descripción: Almacena los detalles de cada producto en una orden de compra.

Columnas:
id: Identificador único del detalle (clave primaria).
idPurchaseOrder: Referencia a la orden de compra en purchaseOrders.

idProduct: Referencia al producto en products.
quantity: Cantidad del producto.
price: Precio unitario al momento de la compra.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **historyProductPrices**
Descripción: Guarda el historial de precios de productos.

Columnas:
id: Identificador único del registro de precio (clave primaria).
idProduct: Referencia al producto en products.
price: Precio del producto en un momento específico.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **invoices**
Descripción: Almacena las facturas generadas para los clientes.

Columnas:
id: Identificador único de la factura (clave primaria).
idUser: Referencia al usuario que generó la factura en users.
total: Total de la factura.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **invoicesDetails**
Descripción: Contiene los detalles de los productos en cada factura.

Columnas:
id: Identificador único del detalle (clave primaria).
idInvoice: Referencia a la factura en invoices.
idProduct: Referencia al producto en products.
quantity: Cantidad de producto facturada.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **paymentMethods**
Descripción: Define los métodos de pago (p.ej., tarjeta, efectivo).

Columnas:
id: Identificador único del método de pago (clave primaria).
methodName: Nombre del método de pago.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **locations**

Descripción: Almacena la información de las ubicaciones físicas de almacenamiento.

Columnas:
id: Identificador único de la ubicación (clave primaria).
locationName: Nombre de la ubicación.
idAddress: Referencia a addresses.
phone, email: Información de contacto.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **paymentGateways**
Descripción: Define los gateways de pago disponibles para cada método de pago.

Columnas:
id: Identificador único del gateway de pago (clave primaria).
idPaymentMethod: Referencia a paymentMethods.
gateway: Nombre del gateway.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **paymentGatewaysLocation**
Descripción: Asocia un gateway de pago con una ubicación.

Columnas:
id: Identificador único de la relación (clave primaria).
idLocation: Referencia a locations.
idPaymentGateway: Referencia a paymentGateways.
createdAt, **updatedAt**: Fechas de creación y actualización.

Tabla **payments**
Descripción: Registra los pagos realizados para facturas.

Columnas:
id: Identificador único del pago (clave primaria).
idInvoice: Referencia a la factura en invoices.
amount: Monto del pago.
idPaymentMethod: Método de pago.
paymentDate: Fecha del pago.
status: Estado del pago (exitoso, pendiente, fallido).
idPaymentGatewaysLocation: Gateway de pago y ubicación asociados.
createdAt, **updatedAt**: Fechas de creación y actualización.

Tabla **parameters**
Descripción: Define parámetros adicionales para configuraciones del sistema.

Columnas:
id: Identificador único del parámetro (clave primaria).
code, **description**: Código y descripción del parámetro.
createdAt, updatedAt: Fechas de creación y actualización.

