














PROYECTO BASE DE DATOS

DAVID DELGADO
CORREA








2. ÍNDICE





-  1. Portada
-  2. Índice
-  3. Introducción
-  4. Análisis del Enunciado
-  5. Modelo Conceptual
-  6. Modelo Relacional
-  7. Implementación en MySQL
-  8. CONSULTAS SQL
-  9. Ampliación de la Base de Datos
-  10. Vistas y Triggers
-  11. Pruebas y Validación
-  12. Conclusiones
-  13. Anexos

3. INTRODUCCIÓN

En CarRentalX, una empresa líder en el sector de alquiler de coches, estamos comprometidos con la excelencia en nuestro servicio y la satisfacción del cliente. Con el objetivo de mejorar nuestra eficiencia operativa y proporcionar una experiencia de alquiler más fluida, estamos buscando implementar un sistema de gestión de alquiler de coches completo.






Actualmente, manejamos una gran cantidad de información de manera manual, lo que genera retrasos en los procesos de reserva, devolución y facturación. Para solucionar esto, necesitamos un sistema que nos permita gestionar de manera eficiente:

-  Clientes: Información personal y preferencias.
 -  Vehículos: Detalles técnicos, disponibilidad y estado.
 -  Reservas: Fechas, duración, costos y vehículos asociados.
 -  Sucursales: Ubicaciones donde se realizan los alquileres.
 -  Empleados: Personal encargado de gestionar las operaciones.
- Además, el sistema debe ser capaz de generar informes sobre:

-  Vehículos más alquilados.
-  Clientes frecuentes.
-  Ingresos mensuales por sucursal.
-  Instrucciones para los Analistas








1. 🔍 Identificación de Entidades y Atributos

Utilizando la descripción del caso, identifiquen las entidades clave que deben formar parte del sistema. Estas entidades deben incluir, como mínimo:

-  Clientes: Información personal (nombre, DNI, dirección, teléfono, email).
 -  Vehículos: Detalles técnicos (marca, modelo, año, matrícula, tipo de combustible, estado).
 -  Reservas: Fechas de inicio y fin, costo total, estado de la reserva.
 -  Sucursales: Ubicación (nombre, dirección, teléfono).
 -  Empleados: Información del personal (nombre, cargo, sucursal asignada).
- Asegúrense de definir los atributos relevantes para cada entidad.

2. 🔄 Relaciones entre Entidades

Determinen cómo se relacionan las entidades entre sí. Por ejemplo:

- Un  cliente realiza una o varias  reservas.
- Una  reserva está asociada a un  vehículo y una  sucursal.
- Un  empleado gestiona las reservas en una  sucursal.

3. 💻 Implementación en MySQL

Una vez definido el modelo conceptual, implementen el modelo relacional en MySQL. Incluyan:

- Scripts de creación de tablas.
- Inserción de datos de prueba.
- Consultas SQL para obtener información relevante (por ejemplo, listado de empleados por departamento, puestos vacantes, etc.).

4. ✨ Ampliación del Sistema

Propongan y desarrollen al menos dos funcionalidades adicionales que mejoren el sistema. Algunas ideas:

Implementación de vistas para simplificar consultas frecuentes.

Creación de triggers para actualizar automáticamente el estado de los vehículos después de una reserva.

Generación de informes de ingresos mensuales por sucursal.

