# Examen FBBDD-2

**ALEXI DURÁN GÓMEZ**

**S2**

**PEDRO FELIPE GÓMEZ BONILLA**

**CAMPUSLANDS**
**RUTA NODE**
**CAJASAN**
**2025**

Tabla de Contenidos

# Introducción

Este documento se ha redactado con el propósito de exponer el diseño del sistema de información propuesto por Alexi Durán Gómez como requerimientos académicos establecidos para aprobar el skill de Introducción al backend del programa de desarrollo de software en Campuslands dictada por el Trainer Pedro Felipe Gómez Bonilla.
Será entregado formalmente.

Está documentación busca abarcar el desarrollo integral del sistema de base de datos, desde la conceptualización inicial hasta la implementación física final, siguiendo metodologías estándar de ingeniería de software y diseño de bases de datos. Acá se presentan todas las fases del proceso de desarrollo del sistema, incluyendo diagramas conceptuales, lógicos y físicos, así como la implementación de las mejores prácticas de normalización.

Este sistema busca el mayor punto de optimización posible, para lo cual se implementan dichas buenas prácticas de normalización y tecnologías que van de la mano para permitir el mayor rendimiento y desempeño del sistema.

Está base de datos se presenta desarrollada con el lenguaje SQL y con el gestor de bases de datos MySQL.

Siendo Sql el lenguaje más utilizado para gestionar y trabajar con bases de datos y MySQL uno de los motores de bases de datos más utilizados desde el inicio del desarrollo de las tecnologías de la información, siendo considerada uno de los aportes pilares de esté campo

**Contexto del problema**
En el competitivo sector de la venta y comercialización de bicicletas, CampusBike enfrenta varios desafíos relacionados con la gestión de la información. La falta de un sistema eficiente y centralizado para organizar, analizar y utilizar datos afecta negativamente diversos aspectos del negocio. Entre los problemas más destacados se encuentran:

1. **Gestión Deficiente de Clientes:** La ausencia de un sistema estructurado impide un seguimiento adecuado de las interacciones con los clientes, lo que dificulta la personalización de servicios y la fidelización.

2. **Ineficiencia en la Cadena de Suministro:** La gestión manual o mediante sistemas dispersos de proveedores, inventarios y pedidos genera errores y retrasos, afectando la disponibilidad de productos y la capacidad de respuesta ante la demanda.

3. **Dificultades en el Análisis de Ventas y Compras:** Sin una base de datos robusta, analizar tendencias de ventas y compras resulta complicado, lo que impide tomar decisiones estratégicas basadas en datos precisos y actualizados.

4. **Gestión Ineficiente de Repuestos:** La falta de un control efectivo sobre los repuestos y accesorios de bicicletas conduce a problemas de inventario, como exceso de stock o desabastecimiento, afectando la calidad del servicio al cliente.

**Solución Propuesta**
Para abordar estos desafíos, CampusBike solicita el diseño y desarrollo de una base de datos robusta y altamente funcional que incluya los siguientes componentes:

- **Gestión de Clientes:** Registro detallado de información de clientes, historial de compras e interacciones, permitiendo un servicio más personalizado y eficiente.

- **Gestión de Proveedores:** Información centralizada sobre proveedores, términos de suministro y seguimiento de pedidos, mejorando la eficiencia de la cadena de suministro.

- **Gestión de Ventas y Compras:** Registro y análisis detallado de ventas y compras, facilitando la toma de decisiones estratégicas basadas en datos reales.

- **Gestión de Inventarios y Repuestos:** Control preciso de inventarios de bicicletas, repuestos y accesorios, optimizando la disponibilidad de productos y reduciendo costos asociados a exceso de stock o desabastecimiento.

# Caso de Estudio

Las bases de datos en los sistemas de ventas e inventarios son esenciales para almacenar, gestionar y centralizar de forma eficiente la información sobre todo el negocio, ventas, servicio, stock, clientes, pedidos y personal, permitiendo una operación eficiente, lo que hace que se realicen ágilmente los procesos y mejora la experiencia de todo el personal.

