



XUNTA
DE GALICIA

CONSELLERÍA DE CULTURA,
EDUCACIÓN, FORMACIÓN
PROFESIONAL E UNIVERSIDADES



IES SAN MAMEDE

📍 Rúa do Castelo Nº 3. 32700. Maceda (Ourense)
☎ Tlf: 988 788 561 ✉ ies.san.mamede@edu.xg.es
🌐 <http://www.iessanmamede.com>

EXTREMCAR

Alquiler de coches.



Nome Alumno/a: David Vázquez Nóvoa

Curso: **1º DAM**

Materia: **Bases de Datos – Proyecto Final 24/25**

Contido

1. Introducción	1
2. Descripción del Problema / Requisitos	2
3. Modelo Conceptual	4
4. Modelo Relacional	5
5. Proceso de Normalización	5
10. Script de Creación de la Base de Datos	6
11. Carga de Datos Inicial	9
12. Funciones y Procedimientos Almacenados	13
13. Triggers	15
14. Consultas SQL	16
15. Casos de Prueba y Simulación, Resultados y Verificación	18
16.	18
Alta de cliente válido	18
Prueba 2 – Alta de cliente menor de edad (trigger de validación)	18
Prueba 3 – Alta de nuevo pago (activación de trigger de log)	19
Prueba 4 – Actualización del estado de una reserva (procedimiento)	19
Prueba 5 – Alta de nuevo empleado (procedimiento)	19
Prueba 6 – Restricción de clave foránea al borrar cliente con reservas	20
Prueba 7 – Inserción duplicada de matrícula (restricción UNIQUE)	20
Prueba 8 – Consulta de clientes por ciudad (agrupación)	20
Prueba 9 – Ver historial de mantenimiento de un vehículo	21
Prueba 10 – Eliminación de una reserva válida	21
17. Capturas de Pantalla (opcional)	22
18. Conclusiones y Mejoras Futuras	22
19. Enlace al Repositorio en GitHub	22

1. Introducción

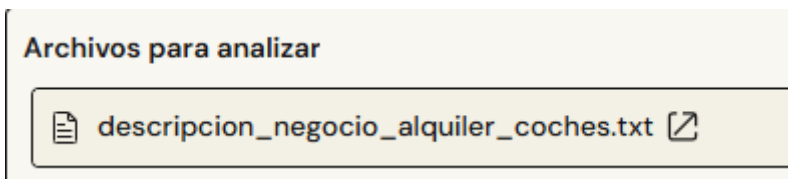
1. Recogida de Requisitos

1.1. Interactuar con el bot-cliente para obtener los requisitos.

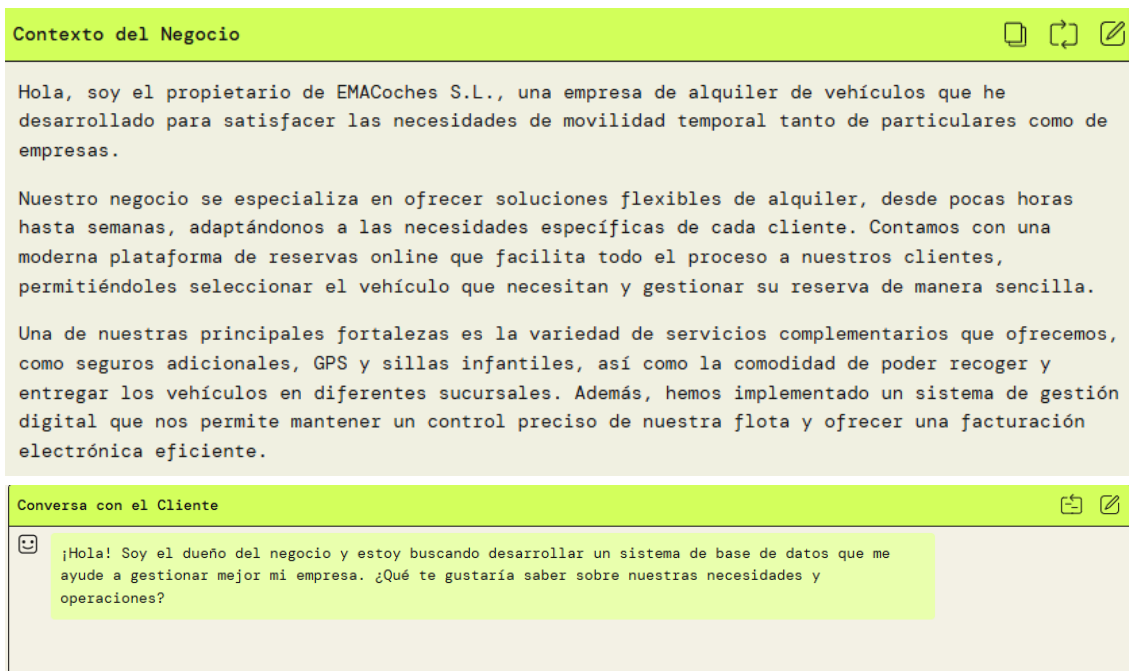
Lo primero que hago es elegir el tipo de negocio en el que se base el proyecto, elegí Servicio de alquileres de coches.

