CENTRO UNIVERSITARIO TECNOLOGICO

(CEUTEC)

DESARROLLO DE APLICACIONES WEB 1

TEMA y PARCIAL:

"MI ALMACEN WEB"

SUSTENTADO POR:

FABRICIO OBED GARCIA MARTINEZ

T32421004

PEDRO MOISES FIGUEROA MAZARIEGOS

T32451155

ROGER AUGUSTO PINEDA BANEGAS

T32511084

EDGARDO ALEJANDRO PEREZ ESCOBER

T32451165

CATEDRÁTICO:

ING.JONIE MIRALDA CRUZ

FECHA:

19 / 11 / AÑO 2025

LUGAR:

CAMPUS TEGUCIGALPA Y SAN PEDRO SULA-

Introducción

El presente informe corresponde al desarrollo de la primera etapa del proyecto Sistema de Inventario para Almacén “Mi Almacén Web”, realizado para la clase de Diseño de Aplicaciones Web 1. El propósito fundamental de esta fase es diseñar y construir la base de datos que servirá como estructura central para el futuro sistema web que permitirá administrar el inventario del almacén.

Actualmente, la empresa maneja su información mediante hojas de cálculo y registros manuales, lo cual genera errores frecuentes como diferencias entre el inventario físico y el digital, productos agotados sin registro oportuno, compras duplicadas y dificultades para rastrear movimientos. Ante esta problemática, se requiere una solución tecnológica que permita gestionar los productos, proveedores, categorías, usuarios y movimientos de stock de manera ordenada, precisa y confiable.

En esta etapa se trabajó el modelo de datos, el diagrama relacional y el script DDL, con el fin de crear una base de datos robusta, normalizada y preparada para ser utilizada después a través de una API desarrollada en Node.js con Express.

Objetivo General

Diseñar y construir una base de datos relacional que permita gestionar de forma eficiente el inventario

Objetivo Específicos

- Analizar las necesidades del almacén para identificar las entidades clave del sistema, sus atributos y las relaciones necesarias.
- Crear un diagrama relacional claro y estructurado, que represente visualmente las tablas, claves primarias, claves foráneas y cardinalidades.
- Desarrollar los scripts DDL en MySQL, incluyendo la creación de la base de datos, definición de tablas, restricciones, claves primarias y claves foráneas

Diagrama relacional

Tabla usuarios

La tabla Usuarios almacena la información de los operadores del sistema, quienes serán responsables de registrar movimientos de inventario y realizar acciones administrativas dentro de la plataforma. Incluye datos básicos como nombre completo, correo electrónico y nombre de usuario, así como el rol y el estado de la cuenta

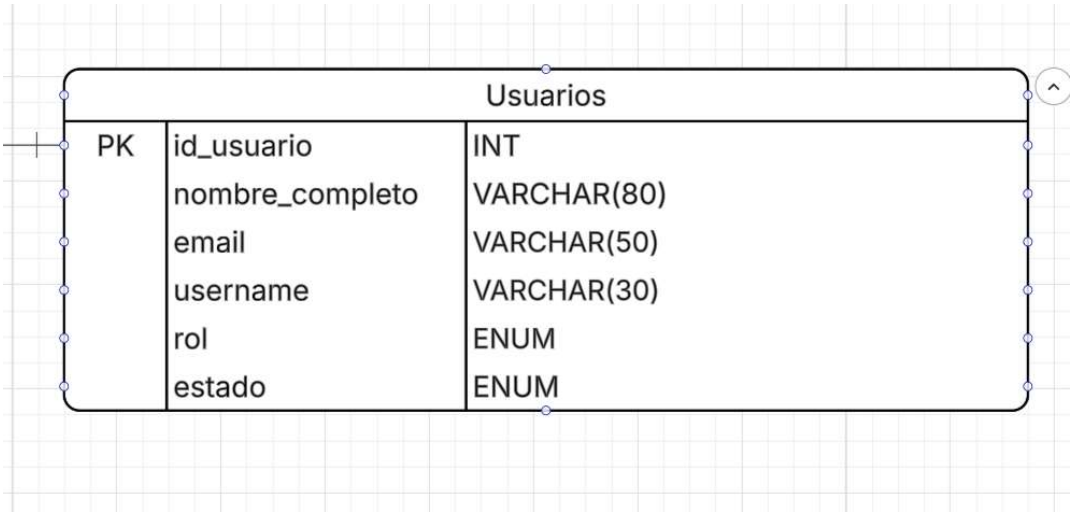

Tabla categorías

La tabla categorías agrupa los productos según su tipo, permitiendo una mejor organización del inventario. Cada producto debe estar asociado a una categoría.

