# UrbanPark - Sistema de Gestión de Parqueaderos

Autores: Alejandro Tafur Jojoa
Juan Camilo Penagos Molina

Universidad Corhuila

Materia: Modelado y Gestión de Bases de Datos

Docente: Jesus Ariel Gonzales Bonilla

Neiva - Huila

2025

**Tabla de contenido**

# 1. Introducción

## 1.1. Proposito

UrbanPark es un sistema diseñado para la administración integral de parqueaderos urbanos. Permite gestionar clientes, vehículos, reservas, tarifas, facturación, auditoría de eventos y control de acceso por roles.

El proyecto incluye diseño de base de datos en MySQL, construcción de MER y MR, creación de tablas, funciones, procedimientos, triggers y consultas JOIN.

# 2. Objetivos del proyecto

## 2.1. Objetivo general

Diseñar e implementar una base de datos relacional optimizada para un sistema de gestión de parqueaderos que cubra los procesos de reserva, facturación, registro de usuarios, tarifas y auditoría.

## 2.2. Objetivos específicos

- ✓ Modelar el dominio mediante un Modelo Entidad-Relación (MER) y un Modelo Relacional (MR).
- ✓ Implementar la base de datos en MySQL siguiendo buenas prácticas de normalización.
- ✓ Crear procedimientos almacenados, funciones, triggers y consultas complejas.
- ✓ Poblar la base de datos con mínimo 20 registros por tabla.
- ✓ Garantizar integridad referencial y seguridad mediante roles y auditoría.

## 3. Alcance del sistema:

UrbanPark permite gestionar:

### 3.1. Módulos Principales

- Clientes
- Vehículos
- Reservas y gestión de parqueaderos
- Facturación
- Tarifas
- Usuarios, roles y seguridad
- Auditoría de eventos
- Sesiones activas

### 3.2. Procesos Cubiertos

- Registro de cliente y sus vehículos.

- Creación y cierre de reservas.

- Calculo automático del costo según tarifas.

- Facturación automática.

- Control de acceso según rol.

- Registro automático de eventos mediante triggers.

## 4. Requerimientos

### 4.1. Requerimientos Funcionales

**RF01 – Gestión de Clientes**

El sistema debe permitir registrar, actualizar, consultar y eliminar información de los clientes.

Incluye:

- ✓ Validación de correo único.

- ✓ Múltiples vehículos asociados al cliente.

- ✓ Búsqueda por nombre, teléfono o correo.

**RF02 – Gestión de Vehículos**

El sistema debe registrar los vehículos de los clientes, permitiendo:

- ✓ Registrar placa, tipo y cliente asociado.

- ✓ Validar unicidad de la placa.

- ✓ Consultar historial de reservas del vehículo.

**RF03 – Gestión de Reservas**

El sistema debe permitir crear, actualizar, consultar y cerrar reservas.

Incluye:

- ✓ Validar disponibilidad del parqueadero.

- ✓ Registrar fecha/hora entrada y salida.

- ✓ Estados: Activa, Finalizada, Cancelada.

- ✓ Relación directa con el vehículo y parqueadero.

**RF04 – Facturación Automática**

Al finalizar una reserva, el sistema debe generar de forma automática la factura correspondiente.

Incluye:

- ✓ Cálculo de horas usando la función interna.

- ✓ Aplicación de tarifa según tipo de vehículo.

- ✓ Registro del método y fecha de pago.

**RF05 – Gestión de Usuarios y Roles**

El sistema debe gestionar usuarios con distintos niveles de permisos.

Incluye:

- ✓ Crear, editar y eliminar usuarios.

- ✓ Asignar roles (Administrador, Operador, Auditor).

- ✓ Validación de credenciales y control de autenticación.

**RF06 – Auditoría de Operaciones**

El sistema debe registrar automáticamente las acciones críticas del sistema.

Incluye:

- ✓ Uso de triggers para registrar inserciones, actualizaciones y eliminaciones.

- ✓ Log detallado con usuario, fecha y tipo de acción.

- ✓ Auditoría vinculada a las reservas y usuarios que ejecutan cambios.

### 4.2. Requerimientos no funcionales

**RNF01 – Seguridad**

El sistema debe garantizar protección de la información mediante:

- ✓ Contraseñas cifradas.
- ✓ Roles con permisos definidos.
- ✓ Auditoría obligatoria de operaciones críticas.

**RNF02 – Rendimiento**

El sistema debe responder de manera eficiente.

Incluye:

- ✓ Tiempo de respuesta < 2 segundos en consultas frecuentes.
- ✓ Uso de índices en claves primarias y foráneas.
- ✓ Optimización de consultas JOIN.

**RNF03 – Escalabilidad**

El sistema debe permitir crecimiento sin cambios estructurales.

Incluye:

- ✓ Capacidad de agregar más parqueaderos.

- ✓ Soporte para miles de reservas diarias.

- ✓ Extensión para nuevos tipos de tarifa y reglas de negocio.

**RNF04 – Disponibilidad**

El sistema debe estar disponible para su uso continuo.