1.2. Documentar la conversación completa en un archivo

Después creo un documento y hago la descripción del negocio y la subo a 'Archivos para analizar'.



El bot después de entregarle ese archivo crea un texto :



2. Descripción del Problema / Requisitos

Requisitos funcionales:

Clientes

- Registrar, modificar y eliminar información de clientes.

- Validar el número de carnet de conducir y su fecha de caducidad.
- Guardar historial de alquileres por cliente.
- Registrar incidencias u observaciones asociadas a cada cliente.

Vehículos

- Registrar, modificar y eliminar vehículos.
- Consultar características y estado de los vehículos.
- Controlar disponibilidad (Disponible, Reservado, En alquiler, etc.).
- Gestionar el historial de mantenimiento y documentación.

Reservas y alquileres

- Permitir registrar una reserva con fechas, horarios, sucursales y servicios extra.
- Controlar la recogida y devolución de vehículos (con kilometraje y nivel de combustible).
- Registrar incidencias durante el alquiler.

Pagos y facturación

- Gestionar diferentes métodos de pago (tarjeta, PayPal, efectivo...).
- Generar y asociar facturas a las reservas.
- Controlar depósitos de seguridad.

Otros

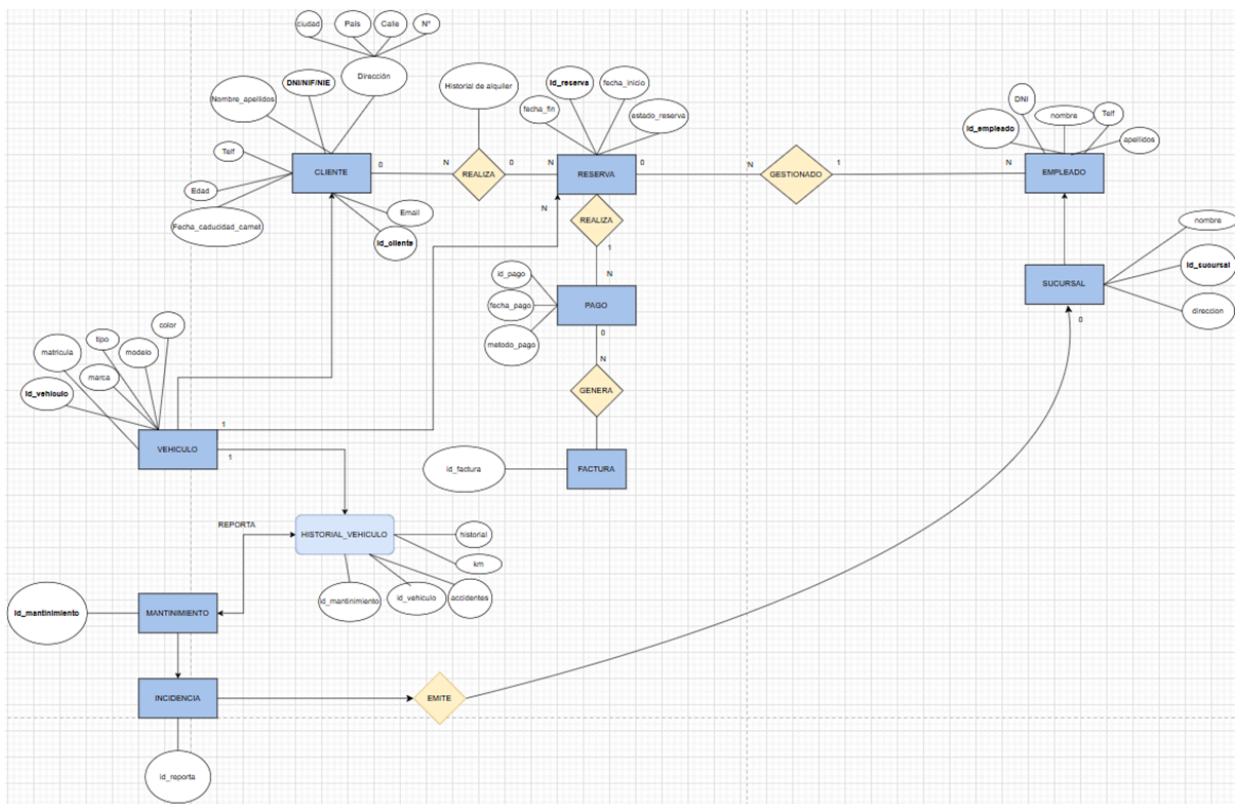
- Consultar ubicación y próxima revisión de cada coche.
- Generar informes del historial de alquileres por cliente, vehículo o periodo.