Tabla proveedores

La tabla Proveedores almacena los datos de las empresas o personas que abastecen los productos. Esto permite identificar el origen de cada artículo y vincularlo con la tabla de productos.

Tabla productos

La tabla Productos contiene los datos principales de los artículos disponibles en el almacén. Cada producto se encuentra asociado a una categoría y a un proveedor mediante claves foráneas. Además, incluye precios, códigos internos y estado del producto.

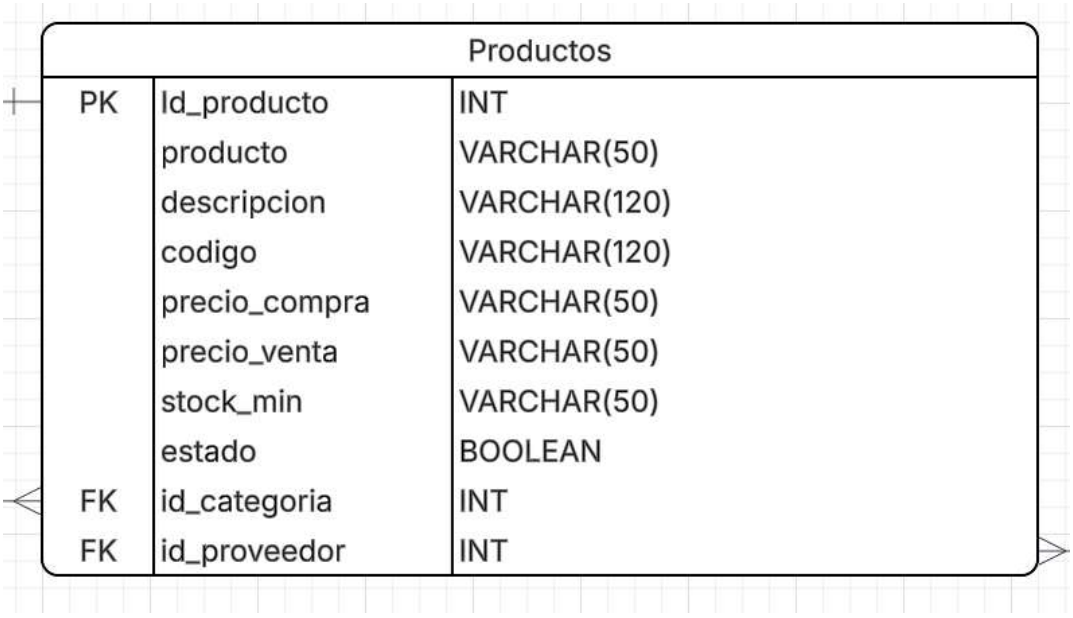

Tabla movimiento de inventario

La tabla Movimientosinventario registra todas las entradas y salidas de productos del almacén. Esta información es fundamental para calcular el stock y mantener el control operativo del inventario.

Tabla devoluciones

La tabla Devoluciones registra los casos en los que un producto debe ser devuelto, ya sea por fallas, errores en un pedido o ajustes administrativos. Cada devolución se encuentra asociada a un movimiento previamente registrado