5. 📄 Documentación

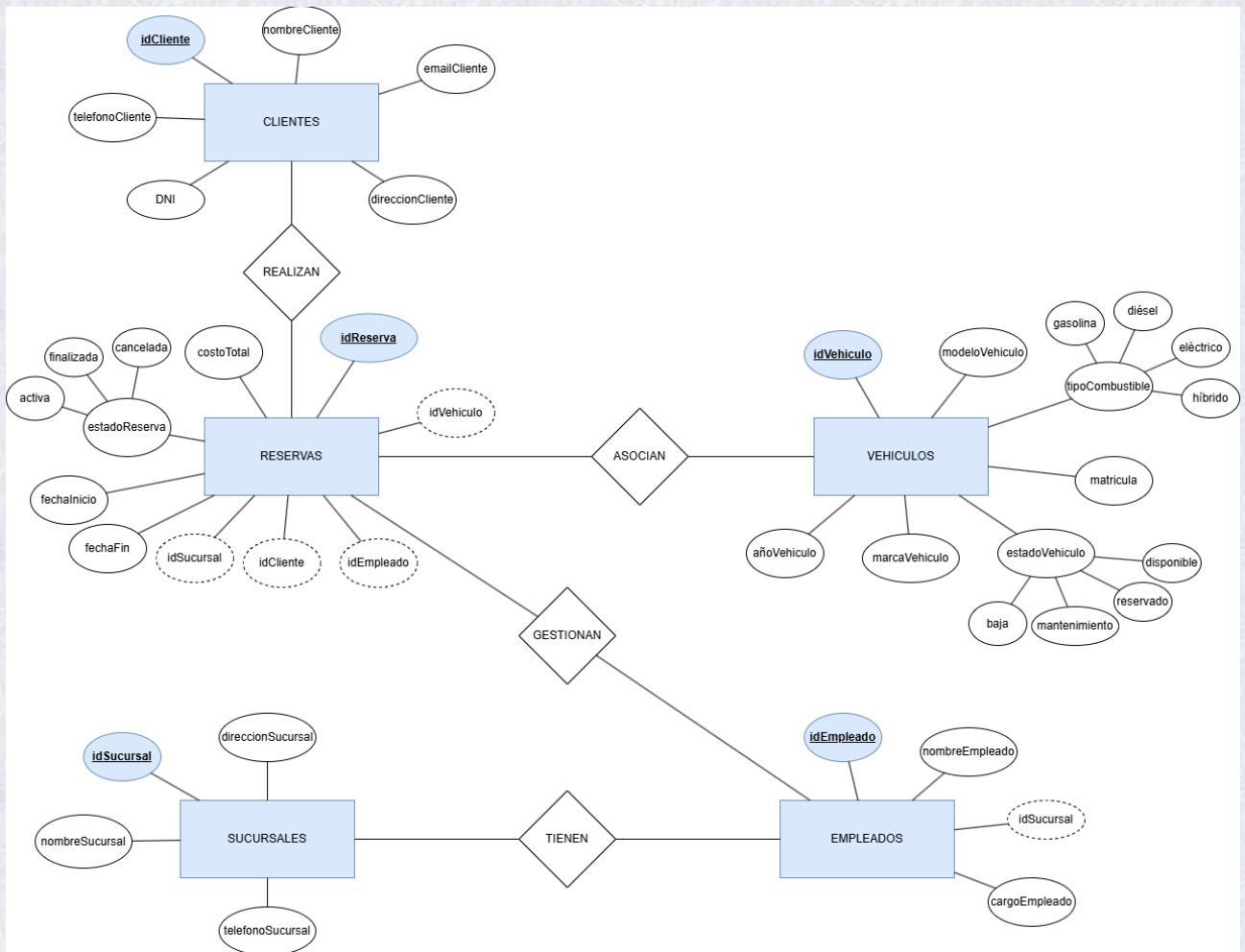
📄 Documentación Deberan Elaborar un informe técnico, tienen las instrucciones en el readme de la carpeta