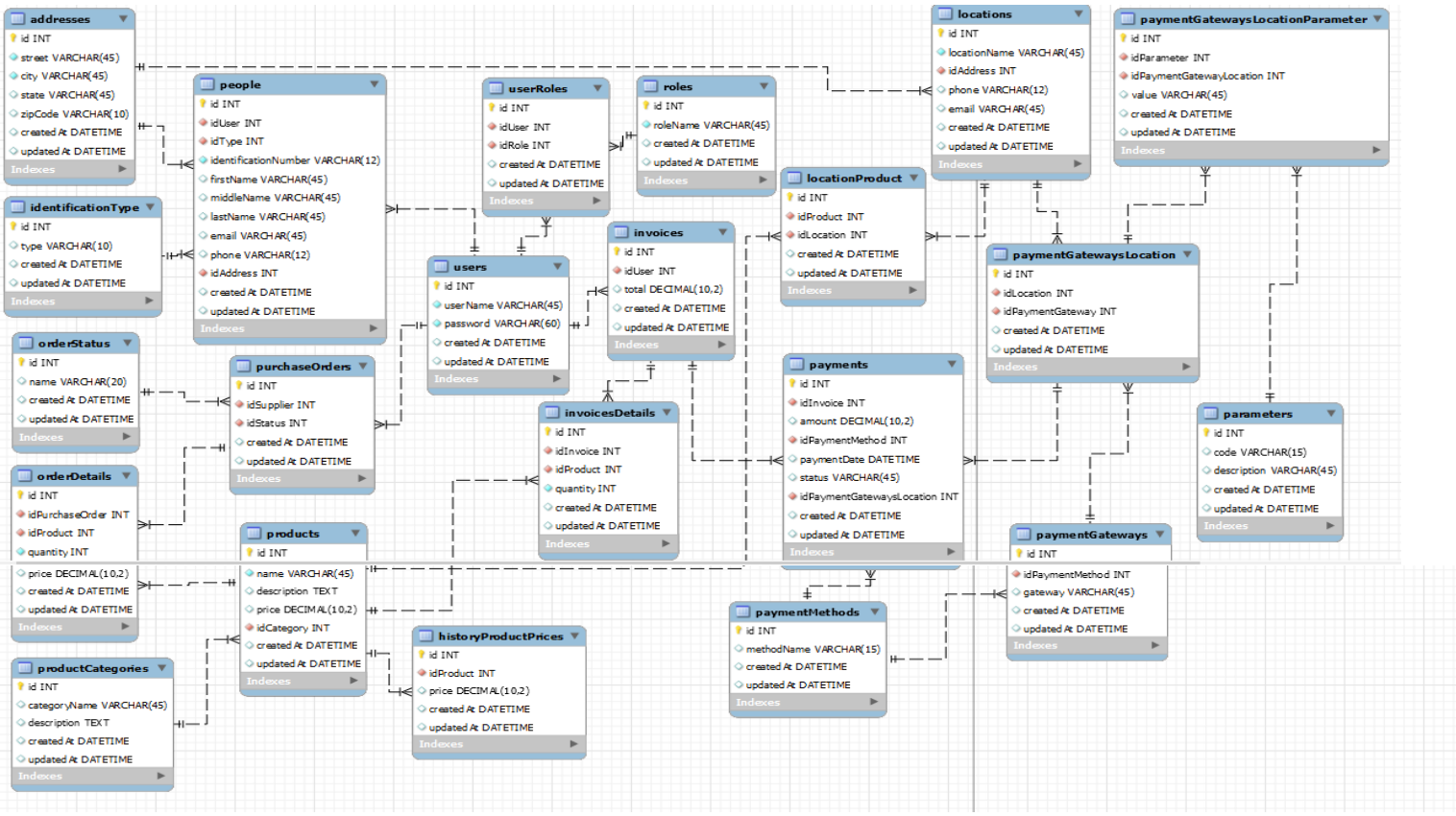
Tabla **paymentGatewaysLocationParameter**
Descripción: Define parámetros específicos para cada gateway de pago y ubicación.

Columnas:
id: Identificador único (clave primaria).
idParameter: Referencia a parameters.
idPaymentGatewayLocation: Gateway y ubicación asociados.
value: Valor del parámetro.
createdAt, updatedAt: Fechas de creación y actualización.

Tabla **locationProduct**
Descripción: Asocia productos con ubicaciones de almacenamiento.

Columnas:
id: Identificador único de la relación (clave primaria).
idProduct: Referencia a products.
idLocation: Referencia a locations.
createdAt, **updatedAt**: Fechas de creación y actualización.

DIAGRAMA DE ENTIDAD-RELACIÓN (ERD):



INSTRUCCIONES DE CONFIGURACIÓN

Instalación y Configuración: Crear la base de datos con CREATE DATABASE Inventory3 y ejecutar las instrucciones SQL.
Seguridad: Configurar permisos de usuario en MySQL para restringir el acceso.
Copia de Seguridad: Utilizar mysqldump para respaldar la base de datos periódicamente.