En este caso se usa para gestionar a los clientes, proveedores, ventas, compras, inventario y presupuesto. Lo que permite un mejor manejo de la data, ya que esté sistema surge de dicha necesidad de gestionar, monitorear y analizar los datos de compras y ventas de los productos generales, ya que está información permite conocer en tiempo real el estado del stock, precios, productos con mayor demanda, cantidad de compras diarias, mensuales, anuales, ganancias y pérdidas, entre otras muchas cosas como la información para una adecuada toma de decisiones.

Esté proceso debería entregar una poderosa herramienta que sea capaz de hacer todo lo mencionado anteriormente, además de organizar, administrar fácilmente los datos, facilidad en su acceso, análisis, consultas, manipulación en tiempo real y por ende en generar un control general de la data.

# Planificación

## Construcción del Modelo Conceptual

### Descripción

Este modelo conceptual representa la estructura fundamental de la información para la gestión de proveedores, compras, ventas, inventario y presupuesto, mediante un diagrama que identifica las entidades principales, sus atributos y las relaciones que existen entre ellas.

**Entidades identificadas:**
**1. Cliente**
Representa a los clientes que realizan compras en la tienda de bicicletas.
Atributos:

- id_cliente (PK): Identificador único de cada cliente.
- nombre_cliente: Nombre del cliente.
- email: Correo electrónico del cliente.
- telefono: Número de contacto del cliente.

Entidad de referencia para vincular las ventas a un cliente específico.

**2. Venta**

Representa cada transacción de venta realizada a los clientes.
Atributos:

- id_venta (PK): Identificador único de la venta.
- cliente_id (FK): Cliente que realizó la compra.
- fecha: Fecha en que se registra la venta.
- cantidad: Número de repuestos vendidos en la transacción.
- precio_total: Valor total de la venta.

**3. Repuesto**
Representa los repuestos y accesorios de bicicletas disponibles en la tienda.
Atributos:

- id_repuesto (PK): Identificador único de cada repuesto.
- nombre: Nombre del repuesto.
- precio_unitario: Valor unitario del repuesto.
- compra_id (FK): Referencia a la compra que abasteció este repuesto.

Entidad central para gestionar inventario, ventas y compras.

**4. Inventario**
Controla el stock y estado de los repuestos en bodega.
Atributos:

- id: Identificador único de cada registro de inventario.
- repuesto_id (FK): Repuesto asociado en el inventario.
- stock: Cantidad disponible en inventario.
- estado: Estado del repuesto (ej. disponible, agotado, en pedido).