4. Análisis del Enunciado

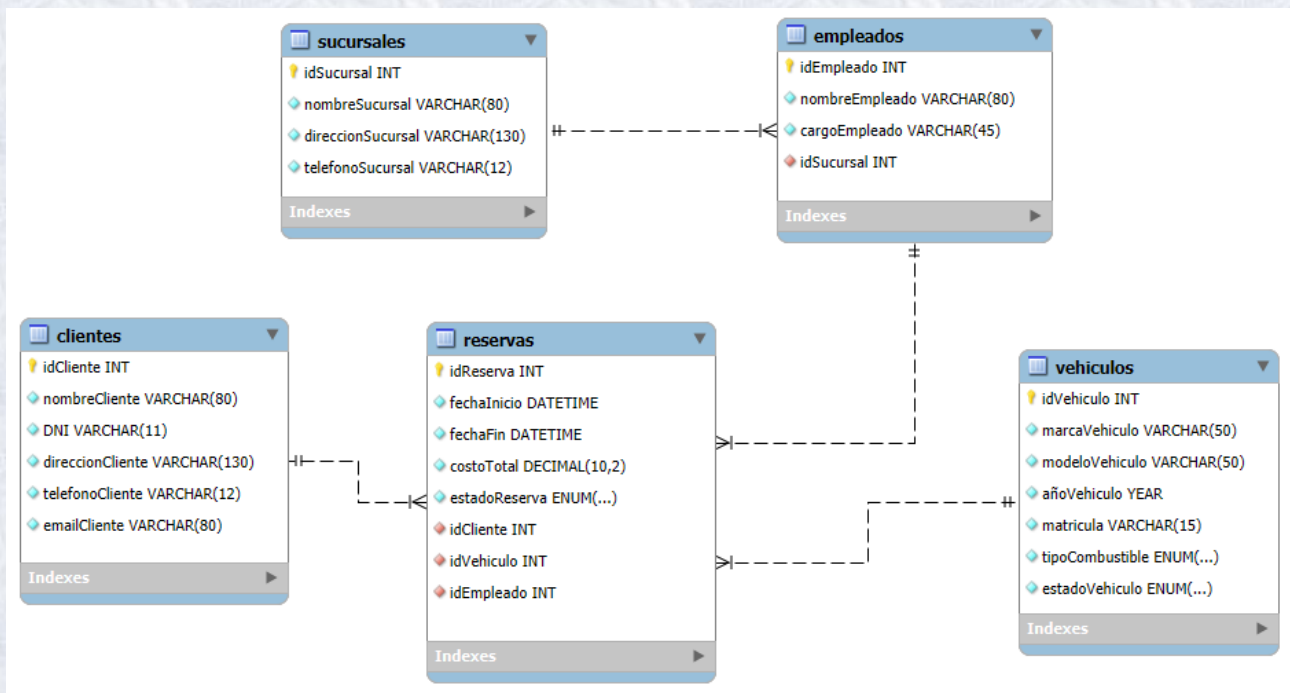
En este proyecto se nos solicita desarrollar un sistema de gestión para Car_Rental, una empresa de alquiler de coches que actualmente maneja su información de forma manual. Nuestro objetivo es automatizar y optimizar procesos clave como la gestión de clientes, vehículos, reservas, sucursales y empleados, así como permitir la generación de informes importantes, como los vehículos más alquilados, los clientes frecuentes y los ingresos mensuales por sucursal.

Para ello, debemos identificar las entidades y sus atributos, definir las relaciones entre ellas e implementar el modelo en MySQL a través de scripts de creación de tablas, inserción de datos y consultas útiles. Además, se nos pide proponer al menos dos funcionalidades adicionales que mejoren el sistema, como la creación de vistas o triggers, y documentar todo el proceso en un informe técnico.