GUÍA DE OPERACIÓN

Tablas Principales

1. users
Propósito: Almacena información básica de usuarios del sistema.

Relaciones:

- Relacionada con userRoles para asignar roles.
- Referenciada por invoices para identificar al cliente

2. roles
Propósito: Define los roles que pueden asignarse a los usuarios.

3. userRoles
Propósito: Relaciona usuarios con roles específicos.
Relaciones:
idUser referencia a users(id).
idRole referencia a roles(id).

Tablas de Gestión de Productos

4. products
Propósito: Registro de productos disponibles en el inventario.

Relaciones:

- Relacionada con productCategories para clasificación.
- Referenciada por orderDetails, historyProductPrices, invoicesDetails.

5. productCategories
Propósito: Agrupa productos en categorías específicas.

Tablas de Facturación y Pagos
6. invoices
Propósito: Almacena la información de las facturas generadas.

7. payments
Propósito: Registro de pagos realizados para facturas.

Tablas de Órdenes de Compra

8. purchaseOrders
Propósito: Registro de órdenes de compra realizadas a proveedores.
Relaciones:
idSupplier referencia a users(id) para identificar al proveedor.
idStatus referencia a orderStatus(id) para el estado de la orden.

9. orderDetails
Propósito: Detalle de productos en cada orden de compra.
Relaciones:
idPurchaseOrder referencia a purchaseOrders(id).
idProduct referencia a products(id).

OPERACIONES BÁSICAS

- Consultas Comunes:

Obtener productos por categoría:

SELECT * FROM products WHERE idCategory = 1;

Buscar usuarios por rol
SELECT u.userName, r.roleName FROM users u
JOIN userRoles ur ON u.id = ur.idUser
JOIN roles r ON ur.idRole = r.id;

- Consultas con relaciones

Listar Productos por Categoría

SELECT p.name, c.categoryName
FROM products p
JOIN productCategories c ON p.idCategory = c.id;

Obtener Facturas y Pagos Asociados
SELECT i.id AS InvoiceID, i.total, p.amount, p.status
FROM invoices i
LEFT JOIN payments p ON i.id = p.idInvoice;

Historial de Precios de un Producto
SELECT p.name, h.price, h.createdAt
FROM products p
JOIN historyProductPrices h ON p.id = h.idProduct
WHERE p.id = 1;

PROCEDIMIENTOS DE ACTUALIZACIÓN Y MIGRACIÓN

Actualización: Al modificar estructuras de tablas, asegúrate de actualizar cualquier índice o clave foránea afectada.

Migración de Datos: Ejecutar en un entorno de pruebas antes de la producción.

Mantenimiento

Respaldo y Restauración

Respaldo: Utiliza mysqldump para exportar la base de datos completa.
Se hace uso del siguiente código sql para un respaldo completo:
mysqldump -u [usuario] -p Inventory3 > backup_inventory3.sql

para un respaldo de una tabla específica:
mysqldump -u [usuario] -p Inventory3 products > backup_products.sql

Restauración: Restaurar el respaldo completo se usa el comando:
mysql -u [usuario] -p Inventory3 < backup_inventory3.sql

SOLUCIÓN DE PROBLEMAS COMUNES

Error en claves foráneas: Revisar que los datos referenciados existan antes de insertarlos.
Causa: Intento de insertar un valor en una tabla relacionada que no existe en la tabla principal.
Solución: Verifica la existencia de valores en la tabla principal antes de insertar:
SELECT * FROM productCategories WHERE id = 1;

Problemas de rendimiento: Usar índices en campos clave como idUser en userRoles o idCategory en products.
Podemos crear índices para columnas utilizadas frecuentemente en búsquedas, como idCategory en products.
Ejemplo:
Agregar índices para columnas frecuentemente utilizadas en búsquedas:
CREATE INDEX idx_product_name ON products(name);

Error: "Cannot Add or Update a Child Row": verifica que las claves foráneas estén correctamente relacionadas y los valores existan en las tablas referenciadas.