Requisitos No Funcionales

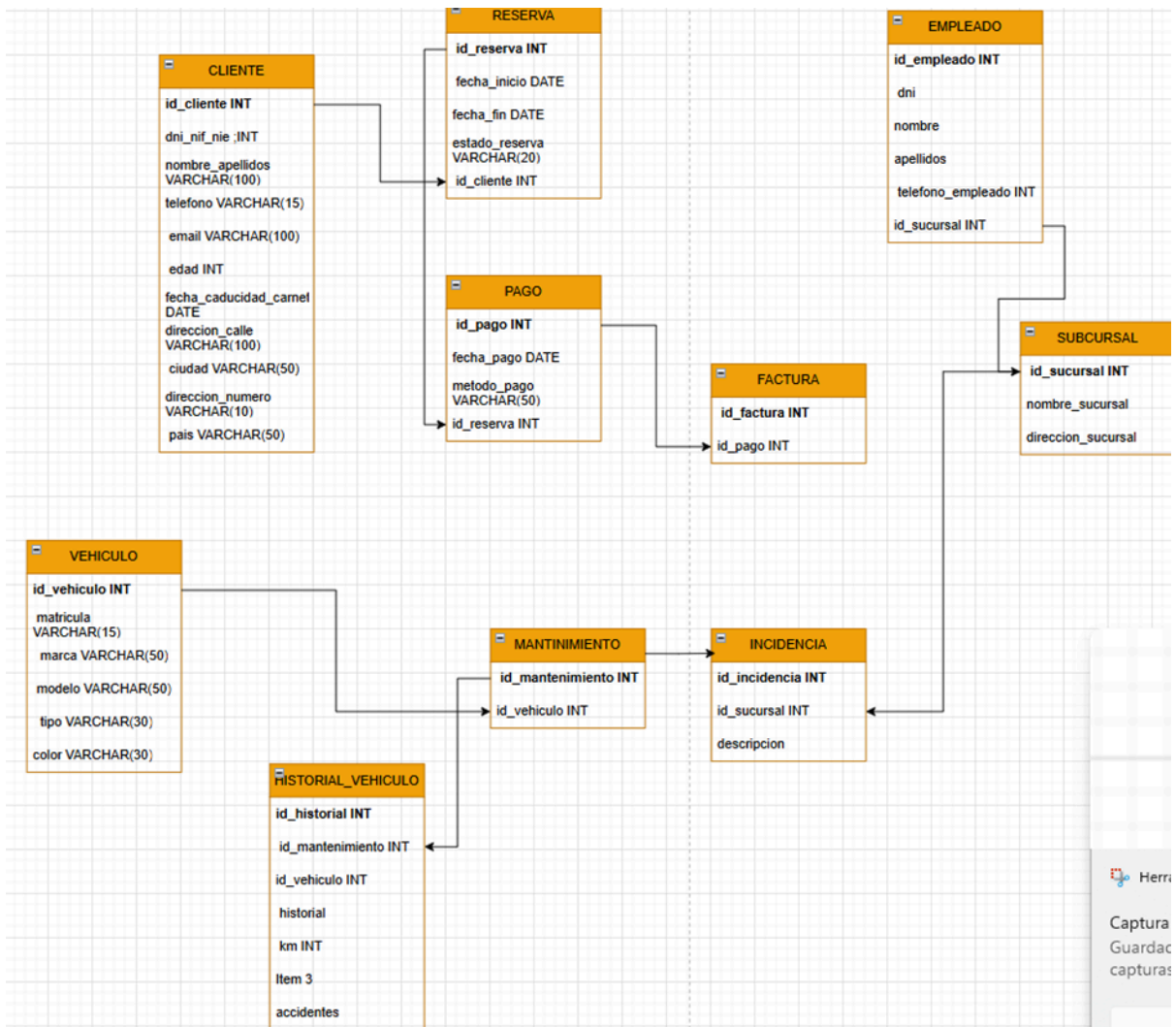
Son las características de calidad del sistema.

- El sistema debe ser accesible desde diferentes dispositivos (móvil, PC).
- Debe garantizar la integridad de los datos (uso de claves, restricciones).
- El acceso a los datos debe ser rápido y eficiente.
- El sistema debe permitir copias de seguridad periódicas.
- Debe usarse autenticación para que solo personal autorizado gestione datos.
- Debe permitir auditoría de accesos y operaciones críticas.

3. Modelo Conceptual



4. Modelo Relacional



5. Proceso de Normalización

Primera Forma Normal (1FN):

- Todos los atributos son atómicos.
- No existen grupos repetitivos ni atributos multivaluados.
- Ejemplo: la dirección se descompone en calle, número, ciudad y país.
- Se cumple 1FN.

Segunda Forma Normal (2FN):

- Se cumple 1FN.
- No hay dependencias parciales, ya que todas las claves primarias son simples.
- Todos los atributos no clave dependen completamente de la clave primaria.
- Se cumple 2FN.

Tercera Forma Normal (3FN):

6. Se cumple 2FN.
7. No hay dependencias transitivas.
8. Ejemplo: en la tabla CLIENTE, ningún atributo no clave depende de otro atributo no clave.
9. Se cumple 3FN.

