Diseño de Base de Datos - TaskManager

**Autor: Carlos Alberto Bolaños Gamarra**

**Rodrigo Holgado Quispe**

**Fabricio Huillca Perez**

**Raul Ppacsi Chillihuani**

**Versión:** 2.2

# 1. Introducción

Este documento describe el modelo de base de datos relacional diseñado para el sistema Agencia de Viaje**s**, cuyo propósito es gestionar de forma integral las operaciones de ventas, clientes, paquetes turísticos, pagos, reportes contables, logística, operaciones y marketing. El diseño garantiza integridad de datos, escalabilidad y soporte a los diferentes procesos administrativos, comerciales y operativos de la empresa.

El diseño de la Base de Datos de TaskManager es esencial para asegurar que el sistema funcione de manera eficiente y escalable. En este documento, se presenta la estructura que soporta todas las funciones del sistema, como la gestión de tareas y usuarios. Utilizando PostgreSQL como sistema de gestión de bases de datos, se garantiza la integridad de los datos y la capacidad de manejar grandes volúmenes a medida que el sistema crece

La estructura está pensada para ser sencilla pero flexible, permitiendo su expansión conforme se agreguen más funcionalidades. Además, se ha integrado una capa de seguridad que protege la información y garantiza el cumplimiento de buenas prácticas para el manejo de datos sensibles.

En resumen, este diseño de base de datos proporciona una solución robusta y segura que facilita la administración de las tareas y recursos dentro del sistema TaskManager, mientras que mantiene la eficiencia y la facilidad de uso para los desarrolladores y usuarios finales.

# 2. Diagrama Entidad-Relación (ERD)

## 2.1. Entidades y sus Atributos

### 2.1. Entidades y sus atributos

**Usuarios**

* id\_usuario (PK)
* nombre
* email (único)
* rol *(gerente, ventas, contabilidad, operaciones, logistica, marketing)*
* fecha\_registro

**Clientes**

* **id\_cliente (PK)**
* **nombres**
* **apellidos**
* **email**
* **teléfono**
* **documento**

**Paquetes**

* **id\_paquete (PK)**
* **nombre**
* **descripcion**
* **precio**
* **disponibilidad**
* **condiciones**

**Ventas**

* **id\_venta (PK)**
* **id\_cliente (FK → Clientes)**
* **id\_asesor (FK → Usuarios)**
* **es\_grupal**
* **num\_personas**
* **estado *(en\_proceso, pendiente\_pago, finalizado, anulado)***
* **total**
* **fecha\_creacion**

**Venta\_Detalles *(resuelve N:M Ventas–Paquetes)***

* **id\_detalle (PK)  
  id\_venta (FK → Ventas)**
* **id\_paquete (FK → Paquetes)**
* **cantidad**
* **precio\_unit**
* **subtotal**

**Pagos**

* **id\_pago (PK)**
* **id\_venta (FK → Ventas)**
* **monto**
* **estado *(pendiente, parcial, pagado, fallido, devuelto)***
* **medio**
* **referencia**
* **fecha\_pago**

**Reportes\_Contables**

* **id\_reporte (PK)**
* **fecha**
* **total\_ventas**
* **total\_pagos**
* **total\_deudas**
* **generado\_por (FK → Usuarios)**

**Activos**

* **id\_activo (PK)**
* **tipo *(vehiculo, mobiliario, equipo)***
* **codigo (único)**
* **descripcion**
* **ubicacion**
* **estado**
* **responsable\_id (FK → Usuarios)**

**Mantenimientos**

* **id\_mantenimiento (PK)**
* **id\_activo (FK → Activos)**
* **tipo *(preventivo, correctivo)***
* **estado *(programado, en\_ejecucion, culminado, cancelado)***
* **programado\_para**
* **iniciado\_en**
* **finalizado\_en**
* **responsable\_id (FK → Usuarios)**
* **notas**

**Inventario**

* **id\_item (PK)**
* **sku (único)**
* **nombre**
* **descripcion**
* **unidad**
* **stock**
* **stock\_min**
* **ubicacion**
* **responsable\_id (FK → Usuarios)**

**Movimientos\_Inventario**

* **id\_mov (PK)**
* **id\_item (FK → Inventario)**
* **tipo *(IN, OUT, AJUSTE)***
* **cantidad**
* **motivo**
* **referencia**
* **realizado\_por (FK → Usuarios)**
* **fecha\_mov**

**Campañas**

* **id\_campaña (PK)**
* **nombre**
* **canal *(social, email, afiliados, interno, otro)***
* **mensaje**
* **fecha\_inicio**
* **fecha\_fin**
* **presupuesto**
* **creado\_por (FK → Usuarios)**

**Campaña\_Assets**

* **id\_asset (PK)**
* **id\_campaña (FK → Campañas)**
* **tipo *(imagen, video, copy, landing, promo)***
* **url**
* **descripcion**

## 2.2. Relaciones

A continuación se detallan las relaciones entre las entidades descritas en la base de datos, incluyendo las claves primarias (PK) y las claves foráneas (FK) que aseguran la integridad referencial:

1. Usuarios - Ventas

* Un usuario (asesor de ventas) puede tener varias ventas asociadas.
* Relación: Usuarios (id\_usuario) → Ventas (id\_asesor) (1:N)
* La clave foránea id\_asesor en la tabla Ventas hace referencia a id\_usuario en la tabla Usuarios.

1. Clientes - Ventas

* Un cliente puede realizar múltiples ventas a lo largo del tiempo.
* Relación: Clientes (id\_cliente) → Ventas (id\_cliente) (1:N)
* La clave foránea id\_cliente en la tabla Ventas hace referencia a id\_cliente en la tabla Clientes.

1. Ventas - Paquetes (a través de Venta\_Detalles)

* Una venta puede involucrar múltiples paquetes turísticos y un paquete puede estar en muchas ventas.
* Relación: Ventas (id\_venta) ↔ Paquetes (id\_paquete) (N:M)
* La tabla Venta\_Detalles resuelve esta relación muchos a muchos, con las claves foráneas id\_venta y id\_paquete.

1. Ventas - Pagos

* Una venta puede tener varios pagos asociados.
* Relación: Ventas (id\_venta) → Pagos (id\_venta) (1:N)
* La clave foránea id\_venta en la tabla Pagos hace referencia a id\_venta en la tabla Ventas.

1. Usuarios - Reportes\_Contables

* Un usuario (generalmente el encargado de contabilidad) puede generar múltiples reportes contables.
* Relación: Usuarios (id\_usuario) → Reportes\_Contables (generado\_por) (1:N)
* La clave foránea generado\_por en la tabla Reportes\_Contables hace referencia a id\_usuario en la tabla Usuarios.