5. MODELO CONCEPTUAL

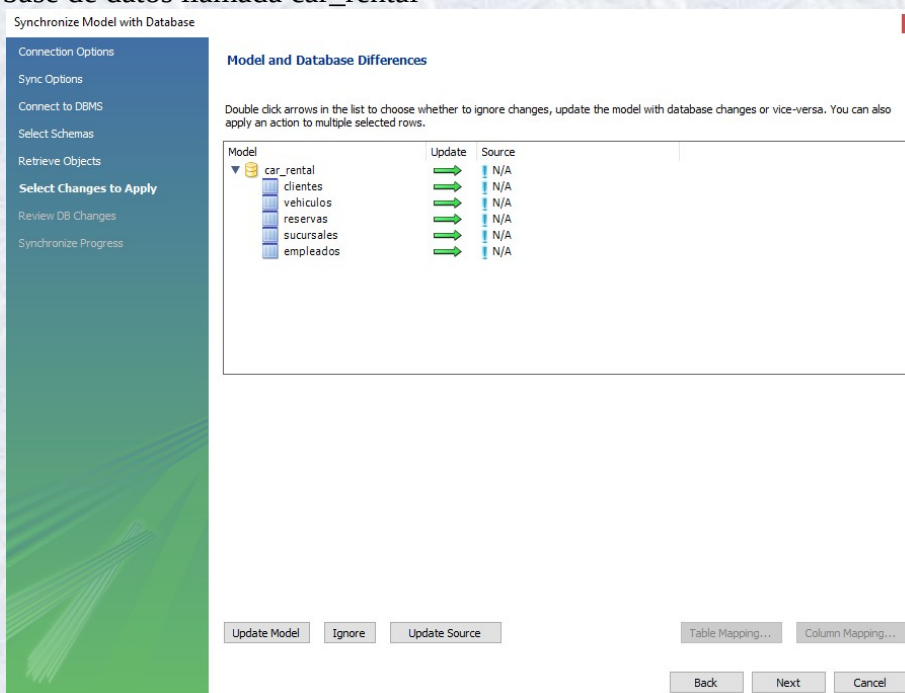


6. MODELO RELACIONAL



7. IMPLEMENTACIÓN EN MYSQL

Llevamos a cabo la implementación del modelo relacional y sincronizando el modelo creamos la base de datos llamada `car_rental`



INSERCIÓN DE DATOS

Insertamos datos en la base de datos para el correcto funcionamiento y validación del sistema. Esto nos permite:

- Validar el modelo de datos: Al poblar las tablas con información realista, podemos verificar que las relaciones entre las entidades (clientes, empleados, vehículos, reservas, etc.) están correctamente estructuradas y que el modelo relacional es coherente con los requisitos del sistema.
- Ejecutar consultas SQL: La base de datos con datos completos nos permite realizar pruebas de consultas, filtros, uniones y agregaciones. Esto es crucial para simular escenarios de uso real y para asegurar que las consultas de gestión y reporte son efectivas.
- Simular casos de uso reales: Con los datos insertados, podemos simular de forma precisa cómo los usuarios interactuarían con el sistema, como la creación de reservas, la asignación de vehículos a clientes o la actualización de estados de las reservas. Esto ayuda a detectar posibles problemas antes de la implementación final.
- Desarrollar y probar características adicionales: Los datos servirán en el futuro para crear vistas y triggers que automatizan tareas, optimizan consultas complejas y aseguran la integridad de los datos en tiempo real.

8. CONSULTAS PROPUESTAS

Estas consultas están diseñadas para optimizar la gestión del sistema de alquiler de vehículos al proporcionar información clave sobre las reservas, la flota de vehículos, el desempeño de los empleados y el comportamiento de los clientes. Facilitan la toma de decisiones informadas y ayudan en la planificación estratégica, el análisis de ingresos y la mejora del servicio al cliente.

1. Listar todas las reservas activas con datos del cliente y vehículo.

Esta consulta permite obtener una visión completa de las reservas activas en el sistema, incluyendo datos de los clientes y los vehículos asignados. Ayuda a los empleados a gestionar las reservas actuales, verificar la disponibilidad y atender a los clientes de manera eficiente.

2. Obtener el número de reservas finalizadas por cliente

Muestra cuántas reservas finalizadas ha tenido cada cliente. Es útil para el análisis de la fidelidad de los clientes y para identificar a aquellos que han utilizado el servicio con más frecuencia, lo que podría llevar a estrategias de fidelización.

3. Ver el estado actual de todos los vehículos y si están asignados a una reserva activa

Permite conocer la disponibilidad de todos los vehículos en la flota, indicando si están disponibles, asignados a una reserva activa o fuera de servicio (en mantenimiento o baja). Facilita la gestión de la flota y la asignación de vehículos a nuevos clientes.

4. Cantidad de reservas gestionadas por cada empleado (solo roles autorizados)

Permite analizar la carga de trabajo de cada empleado, mostrando cuántas reservas han gestionado. Esto puede ayudar a evaluar el desempeño de los empleados, reconocer a los más productivos y gestionar mejor los recursos humanos en el sistema.