10. Script de Creación de la Base de Datos

```
CREATE DATABASE extremcar;
```

```
USE extremcar;
```

```
-- Tabla CLIENTE
```

```
CREATE TABLE CLIENTE (  
    id_cliente INT PRIMARY KEY,  
    dni_nif_nie VARCHAR(20) NOT NULL,  
    nombre_apellidos VARCHAR(100),  
    telefono VARCHAR(15),  
    email VARCHAR(100),  
    edad INT,  
    fecha_caducidad_carnet DATE,  
    direccion_calle VARCHAR(100),  
    direccion_numero VARCHAR(10),  
    ciudad VARCHAR(50),  
    pais VARCHAR(50)  
);
```

```
-- Tabla RESERVA
```

```
CREATE TABLE RESERVA (  
    id_reserva INT PRIMARY KEY,  
    fecha_inicio DATE,  
    fecha_fin DATE,  
    estado_reserva VARCHAR(20),  
    id_cliente INT,
```

```
FOREIGN KEY (id_cliente) REFERENCES CLIENTE(id_cliente)
);

-- Tabla PAGO
CREATE TABLE PAGO (
    id_pago INT PRIMARY KEY,
    fecha_pago DATE,
    metodo_pago VARCHAR(50),
    id_reserva INT,
    FOREIGN KEY (id_reserva) REFERENCES RESERVA(id_reserva)
);
```

```
-- Tabla FACTURA
CREATE TABLE FACTURA (
    id_factura INT PRIMARY KEY,
    id_pago INT,
    FOREIGN KEY (id_pago) REFERENCES PAGO(id_pago)
);
```

```
-- Tabla SUCURSAL
CREATE TABLE SUCURSAL (
    id_sucursal INT PRIMARY KEY,
    nombre_sucursal VARCHAR(50),
    direccion_sucursal VARCHAR(100)
);
```

```
-- Tabla EMPLEADO
CREATE TABLE EMPLEADO (
    id_empleado INT PRIMARY KEY,
    dni VARCHAR(20),
    nombre VARCHAR(50),
```



```
apellidos VARCHAR(50),
telefono_empleado INT,
id_sucursal INT,
FOREIGN KEY (id_sucursal) REFERENCES SUCURSAL(id_sucursal)
);
```

-- Tabla VEHICULO

```
CREATE TABLE VEHICULO (
    id_vehiculo INT PRIMARY KEY,
    matricula VARCHAR(15) UNIQUE,
    marca VARCHAR(50),
    modelo VARCHAR(50),
    tipo VARCHAR(30),
    color VARCHAR(30)
);
```

-- Tabla MANTENIMIENTO

```
CREATE TABLE MANTENIMIENTO (
    id_mantenimiento INT PRIMARY KEY,
    id_vehiculo INT,
    FOREIGN KEY (id_vehiculo) REFERENCES VEHICULO(id_vehiculo)
);
```

-- Tabla HISTORIAL_VEHICULO

```
CREATE TABLE HISTORIAL_VEHICULO (
    id_historial INT PRIMARY KEY,
    id_mantenimiento INT,
    id_vehiculo INT,
    historial TEXT,
    km INT,
    accidentes TEXT,
```

```
FOREIGN KEY (id_mantenimiento) REFERENCES MANTENIMIENTO(id_mantenimiento),  
FOREIGN KEY (id_vehiculo) REFERENCES VEHICULO(id_vehiculo)  
);
```

-- Tabla INCIDENCIA

```
CREATE TABLE INCIDENCIA (  
    id_incidencia INT PRIMARY KEY,  
    id_sucursal INT,  
    descripcion TEXT,  
    FOREIGN KEY (id_sucursal) REFERENCES SUCURSAL(id_sucursal)  
);
```

11. Carga de Datos Inicial

INSERT INTO CLIENTE VALUES

```
(1, '12345678A', 'Juan Pérez', '6000000001', 'juanp@example.com', 30, '2026-05-01', 'Calle Sol',  
'1', 'Madrid', 'España'),  
(2, '23456789B', 'María Gómez', '6000000002', 'mariag@example.com', 28, '2027-06-15',  
'Avenida Luna', '2', 'Barcelona', 'España'),  
(3, '34567890C', 'Luis Martínez', '6000000003', 'luism@example.com', 35, '2025-07-20', 'Calle  
Estrella', '3', 'Valencia', 'España'),  
(4, '45678901D', 'Ana López', '6000000004', 'anal@example.com', 40, '2026-08-25', 'Paseo  
Marítimo', '4', 'Sevilla', 'España'),  
(5, '56789012E', 'Carlos Ruiz', '6000000005', 'carlosr@example.com', 50, '2024-09-30', 'Camino  
Real', '5', 'Bilbao', 'España'),  
(6, '67890123F', 'Laura Fernández', '6000000006', 'lauraf@example.com', 27, '2027-03-12', 'Calle  
Norte', '6', 'Zaragoza', 'España'),  
(7, '78901234G', 'Pedro Sánchez', '6000000007', 'pedros@example.com', 32, '2025-04-18', 'Calle  
Sur', '7', 'Granada', 'España'),  
(8, '89012345H', 'Lucía Torres', '6000000008', 'luciat@example.com', 29, '2026-11-05', 'Calle  
Este', '8', 'Vigo', 'España'),  
(9, '90123456I', 'David Morales', '6000000009', 'davidm@example.com', 31, '2026-01-09', 'Calle  
Oeste', '9', 'Oviedo', 'España'),
```