**5. Compra**
Registra las adquisiciones de repuestos a los proveedores.
Atributos:

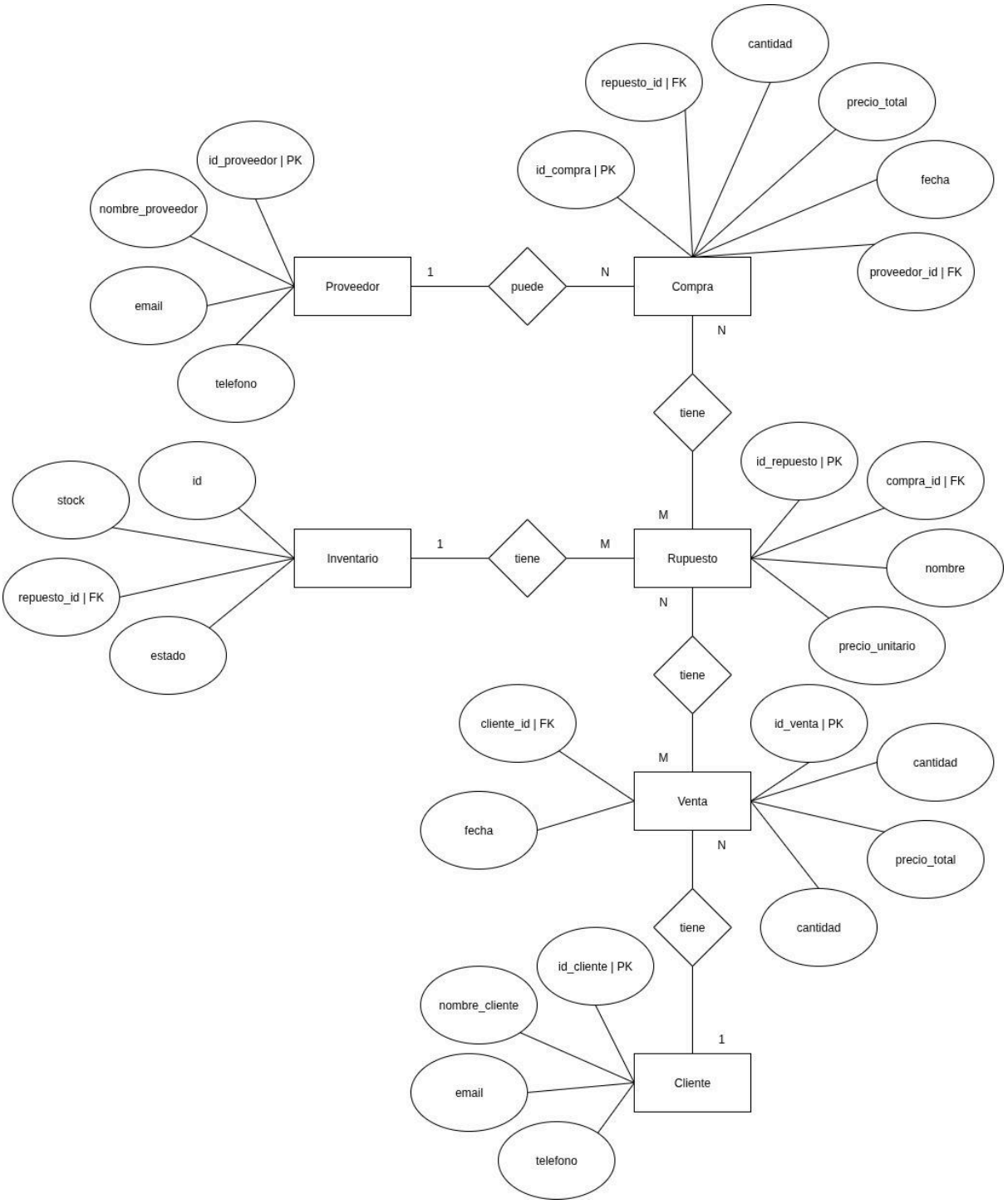
- id_compra (PK): Identificador único de cada compra.
- proveedor_id (FK): Proveedor al que se le compró.
- repuesto_id (FK): Repuesto adquirido.
- fecha: Fecha de la compra.
- cantidad: Cantidad adquirida en la compra.
- precio_total: Valor total de la compra.

**6. Proveedor**
Representa a los proveedores que abastecen de repuestos y accesorios.
Atributos:

- id_proveedor (PK): Identificador único de cada proveedor.
- nombre_proveedor: Nombre del proveedor.
- email: Correo electrónico del proveedor.
- telefono: Número de contacto del proveedor.

Entidad de referencia para vincular compras con el origen de los productos.

## Gráfica




## Descripción Técnica

El modelo conceptual de **CampusBike** representa un sistema de información orientado a la gestión integral de clientes, proveedores, compras, ventas, repuestos e inventario. Cada entidad está definida con atributos específicos que permiten la identificación única y el registro detallado de los datos, garantizando consistencia y trazabilidad dentro del negocio. Por ejemplo, los clientes cuentan con información de contacto vinculada directamente a las ventas, mientras que los proveedores están asociados a las compras realizadas para abastecer el inventario.

Las relaciones reflejan los procesos clave de la empresa: un proveedor puede realizar múltiples compras (1:N), cada compra puede incluir varios repuestos (N:M), y esos repuestos son los que posteriormente se registran en el inventario con control de stock y estado. A su vez, los clientes generan ventas que también están asociadas a múltiples repuestos, lo que configura otra relación de tipo N:M. Estas relaciones permiten modelar de forma precisa la dinámica de abastecimiento y comercialización de la empresa.

En términos técnicos, el modelo asegura escalabilidad y normalización, ya que centraliza la información y resuelve las dependencias mediante claves primarias y foráneas. Las relaciones de muchos a muchos (como en ventas y compras de repuestos) requieren tablas intermedias que almacenen detalles como cantidad, precio unitario y valor total, lo que posibilita análisis detallados de costos, márgenes y movimientos de inventario. Con este diseño, CampusBike puede optimizar el seguimiento de su cadena de suministro, mejorar el control de inventario y disponer de información confiable para la toma de decisiones estratégicas.