1. Usuarios - Activos

* Un usuario (encargado de operaciones) puede ser responsable de varios activos (vehículos, equipos, etc.).
* Relación: Usuarios (id\_usuario) → Activos (responsable\_id) (1:N)
* La clave foránea responsable\_id en la tabla Activos hace referencia a id\_usuario en la tabla Usuarios.

1. Activos - Mantenimientos

* Un activo (por ejemplo, un vehículo o equipo) puede tener varios mantenimientos.
* Relación: Activos (id\_activo) → Mantenimientos (id\_activo) (1:N)
* La clave foránea id\_activo en la tabla Mantenimientos hace referencia a id\_activo en la tabla Activos.

1. Usuarios - Mantenimientos

* Un usuario (responsable de mantenimiento) puede estar asociado a varios mantenimientos.
* Relación: Usuarios (id\_usuario) → Mantenimientos (responsable\_id) (1:N)
* La clave foránea responsable\_id en la tabla Mantenimientos hace referencia a id\_usuario en la tabla Usuarios.

1. Usuarios - Inventario

* Un usuario (responsable de logística) puede ser responsable de varios ítems en el inventario.
* Relación: Usuarios (id\_usuario) → Inventario (responsable\_id) (1:N)
* La clave foránea responsable\_id en la tabla Inventario hace referencia a id\_usuario en la tabla Usuarios.

1. Inventario - Movimientos\_Inventario

* Un ítem de inventario puede tener múltiples movimientos (entradas, salidas, ajustes).
* Relación: Inventario (id\_item) → Movimientos\_Inventario (id\_item) (1:N)
* La clave foránea id\_item en la tabla Movimientos\_Inventario hace referencia a id\_item en la tabla Inventario.

1. Usuarios - Movimientos\_Inventario

* Un usuario (responsable de inventario) puede registrar varios movimientos de inventario.
* Relación: Usuarios (id\_usuario) → Movimientos\_Inventario (realizado\_por) (1:N)
* La clave foránea realizado\_por en la tabla Movimientos\_Inventario hace referencia a id\_usuario en la tabla Usuarios.

1. Campañas - Usuarios

* Un usuario (generalmente del área de marketing) puede crear y gestionar varias campañas publicitarias.
* Relación: Usuarios (id\_usuario) → Campañas (creado\_por) (1:N)
* La clave foránea creado\_por en la tabla Campañas hace referencia a id\_usuario en la tabla Usuarios.

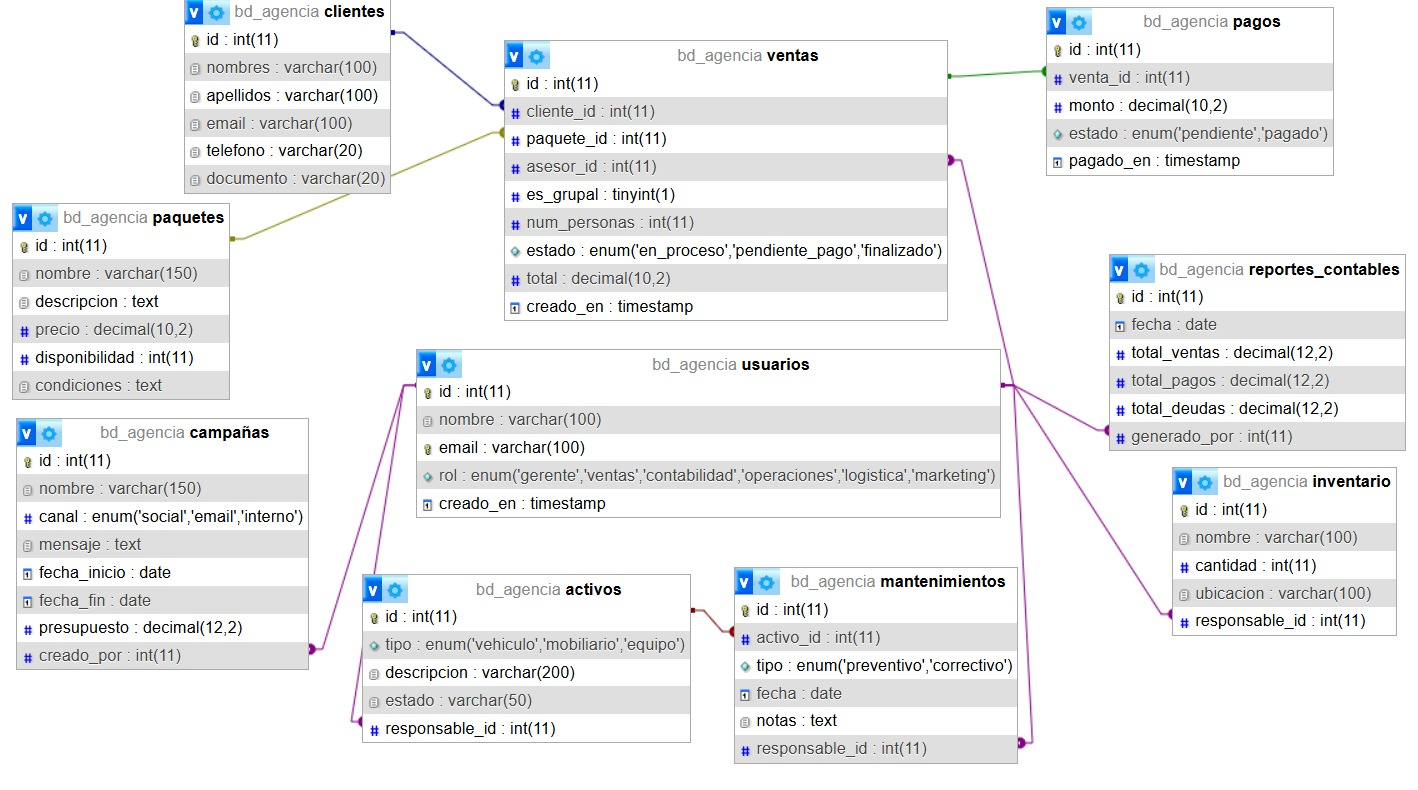
1. Campañas - Campaña\_Assets

* Una campaña puede tener varios activos asociados (imágenes, videos, copys, etc.).
* Relación: Campañas (id\_campaña) → Campaña\_Assets (id\_campaña) (1:N)
* La clave foránea id\_campaña en la tabla Campaña\_Assets hace referencia a id\_campaña en la tabla Campañas.

