

**Concesionario CampusCar**

**ALEJANDRA MACHUCA MOLINA**

**DOCENTE**

**PEDRO FELIPE GÓMEZ BONILLA**

**CAMPUSLANDS  
SANDBOX GRUPO T2  
RUTA BASE DE DATOS  
TIBÚ  
2024**

## Tabla de Contenidos

<b>Introducción</b>	<b>3</b>
<b>Caso de Estudio</b>	<b>4</b>
<b>Planificación</b>	<b>4</b>
Construcción del Modelo Conceptual	4
Descripción	5
Gráfica	5
Construcción del Modelo Lógico	6
Descripción	6
Gráfica	8
Normalización del Modelo Lógico	8
Primera Forma Normal (1FN)	8
Descripción	8
Segunda Forma Normal (2FN)	9
Descripción	9
Tercera Forma Normal (3FN)	9
Descripción	9
Construcción del Modelo Físico	9
Descripción	9

# Introducción

Este proyecto busca crear una base de datos para el concesionario "CampusCar", que permite gestionar de forma ordenada la información de vehículos, clientes, ventas y servicios de mantenimiento. Con esta base de datos, el concesionario podrá llevar un mejor control de su inventario y sus transacciones, optimizando sus procesos.

Este documento explica el diseño y desarrollo de la base de datos, desde el modelo conceptual hasta el físico. Además, se normaliza el modelo para evitar duplicaciones y asegurar la integridad de la información.

# Caso de Estudio

Este proyecto se centra en diseñar una base de datos para el concesionario CampusCar, que permite organizar y manejar mejor la información sobre vehículos en stock, clientes, ventas y servicios de mantenimiento. La base de datos centralizará estos datos para facilitar la gestión de inventario, el seguimiento de clientes y la administración de ventas y servicios.

Se han identificado las entidades clave (Vehículos, Clientes, Vendedores, Ventas y Mantenimiento) y sus relaciones, normalizando el diseño para evitar redundancias y asegurar la integridad de los datos. Además, se ha creado un diagrama E-R que muestra cómo se conectan las distintas partes de la base de datos, y se ha preparado el modelo físico para su implementación en un SGBD. Con esta base de datos, CampusCar podrá agilizar sus operaciones y mejorar el control de su información.

## Planificación

1. **Identificar Necesidades:** Primero, definimos qué información necesita el concesionario para gestionar vehículos, clientes, ventas y servicios de mantenimiento.
2. **Diseño Conceptual:** Luego, armamos un esquema básico con las entidades principales (como Vehículos, Clientes, Ventas, etc.) y sus relaciones. Aquí creamos un diagrama E-R para visualizar cómo se conecta todo.
3. **Diseño Lógico:** Convertimos el esquema en un diseño más detallado, definiendo las llaves primarias y foráneas para asegurar que todo esté bien relacionado.
4. **Normalización:** Aplicamos las reglas de normalización (1FN, 2FN y 3FN) para evitar datos repetidos y hacer que la base de datos sea más eficiente.
5. **Diseño Físico:** Pasamos el diseño a un SGBD como MySQL, definiendo las tablas, columnas y relaciones que se usarán en la base de datos final.

## Construcción del Modelo Conceptual

Las entidades representan los objetos de interés para el sistema, estas contienen atributos que describen sus características. Las relaciones representan las asociaciones entre las entidades (si aplica o si existe).

## Descripción

- **Vehículos:** Id, marca, modelo, año, número de serie único, precio, color, tipo de combustible, tipo de transmisión, estado
- **Clientes:** Id\_cliente, nombre, teléfono, correo electrónico, dirección
- **Vendedores:** Id\_vendedor, nombre, número de empleado, fecha de contratación, múltiples ventas, transacciones de ventas}
- **Ventas:** id\_venta, cliente, vendedor, vehículos, fecha de la venta, total transacción, método de pago
- **Mantenimiento:** id\_mantenimiento, mantenimiento, tipo de servicio, costo, fecha del servicio, cliente, vehículo

## Relaciones y Cardinalidad

### 1. Clientes

- **Relación:** “tiene” Un cliente puede tener muchos vehículos
- **Cardinalidad:** 1:N

### 2. Vendedores

- **Relación:** “Tiene” Un vendedor tiene un cliente por venta
- **Cardinalidad:** 1:1