# Construcción del Modelo Lógico

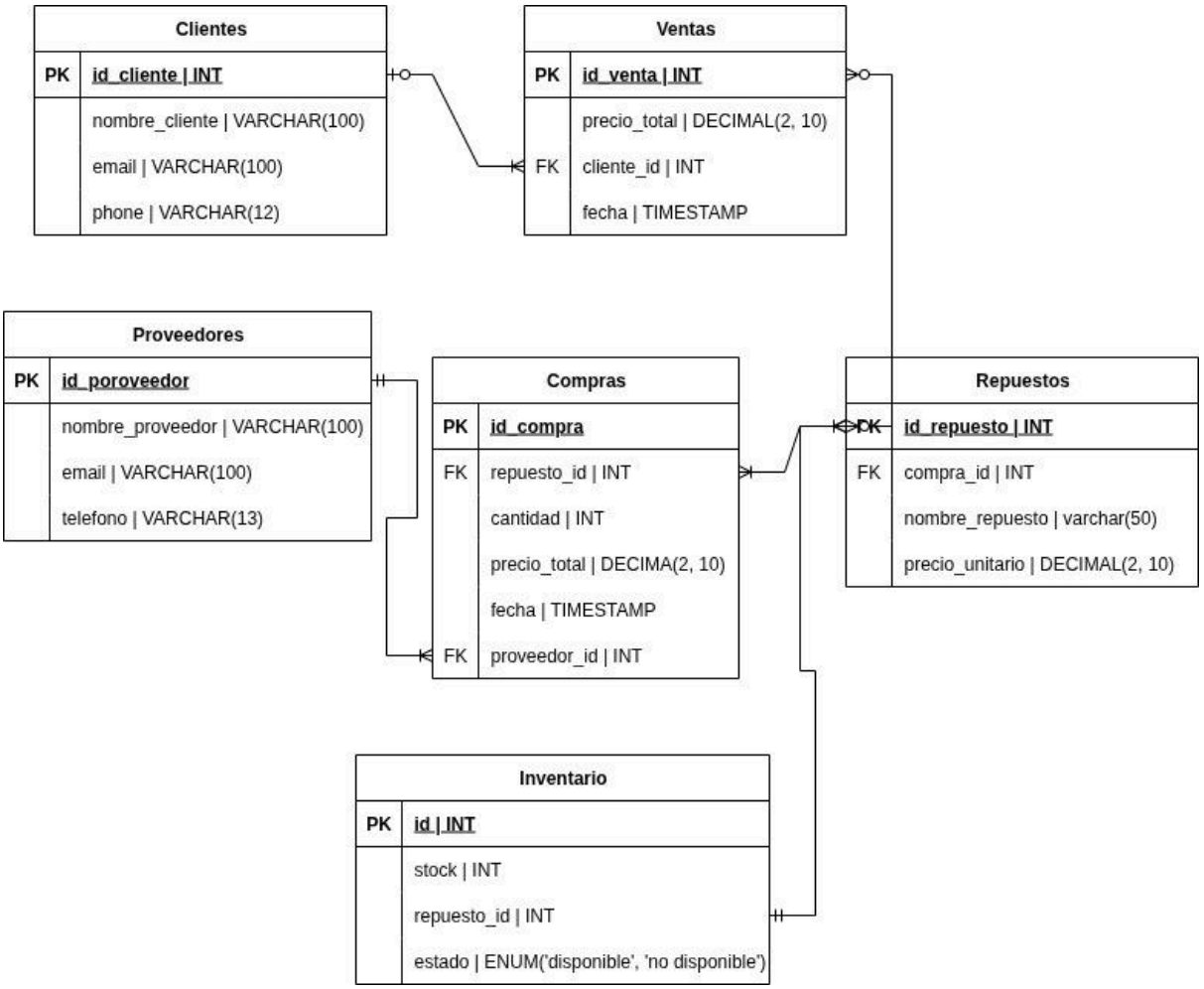
Un modelo de base de datos muestra la estructura lógica de la base, incluidas las relaciones y limitaciones que determinan cómo se almacenan los datos y cómo se accede a ellos. Los modelos de bases de datos individuales se diseñan en base a las reglas y los conceptos de cualquier modelo de datos más amplio que los diseñadores adopten. La mayoría de los modelos de datos se pueden representar por medio de un diagrama de base de datos acompañante.