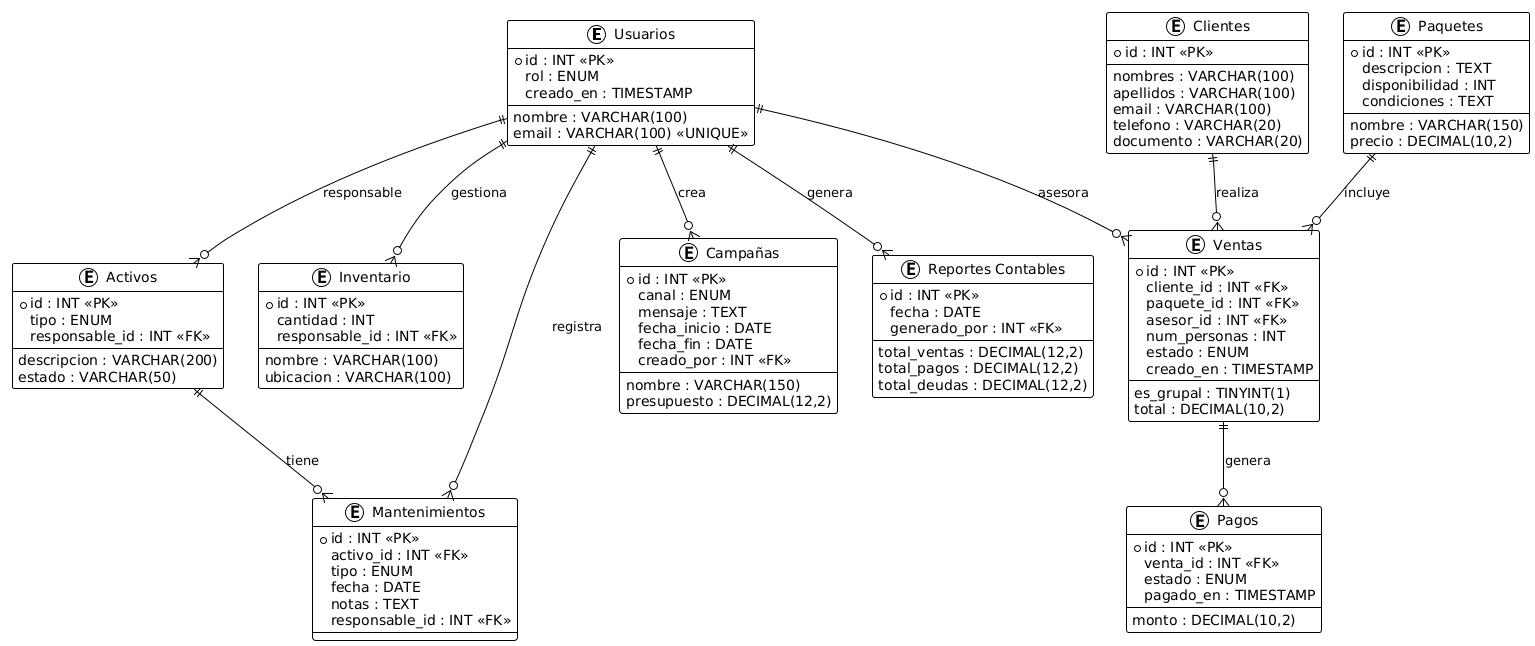
## 2.3. Reglas y Restricciones

* El **email de usuarios** debe ser único.
* Cada **venta** debe tener un cliente, un paquete y un asesor asignado.
* Los **pagos** dependen de ventas registradas.
* Los **reportes contables** sólo pueden ser generados por usuarios con rol de contabilidad.
* Un **activo o inventario** siempre debe estar bajo responsabilidad de un usuario.  
  Se evita duplicidad en asignación de roles críticos (ej. un mismo pago no puede repetirse en la misma venta).

## 2.4. Diagrama de Base de Datos (E/R)

* **Usuarios (1) → (N) Ventas (como asesores).**
* **Clientes (1) → (N) Ventas.**
* **Paquetes (1) → (N) Ventas.**
* **Ventas (1) → (N) Pagos.**
* **Usuarios (1) → (N) Reportes Contables.**
* **Usuarios (1) → (N) Activos.**
* **Activos (1) → (N) Mantenimientos.**
* **Usuarios (1) → (N) Inventario.**
* **Usuarios (1) → (N) Campañas.**

# 3. Diagrama Relacional



# 4. Diccionario de Datos

**Tabla Usuarios**

| **Campo** | **Tipo de Dato** | **Descripción** | **Restricciones** |
| --- | --- | --- | --- |
| id | INT | Identificador único del usuario del sistema. | PRIMARY KEY, AUTO\_INCREMENT |
| nombre | VARCHAR(100) | Nombre completo del usuario. | NOT NULL |
| email | VARCHAR(100) | Dirección de correo electrónico del usuario. | UNIQUE, NOT NULL |
| rol | ENUM(...) | Rol del usuario en la agencia (gerente, ventas, contabilidad, operaciones, logística, marketing). | NOT NULL |
| creado\_en | TIMESTAMP | Fecha y hora de creación del registro. | DEFAULT CURRENT\_TIMESTAMP |

**Tabla Clientes**

| **Campo** | **Tipo de Dato** | **Descripción** | **Restricciones** |
| --- | --- | --- | --- |
| id | INT | Identificador único del cliente. | PRIMARY KEY, AUTO\_INCREMENT |
| nombres | VARCHAR(100) | Nombres del cliente. | NOT NULL |
| apellidos | VARCHAR(100) | Apellidos del cliente. | NULLABLE |
| email | VARCHAR(100) | Correo electrónico del cliente. | NULLABLE |
| telefono | VARCHAR(20) | Número de teléfono del cliente. | NULLABLE |
| documento | VARCHAR(20) | Documento de identidad del cliente. | NULLABLE |

**Tabla Paquetes**

| **Campo** | **Tipo de Dato** | **Descripción** | **Restricciones** |
| --- | --- | --- | --- |
| id | INT | Identificador único del paquete turístico. | PRIMARY KEY, AUTO\_INCREMENT |
| nombre | VARCHAR(150) | Nombre del paquete turístico. | NOT NULL |
| descripcion | TEXT | Descripción detallada del paquete. | NULLABLE |
| precio | DECIMAL(10,2) | Precio del paquete turístico. | NOT NULL |
| disponibilidad | INT | Número de plazas o cupos disponibles. | NOT NULL |
| condiciones | TEXT | Condiciones o restricciones del paquete. | NULLABLE |

**Tabla Ventas**

| **Campo** | **Tipo de Dato** | **Descripción** | **Restricciones** |
| --- | --- | --- | --- |
| id | INT | Identificador único de la venta. | PRIMARY KEY, AUTO\_INCREMENT |
| cliente\_id | INT | Cliente que realiza la compra. | FOREIGN KEY (Clientes.id), NOT NULL |
| paquete\_id | INT | Paquete adquirido en la venta. | FOREIGN KEY (Paquetes.id), NOT NULL |
| asesor\_id | INT | Asesor de ventas responsable. | FOREIGN KEY (Usuarios.id), NOT NULL |
| es\_grupal | TINYINT(1) | Indica si la venta es grupal (1) o individual (0). | DEFAULT 0 |
| num\_personas | INT | Número de personas incluidas en la venta grupal. | NULLABLE |
| estado | ENUM(...) | Estado de la venta (en\_proceso, pendiente\_pago, finalizado). | DEFAULT 'en\_proceso' |
| total | DECIMAL(10,2) | Monto total de la venta. | NULLABLE |
| creado\_en | TIMESTAMP | Fecha y hora de registro de la venta. | DEFAULT CURRENT\_TIMESTAMP |