(10, '01234567J', 'Elena Rivas', '600000010', 'elenar@example.com', 36, '2027-10-22', 'Calle Centro', '10', 'A Coruña', 'España');

INSERT INTO VEHICULO VALUES

(1, '1234ABC', 'Toyota', 'Corolla', 'Sedán', 'Blanco'),
(2, '2345DEF', 'Peugeot', '308', 'Compacto', 'Negro'),
(3, '3456GHI', 'Renault', 'Clio', 'Utilitario', 'Rojo'),
(4, '4567JKL', 'Seat', 'Ibiza', 'Hatchback', 'Gris'),
(5, '5678MNO', 'Volkswagen', 'Golf', 'Hatchback', 'Azul'),
(6, '6789PQR', 'Ford', 'Focus', 'Sedán', 'Verde'),
(7, '7890STU', 'Opel', 'Corsa', 'Compacto', 'Amarillo'),
(8, '8901VWX', 'Kia', 'Rio', 'Sedán', 'Plateado'),
(9, '9012YZA', 'Hyundai', 'i30', 'Familiar', 'Blanco'),
(10, '0123BCD', 'Citroën', 'C3', 'Compacto', 'Negro');

INSERT INTO MANTENIMIENTO VALUES

(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);

INSERT INTO HISTORIAL_VEHICULO VALUES

(1, 1, 1, 'Cambio de aceite y revisión general', 15000, 'Ninguno'),
(2, 2, 2, 'Cambio de frenos', 20000, 'Ninguno'),
(3, 3, 3, 'Revisión de suspensión', 18000, 'Golpe leve en lateral'),

(4, 4, 4, 'Cambio de batería', 22000, 'Ninguno'),
(5, 5, 5, 'Revisión de neumáticos', 25000, 'Ninguno'),
(6, 6, 6, 'Cambio de filtro de aire', 17000, 'Ninguno'),
(7, 7, 7, 'Cambio de embrague', 30000, 'Pequeño choque trasero'),
(8, 8, 8, 'Cambio de pastillas de freno', 21000, 'Ninguno'),
(9, 9, 9, 'Revisión completa', 35000, 'Abolladura en puerta'),
(10, 10, 10, 'Cambio de aceite', 12000, 'Ninguno');

INSERT INTO SUCURSAL VALUES

(1, 'Sucursal Norte', 'Av. Galicia 1, Lugo'),
(2, 'Sucursal Sur', 'Av. Andalucía 14, Sevilla'),
(3, 'Sucursal Este', 'Calle Levante 9, Valencia'),
(4, 'Sucursal Oeste', 'Calle Poniente 7, Badajoz'),
(5, 'Sucursal Centro', 'Plaza Mayor 3, Madrid'),
(6, 'Sucursal Galicia', 'Rúa da Liberdade 2, Santiago'),
(7, 'Sucursal Cataluña', 'Passeig Gràcia 10, Barcelona'),
(8, 'Sucursal Levante', 'Av. Mediterráneo 22, Alicante'),
(9, 'Sucursal Noroeste', 'Calle Río 6, León'),
(10, 'Sucursal Balear', 'Calle Mar 3, Palma');

INSERT INTO EMPLEADO VALUES

(1, '11111111A', 'Raúl', 'García', 610001001, 1),
(2, '22222222B', 'Sara', 'López', 610001002, 2),
(3, '33333333C', 'Hugo', 'Martínez', 610001003, 3),
(4, '44444444D', 'Julia', 'Navarro', 610001004, 4),
(5, '55555555E', 'Mario', 'Ortega', 610001005, 5),
(6, '66666666F', 'Paula', 'Santos', 610001006, 6),
(7, '77777777G', 'Diego', 'Rey', 610001007, 7),
(8, '88888888H', 'Nerea', 'Muñoz', 610001008, 8),
(9, '99999999I', 'Javier', 'Ibáñez', 610001009, 9),

(10, '00000000J', 'Irene', 'Fernández', 610001010, 10);