5. Ingresos generados por mes a partir de las reservas finalizadas

Muestra los ingresos generados por las reservas finalizadas en cada mes. Es útil para el análisis financiero, ayudando a identificar patrones estacionales, hacer previsiones de ingresos y tomar decisiones estratégicas basadas en la rentabilidad.

6. Ver el historial completo de reservas de un cliente específico

Permite consultar todo el historial de reservas de un cliente en particular, lo que es útil para ofrecer un servicio personalizado, verificar el comportamiento del cliente o gestionar posibles incidencias relacionadas con el historial de alquileres.

9. AMPLIACIÓN DE LA BASE DE DATOS

1. Añadimos una columna kilometraje a la tabla vehículos

Nos permite registrar el número de kilómetros recorridos por cada vehículo.

Planificar mantenimientos preventivos.

Evaluar el desgaste o la antigüedad del vehículo.

Hacer seguimiento del uso en función de las reservas.

2. Añadimos la columna fechaRegistro a la tabla clientes

Nos indica la fecha en que un cliente empezó a usar el servicio (tomando como referencia su primera reserva). Esto nos ayuda a:

Analizar la antigüedad y fidelidad de los clientes.

Enviar promociones a clientes nuevos o veteranos.

Hacer segmentaciones en reportes o análisis de comportamiento.

3. Creamos la nueva tabla mantenimientos

Para registrar todas las acciones de mantenimiento realizadas a los vehículos: tipo, fecha, descripción y costo. Esto es clave para:

Tener un historial técnico de cada coche.

Calcular gastos por vehículo o por periodo.

Garantizar que solo vehículos en condiciones óptimas estén disponibles para alquilar.

Con estos cambios, nuestra base de datos se vuelve más completa y funcional abriendo puertas a crear vistas y triggers útiles que veremos a continuación.

10. VISTAS Y TRIGGERS

Para facilitar la gestión de la base de datos vamos a crear unas vistas y unos triggers que convierten la base de datos en un sistema más robusto, eficiente y útil, tanto para la gestión diaria como para el análisis estratégico.

Vista 1: vista_historial_completo_clientes

Esta vista muestra un historial completo de los clientes, incluyendo su información de contacto, la cantidad total de reservas que han hecho, el dinero que han gastado en total y la fecha de su primera y última reserva.

Esta vista nos sirve para:

- Analizar el comportamiento de cada cliente a lo largo del tiempo.
- Detectar clientes fieles o con mayor valor económico.
- Evaluar posibles estrategias de retención o recompensas.

```
2  ## Vista 1: vista_historial_completo_clientes
3  • CREATE VIEW vista_historial_completo_clientes AS
4  SELECT c.idCliente, c.nombreCliente, c.emailCliente,
5  COUNT(r.idReserva) AS totalReservas,
6  SUM(r.costTotal) AS gastoTotal,
7  MIN(r.fechaInicio) AS primeraReserva,
8  MAX(r.fechaFin) AS ultimaReserva
9  FROM clientes c
10 LEFT JOIN reservas r ON c.idCliente = r.idCliente
11 GROUP BY c.idCliente, c.nombreCliente, c.emailCliente
12 ORDER BY gastoTotal DESC;
13
14 • SELECT * FROM vista_historial_completo_clientes;
15
```

	idCliente	nombreCliente	emailCliente	totalReservas	gastoTotal	primeraReserva	ultimaReserva
▶	7	Isabel Herrera	isabel.herrera@gmail.com	3	570.00	2024-05-15 09:00:00	2025-02-13 00:00:00
	6	Francisco Delgado	fran.delgado@hotmail.com	3	490.00	2024-06-01 10:30:00	2025-04-18 16:00:00
	5	Elena Sánchez López	elena.sanchez@protonmail.com	1	400.00	2025-04-01 00:00:00	2025-04-10 00:00:00
	4	Miguel Guzmán González	miguel.guzman@hotmail.com	2	360.00	2025-03-18 00:00:00	2025-04-19 18:00:00
	3	Maria Pilar Lucena	mariapilar@gmail.com	2	335.00	2024-08-28 12:30:00	2025-03-25 00:00:00
	1	David Delgado Correa	daviddc@outlook.com	1	210.00	2025-02-01 00:00:00	2025-02-07 00:00:00
	2	Carlos Pérez Pérez	carlos.perez@gmail.com	1	180.00	2025-02-10 00:00:00	2025-02-15 00:00:00