## Descripción

Este modelo representa cómo se organiza la información en una empresa que vende y compra repuestos. La base de datos guarda datos de los clientes y de las ventas que ellos realizan, incluyendo los productos que compran. También se registran los proveedores que entregan los repuestos, junto con las compras que se hacen a ellos.

Además, se maneja un inventario, donde se controla la cantidad de repuestos disponibles y si están en estado de “disponible” o “no disponible”. De esta manera, el sistema permite llevar un control completo de todo el proceso: desde que se compra un repuesto a un proveedor, hasta que se vende a un cliente, asegurando siempre que la información esté bien organizada y actualizada.

## Gráfica




## Descripción Técnica

El modelo lógico presentado describe un sistema de gestión para una empresa que maneja clientes, proveedores, ventas, compras, repuestos e inventario. Cada entidad está normalizada y definida con sus respectivas claves primarias (PK) y claves foráneas (FK), lo que permite mantener la integridad referencial en la base de datos.

La entidad **Clientes** registra información básica como nombre, correo y teléfono, y se relaciona con **Ventas**, que almacena el total de la transacción, la fecha y el cliente asociado. A su vez, las ventas están conectadas con la entidad **Repuestos**, donde se gestionan los productos vendidos, incluyendo su nombre y precio unitario. Por otro lado, la entidad **Proveedores** se vincula con **Compras**, que registra la adquisición de repuestos, con atributos como cantidad, precio total, fecha y el proveedor correspondiente.

Finalmente, la entidad **Inventario** asegura el control del stock de cada repuesto adquirido, relacionándose directamente con la entidad **Repuestos**. En ella se define la cantidad disponible y el estado de los productos, ya sea “disponible” o “no disponible”. En conjunto, este modelo lógico está diseñado para gestionar de manera integral el ciclo de abastecimiento y venta de repuestos, garantizando trazabilidad desde la compra a proveedores hasta la venta al cliente final.

# Normalización del Modelo Lógico

Normalizar una base de datos significa organizar sus tablas y relaciones de manera que no existan datos repetidos ni dependencias innecesarias. El objetivo es mejorar la eficiencia del almacenamiento y garantizar que la información se mantenga consistente y sin errores.

Al aplicar la normalización se reducen los problemas de duplicación de datos, se facilita la actualización de la información y se evita la aparición de inconsistencias. Además, la base de datos se vuelve más flexible y escalable, lo que permite adaptarla mejor a futuros cambios o necesidades del sistema.

## Primera Forma Normal (1FN)

La **Primera Forma Normal (1FN)** exige que cada tabla tenga valores atómicos, es decir, que no existan campos que almacenen múltiples datos en una misma columna ni repeticiones de grupos de atributos. En tu diseño, todas las tablas cumplen con esta condición porque los atributos están bien definidos: por ejemplo, en la tabla *Clientes* cada campo (nombre, email, teléfono) almacena un solo dato y no listas de información. Esto asegura que la base de datos pueda consultarse de manera clara y sin ambigüedades.

## Segunda Forma Normal (2FN)

La **Segunda Forma Normal (2FN)** establece que, además de cumplir con la 1FN, todos los campos de una tabla deben depender completamente de la clave primaria, evitando dependencias parciales. En tu modelo, este principio se respeta: por ejemplo, en la tabla *Compras*, los atributos como cantidad, precio_total y fecha dependen únicamente de *id_compra*, que es la clave primaria. No existen campos que dependan solo de una parte de la clave o de otra relación externa. Esto elimina redundancias y asegura consistencia en la información.

## Tercera Forma Normal (3FN)

La **Tercera Forma Normal (3FN)** indica que, además de cumplir con la 2FN, ningún campo debe depender de otro atributo que no sea la clave primaria, evitando dependencias transitivas. En tu diseño, esta regla también se cumple: por ejemplo, en la tabla *Repuestos*, el nombre y precio_unitario dependen únicamente de *id_repuesto* y no de otros atributos como el id de compra o proveedor. Esto garantiza que los datos estén completamente separados por entidades, evitando duplicación de información y facilitando el mantenimiento de la base de datos.