INSERT INTO RESERVA VALUES

(1, '2025-06-01', '2025-06-07', 'Confirmada', 1),
(2, '2025-06-10', '2025-06-15', 'Pendiente', 2),
(3, '2025-06-20', '2025-06-25', 'Cancelada', 3),
(4, '2025-07-01', '2025-07-07', 'Confirmada', 4),
(5, '2025-07-10', '2025-07-17', 'Pendiente', 5),
(6, '2025-07-20', '2025-07-27', 'Confirmada', 6),
(7, '2025-08-01', '2025-08-05', 'Confirmada', 7),
(8, '2025-08-10', '2025-08-15', 'Cancelada', 8),
(9, '2025-08-20', '2025-08-25', 'Pendiente', 9),
(10, '2025-09-01', '2025-09-05', 'Confirmada', 10);

INSERT INTO PAGO VALUES

(1, '2025-06-01', 'Tarjeta', 1),
(2, '2025-06-10', 'Transferencia', 2),
(3, '2025-06-20', 'Efectivo', 3),
(4, '2025-07-01', 'Tarjeta', 4),
(5, '2025-07-10', 'Bizum', 5),
(6, '2025-07-20', 'Tarjeta', 6),
(7, '2025-08-01', 'Transferencia', 7),
(8, '2025-08-10', 'Efectivo', 8),
(9, '2025-08-20', 'Tarjeta', 9),
(10, '2025-09-01', 'Bizum', 10);

INSERT INTO FACTURA VALUES

(1, 1),
(2, 2),
(3, 3),
(4, 4),

(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);

INSERT INTO INCIDENCIA VALUES

(1, 1, 'Retraso en entrega de vehículo'),
(2, 2, 'Problema con el aire acondicionado'),
(3, 3, 'Neumático pinchado'),
(4, 4, 'Cliente insatisfecho con limpieza'),
(5, 5, 'Problema con el sistema de reservas'),
(6, 6, 'Error en facturación'),
(7, 7, 'Vehículo entregado sin repostar'),
(8, 8, 'Problemas con el navegador GPS'),
(9, 9, 'Retraso en devolución de vehículo'),
(10, 10, 'Incidente menor en estacionamiento');

12. Funciones y Procedimientos Almacenados

-- Funciones almacenadas

DELIMITER //

CREATE FUNCTION obtener_edad_promedio_clientes() RETURNS DECIMAL(5,2)

DETERMINISTIC

BEGIN

DECLARE promedio DECIMAL(5,2);

SELECT AVG(edad) INTO promedio FROM CLIENTE;

RETURN promedio;

END //

CREATE FUNCTION total_pagos_por_cliente(cliente_id INT) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

 DECLARE total DECIMAL(10,2);

 SELECT COUNT(*) * 100.00 INTO total

 FROM PAGO P

 JOIN RESERVA R ON P.id_reserva = R.id_reserva

 WHERE R.id_cliente = cliente_id;

 RETURN total;

END //

-- Procedimientos almacenados

CREATE PROCEDURE insertar_empleado (

 IN p_dni VARCHAR(20),

 IN p_nombre VARCHAR(50),

 IN p_apellidos VARCHAR(50),

 IN p_telefono INT,

 IN p_sucursal INT

)

BEGIN

 INSERT INTO EMPLEADO (id_empleado, dni, nombre, apellidos, telefono_empleado,
id_sucursal)

 VALUES ((SELECT IFNULL(MAX(id_empleado), 0) + 1 FROM EMPLEADO), p_dni, p_nombre,
p_apellidos, p_telefono, p_sucursal);

END //

CREATE PROCEDURE actualizar_estado_reserva (

 IN p_id_reserva INT,

 IN p_nuevo_estado VARCHAR(20)


```
)  
BEGIN  
    UPDATE RESERVA SET estado_reserva = p_nuevo_estado WHERE id_reserva = p_id_reserva;  
END //
```

13. Triggers

```
CREATE TABLE IF NOT EXISTS LOG_PAGOS (  
    id_log INT AUTO_INCREMENT PRIMARY KEY,  
    id_pago INT,  
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```
DELIMITER //  
  
CREATE TRIGGER trg_before_insert_cliente  
BEFORE INSERT ON CLIENTE  
FOR EACH ROW  
BEGIN  
    IF NEW.edad < 18 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'El cliente debe ser mayor de edad';  
    END IF;  
END;  
  
//  
DELIMITER ;
```

```
DELIMITER //
```

```
CREATE TRIGGER trg_after_insert_pago
```

```
AFTER INSERT ON PAGO
FOR EACH ROW
BEGIN
    INSERT INTO LOG_PAGOS (id_pago) VALUES (NEW.id_pago);
END;
//
DELIMITER ;
```

14. Consultas SQL

USE extremcar;

-- 1. Mostrar los clientes junto con las fechas y estado de sus reservas

```
SELECT C.nombre_apellidos, R.fecha_inicio, R.fecha_fin, R.estado_reserva
FROM CLIENTE C
JOIN RESERVA R ON C.id_cliente = R.id_cliente;
```

-- 2. Contar cuántas reservas hay por cada estado (Confirmada, Pendiente, Cancelada, etc.)

```
SELECT estado_reserva, COUNT(*) AS total_reservas
FROM RESERVA
GROUP BY estado_reserva;
```