Vista 2: vista_mantenimientos_recientes

Esta vista muestra los últimos mantenimientos realizados, con información del vehículo, el tipo de mantenimiento y su fecha. Muy útil para gestionar la salud de la flota.

Esta vista nos sirve para:

- Ver rápidamente qué vehículos han pasado por mantenimiento.
- Controlar los gastos recientes de la flota.
- Anticipar futuras revisiones o inspecciones técnicas.

```
16  ## Vista 2: vista_mantenimientos_recientes
17  • CREATE VIEW vista_mantenimientos_recientes AS
18  SELECT m.idMantenimiento, v.marcaVehiculo, v.modeloVehiculo, m.tipoMantenimiento, m.fechaMantenimiento, m.costo, m.descripcion
19  FROM mantenimientos m
20  JOIN vehiculos v ON m.idVehiculo = v.idVehiculo
21  ORDER BY m.fechaMantenimiento DESC;
22
23  • SELECT * FROM vista_mantenimientos_recientes;
24
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [A](#)

	idMantenimiento	marcaVehiculo	modeloVehiculo	tipoMantenimiento	fechaMantenimiento	costo	descripcion
▶	4	Toyota	Corolla	Mantenimiento preventivo	2025-04-25	100.00	Revisión general
	3	Peugeot	308	Cambio de neumáticos	2025-04-19	300.00	Reemplazo de los cuatro neumáticos por desgaste
	2	Ford	Focus	Revisión general	2025-04-18	220.00	Revisión completa de 80.000 km, cambio de ace...
	1	Volkswagen	Golf	Cambio de frenos	2025-04-15	160.00	Sustitución de pastillas de freno delanteras y tr...



Vista 3: vista_ingresos_mensuales

Esta vista agrupa los ingresos generados por las reservas finalizadas en función del mes. Es ideal para reportes financieros o para evaluar el rendimiento del negocio mes a mes.

Esta vista nos sirve para:

- Controlar los ingresos mensuales generados por las reservas.
- Comparar el rendimiento entre distintos meses.
- Tomar decisiones estratégicas en función de la temporada alta/baja.

```
25  ## Vista 3: vista_ingresos_mensuales_sucursal
26  ● CREATE VIEW vista_ingresos_mensuales_sucursal AS
27  SELECT s.nombreSucursal,
28  DATE_FORMAT(r.fechaFin, '%Y-%m') AS mes,
29  SUM(r.costoTotal) AS ingresos_mensuales
30  FROM reservas r
31  JOIN empleados e ON r.idEmpleado = e.idEmpleado
32  JOIN sucursales s ON e.idSucursal = s.idSucursal
33  WHERE r.estadoReserva = 'Finalizada'
34  GROUP BY s.nombreSucursal, mes
35  ORDER BY mes, s.nombreSucursal;
36
37  ● SELECT * FROM vista_ingresos_mensuales_sucursal;
38
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	nombreSucursal	mes	ingresos_mensuales
►	Sucursal La Laguna	2024-05	200.00
	Sucursal Costa Adeje	2024-06	250.00
	Sucursal Santa Cruz	2024-06	220.00
	Sucursal Santa Cruz	2024-07	130.00
	Sucursal La Laguna	2024-08	85.00
	Sucursal Costa Adeje	2025-02	120.00
	Sucursal Santa Cruz	2025-02	210.00

Trigger 1: trg_after_insert_actualizar_estado_vehiculo

Creamos un trigger que cuando se inserta una nueva reserva activa, este trigger actualiza automáticamente el estado del vehículo a 'Reservado' si no está ya en mantenimiento o baja. Esto evita inconsistencias si alguien se olvida de actualizar manualmente el estado del coche.