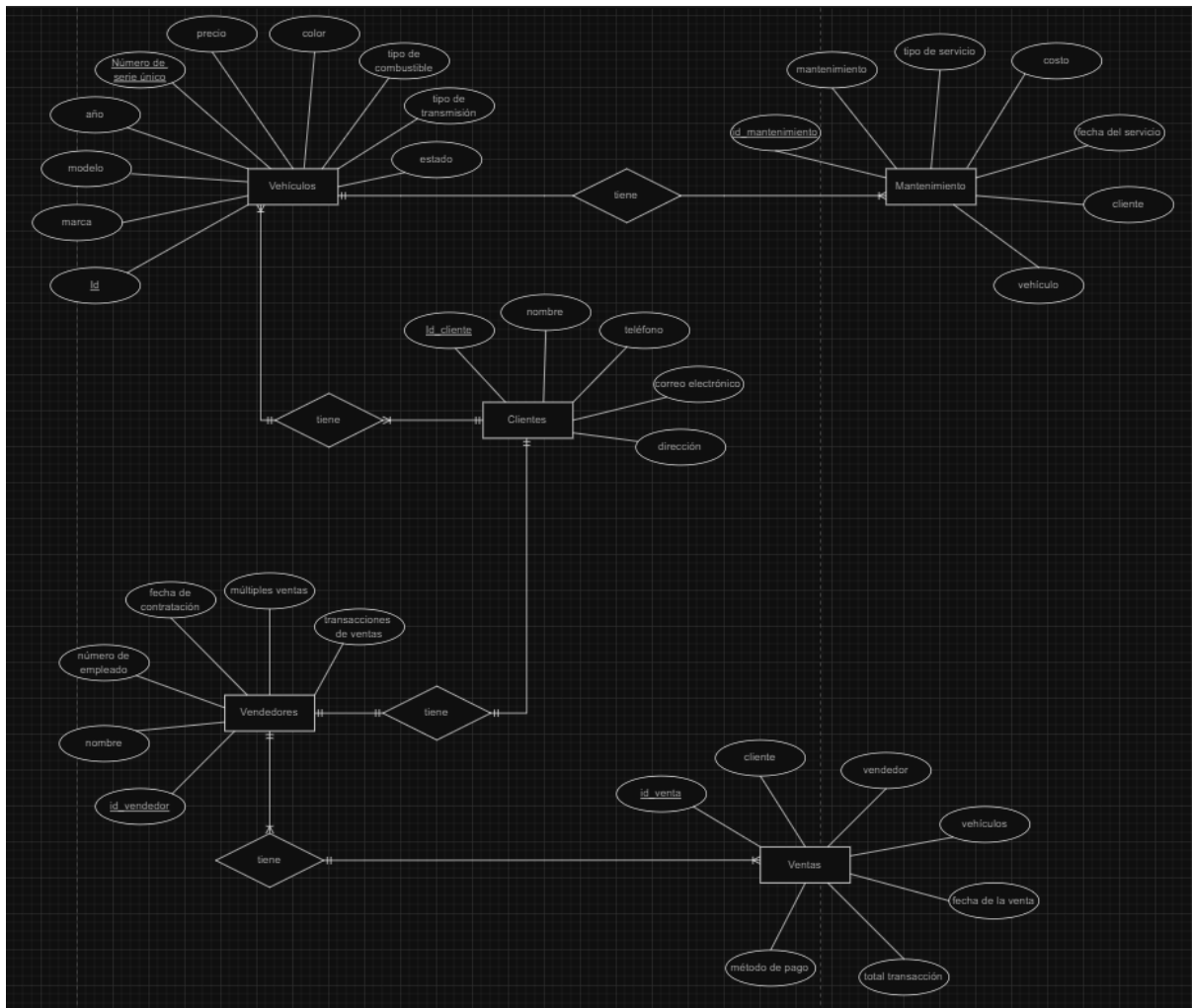
### 3. Vehículos

- **Relación:** “Tiene” Un vehículo puede tener muchos mantenimientos
- **Cardinalidad:** 1:N

### 4. Vendedores

- **Relación:** “Tiene” Un vendedor puede tener muchas ventas
- **Cardinalidad:** 1:N

## Gráfica



## Construcción del Modelo Lógico

Contiene distintas entidades dadas en el caso de estudio. Estas entidades tienen atributos que representan sus características.