GLOSARIO

Primary Key:Campo o conjunto de campos que identifica de manera única cada registro dentro de una tabla. Es un identificador único que no puede ser nulo ni repetido.

Foreign Key: Campo que establece relaciones entre dos tablas al referenciar la clave primaria de otra tabla. Esto garantiza la integridad referencial al evitar que existan valores en la tabla hija que no correspondan a registros en la tabla padre.

Index (Índice): Estructura que optimiza la velocidad de las consultas en una tabla. Los índices permiten acceder más rápido a los datos de columnas específicas, especialmente en tablas con gran cantidad de registros. Sin embargo, deben usarse con moderación ya que pueden aumentar el tiempo de inserción y actualización.

Normalization (Normalización): Proceso de estructurar las tablas de una base de datos para minimizar la redundancia y garantizar la integridad de los datos. Implica dividir la información en tablas relacionadas y eliminar dependencias innecesarias.

Integrity Constraints (Restricciones de Integridad): Reglas que garantizan la precisión y consistencia de los datos en la base de datos. Estas incluyen claves primarias, foráneas, restricciones de unicidad (UNIQUE), y reglas de validación (CHECK).

Auto-Increment: Propiedad que permite generar automáticamente valores únicos y consecutivos para un campo en una tabla, generalmente usado en claves primarias.