Incluye:

- ✓ Baja probabilidad de fallos.

- ✓ Soporte para respaldos periódicos.

- ✓ Manejo correcto de sesiones activas.

**RNF05 – Mantenibilidad**

El sistema debe permitir actualización y mantenimiento sencillo.

Incluye:

- ✓ Estructura de base de datos bien documentada.

- ✓ Uso de funciones y procedimientos para encapsular lógica.

- ✓ Nombres estandarizados en tablas, campos y relaciones.

**RNF06 – Usabilidad**

La interfaz del sistema debe ser intuitiva y clara para diferentes usuarios.

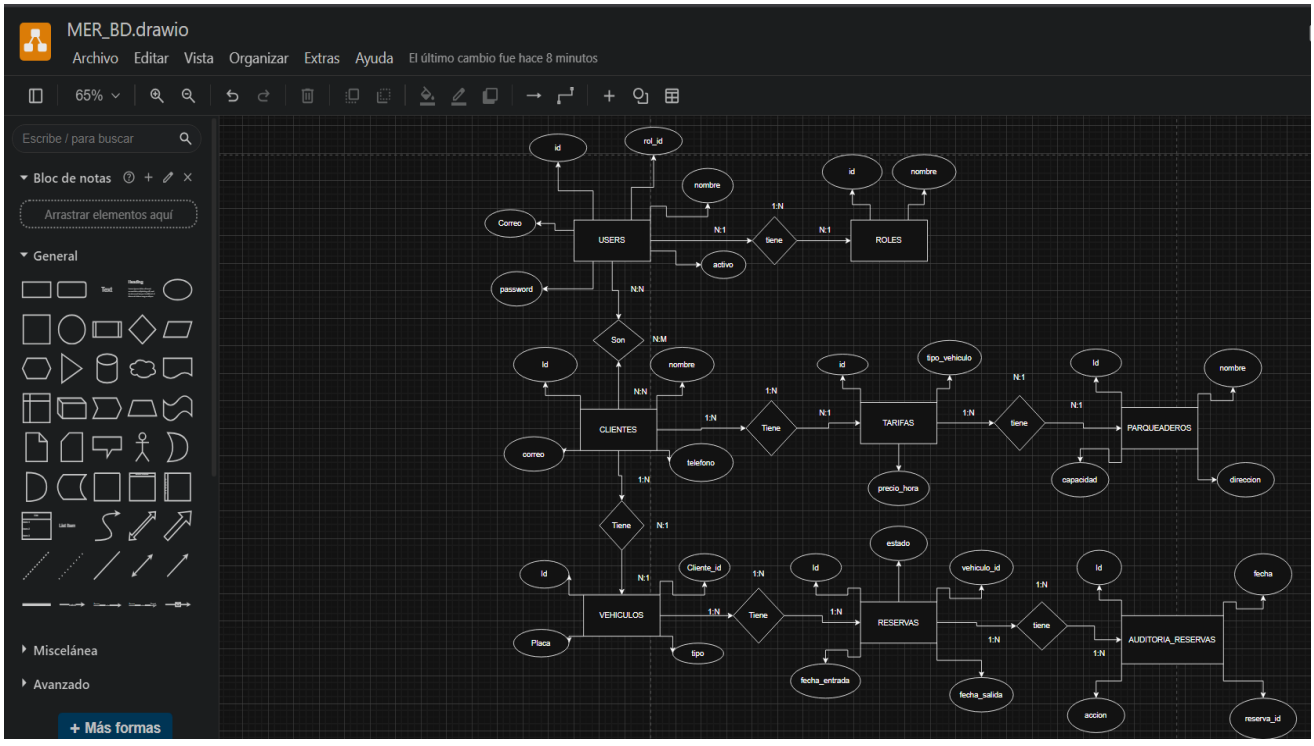
Incluye:

✓ Formularios simples.