Ventajas de este trigger:

- Automatiza la gestión del estado de los vehículos.
- Mejora la integridad del sistema.
- Reduce errores humanos.

```
39  ## Trigger 1: trg_actualizar_estado_vehiculo
40  DELIMITER $$
41
42  • CREATE TRIGGER trg_actualizar_estado_vehiculo
43  AFTER INSERT ON reservas
44  FOR EACH ROW
45  BEGIN
46  IF NEW.estadoReserva = 'Activa' THEN
47      UPDATE vehiculos
48      SET estadoVehiculo = 'Reservado'
49      WHERE idVehiculo = NEW.idVehiculo
50      AND estadoVehiculo NOT IN ('Mantenimiento', 'Baja');
51  END IF;
52  END $$
53
54  DELIMITER ;
55
56  • ## Prueba y validación del trigger
57  SELECT * FROM vehiculos WHERE idVehiculo = 9;
58
59  • INSERT INTO reservas (fechaInicio, fechaFin, costoTotal, estadoReserva, idCliente, idEmpleado, idVehiculo)
60  VALUES ('2025-04-17 12:00:00', '2025-04-18 16:00:00', 140.00, 'Activa', 6, 9, 9);
61
62  • SELECT * FROM vehiculos WHERE idVehiculo = 9;
```

Result Grid								
Filter Rows: <input type="text"/>								
Edit: Export/Import: Wrap Cell Content:								
	idVehiculo	marcaVehiculo	modeloVehiculo	añoVehiculo	matricula	tipoCombustible	estadoVehiculo	kilometraje
▶	9	Opel	Astra	2023	2164PMM	Gasolina	Reservado	30500
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Trigger 2: trg_after_insert_aud_mantenimiento

Creamos un trigger que registre automáticamente cada inserción en la tabla mantenimientos dentro de una nueva tabla llamada, aud_mantenimientos. Esto nos permitirá tener un historial de acciones para auditorías o seguimientos.

-Creamos la tabla aud_mantenimientos en la que guardaremos un log de cuando y que vehículo entró en mantenimiento.

-Creamos el trigger el cual se ejecuta después de insertar un nuevo mantenimiento y comprobamos y validamos el trigger.

```
64  ## Trigger 2:
65  ## Creamos la tabla aud_mantenimientos
66  ● CREATE TABLE aud_mantenimientos (
67      idLog INT AUTO_INCREMENT PRIMARY KEY,
68      idVehiculo INT NOT NULL,
69      fechaMantenimiento DATETIME NOT NULL,
70      descripcion TEXT,
71      fechaRegistro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
72  );
73
74  ## Creamos el trigger
75  DELIMITER //
76
77  ● CREATE TRIGGER trg_after_insert_aud_mantenimiento
78  AFTER INSERT ON mantenimientos
79  FOR EACH ROW
80  BEGIN
81      INSERT INTO aud_mantenimientos (idVehiculo, fechaMantenimiento, descripcion)
82      VALUES (NEW.idVehiculo, NEW.fechaMantenimiento, NEW.descripcion);
83  END;
84  //
85
86  DELIMITER ;
87
88  ● ## Prueba y validación del trigger
89  INSERT INTO mantenimientos (idVehiculo, fechaMantenimiento, tipoMantenimiento, descripcion, costo)
90  VALUES (1, '2025-04-25 10:00:00', 'Mantenimiento preventivo', 'Revisión general', 100.00);
91
92  ● SELECT * FROM aud_mantenimientos;
```

Result Grid | | Filter Rows: | Edit: | Export/Import: | Wrap

idLog	idVehiculo	fechaMantenimiento	descripcion	fechaRegistro
1	1	2025-04-25 00:00:00	Revisión general	2025-04-20 16:53:23
NULL	NULL	NULL	NULL	NULL

Trigger 3: Evitar reservar vehículos que no están disponibles

Creamos este trigger que se activará antes de insertar una nueva reserva y comprobará que el vehículo esté en estado Disponible. Si no lo está, lanza un error.