Diagrama de relaciones

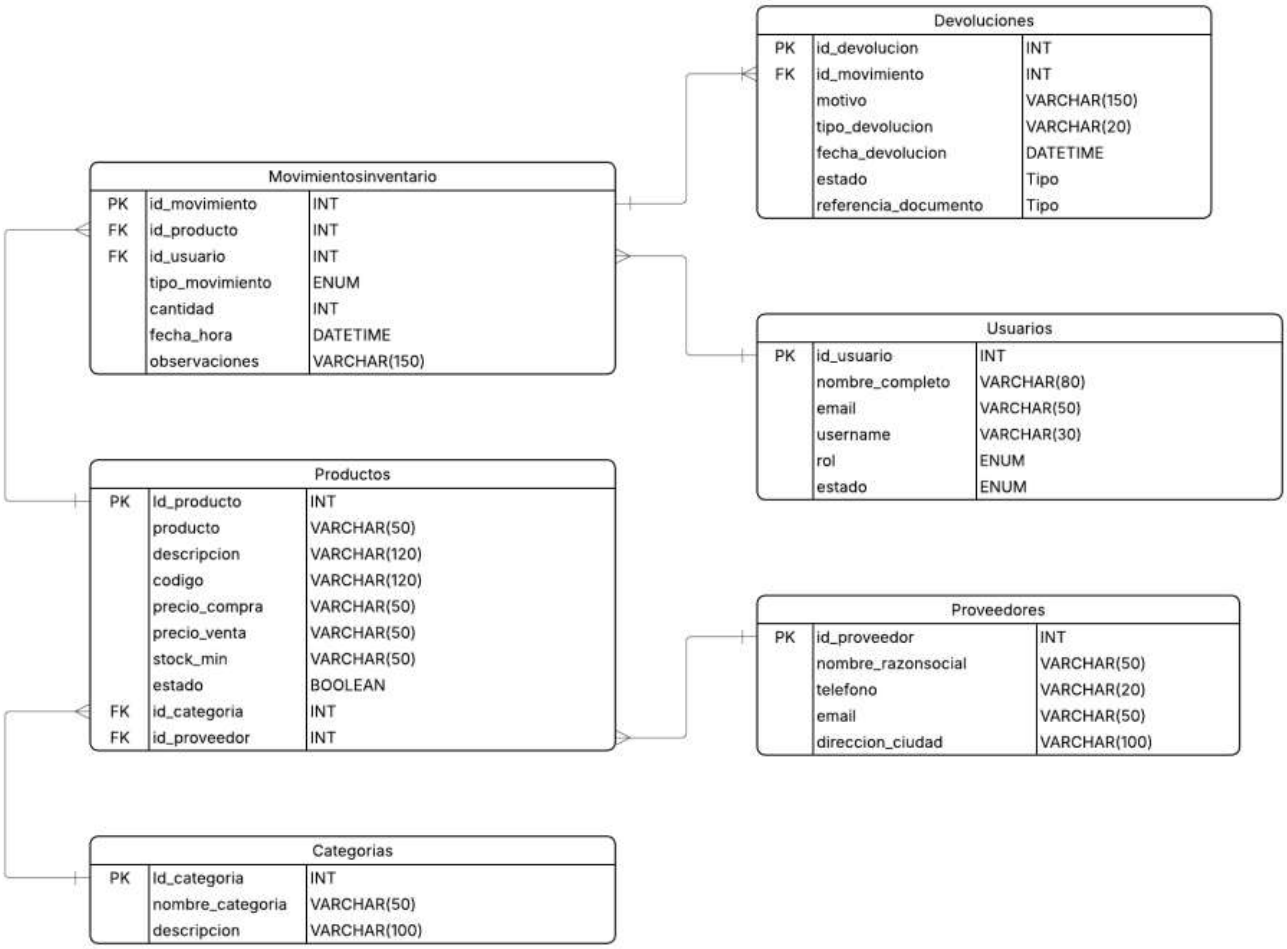

Tabla Principal	Tabla relacionada	Tipo de relación	Propósito
Productos	Categorías	N:1	Clasificar productos
Productos	Proveedores	N:1	Registrar proveedor principal
Movimientos	Productos	N:1	Controlar stock por producto
Movimientos	Usuarios	N:1	Registrar quién realiza el movimiento
Devoluciones	Movimientos	1:N	Documentar devoluciones por movimiento