# Construcción del Modelo Físico

El **modelo físico** es la etapa donde el diseño de la base de datos se lleva a un nivel concreto y técnico, convirtiendo el modelo lógico en estructuras reales dentro de un motor de base de datos (por ejemplo, MySQL, PostgreSQL o SQL Server). Aquí se definen con precisión los tipos de datos, las restricciones, los índices, las claves primarias y foráneas, así como las reglas de integridad referencial. En otras palabras, es la traducción del diseño conceptual y lógico a sentencias SQL que crearán las tablas y relaciones en un sistema de gestión de bases de datos.

## Descripción

En el caso de tu estudio, el modelo físico aplicaría directamente a las tablas que diseñaste en el modelo lógico (*Clientes, Ventas, Proveedores, Compras, Repuestos, Inventario*). Esto significa definir sus campos con los tipos de datos adecuados (ejemplo: `VARCHAR(100)` para nombres, `DECIMAL(10,2)` para precios, `TIMESTAMP` para fechas), establecer las claves primarias (`PRIMARY KEY`), claves foráneas (`FOREIGN KEY`) y restricciones como `NOT NULL` o `UNIQUE` cuando sea necesario. También aquí podrías agregar optimizaciones como índices para mejorar la velocidad de consulta sobre columnas que se usen mucho en búsquedas.

Aplicar este modelo físico te permitirá no solo tener la estructura organizada y lista para almacenar datos, sino también asegurar la integridad entre las entidades. Por ejemplo, evitar que se registre una *Venta* sin que exista un *Cliente*, o impedir que un *Repuesto* quede en inventario si no fue comprado previamente a un *Proveedor*. Así, el modelo físico convierte tu diseño en una base de datos funcional y utilizable en un entorno real.

## Código

```sql
-- CREACIÓN DE LA BASE DE DATOS
CREATE DATABASE campusbike_db;
USE campusbike_db;

-- TABLA CLIENTES
CREATE TABLE Clientes (
   id_cliente INT AUTO_INCREMENT PRIMARY KEY,
   nombre_cliente VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   telefono VARCHAR(20)
);

-- TABLA PROVEEDORES
CREATE TABLE Proveedores (
   id_proveedor INT AUTO_INCREMENT PRIMARY KEY,
   nombre_proveedor VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   telefono VARCHAR(20)
);

-- TABLA REPUESTOS
CREATE TABLE Repuestos (
   id_repuesto INT AUTO_INCREMENT PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL,
   precio_unitario DECIMAL(10,2) NOT NULL
);
```

```sql
-- TABLA INVENTARIO
CREATE TABLE Inventario (
   id_inventario INT AUTO_INCREMENT PRIMARY KEY,
   id_repuesto INT NOT NULL,
   stock INT NOT NULL DEFAULT 0,
   estado ENUM('disponible','agotado','en pedido') DEFAULT
'disponible',
   FOREIGN KEY (id_repuesto) REFERENCES Repuestos(id_repuesto)
);

-- TABLA VENTAS
CREATE TABLE Ventas (
   id_venta INT AUTO_INCREMENT PRIMARY KEY,
   id_cliente INT NOT NULL,
   fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   precio_total DECIMAL(10,2) NOT NULL,
   FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente)
);

-- TABLA DETALLE VENTAS (RESUELVE N:M ENTRE VENTAS Y REPUESTOS)
CREATE TABLE DetalleVenta (
   id_detalle_venta INT AUTO_INCREMENT PRIMARY KEY,
   id_venta INT NOT NULL,
   id_repuesto INT NOT NULL,
   cantidad INT NOT NULL,
   precio_unitario DECIMAL(10,2) NOT NULL,
   precio_total DECIMAL(10,2) NOT NULL,
   FOREIGN KEY (id_venta) REFERENCES Ventas(id_venta),
   FOREIGN KEY (id_repuesto) REFERENCES Repuestos(id_repuesto)
);

-- TABLA COMPRAS
CREATE TABLE Compras (
   id_compra INT AUTO_INCREMENT PRIMARY KEY,
   id_proveedor INT NOT NULL,
   fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   precio_total DECIMAL(10,2) NOT NULL,
   FOREIGN KEY (id_proveedor) REFERENCES Proveedores(id_proveedor)
);

-- TABLA DETALLE COMPRAS (RESUELVE N:M ENTRE COMPRAS Y REPUESTOS)
CREATE TABLE DetalleCompra (
   id_detalle_compra INT AUTO_INCREMENT PRIMARY KEY,
```

```
    id_compra INT NOT NULL,
    id_repuesto INT NOT NULL,
    cantidad INT NOT NULL,
    precio_unitario DECIMAL(10,2) NOT NULL,
    precio_total DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (id_compra) REFERENCES Compras(id_compra),
    FOREIGN KEY (id_repuesto) REFERENCES Repuestos(id_repuesto)
);
```