## Descripción

### 1. Vehículos:

- Su llave principal es id\_vehículo
- Sus atributos son: marca, modelo, año, número de serie único, precio, color, tipo de combustible, tipo de transmisión, estado.

### 2. Clientes:

- Su llave principal es id\_cliente

- Sus atributos son: nombre, teléfono, correo electrónico, dirección
3. Vendedores:
- Su llave principal es id\_vendedor
  - Sus atributos son: nombre, número de empleado, fecha de contratación, múltiples ventas, transacciones de ventas.
4. Ventas:
- Su llave principal es id\_venta
  - Su llave foránea es id\_vehículo
  - Sus atributos son: cliente, vendedor, vehículos, fecha de la venta, total transacción, método de pago
5. Mantenimiento:
- Su llave principal es id\_mantenimiento
  - Su llave foránea es id\_vehículo
  - Sus atributos son: mantenimiento, tipo de servicio, costo, fecha del servicio, cliente, vehículo

## Relaciones y Cardinalidad

### 1. Clientes - Vehículos:

- Un cliente tiene muchos vehículos



### 2. Vendedor - Cliente:

- Un vendedor tiene un cliente por venta



### 3. Vehículos - Mantenimiento

- Un cliente tiene muchos mantenimientos

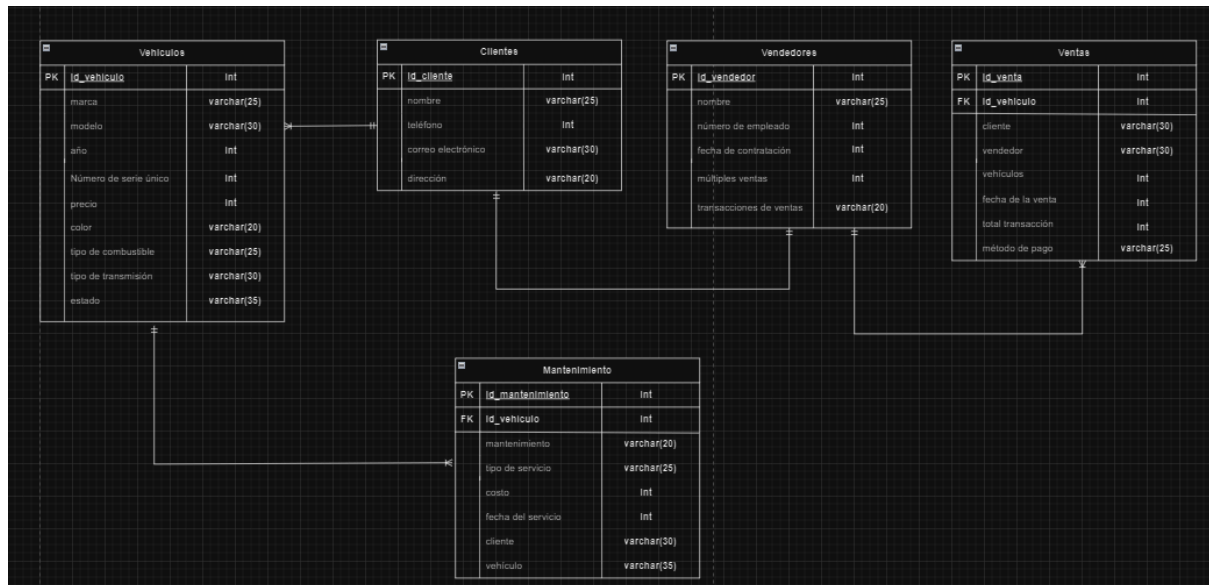


### 4. Vendedores - Ventas

- Un vendedor tiene muchas ventas



## Gráfica



## Normalización del Modelo Lógico

La normalización es el proceso de organizar las tablas y relaciones en la base de datos para evitar datos repetidos y asegurar la integridad. Esto se logra aplicando tres formas normales (1FN, 2FN y 3FN), cada una con reglas que hacen que la base de datos sea más eficiente y fácil de manejar.