ANEXOS

Script de la base de datos inventory3:

```
DROP SCHEMA IF EXISTS Inventory3;
CREATE DATABASE IF NOT EXISTS Inventory3;
USE Inventory3;
-- Crear la tabla de direcciones
CREATE TABLE addresses (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    street VARCHAR(45) NOT NULL,
    city VARCHAR(45) NOT NULL,
    state VARCHAR(45),
    zipCode VARCHAR(10),
    createdAt DATETIME,
    updatedAt DATETIME
);

-- Tabla identificationType
CREATE TABLE identificationType (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    `type` VARCHAR(10),
    createdAt DATETIME,
    updatedAt DATETIME
);

-- Tabla roles
CREATE TABLE roles (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    roleName VARCHAR(45) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME
);

-- Tabla users
CREATE TABLE users (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    userName VARCHAR(45) NOT NULL,
    `password` VARCHAR(60) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME
);

-- Tabla userRoles (relación entre users y roles)
CREATE TABLE userRoles (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idUser INT(20) NOT NULL,
```

```
    idRole INT(20) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idUser) REFERENCES users(id),
    FOREIGN KEY (idRole) REFERENCES roles(id)
);

-- Tabla people
CREATE TABLE people (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idUser INT(20) NOT NULL,
    idType INT(20) NOT NULL,
    identificationNumber VARCHAR(12) NOT NULL,
    firstName VARCHAR(45),
    middleName VARCHAR(45),
    lastName VARCHAR(45),
    email VARCHAR(45),
    phone VARCHAR(12),
    idAddress INT(20) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idUser) REFERENCES users(id),
    FOREIGN KEY (idType) REFERENCES identificationType(id),
    FOREIGN KEY (idAddress) REFERENCES addresses(id)
);

-- Tabla orderStatus
CREATE TABLE orderStatus (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(20),
    createdAt DATETIME,
    updatedAt DATETIME
);

-- Tabla productCategories
CREATE TABLE productCategories (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    categoryName VARCHAR(45),
    `description` TEXT,
    createdAt DATETIME,
    updatedAt DATETIME
);

-- Tabla products
CREATE TABLE products (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(45) NOT NULL,
```

```
    `description` TEXT,
    price DECIMAL(10,2),
    idCategory INT(20) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idCategory) REFERENCES productCategories(id)
);

-- Tabla purchaseOrders
CREATE TABLE purchaseOrders (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idSupplier INT(20) NOT NULL,
    idStatus INT(20) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idSupplier) REFERENCES users(id),
    FOREIGN KEY (idStatus) REFERENCES orderStatus(id)
);

-- Tabla orderDetails
CREATE TABLE orderDetails (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idPurchaseOrder INT(20) NOT NULL,
    idProduct INT(20) NOT NULL,
    quantity INT(20) NOT NULL,
    price DECIMAL(10,2),
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idPurchaseOrder) REFERENCES purchaseOrders(id),
    FOREIGN KEY (idProduct) REFERENCES products(id)
);

-- Tabla historyProductPrices
CREATE TABLE historyProductPrices (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idProduct INT(20) NOT NULL,
    price DECIMAL(10,2),
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idProduct) REFERENCES products(id)
);

-- Tabla invoices
CREATE TABLE invoices (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idUser INT(20) NOT NULL,
    total DECIMAL(10,2),
```

```
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idUser) REFERENCES users(id)
);

-- Tabla invoicesDetails
CREATE TABLE invoicesDetails (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idInvoice INT(20) NOT NULL,
    idProduct INT(20) NOT NULL,
    quantity INT(20) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idInvoice) REFERENCES invoices(id),
    FOREIGN KEY (idProduct) REFERENCES products(id)
);

-- Tabla pay_methods
CREATE TABLE paymentMethods (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    methodName VARCHAR(15),
    createdAt DATETIME,
    updatedAt DATETIME
);

-- Tabla locations
CREATE TABLE locations (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    locationName VARCHAR(45) NOT NULL,
    idAddress INT(20) NOT NULL,
    phone VARCHAR(12),
    email VARCHAR(45),
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idAddress) REFERENCES addresses(id)
);

-- Tabla paymentGateways
CREATE TABLE paymentGateways (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idPaymentMethod INT(20) NOT NULL,
    gateway VARCHAR(45),
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idPaymentMethod) REFERENCES paymentMethods(id)
);
```

```
-- Tabla paymentGatewaysLocation
CREATE TABLE paymentGatewaysLocation (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idLocation INT(20) NOT NULL,
    idPaymentGateway INT(20) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idLocation) REFERENCES locations(id),
    FOREIGN KEY (idPaymentGateway) REFERENCES paymentGateways(id)
);

-- Tabla payments
CREATE TABLE payments (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idInvoice INT(20) NOT NULL,
    amount DECIMAL(10,2),
    idPaymentMethod INT(20) NOT NULL,
    paymentDate DATETIME,
    `status` VARCHAR(45), -- exitoso, pendiente o fallido
    idPaymentGatewaysLocation INT(20) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idInvoice) REFERENCES invoices(id),
    FOREIGN KEY (idPaymentGatewaysLocation) REFERENCES
paymentGatewaysLocation(Id),
    FOREIGN KEY (idPaymentMethod) REFERENCES paymentMethods(id)
);

-- Tabla parameters
CREATE TABLE parameters (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    `code` VARCHAR(15),
    `description` VARCHAR(45),
    createdAt DATETIME,
    updatedAt DATETIME
);

-- Tabla paymentGatewaysLocationParameter
CREATE TABLE paymentGatewaysLocationParameter (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idParameter INT(20) NOT NULL,
    idPaymentGatewayLocation INT(20) NOT NULL,
    `value` VARCHAR(45),
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idParameter) REFERENCES parameters(id),
```

```
    FOREIGN KEY (idPaymentGatewayLocation) REFERENCES paymentGatewaysLocation(id)
);
-- Tabla locationProduct
CREATE TABLE locationProduct (
    id INT(20) PRIMARY KEY AUTO_INCREMENT,
    idProduct INT(20) NOT NULL,
    idLocation INT(20) NOT NULL,
    createdAt DATETIME,
    updatedAt DATETIME,
    FOREIGN KEY (idProduct) REFERENCES products(id),
    FOREIGN KEY (idLocation) REFERENCES locations(id)
);
```