Insertions .sql

```
-- INSERCIONES DE PRUEBA PARA CAMPUSBIKE
USE campusbike_db;

-- TABLA Clientes
INSERT INTO Clientes (id_cliente, nombre_cliente, email, telefono)
VALUES
(1, 'Juan Pérez', 'juan.perez@example.com', '3001234567'),
(2, 'María López', 'maria.lopez@example.com', '3019876543'),
(3, 'Carlos Gómez', 'carlos.gomez@example.com', '3024567890'),
(4, 'Ana Torres', 'ana.torres@example.com', '3101237894'),
(5, 'Luis Rodríguez', 'luis.rodriguez@example.com', '3159876541'),
(6, 'Paula Jiménez', 'paula.jimenez@example.com', '3162345678'),
(7, 'Andrés Castro', 'andres.castro@example.com', '3174561234'),
(8, 'Laura Herrera', 'laura.herrera@example.com', '3186549871'),
(9, 'Diego Martínez', 'diego.martinez@example.com', '3198527412'),
(10, 'Sofía Ramírez', 'sofia.ramirez@example.com', '3209632587');

-- TABLA Proveedores
INSERT INTO Proveedores (id_proveedor, nombre_proveedor, email,
telefono) VALUES
(1, 'BiciParts Co', 'contact@biciparts.com', '601123456'),
(2, 'Repuestos Andinos', 'ventas@andinos.com', '602987654'),
(3, 'CycleSupply', 'info@cyclesupply.com', '603456789'),
(4, 'GlobalBikes', 'sales@globalbikes.com', '604123987'),
(5, 'Ruta Proveedores', 'contacto@rutaproveedores.com', '605987321'),
(6, 'Mundo Repuestos', 'hola@mundorepuestos.com', '606555444'),
(7, 'BikeHub', 'support@bikehub.com', '607333222'),
(8, 'Ciclos y Más', 'info@ciclosymas.com', '608777666'),
(9, 'Accesorios Norte', 'ventas@norteaccesorios.com', '609111222'),
(10, 'Taller Suministros', 'taller@suministros.com', '610888999');
```

```sql
-- TABLA Repuestos
INSERT INTO Repuestos (id_repuesto, nombre, precio_unitario) VALUES
(1, 'Pastillas de freno', 25.50),
(2, 'Cadena', 12.00),
(3, 'Neumático 26\"', 45.00),
(4, 'Cámara', 8.50),
(5, 'Asiento (saddle)', 35.00),
(6, 'Pedal', 20.00),
(7, 'Juego de cambios (grupo)', 150.00),
(8, 'Manubrio', 40.00),
(9, 'Llanta (rim)', 70.00),
(10, 'Radio (spoke)', 2.50);

-- TABLA Inventario
INSERT INTO Inventario (id_inventario, id_repuesto, stock, estado)
VALUES
(1, 1, 50, 'disponible'),
(2, 2, 100, 'disponible'),
(3, 3, 30, 'disponible'),
(4, 4, 200, 'disponible'),
(5, 5, 15, 'disponible'),
(6, 6, 60, 'disponible'),
(7, 7, 5, 'en pedido'),
(8, 8, 20, 'disponible'),
(9, 9, 8, 'disponible'),
(10, 10, 500, 'disponible');

-- TABLA Compras
INSERT INTO Compras (id_compra, id_proveedor, fecha, precio_total)
VALUES
(1, 1, '2025-09-01 10:00:00', 510.00),
(2, 2, '2025-09-02 11:15:00', 600.00),
(3, 3, '2025-09-03 09:30:00', 450.00),
(4, 4, '2025-09-04 14:45:00', 850.00),
(5, 5, '2025-09-05 16:00:00', 350.00),
(6, 6, '2025-09-06 08:20:00', 800.00),
(7, 7, '2025-09-07 13:10:00', 300.00),
(8, 8, '2025-09-08 12:00:00', 400.00),
(9, 9, '2025-09-09 15:30:00', 420.00),
(10, 10, '2025-09-10 17:50:00', 750.00);

-- TABLA DetalleCompra
```

```sql
INSERT INTO DetalleCompra (id_detalle_compra, id_compra, id_repuesto,
cantidad, precio_unitario, precio_total) VALUES
(1, 1, 1, 20, 25.50, 510.00),
(2, 2, 2, 50, 12.00, 600.00),
(3, 3, 3, 10, 45.00, 450.00),
(4, 4, 4, 100, 8.50, 850.00),
(5, 5, 5, 10, 35.00, 350.00),
(6, 6, 6, 40, 20.00, 800.00),
(7, 7, 7, 2, 150.00, 300.00),
(8, 8, 8, 10, 40.00, 400.00),
(9, 9, 9, 6, 70.00, 420.00),
(10, 10, 10, 300, 2.50, 750.00);

-- TABLA Ventas
INSERT INTO Ventas (id_venta, id_cliente, fecha, precio_total) VALUES
(1, 1, '2025-09-11 10:10:00', 51.00),
(2, 2, '2025-09-11 11:20:00', 12.00),
(3, 3, '2025-09-12 09:05:00', 45.00),
(4, 4, '2025-09-12 13:25:00', 25.50),
(5, 5, '2025-09-13 15:00:00', 35.00),
(6, 6, '2025-09-13 16:40:00', 40.00),
(7, 7, '2025-09-14 12:30:00', 150.00),
(8, 8, '2025-09-14 14:15:00', 40.00),
(9, 9, '2025-09-15 09:45:00', 70.00),
(10, 10, '2025-09-15 11:55:00', 25.00);

-- TABLA DetalleVenta
INSERT INTO DetalleVenta (id_detalle_venta, id_venta, id_repuesto,
cantidad, precio_unitario, precio_total) VALUES
(1, 1, 1, 2, 25.50, 51.00),
(2, 2, 2, 1, 12.00, 12.00),
(3, 3, 3, 1, 45.00, 45.00),
(4, 4, 4, 3, 8.50, 25.50),
(5, 5, 5, 1, 35.00, 35.00),
(6, 6, 6, 2, 20.00, 40.00),
(7, 7, 7, 1, 150.00, 150.00),
(8, 8, 8, 1, 40.00, 40.00),
(9, 9, 9, 1, 70.00, 70.00),
(10, 10, 10, 10, 2.50, 25.00);
```