### Primera Forma Normal (1FN)

Se establece que los atributos deben ser atómicos, es decir que el número de valores que puede tomar un atributo debe limitarse a uno. Este se basa en las características del modelo relacional. Algunas de estas son que cada fila de la misma tabla debe ser única y que debe prevalecer un crecimiento vertical de los datos y no horizontal.

### Descripción

Se asegura que cada atributo en una tabla tenga un solo valor (atómico) y no se repita. Por ejemplo, en la tabla **Clientes**, el campo "teléfono" no puede almacenar más de un número, por lo que si un cliente tiene varios números, estos deben almacenarse en registros diferentes o en una tabla adicional si es necesario.



## Segunda Forma Normal (2FN)

Se debe tener una clave clave principal, de preferencia simple. Cada atributo de la tabla debe depender totalmente del atributo clave. Este se basa en el concepto de dependencias funcionales.

### Descripción

Aplica a tablas con claves compuestas. Asegura que cada campo en la tabla dependa completamente de la clave primaria. En la tabla **Ventas**, por ejemplo, cada registro debe depender completamente de "id\_venta", evitando que se almacenen datos que no estén directamente relacionados con esa venta específica.

## Tercera Forma Normal (3FN)

Se basa en el concepto de dependencias transitivas. Para esto, debe estar primero en 2FN y no deben existir atributos no principales que dependan transitivamente del atributo clave.

### Descripción

Elimina dependencias transitivas. Esto significa que los atributos que no forman parte de la clave primaria no deben depender de otros atributos que tampoco son claves. En la tabla **Mantenimiento**, el "costo" debe depender solo de "id\_mantenimiento" y no de otros atributos de la tabla, garantizando que el costo está relacionado sólo con el servicio específico.

## Construcción del Modelo Físico

Ya esta es la última etapa del proceso, se basa en el modelo lógico y se utiliza para implementar dicho modelo en una base de datos real. Este modelo consta de tablas, columnas y relaciones.

### Descripción

El modelo físico se diseñó para funcionar en MySQL, donde cada entidad se representa como una tabla compuesta por sus atributos correspondientes, organizados en columnas con tipos de datos específicos según sea necesario.

## Tablas

Para crear la base de datos utilice el siguiente comando:

```
CREATE DATABASE Concesionario;
```

Para utilizar la base de datos ocupe el siguiente comando:

```
USE Concesionario;
```

Comenzaremos creando las tablas junto con sus tipos de datos correspondientes. Para esto, utilicé los siguientes comandos:

### 1. Creación de tabla Vehículos

```
CREATE TABLE Vehiculos(  
    id_vehiculo INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    marca VARCHAR(25) NOT NULL,  
    modelo VARCHAR(30) NOT NULL,  
    año INT NOT NULL,  
    numero_de_serie_unico INT NOT NULL,  
    precio INT NOT NULL,  
    color VARCHAR(20) NOT NULL,  
    tipo_de_combustible VARCHAR(25) NOT NULL,  
    tipo_de_transmisión VARCHAR(30) NOT NULL,  
    estado VARCHAR(35) NOT NULL  
);
```

### 2. Creación de tabla Clientes

```
CREATE TABLE Clientes(  
    id_cliente INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(25) NOT NULL,  
    telefono INT NOT NULL,  
    correo_electronico VARCHAR(30) NOT NULL,  
    direccion VARCHAR(20) NOT NULL  
);
```

### 3. Creación de tabla Vendedores

```
CREATE TABLE Vendedores(  
    id_vendedor INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(25) NOT NULL,  
    numero_de_empleado INT NOT NULL,  
    fecha_de_contratacion INT NOT NULL,  
    multiples_ventas INT NOT NULL,  
    transacciones_de_ventas VARCHAR(20) NOT NULL  
);
```

### 4. Creación de tabla Ventas

```
CREATE TABLE Ventas(  
    id_venta INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    cliente VARCHAR(30) NOT NULL,  
    vendedor VARCHAR(30) NOT NULL,  
    vehiculos INT NOT NULL,  
    fecha_de_la_venta INT NOT NULL,  
    total_transaccion INT NOT NULL,  
    metodo_de_pago VARCHAR(25) NOT NULL,  
    FOREIGN KEY(Vehiculos) REFERENCES Vehiculos(id_vehiculo)  
);
```