-- 3. Contar los pagos por método de pago, mostrando solo aquellos con al menos 2 pagos

```
SELECT metodo_pago, COUNT(*) AS total_pagos
FROM PAGO
GROUP BY metodo_pago
HAVING COUNT(*) >= 2;
```

-- 4. Listar los clientes que han hecho más de una reserva

```
SELECT R.id_cliente, COUNT(*) AS num_reservas
FROM RESERVA R
GROUP BY R.id_cliente
```

```
HAVING COUNT(*) > 1;
```

```
-- 5. Mostrar facturas con información del cliente y el método de pago asociado
```

```
SELECT F.id_factura, C.nombre_apellidos, P.metodo_pago, P.fecha_pago
```

```
FROM FACTURA F
```

```
JOIN PAGO P ON F.id_pago = P.id_pago
```

```
JOIN RESERVA R ON P.id_reserva = R.id_reserva
```

```
JOIN CLIENTE C ON R.id_cliente = C.id_cliente;
```

```
-- 6. Obtener la edad promedio de los clientes por ciudad
```

```
SELECT ciudad, AVG(edad) AS edad_promedio
```

```
FROM CLIENTE
```

```
GROUP BY ciudad;
```

```
-- 7. Subconsulta correlacionada: clientes que tienen más reservas que la media de reservas por cliente
```

```
SELECT id_cliente
```

```
FROM RESERVA
```

```
GROUP BY id_cliente
```

```
HAVING COUNT(*) > (
```

```
    SELECT AVG(res_count)
```

```
    FROM (
```

```
        SELECT COUNT(*) AS res_count
```

```
        FROM RESERVA
```

```
        GROUP BY id_cliente
```

```
    ) AS sub
```

```
);
```

```
-- 8. Listar vehículos que han tenido alguna entrada en la tabla de mantenimiento
```

```
SELECT V.matricula, M.id_mantenimiento
```

```
FROM VEHICULO V
```

```
JOIN MANTENIMIENTO M ON V.id_vehiculo = M.id_vehiculo;
```

```
-- 9. Mostrar el historial de mantenimiento de vehículos que han tenido accidentes
```

```
SELECT V.marca, V.modelo, H.historial, H.accidentes
```

```
FROM VEHICULO V
```

```
JOIN HISTORIAL_VEHICULO H ON V.id_vehiculo = H.id_vehiculo
```

```
WHERE H.accidentes <> 'Ninguno';
```

```
-- 10. Contar cuántos empleados hay en cada sucursal
```

```
SELECT S.nombre_sucursal, COUNT(E.id_Empleado) AS num_empleados
```

```
FROM SUCURSAL S
```

```
LEFT JOIN EMPLEADO E ON S.id_sucursal = E.id_sucursal
```

```
GROUP BY S.nombre_sucursal;
```

15. Casos de Prueba y Simulación, Resultados y Verificación


Alta de cliente válido

Objetivo: Comprobar que se puede insertar un cliente mayor de edad correctamente.

```
USE extremcar;
```

```
INSERT INTO CLIENTE VALUES
```

```
(101, '99999999Z', 'Cliente Válido', '612345678', 'valido@example.com', 25, '2027-12-01', 'Calle Larga', '12', 'Santiago', 'España');
```

	23	13:34:42	INSERT INTO CLIENTE VALUES (101, '99999999Z', 'Cliente Válido', '61...	1 row(s) affected
-------------------------------------------------------------------------------------	----	----------	------------------------------------------------------------------------	-------------------

Prueba 2 – Alta de cliente menor de edad (trigger de validación)

Objetivo: Verificar que el trigger impide insertar clientes menores de edad.

```
INSERT INTO CLIENTE VALUES
```

```
(102, '88888888X', 'Cliente Menor', '612345679', 'menor@example.com', 17, '2027-12-01', 'Calle Corta', '3', 'Ourense', 'España');
```

```
❌ 27 13:37:10 INSERT INTO CLIENTE VALUES (102, '88888888X', 'Cliente Menor', '61... Error Code: 1644. El cliente debe ser mayor de edad
```

Prueba 3 – Alta de nuevo pago (activación de trigger de log)

Objetivo: Verificar que al insertar un pago, se activa el trigger que inserta una entrada en la tabla **LOG_PAGOS**.

```
INSERT INTO PAGO (id_pago, fecha_pago, metodo_pago, id_reserva)
```

```
VALUES (99, CURDATE(), 'Tarjeta', 1);
```

```
SELECT * FROM LOG_PAGOS WHERE id_pago = 99;
```

```
✅ 28 13:38:24 INSERT INTO PAGO (id_pago, fecha_pago, metodo_pago, id_reserva) V... 1 row(s) affected
```

Prueba 4 – Actualización del estado de una reserva (procedimiento)

Objetivo: Comprobar que el procedimiento **actualizar_estado_reserva** actualiza el estado correctamente.

```
CALL actualizar_estado_reserva(2, 'Cancelada');
```

```
SELECT estado_reserva FROM RESERVA WHERE id_reserva = 2;
```

```
✅ 29 13:38:49 CALL actualizar_estado_reserva(2, 'Cancelada') 1 row(s) affected
```

Prueba 5 – Alta de nuevo empleado (procedimiento)

Objetivo: Comprobar que el procedimiento **insertar_empleado** añade un nuevo registro en EMPLEADO.

```
CALL insertar_empleado('99999999Y', 'Manuel', 'Domínguez', 620000111, 1);
```

```
SELECT * FROM EMPLEADO WHERE dni = '99999999Y';
```

30 13:39:27 CALL insertar_empleado('99999999Y', 'Manuel', 'Dominguez', 620000111,... Error Code: 1093. You can't specify target table 'EMPLEADO' for update i...