DDL EN MYQL

Durante el proceso de conversión del diagrama ERD elaborado en Lucidchart hacia el código SQL definitivo para MySQL, fue necesario realizar varios ajustes para garantizar que la estructura de la base de datos cumpliera con los requisitos técnicos, de integridad y normalización establecidos para el proyecto. Estos ajustes fueron la agregación de atributos a los tipo de datos pero no modifican el diseño conceptual original.

```
-- DDL

CREATE database G4Inventario_Almacen;


USE G4Inventario_Almacen;


CREATE TABLE `Usuarios` (
  `id_usuario` INT AUTO_INCREMENT,
  `nombre_completo` VARCHAR(80)NOT NULL,
  `email` VARCHAR(50)NOT NULL,
  `username` VARCHAR(30)NOT NULL,
  `rol` ENUM("Administrador", "Vendedor")NOT NULL,
  `estado` ENUM("Activo", "Inactivo")NOT NULL,
  PRIMARY KEY (`id_usuario`)
);
-- Comentario: almacena información de los usuarios del sistema.


CREATE TABLE `Categorias` (
  `Id_categoria` INT AUTO_INCREMENT,
```

```
  `nombre_categoria` VARCHAR(50) NOT NULL,
  `descripcion` VARCHAR(100),
  PRIMARY KEY (`Id_categoria`)
);

-- Comentario: categorías de los productos.

CREATE TABLE `Proveedores` (
  `id_proveedor` INT AUTO_INCREMENT,
  `nombre_razonsocial` VARCHAR(50) NOT NULL,
  `telefono` VARCHAR(20),
  `email` VARCHAR(50),
  `direccion_ciudad` VARCHAR(100),
  PRIMARY KEY (`id_proveedor`)
);

-- Comentario: datos de proveedores

CREATE TABLE `Productos` (
  `Id_producto` INT AUTO_INCREMENT,
  `producto` VARCHAR(50) NOT NULL,
  `descripcion` VARCHAR(120) ,
  `codigo` VARCHAR(120) NOT NULL,
  `precio_compra` VARCHAR(50) NOT NULL,
```

```
  `precio_venta` VARCHAR(50) NOT NULL,
  `stock_min` VARCHAR(50) NOT NULL,
  `estado` BOOLEAN NOT NULL,
  `id_categoria` INT NOT NULL,
  `id_proveedor` INT NOT NULL,
  PRIMARY KEY (`Id_producto`),
  FOREIGN KEY (`id_categoria`)
    REFERENCES `Categorias`(`Id_categoria`),
  FOREIGN KEY (`id_proveedor`)
    REFERENCES `Proveedores`(`id_proveedor`)
);

-- Comentario: almacenamiento de todos los productos.

CREATE TABLE `Movimientosinventario` (
  `id_movimiento` INT AUTO_INCREMENT,
  `id_producto` INT NOT NULL,
  `id_usuario` INT NOT NULL,
  `tipo_movimiento` ENUM("E", "S") NOT NULL, -- E:Entrda | S:Salida
  `cantidad` INT NOT NULL,
  `fecha_hora` DATETIME NOT NULL,
  `observaciones` VARCHAR(150),
  PRIMARY KEY (`id_movimiento`),
  FOREIGN KEY (`id_producto`)
```

```
    REFERENCES `Productos`(`Id_producto`),
  FOREIGN KEY (`id_usuario`)
    REFERENCES `Usuarios`(`id_usuario`)
);
-- Comentario: entradas y salidas de inventario.

CREATE TABLE `Devoluciones` (
  `id_devolucion` INT AUTO_INCREMENT,
  `id_movimiento` INT,
  `motivo` VARCHAR(150) NOT NULL,
  `tipo_devolucion` VARCHAR(20) NOT NULL,
  `fecha_devolucion` DATETIME NOT NULL,
  `estado` VARCHAR(15) NOT NULL,
  `referencia_documento` VARCHAR(50),
  PRIMARY KEY (`id_devolucion`),
  FOREIGN KEY (`id_movimiento`)
    REFERENCES `Movimientosinventario`(`id_movimiento`)
);
-- Comentario: devoluciones asociadas a movimientos.
```

Futuros endpoints

Como parte de la planificación a futuro del proyecto, se definió una lista de posibles endpoints REST que podrán implementarse en una API desarrollada con Node.js y Express. Estos endpoints permitirán realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre las principales entidades de la base de datos: productos, categorías, proveedores, usuarios, movimientos de inventario y devoluciones.

1. Endpoints para Productos

Estos endpoints permiten administrar la información de los productos del almacén.