**Tabla Pagos**

| **Campo** | **Tipo de Dato** | **Descripción** | **Restricciones** |
| --- | --- | --- | --- |
| id | INT | Identificador único del pago. | PRIMARY KEY, AUTO\_INCREMENT |
| venta\_id | INT | Venta a la que pertenece el pago. | FOREIGN KEY (Ventas.id), NOT NULL |
| monto | DECIMAL(10,2) | Monto pagado. | NOT NULL |
| estado | ENUM(...) | Estado del pago (pendiente, pagado). | DEFAULT 'pendiente' |
| pagado\_en | TIMESTAMP | Fecha y hora en que se realizó el pago. | NULLABLE |

**Tabla Reportes Contables**

| **Campo** | **Tipo de Dato** | **Descripción** | **Restricciones** |
| --- | --- | --- | --- |
| id | INT | Identificador único del reporte contable. | PRIMARY KEY, AUTO\_INCREMENT |
| fecha | DATE | Fecha del reporte. | NOT NULL |
| total\_ventas | DECIMAL(12,2) | Monto total de ventas registradas. | NULLABLE |
| total\_pagos | DECIMAL(12,2) | Monto total de pagos recibidos. | NULLABLE |
| total\_deudas | DECIMAL(12,2) | Monto total de deudas pendientes. | NULLABLE |
| generado\_por | INT | Usuario del área contable que genera el reporte. | FOREIGN KEY (Usuarios.id) |

**Tabla Campaña**

| **Campo** | **Tipo de Dato** | **Descripción** | **Restricciones** |
| --- | --- | --- | --- |
| id | INT | Identificador único de la campaña. | PRIMARY KEY, AUTO\_INCREMENT |
| nombre | VARCHAR(150) | Nombre de la campaña de marketing. | NOT NULL |
| canal | ENUM(...) | Canal de difusión (social, email, interno). | NOT NULL |
| mensaje | TEXT | Mensaje de la campaña. | NULLABLE |
| fecha\_inicio | DATE | Fecha de inicio de la campaña. | NULLABLE |
| fecha\_fin | DATE | Fecha de finalización de la campaña. | NULLABLE |
| presupuesto | DECIMAL(12,2) | Presupuesto asignado a la campaña. | NULLABLE |
| creado\_por | INT | Usuario creador de la campaña. | FOREIGN KEY (Usuarios.id) |

## 4.1. Relaciones y Restricciones:

La estructura de la base de datos de TaskManager está diseñada para garantizar la integridad de los datos mediante relaciones claras entre las entidades, así como restricciones que aseguran la consistencia y seguridad de la información almacenada. A continuación, se detallan las relaciones y restricciones implementadas en el sistema:

Relaciones

1. Usuarios ↔ Ventas

* Un usuario (asesor de ventas) puede estar relacionado con varias ventas, pero una venta solo puede ser realizada por un único usuario.
* Restricción: La clave foránea id\_asesor en la tabla Ventas debe hacer referencia a un id\_usuario válido en la tabla Usuarios.

1. Clientes ↔ Ventas

* Un cliente puede realizar múltiples ventas, pero una venta está asociada a un solo cliente.
* Restricción: La clave foránea id\_cliente en la tabla Ventas debe hacer referencia a un id\_cliente válido en la tabla Clientes.

1. Ventas ↔ Paquetes (a través de Venta\_Detalles)

* Una venta puede involucrar varios paquetes, y un paquete puede estar asociado a múltiples ventas. La relación muchos a muchos se resuelve a través de la tabla Venta\_Detalles.
* Restricción: Las claves foráneas id\_venta en Venta\_Detalles deben hacer referencia a un id\_venta válido en la tabla Ventas, y las claves foráneas id\_paquete en Venta\_Detalles deben hacer referencia a un id\_paquete válido en la tabla Paquetes.

1. Ventas ↔ Pagos

* Una venta puede tener varios pagos asociados (por ejemplo, pagos parciales o completos), pero cada pago se refiere a una sola venta.
* Restricción: La clave foránea id\_venta en la tabla Pagos debe hacer referencia a un id\_venta válido en la tabla Ventas.

1. Usuarios ↔ Reportes\_Contables

* Un usuario (generalmente el encargado de contabilidad) puede generar varios reportes contables. Cada reporte contable está asociado a un único usuario.
* Restricción: La clave foránea generado\_por en la tabla Reportes\_Contables debe hacer referencia a un id\_usuario válido en la tabla Usuarios.

1. Usuarios ↔ Activos

* Un usuario (responsable de operaciones) puede ser responsable de varios activos. Cada activo tiene asignado un único usuario responsable.
* Restricción: La clave foránea responsable\_id en la tabla Activos debe hacer referencia a un id\_usuario válido en la tabla Usuarios.

1. Activos ↔ Mantenimientos

* Un activo puede tener múltiples mantenimientos (por ejemplo, mantenimiento preventivo o correctivo), pero cada mantenimiento está asociado a un solo activo.
* Restricción: La clave foránea id\_activo en la tabla Mantenimientos debe hacer referencia a un id\_activo válido en la tabla Activos.

1. Usuarios ↔ Mantenimientos

* Un usuario (encargado de mantenimiento) puede estar relacionado con múltiples mantenimientos. Cada mantenimiento es realizado por un único usuario responsable.
* Restricción: La clave foránea responsable\_id en la tabla Mantenimientos debe hacer referencia a un id\_usuario válido en la tabla Usuarios.

1. Usuarios ↔ Inventario

* Un usuario (responsable de logística) puede estar encargado de varios ítems en el inventario. Cada ítem tiene asignado un usuario responsable.
* Restricción: La clave foránea responsable\_id en la tabla Inventario debe hacer referencia a un id\_usuario válido en la tabla Usuarios.

1. Inventario ↔ Movimientos\_Inventario

* Un ítem de inventario puede tener múltiples movimientos (entradas, salidas o ajustes), pero cada movimiento está asociado a un solo ítem de inventario.
* Restricción: La clave foránea id\_item en la tabla Movimientos\_Inventario debe hacer referencia a un id\_item válido en la tabla Inventario.