Para que nos sirva este trigger:

- Evita que se asignen vehículos en Mantenimiento, Reservado o Baja a nuevas reservas.
- Mejora la integridad del sistema y la gestión de flota.

```
94  ## Trigger 3: Evitar reservar vehículos que no están disponibles
95
96  DELIMITER $$
97
98  • CREATE TRIGGER before_insert_reserva
99  BEFORE INSERT ON reservas
100  FOR EACH ROW
101  BEGIN
102      DECLARE estado_actual ENUM('Disponible', 'Reservado', 'Mantenimiento', 'Baja');
103
104      SELECT estadoVehiculo INTO estado_actual
105      FROM vehiculos
106      WHERE idVehiculo = NEW.idVehiculo;
107
108      IF estado_actual != 'Disponible' THEN
109          SIGNAL SQLSTATE '45000'
110          SET MESSAGE_TEXT = 'Error: El vehículo no está disponible para reservas.';
111      END IF;
112  END $$
113
114  DELIMITER ;
115
116  • ## Prueba y validación del trigger
117  INSERT INTO reservas (fechaInicio, fechaFin, costoTotal, estadoReserva, idCliente, idEmpleado, idVehiculo)
118  VALUES ('2025-05-01 10:00:00', '2025-05-03 12:00:00', 150.00, 'Activa', 1, 2, 2);
119
120
121
122
123
124
125
```

Output

#	Time	Action	Message
66	16:53:23	INSERT INTO mantenimientos (idVehiculo, fechaMantenimiento, tipoMantenimiento, descripcion, costo) VALUES (1, '2025-04-25 10:00:00', 'Mantenimi...	1 row(s) affected, 1 warning(s): 1292 Incorrect date value: '2025-04-25'
67	17:01:11	SELECT * FROM aud_mantenimientos LIMIT 0, 1000	1 row(s) returned
68	17:34:37	SELECT * FROM aud_mantenimientos LIMIT 0, 1000	1 row(s) returned
69	17:34:42	CREATE TRIGGER before_insert_reserva BEFORE INSERT ON reservas FOR EACH ROW BEGIN DECLARE estado_actual ENUM('Disponible', '...	0 row(s) affected
70	17:42:37	SELECT * FROM vehiculos LIMIT 0, 1000	9 row(s) returned
71	17:51:28	INSERT INTO reservas (fechaInicio, fechaFin, costoTotal, estadoReserva, idCliente, idEmpleado, idVehiculo) VALUES ('2025-05-01 10:00:00', '2025-0...	Error Code: 1644. Error: El vehículo no está disponible para reservas.

Las vistas y los triggers han mejorado significativamente nuestra base de datos al permitirnos automatizar procesos clave, mantener la integridad de los datos y facilitar el acceso a información relevante de forma más clara y eficiente. Gracias a ellos, ahora contamos con una gestión más organizada, segura y útil para la toma de decisiones en el sistema de alquiler de vehículos.

12. CONCLUSIÓN

Este proyecto ha representado una excelente oportunidad para poner en práctica los conocimientos aprendidos en el ámbito del diseño y gestión de bases de datos. A lo largo del proceso, he desarrollado un sistema completo y funcional para una empresa de alquiler de vehículos, abarcando desde el modelado inicial hasta la implementación de consultas, vistas y triggers que aportan valor real a la gestión de la información.

El uso de MySQL ha permitido estructurar adecuadamente la base de datos, estableciendo relaciones claras entre clientes, empleados, vehículos y reservas. La incorporación de mejoras como la tabla de mantenimientos, nuevas columnas informativas y automatizaciones mediante triggers, ha enriquecido aún más el sistema, facilitando tareas comunes y mejorando el control de operaciones. En conjunto, este proyecto me ha permitido reforzar tanto mis capacidades técnicas como mi visión sobre cómo una base de datos bien diseñada puede optimizar los procesos de una empresa en la vida real.

DAVID DELGADO CORREA