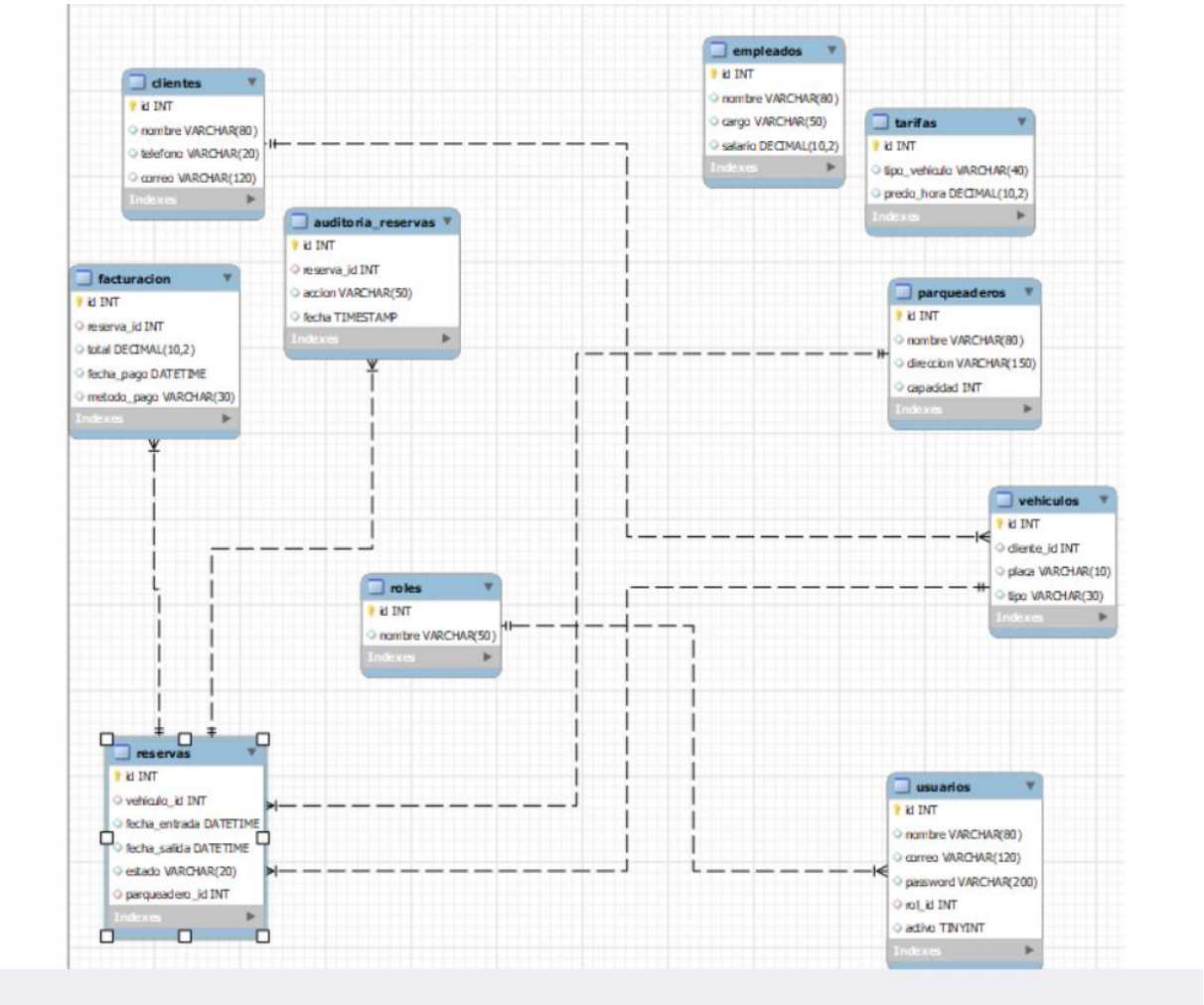
✓ Mensajes de error comprensibles.

✓ Flujo guiado en procesos críticos (reservar, facturar).

## 5. Modelo conceptual (MER)



## 6. Modelo relacional (MR)




## 7. Implementacion SQL

### A. Creación de Tablas (Ejemplos)

La base de datos se crea con el nombre urbanpark.

Tabla clientes:

```
CREATE TABLE clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(80),
    telefono VARCHAR(20),
    correo VARCHAR(120)
```

);

Tabla vehiculos:

```
CREATE TABLE vehiculos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT,
    placa VARCHAR(10),
    tipo VARCHAR(30),
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)
);
```

**B. Procedimientos, funciones y triggers**

| Tipo | Nombre | Descripción y Muestra (Lógica) |
|---|---|---|
| **Procedimiento** | crear_reserva | Inserta una nueva reserva con la hora actual y estado 'Activa' al recibir un vehiculo_id19. |
| **Función** | calcular_horas | Calcula y retorna la diferencia en **horas** enteras entre una fecha_inicio y una fecha_fin20. |
| **Trigger** | tr_reserva_insert | Se ejecuta **después** de un INSERT en la tabla reservas, registrando el evento en una tabla de logs (logs_triggers)21. |

**C. Consultas JOIN(ejemplos de reportes)**

Se implementaron consultas JOIN para generar reportes, cruciales para la gestión del negocio.

- Cliente y sus Vehículos: Une clientes y vehiculos para listar el nombre del cliente, la placa y el tipo de vehículo.

- Detalle de Facturación de Reservas: Une reservas y facturacion para obtener el ID de

reserva y el total facturado.

- Reporte Completo de Reserva/Factura/Tarifa: Une reservas, vehiculos, tarifas y facturacion para obtener el ID de reserva, tipo de vehículo, precio por hora y total facturado.

- Usuarios y su Rol: Une usuarios y roles para mostrar el nombre del usuario y el rol asignado.

**8. Conclusiones**

- ✓ Cumplimiento: El proyecto UrbanPark ha completado el diseño relacional, la implementación de estructuras, lógica de negocio (procedimientos, funciones, triggers) y la dotación de datos de prueba.

- ✓ Características Clave: La base de datos está diseñada para soportar auditoría, control de accesos y la generación de reportes complejos mediante consultas JOIN.