1. Usuarios ↔ Movimientos\_Inventario

* Un usuario (responsable de inventario) puede registrar varios movimientos en el inventario. Cada movimiento está realizado por un único usuario.
* Restricción: La clave foránea realizado\_por en la tabla Movimientos\_Inventario debe hacer referencia a un id\_usuario válido en la tabla Usuarios.

1. Usuarios ↔ Campañas

* Un usuario (del área de marketing) puede crear y gestionar varias campañas publicitarias. Cada campaña tiene un único usuario creador.
* Restricción: La clave foránea creado\_por en la tabla Campañas debe hacer referencia a un id\_usuario válido en la tabla Usuarios.

1. Campañas ↔ Campaña\_Assets

* Una campaña puede tener varios activos asociados (como imágenes, videos, copys, etc.), pero cada activo está relacionado con una sola campaña.
* Restricción: La clave foránea id\_campaña en la tabla Campaña\_Assets debe hacer referencia a un id\_campaña válido en la tabla Campañas.

# 5. Consideraciones de Escalabilidad

Uso de **índices** en campos de búsqueda frecuente (email, cliente\_id, paquete\_id).

**ON DELETE CASCADE** en claves foráneas críticas (ej. pagos al eliminar una venta).

La base de datos está diseñada para **escalar horizontalmente** en caso de alto volumen de ventas o clientes.

División lógica entre áreas: ventas, contabilidad, logística, marketing, lo que facilita microservicios futuros.

# 6. Conclusiones

El modelo relacional de la **Agencia de Viajes** permite cubrir las áreas críticas de gestión:

* Comercial (clientes, ventas, pagos).
* Contable (reportes financieros).
* Operativa (activos, mantenimientos, inventario).
* Estratégica (campañas de marketing).

Con este diseño, la base de datos soporta la **integridad referencial**, garantiza seguridad en la trazabilidad de operaciones y está preparada para crecer en complejidad sin perder consistencia.

ANEXO. BASE DE DATOS

-- =====================================================

-- BASE DE DATOS AGENCIA DE VIAJES - VERSIÓN PROFESIONAL

-- =====================================================

USE agencia\_viajes;

-- Configuración de motor de almacenamiento y codificación

SET foreign\_key\_checks = 0;

-- =====================================================

-- TABLAS DE CONFIGURACIÓN Y CATÁLOGOS

-- =====================================================

-- Catálogo de países

CREATE TABLE IF NOT EXISTS paises (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_iso CHAR(2) NOT NULL UNIQUE,

nombre VARCHAR(100) NOT NULL,

activo BOOLEAN DEFAULT TRUE,

INDEX idx\_codigo\_iso (codigo\_iso)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Catálogo de ciudades

CREATE TABLE IF NOT EXISTS ciudades (

id INT AUTO\_INCREMENT PRIMARY KEY,

pais\_id INT NOT NULL,

nombre VARCHAR(100) NOT NULL,

activo BOOLEAN DEFAULT TRUE,

INDEX idx\_pais\_id (pais\_id),

FOREIGN KEY (pais\_id) REFERENCES paises(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE USUARIOS Y EMPLEADOS

-- =====================================================

CREATE TABLE IF NOT EXISTS departamentos (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(50) NOT NULL UNIQUE,

descripcion TEXT,

activo BOOLEAN DEFAULT TRUE,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS usuarios (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_empleado VARCHAR(20) NOT NULL UNIQUE,

nombres VARCHAR(100) NOT NULL,

apellidos VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

telefono VARCHAR(20),

departamento\_id INT NOT NULL,

rol ENUM('gerente','supervisor','ventas','contabilidad','operaciones','logistica','marketing') NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

activo BOOLEAN DEFAULT TRUE,

ultimo\_acceso TIMESTAMP NULL,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

actualizado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_email (email),

INDEX idx\_departamento (departamento\_id),

INDEX idx\_rol (rol),

FOREIGN KEY (departamento\_id) REFERENCES departamentos(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE CLIENTES

-- =====================================================

CREATE TABLE IF NOT EXISTS tipos\_documento (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo VARCHAR(10) NOT NULL UNIQUE,

descripcion VARCHAR(50) NOT NULL,

activo BOOLEAN DEFAULT TRUE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS clientes (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_cliente VARCHAR(20) NOT NULL UNIQUE,

nombres VARCHAR(100) NOT NULL,

apellidos VARCHAR(100) NOT NULL,

email VARCHAR(100),

telefono VARCHAR(20),

telefono\_alternativo VARCHAR(20),

tipo\_documento\_id INT,

numero\_documento VARCHAR(20),

fecha\_nacimiento DATE,

direccion TEXT,

ciudad\_id INT,

preferencias\_viaje TEXT,

activo BOOLEAN DEFAULT TRUE,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

actualizado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_email (email),

INDEX idx\_documento (tipo\_documento\_id, numero\_documento),

INDEX idx\_codigo\_cliente (codigo\_cliente),

FOREIGN KEY (tipo\_documento\_id) REFERENCES tipos\_documento(id) ON DELETE RESTRICT,

FOREIGN KEY (ciudad\_id) REFERENCES ciudades(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE PRODUCTOS TURÍSTICOS

-- =====================================================

CREATE TABLE IF NOT EXISTS categorias\_paquetes (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(100) NOT NULL UNIQUE,

descripcion TEXT,

activo BOOLEAN DEFAULT TRUE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS proveedores (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_proveedor VARCHAR(20) NOT NULL UNIQUE,

razon\_social VARCHAR(200) NOT NULL,

ruc VARCHAR(20),

telefono VARCHAR(20),

email VARCHAR(100),

direccion TEXT,

ciudad\_id INT,

activo BOOLEAN DEFAULT TRUE,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_codigo\_proveedor (codigo\_proveedor),

FOREIGN KEY (ciudad\_id) REFERENCES ciudades(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS paquetes (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_paquete VARCHAR(30) NOT NULL UNIQUE,

nombre VARCHAR(200) NOT NULL,

descripcion TEXT,

categoria\_id INT NOT NULL,

duracion\_dias INT NOT NULL DEFAULT 1,

precio\_base DECIMAL(12,2) NOT NULL,

precio\_venta DECIMAL(12,2) NOT NULL,

cupo\_minimo INT DEFAULT 1,

cupo\_maximo INT DEFAULT 50,

disponibilidad\_actual INT NOT NULL DEFAULT 0,

fecha\_inicio\_vigencia DATE,

fecha\_fin\_vigencia DATE,

incluye TEXT,

no\_incluye TEXT,

condiciones\_cancelacion TEXT,

proveedor\_id INT,

activo BOOLEAN DEFAULT TRUE,

creado\_por INT NOT NULL,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

actualizado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_codigo\_paquete (codigo\_paquete),