### 5. Creación de tabla Mantenimiento

```
CREATE TABLE Mantenimiento(  
    id_mantenimiento INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    mantenimiento VARCHAR(20) NOT NULL,  
    tipo_de_servicio VARCHAR(25) NOT NULL,  
    costo INT NOT NULL,  
    fecha_del_servicio INT NOT NULL,  
    cliente VARCHAR(30) NOT NULL,  
    vehiculos INT NOT NULL,  
    FOREIGN KEY(Vehiculos) REFERENCES Vehiculos(id_vehiculo)  
);
```

## Construcción del Diagrama UML

Se ha diseñado un diagrama UML tomando como referencia la normalización para entender mejor los diseños, la arquitectura del código y la implementación propuesta. Este enfoque nos permitirá tener una visión clara y detallada de cómo se manejan cada una de las consultas, funcionalidades y los usuarios en la base de datos.

### Descripción

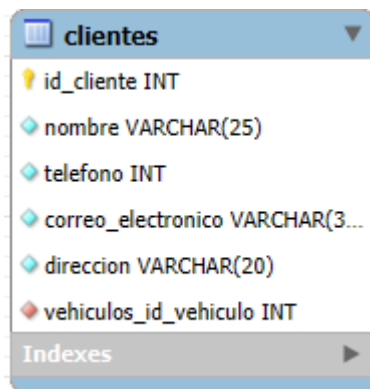
El diagrama UML se ha diseñado con el objetivo de representar detalladamente la estructura de cada tabla y sus relaciones. Este diagrama ilustra claramente el tipo de dato correspondiente a cada atributo, así como la identificación de claves primarias (primary keys) y claves foráneas (foreign keys).

Comenzaremos creando las tablas junto con sus tipos de datos correspondientes:

### 1. Tabla Vehículos



## 2. Tabla Clientes

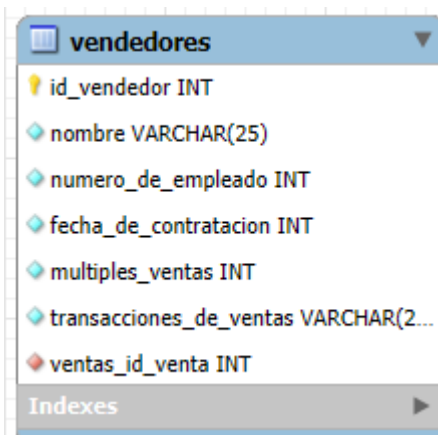


The screenshot shows the 'clientes' table with the following fields:

Field Name	Field Type
id_cliente	INT
nombre	VARCHAR(25)
telefono	INT
correo_electronico	VARCHAR(300)
direccion	VARCHAR(200)
vehiculos_id_vehiculo	INT

Below the fields is a tab labeled 'Indexes' with a right-pointing arrow.

## 3. Tabla Vendedores



The screenshot shows the 'vendedores' table with the following fields:

Field Name	Field Type
id_vendedor	INT
nombre	VARCHAR(25)
numero_de_empleado	INT
fecha_de_contratacion	INT
multiples_ventas	INT
transacciones_de_ventas	VARCHAR(200)
ventas_id_venta	INT

Below the fields is a tab labeled 'Indexes' with a right-pointing arrow.

## 4. Tabla Ventas

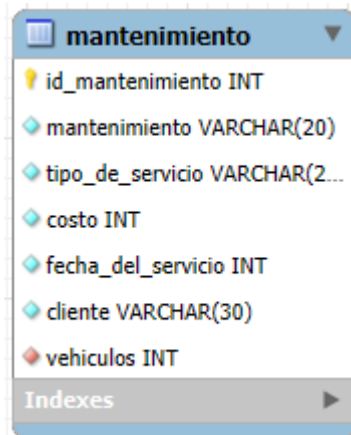


The screenshot shows the 'ventas' table with the following fields:

Field Name	Field Type
id_venta	INT
cliente	VARCHAR(30)
vendedor	VARCHAR(30)
vehiculos	INT
fecha_de_la_venta	INT
total_transaccion	INT
metodo_de_pago	VARCHAR(200)

Below the fields is a tab labeled 'Indexes' with a right-pointing arrow.

## 5. Tabla Mantenimiento



## Gráfica