- **GET /api/productos**
  Permite obtener el listado completo de productos registrados en el sistema.
- **GET /api/productos/:id**
  Devuelve la información detallada de un producto específico, identificado por su id.
- **POST /api/productos**
  Permite registrar un nuevo producto en la base de datos, enviando los datos en el cuerpo de la petición (JSON).
- **PUT /api/productos/:id**
  Actualiza los datos de un producto existente, identificado por su id.
- **DELETE /api/productos/:id**
  Elimina un producto específico.

2. Endpoints para Categorías

Estos endpoints gestionan las categorías a las que pertenecen los productos.

- **GET /api/categorias**

  Lista todas las categorías disponibles.
- **GET /api/categorias/:id**

  Muestra la información de una categoría específica.
- **POST /api/categorias**

  Crea una nueva categoría en el sistema.

- **PUT /api/categorias/:id**
  Modifica los datos de una categoría existente.
- **DELETE /api/categorias/:id**
  Elimina una categoría determinada, siempre que no afecte la integridad de los productos asociados.

3. Endpoints para Proveedores

Se utilizan para administrar los proveedores que abastecen al almacén.

- **GET /api/proveedores**
  Devuelve el listado de todos los proveedores registrados.
- **GET /api/proveedores/:id**
  Muestra los datos de un proveedor específico.
- **POST /api/proveedores**
  Registra un nuevo proveedor en la base de datos.
- **PUT /api/proveedores/:id**
  Actualiza la información de un proveedor existente.
- **DELETE /api/proveedores/:id**
  Elimina un proveedor determinado, considerando las relaciones con productos.

4. Endpoints para Usuarios

Estos endpoints permiten gestionar los usuarios (operadores) del sistema.

- **GET /api/usuarios**
  Lista todos los usuarios registrados.
- **GET /api/usuarios/:id**
  Devuelve la información de un usuario específico.
- **POST /api/usuarios**
  Crea un nuevo usuario en el sistema.

- **PUT /api/usuarios/:id**
  Actualiza los datos de un usuario existente (por ejemplo, nombre, rol o estado).
- **DELETE /api/usuarios/:id**
  Elimina o desactiva un usuario del sistema.

5. Endpoints para Movimientos de Inventario

Estos endpoints se enfocan en el registro y consulta de las entradas y salidas de productos.

- **GET /api/movimientos**
  Lista todos los movimientos de inventario registrados (entradas y salidas).
- **GET /api/movimientos/:id**
  Muestra el detalle de un movimiento específico.
- **POST /api/movimientos**
  Registra un nuevo movimiento de inventario, indicando el producto, la cantidad, el tipo de movimiento (“E” o “S”) y el usuario que lo realiza.

6. Endpoints para Devoluciones

Estos endpoints permiten administrar las devoluciones asociadas a movimientos de inventario.

- **GET /api/devoluciones**
  Devuelve el listado de todas las devoluciones registradas en el sistema.
- **GET /api/devoluciones/:id**
  Muestra la información de una devolución específica, incluyendo su relación con el movimiento original.
- **POST /api/devoluciones**
  Permite registrar una nueva devolución, asociándola a un movimiento de

inventario y especificando el motivo, tipo de devolución y demás datos relevantes.

Conclusión

En conclusión, se desarrolló la base de datos relacional siguiendo los parámetros establecidos en el ejercicio, buscando dar solución a las necesidades planteadas por el cliente. Para ello, se inició con la identificación de las entidades principales del sistema y posteriormente se definieron las relaciones entre ellas, garantizando un modelo estructurado y coherente.

Seguidamente, se seleccionaron y asignaron los tipos de datos adecuados para cada atributo, y se representó el modelo mediante un diagrama relacional que permitió visualizar de manera clara las conexiones entre las tablas. Luego, dicho modelo fue llevado a su implementación en MySQL mediante el uso de sentencias DDL, incorporando claves primarias, claves foráneas, restricciones y comentarios que fortalecen la integridad del sistema.

Finalmente, se elaboró una lista preliminar de los futuros endpoints que podrían implementarse en una API desarrollada con Node.js y Express, sentando así las bases para la siguiente etapa del proyecto. Gracias a este proceso, se obtuvo una base de datos completa, organizada y lista para integrarse en el sistema web de gestión de inventario.