INDEX idx\_categoria (categoria\_id),

INDEX idx\_vigencia (fecha\_inicio\_vigencia, fecha\_fin\_vigencia),

INDEX idx\_precios (precio\_base, precio\_venta),

FOREIGN KEY (categoria\_id) REFERENCES categorias\_paquetes(id) ON DELETE RESTRICT,

FOREIGN KEY (proveedor\_id) REFERENCES proveedores(id) ON DELETE RESTRICT,

FOREIGN KEY (creado\_por) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Destinos incluidos en cada paquete

CREATE TABLE IF NOT EXISTS paquete\_destinos (

id INT AUTO\_INCREMENT PRIMARY KEY,

paquete\_id INT NOT NULL,

ciudad\_id INT NOT NULL,

orden\_visita INT DEFAULT 1,

dias\_estancia INT DEFAULT 1,

notas TEXT,

FOREIGN KEY (paquete\_id) REFERENCES paquetes(id) ON DELETE CASCADE,

FOREIGN KEY (ciudad\_id) REFERENCES ciudades(id) ON DELETE RESTRICT,

UNIQUE KEY unique\_paquete\_ciudad (paquete\_id, ciudad\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE VENTAS Y COTIZACIONES

-- =====================================================

CREATE TABLE IF NOT EXISTS cotizaciones (

id INT AUTO\_INCREMENT PRIMARY KEY,

numero\_cotizacion VARCHAR(30) NOT NULL UNIQUE,

cliente\_id INT NOT NULL,

asesor\_id INT NOT NULL,

fecha\_cotizacion DATE NOT NULL,

fecha\_vencimiento DATE NOT NULL,

subtotal DECIMAL(12,2) NOT NULL DEFAULT 0,

descuento DECIMAL(12,2) DEFAULT 0,

impuestos DECIMAL(12,2) DEFAULT 0,

total DECIMAL(12,2) NOT NULL DEFAULT 0,

estado ENUM('borrador','enviada','aprobada','rechazada','vencida') DEFAULT 'borrador',

observaciones TEXT,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

actualizado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_numero\_cotizacion (numero\_cotizacion),

INDEX idx\_cliente (cliente\_id),

INDEX idx\_asesor (asesor\_id),

INDEX idx\_estado (estado),

INDEX idx\_fecha\_vencimiento (fecha\_vencimiento),

FOREIGN KEY (cliente\_id) REFERENCES clientes(id) ON DELETE RESTRICT,

FOREIGN KEY (asesor\_id) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS cotizacion\_detalles (

id INT AUTO\_INCREMENT PRIMARY KEY,

cotizacion\_id INT NOT NULL,

paquete\_id INT NOT NULL,

cantidad\_personas INT NOT NULL DEFAULT 1,

precio\_unitario DECIMAL(12,2) NOT NULL,

descuento\_unitario DECIMAL(12,2) DEFAULT 0,

subtotal DECIMAL(12,2) NOT NULL,

observaciones TEXT,

FOREIGN KEY (cotizacion\_id) REFERENCES cotizaciones(id) ON DELETE CASCADE,

FOREIGN KEY (paquete\_id) REFERENCES paquetes(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS ventas (

id INT AUTO\_INCREMENT PRIMARY KEY,

numero\_venta VARCHAR(30) NOT NULL UNIQUE,

cotizacion\_id INT,

cliente\_id INT NOT NULL,

asesor\_id INT NOT NULL,

fecha\_venta DATE NOT NULL,

fecha\_viaje DATE,

numero\_personas INT NOT NULL DEFAULT 1,

subtotal DECIMAL(12,2) NOT NULL DEFAULT 0,

descuento DECIMAL(12,2) DEFAULT 0,

impuestos DECIMAL(12,2) DEFAULT 0,

total DECIMAL(12,2) NOT NULL DEFAULT 0,

estado ENUM('confirmada','en\_proceso','pagada','cancelada','finalizada') DEFAULT 'confirmada',

tipo\_venta ENUM('individual','grupal','corporativa') DEFAULT 'individual',

observaciones TEXT,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

actualizado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_numero\_venta (numero\_venta),

INDEX idx\_cliente (cliente\_id),

INDEX idx\_asesor (asesor\_id),

INDEX idx\_estado (estado),

INDEX idx\_fecha\_venta (fecha\_venta),

INDEX idx\_fecha\_viaje (fecha\_viaje),

FOREIGN KEY (cotizacion\_id) REFERENCES cotizaciones(id) ON DELETE RESTRICT,

FOREIGN KEY (cliente\_id) REFERENCES clientes(id) ON DELETE RESTRICT,

FOREIGN KEY (asesor\_id) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS venta\_detalles (

id INT AUTO\_INCREMENT PRIMARY KEY,

venta\_id INT NOT NULL,

paquete\_id INT NOT NULL,

cantidad\_personas INT NOT NULL DEFAULT 1,

precio\_unitario DECIMAL(12,2) NOT NULL,

descuento\_unitario DECIMAL(12,2) DEFAULT 0,

subtotal DECIMAL(12,2) NOT NULL,

observaciones TEXT,

FOREIGN KEY (venta\_id) REFERENCES ventas(id) ON DELETE CASCADE,

FOREIGN KEY (paquete\_id) REFERENCES paquetes(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE PAGOS Y FINANZAS

-- =====================================================

CREATE TABLE IF NOT EXISTS metodos\_pago (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo VARCHAR(10) NOT NULL UNIQUE,

descripcion VARCHAR(50) NOT NULL,

requiere\_referencia BOOLEAN DEFAULT FALSE,

activo BOOLEAN DEFAULT TRUE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS pagos (

id INT AUTO\_INCREMENT PRIMARY KEY,

numero\_pago VARCHAR(30) NOT NULL UNIQUE,

venta\_id INT NOT NULL,

metodo\_pago\_id INT NOT NULL,

monto DECIMAL(12,2) NOT NULL,

fecha\_programada DATE,

fecha\_pago DATE,

numero\_referencia VARCHAR(100),

estado ENUM('programado','recibido','confirmado','rechazado','anulado') DEFAULT 'programado',

observaciones TEXT,

procesado\_por INT,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

actualizado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_numero\_pago (numero\_pago),

INDEX idx\_venta (venta\_id),

INDEX idx\_estado (estado),

INDEX idx\_fecha\_programada (fecha\_programada),

INDEX idx\_fecha\_pago (fecha\_pago),

FOREIGN KEY (venta\_id) REFERENCES ventas(id) ON DELETE RESTRICT,

FOREIGN KEY (metodo\_pago\_id) REFERENCES metodos\_pago(id) ON DELETE RESTRICT,

FOREIGN KEY (procesado\_por) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE REPORTES Y CONTABILIDAD