## Descripción Técnica

El modelo físico de `campusbike_db` está diseñado para la gestión de clientes, proveedores, repuestos, inventario, ventas y compras dentro de un sistema de control de repuestos de bicicletas. Cada entidad se materializa en una tabla con **claves primarias** (`PRIMARY KEY`) que garantizan la unicidad de los registros, y **claves foráneas** (`FOREIGN KEY`) que mantienen la integridad referencial entre las entidades relacionadas.

Las tablas principales son `Clientes`, `Proveedores` y `Repuestos`, que funcionan como entidades base del sistema. A partir de ellas se construyen tablas dependientes como `Inventario`, que gestiona la disponibilidad de repuestos mediante el campo `estado`, y las tablas de transacciones `Ventas` y `Compras`, que permiten registrar operaciones comerciales vinculadas a clientes y proveedores respectivamente.

Para resolver las relaciones de muchos a muchos entre transacciones y productos, el modelo incluye tablas intermedias: `DetalleVenta` y `DetalleCompra`. Estas tablas no solo vinculan las entidades, sino que también almacenan información adicional como `cantidad`, `precio_unitario` y `precio_total`, lo que garantiza un control detallado de cada operación. Además, los tipos de datos (`VARCHAR`, `DECIMAL`, `TIMESTAMP`, `ENUM`) se definieron de acuerdo con la naturaleza de los atributos, optimizando el almacenamiento y asegurando la consistencia de los registros.

En conjunto, el modelo físico garantiza un esquema normalizado, con relaciones bien definidas, soporte para transacciones y control de inventario, asegurando la escalabilidad y confiabilidad del sistema en un entorno de base de datos relacional.

# Referencias