-- =====================================================

CREATE TABLE IF NOT EXISTS tipos\_reporte (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo VARCHAR(20) NOT NULL UNIQUE,

nombre VARCHAR(100) NOT NULL,

descripcion TEXT,

activo BOOLEAN DEFAULT TRUE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS reportes\_contables (

id INT AUTO\_INCREMENT PRIMARY KEY,

numero\_reporte VARCHAR(30) NOT NULL UNIQUE,

tipo\_reporte\_id INT NOT NULL,

periodo\_inicio DATE NOT NULL,

periodo\_fin DATE NOT NULL,

total\_ventas DECIMAL(15,2) DEFAULT 0,

total\_pagos\_recibidos DECIMAL(15,2) DEFAULT 0,

total\_cuentas\_por\_cobrar DECIMAL(15,2) DEFAULT 0,

total\_gastos DECIMAL(15,2) DEFAULT 0,

utilidad\_bruta DECIMAL(15,2) DEFAULT 0,

observaciones TEXT,

generado\_por INT NOT NULL,

aprobado\_por INT,

fecha\_generacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

fecha\_aprobacion TIMESTAMP NULL,

INDEX idx\_numero\_reporte (numero\_reporte),

INDEX idx\_periodo (periodo\_inicio, periodo\_fin),

INDEX idx\_tipo\_reporte (tipo\_reporte\_id),

FOREIGN KEY (tipo\_reporte\_id) REFERENCES tipos\_reporte(id) ON DELETE RESTRICT,

FOREIGN KEY (generado\_por) REFERENCES usuarios(id) ON DELETE RESTRICT,

FOREIGN KEY (aprobado\_por) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE ACTIVOS Y OPERACIONES

-- =====================================================

CREATE TABLE IF NOT EXISTS tipos\_activo (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo VARCHAR(10) NOT NULL UNIQUE,

descripcion VARCHAR(50) NOT NULL,

vida\_util\_años INT DEFAULT 5,

activo BOOLEAN DEFAULT TRUE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS activos (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_activo VARCHAR(30) NOT NULL UNIQUE,

tipo\_activo\_id INT NOT NULL,

descripcion VARCHAR(200) NOT NULL,

marca VARCHAR(100),

modelo VARCHAR(100),

numero\_serie VARCHAR(100),

fecha\_adquisicion DATE,

valor\_adquisicion DECIMAL(12,2),

estado ENUM('operativo','mantenimiento','dañado','dado\_de\_baja') DEFAULT 'operativo',

ubicacion\_actual VARCHAR(100),

responsable\_id INT,

observaciones TEXT,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

actualizado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_codigo\_activo (codigo\_activo),

INDEX idx\_tipo\_activo (tipo\_activo\_id),

INDEX idx\_estado (estado),

INDEX idx\_responsable (responsable\_id),

FOREIGN KEY (tipo\_activo\_id) REFERENCES tipos\_activo(id) ON DELETE RESTRICT,

FOREIGN KEY (responsable\_id) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS mantenimientos (

id INT AUTO\_INCREMENT PRIMARY KEY,

numero\_mantenimiento VARCHAR(30) NOT NULL UNIQUE,

activo\_id INT NOT NULL,

tipo ENUM('preventivo','correctivo','predictivo') NOT NULL,

fecha\_programada DATE NOT NULL,

fecha\_ejecutada DATE,

descripcion TEXT NOT NULL,

costo DECIMAL(10,2) DEFAULT 0,

proveedor\_servicio VARCHAR(200),

estado ENUM('programado','en\_progreso','completado','cancelado') DEFAULT 'programado',

responsable\_id INT NOT NULL,

observaciones TEXT,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_numero\_mantenimiento (numero\_mantenimiento),

INDEX idx\_activo (activo\_id),

INDEX idx\_tipo (tipo),

INDEX idx\_estado (estado),

INDEX idx\_fecha\_programada (fecha\_programada),

FOREIGN KEY (activo\_id) REFERENCES activos(id) ON DELETE RESTRICT,

FOREIGN KEY (responsable\_id) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE INVENTARIO Y LOGÍSTICA

-- =====================================================

CREATE TABLE IF NOT EXISTS categorias\_inventario (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo VARCHAR(10) NOT NULL UNIQUE,

nombre VARCHAR(100) NOT NULL,

descripcion TEXT,

activo BOOLEAN DEFAULT TRUE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS almacenes (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_almacen VARCHAR(20) NOT NULL UNIQUE,

nombre VARCHAR(100) NOT NULL,

direccion TEXT,

ciudad\_id INT,

responsable\_id INT,

activo BOOLEAN DEFAULT TRUE,

INDEX idx\_codigo\_almacen (codigo\_almacen),

FOREIGN KEY (ciudad\_id) REFERENCES ciudades(id) ON DELETE RESTRICT,

FOREIGN KEY (responsable\_id) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS productos\_inventario (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_producto VARCHAR(30) NOT NULL UNIQUE,

nombre VARCHAR(150) NOT NULL,

descripcion TEXT,

categoria\_id INT NOT NULL,

unidad\_medida VARCHAR(20) DEFAULT 'UNIDAD',

stock\_minimo INT DEFAULT 0,

stock\_maximo INT DEFAULT 1000,

precio\_unitario DECIMAL(10,2) DEFAULT 0,

activo BOOLEAN DEFAULT TRUE,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_codigo\_producto (codigo\_producto),

INDEX idx\_categoria (categoria\_id),

FOREIGN KEY (categoria\_id) REFERENCES categorias\_inventario(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS inventario (

id INT AUTO\_INCREMENT PRIMARY KEY,

producto\_id INT NOT NULL,

almacen\_id INT NOT NULL,

cantidad\_actual INT NOT NULL DEFAULT 0,

fecha\_ultimo\_movimiento TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

responsable\_id INT,

FOREIGN KEY (producto\_id) REFERENCES productos\_inventario(id) ON DELETE RESTRICT,

FOREIGN KEY (almacen\_id) REFERENCES almacenes(id) ON DELETE RESTRICT,

FOREIGN KEY (responsable\_id) REFERENCES usuarios(id) ON DELETE RESTRICT,

UNIQUE KEY unique\_producto\_almacen (producto\_id, almacen\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS movimientos\_inventario (

id INT AUTO\_INCREMENT PRIMARY KEY,

numero\_movimiento VARCHAR(30) NOT NULL UNIQUE,

producto\_id INT NOT NULL,

almacen\_id INT NOT NULL,

tipo\_movimiento ENUM('entrada','salida','ajuste','transferencia') NOT NULL,

cantidad INT NOT NULL,

motivo VARCHAR(200),

documento\_referencia VARCHAR(100),

responsable\_id INT NOT NULL,

fecha\_movimiento TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_numero\_movimiento (numero\_movimiento),

INDEX idx\_producto (producto\_id),

INDEX idx\_almacen (almacen\_id),

INDEX idx\_tipo\_movimiento (tipo\_movimiento),

INDEX idx\_fecha\_movimiento (fecha\_movimiento),

FOREIGN KEY (producto\_id) REFERENCES productos\_inventario(id) ON DELETE RESTRICT,

FOREIGN KEY (almacen\_id) REFERENCES almacenes(id) ON DELETE RESTRICT,

FOREIGN KEY (responsable\_id) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- GESTIÓN DE MARKETING Y CAMPAÑAS

-- =====================================================

CREATE TABLE IF NOT EXISTS canales\_marketing (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo VARCHAR(10) NOT NULL UNIQUE,

nombre VARCHAR(50) NOT NULL,

descripcion TEXT,

activo BOOLEAN DEFAULT TRUE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS campañas (

id INT AUTO\_INCREMENT PRIMARY KEY,

codigo\_campaña VARCHAR(30) NOT NULL UNIQUE,

nombre VARCHAR(200) NOT NULL,

objetivo TEXT,

canal\_id INT NOT NULL,

mensaje TEXT,

fecha\_inicio DATE NOT NULL,

fecha\_fin DATE NOT NULL,

presupuesto\_asignado DECIMAL(12,2) DEFAULT 0,

presupuesto\_ejecutado DECIMAL(12,2) DEFAULT 0,

estado ENUM('planificada','activa','pausada','finalizada','cancelada') DEFAULT 'planificada',

publico\_objetivo TEXT,

metricas\_objetivo JSON,

resultados\_obtenidos JSON,

creado\_por INT NOT NULL,

aprobado\_por INT,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

actualizado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_codigo\_campaña (codigo\_campaña),

INDEX idx\_canal (canal\_id),

INDEX idx\_estado (estado),

INDEX idx\_fechas (fecha\_inicio, fecha\_fin),

FOREIGN KEY (canal\_id) REFERENCES canales\_marketing(id) ON DELETE RESTRICT,

FOREIGN KEY (creado\_por) REFERENCES usuarios(id) ON DELETE RESTRICT,

FOREIGN KEY (aprobado\_por) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- =====================================================

-- TRIGGERS PARA MANTENER CONSISTENCIA DE DATOS

-- =====================================================

DELIMITER //

-- Trigger para actualizar el stock cuando se registra un movimiento de inventario

CREATE TRIGGER tr\_actualizar\_stock\_inventario

AFTER INSERT ON movimientos\_inventario

FOR EACH ROW

BEGIN

IF NEW.tipo\_movimiento IN ('entrada', 'ajuste') THEN

UPDATE inventario

SET cantidad\_actual = cantidad\_actual + NEW.cantidad,

fecha\_ultimo\_movimiento = NEW.fecha\_movimiento

WHERE producto\_id = NEW.producto\_id AND almacen\_id = NEW.almacen\_id;

ELSEIF NEW.tipo\_movimiento = 'salida' THEN

UPDATE inventario

SET cantidad\_actual = cantidad\_actual - NEW.cantidad,

fecha\_ultimo\_movimiento = NEW.fecha\_movimiento

WHERE producto\_id = NEW.producto\_id AND almacen\_id = NEW.almacen\_id;

END IF;

END//

-- Trigger para actualizar totales de cotización cuando se modifican los detalles

CREATE TRIGGER tr\_actualizar\_total\_cotizacion

AFTER INSERT ON cotizacion\_detalles

FOR EACH ROW

BEGIN

UPDATE cotizaciones

SET subtotal = (

SELECT COALESCE(SUM(subtotal), 0)

FROM cotizacion\_detalles

WHERE cotizacion\_id = NEW.cotizacion\_id

),

total = subtotal - descuento + impuestos,

actualizado\_en = CURRENT\_TIMESTAMP

WHERE id = NEW.cotizacion\_id;

END//

-- Trigger para actualizar totales de venta cuando se modifican los detalles

CREATE TRIGGER tr\_actualizar\_total\_venta

AFTER INSERT ON venta\_detalles

FOR EACH ROW

BEGIN

UPDATE ventas

SET subtotal = (

SELECT COALESCE(SUM(subtotal), 0)

FROM venta\_detalles

WHERE venta\_id = NEW.venta\_id

),

total = subtotal - descuento + impuestos,

actualizado\_en = CURRENT\_TIMESTAMP

WHERE id = NEW.venta\_id;

END//

DELIMITER ;

-- =====================================================

-- INSERCIÓN DE DATOS INICIALES

-- =====================================================

-- Datos iniciales para departamentos

INSERT IGNORE INTO departamentos (nombre, descripcion) VALUES

('Gerencia', 'Dirección general de la empresa'),

('Ventas', 'Departamento de ventas y atención al cliente'),

('Contabilidad', 'Departamento de finanzas y contabilidad'),

('Operaciones', 'Departamento de operaciones y logística'),

('Marketing', 'Departamento de marketing y publicidad'),

('Sistemas', 'Departamento de tecnología e informática');

-- Datos iniciales para tipos de documento

INSERT IGNORE INTO tipos\_documento (codigo, descripcion) VALUES

('DNI', 'Documento Nacional de Identidad'),

('CE', 'Cédula de Extranjería'),

('PAS', 'Pasaporte'),

('RUC', 'Registro Único de Contribuyentes');

-- Datos iniciales para métodos de pago

INSERT IGNORE INTO metodos\_pago (codigo, descripcion, requiere\_referencia) VALUES

('EFE', 'Efectivo', FALSE),

('TJT', 'Tarjeta de Crédito/Débito', TRUE),

('TRA', 'Transferencia Bancaria', TRUE),

('DEP', 'Depósito Bancario', TRUE),

('CHE', 'Cheque', TRUE);

-- Datos iniciales para tipos de activo

INSERT IGNORE INTO tipos\_activo (codigo, descripcion, vida\_util\_años) VALUES

('VEH', 'Vehículos', 10),

('MOB', 'Mobiliario', 8),

('EQU', 'Equipos', 5),

('TEC', 'Tecnología', 3);

-- Datos iniciales para canales de marketing

INSERT IGNORE INTO canales\_marketing (codigo, nombre) VALUES

('SOC', 'Redes Sociales'),

('EMA', 'Email Marketing'),

('WEB', 'Sitio Web'),

('RAD', 'Radio'),

('TV', 'Televisión'),

('IMP', 'Medios Impresos');

-- Datos iniciales para tipos de reporte

INSERT IGNORE INTO tipos\_reporte (codigo, nombre) VALUES

('VEN', 'Reporte de Ventas'),

('FIN', 'Reporte Financiero'),

('INV', 'Reporte de Inventario'),

('CLI', 'Reporte de Clientes'),

('CAM', 'Reporte de Campañas');

SET foreign\_key\_checks = 1;

-- =====================================================

-- FIN DE SCRIPT

-- =====================================================