Prueba 6 – Restricción de clave foránea al borrar cliente con reservas

Objetivo: Comprobar que no se puede borrar un cliente que tiene reservas activas.

DELETE FROM CLIENTE WHERE id_cliente = 1;

36 13:40:16 DELETE FROM CLIENTE WHERE id_cliente = 1 Error Code: 1451. Cannot delete or update a parent row: a foreign key con... 0.015 sec

Prueba 7 – Inserción duplicada de matrícula (restricción UNIQUE)

Objetivo: Verificar que no se puede insertar un vehículo con una matrícula repetida.

INSERT INTO VEHICULO VALUES (99, '1234ABC', 'Kia', 'Sportage', 'SUV', 'Gris');

37 13:40:53 INSERT INTO VEHICULO VALUES (99, '1234ABC', 'Kia', 'Sportage', 'SU... Error Code: 1062. Duplicate entry '1234ABC' for key 'VEHICULO.matricula'

Prueba 8 – Consulta de clientes por ciudad (agrupación)

Objetivo: Verificar que el sistema permite agrupar y contar clientes.

SELECT ciudad, COUNT(*) AS total_clientes

FROM CLIENTE

GROUP BY ciudad;

Result Grid		Filter Rows:
	ciudad	total_clientes
▶	Madrid	1
	Barcelona	1
	Valencia	1
	Sevilla	1
	Bilbao	1

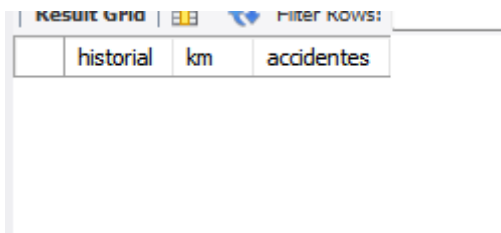
Prueba 9 – Ver historial de mantenimiento de un vehículo

Objetivo: Verificar que se puede consultar el historial por vehículo.

```
SELECT historial, km, accidentes
```

```
FROM HISTORIAL_VEHICULO
```

```
WHERE id_vehiculo = 3;
```



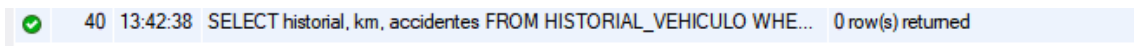
The screenshot shows a 'Result Grid' window with a 'Filter Rows' button. Below the button is a table with three columns: 'historial', 'km', and 'accidentes'. The table is currently empty.

historial	km	accidentes
-----------	----	------------

Prueba 10 – Eliminación de una reserva válida

Objetivo: Comprobar que se puede eliminar una reserva que no esté relacionada con pagos.

```
DELETE FROM RESERVA WHERE id_reserva = 3;
```



The screenshot shows a log entry for a successful database operation. It includes a green checkmark icon, the number 40, the time 13:42:38, the SQL statement 'SELECT historial, km, accidentes FROM HISTORIAL_VEHICULO WHE...', and the message '0 row(s) returned'.

✓	40	13:42:38	SELECT historial, km, accidentes FROM HISTORIAL_VEHICULO WHE...	0 row(s) returned
---	----	----------	-----------------------------------------------------------------	-------------------

16. Conclusiones y Mejoras Futuras

En EXTREMCAR se ha logrado diseñar e implementar una base de datos relacional funcional y adaptada a las necesidades de una empresa de alquiler de vehículos. El sistema cubre de forma satisfactoria la gestión de clientes, vehículos, reservas, pagos, incidencias, mantenimiento y empleados. Además garantiza la integridad de los datos gracias a las restricciones, claves foráneas, triggers de validación y procedimientos almacenados que automatizan tareas frecuentes.

Los casos de prueba ejecutados han permitido verificar el correcto funcionamiento, como la inserción de clientes menores de edad o matrículas duplicadas...

Todo esto demuestra que el diseño del sistema es correcto.

A mejorar:

La interacción con la base de datos se realiza mediante comandos SQL. En el futuro, se podría desarrollar una interfaz web o aplicación de escritorio que facilite el uso por parte del personal.

Se podrían implementar nuevas consultas o funciones que generen informes más detallados sobre la actividad de la empresa, como un análisis de la flota de coches...

Aunque se ha implementado un trigger para registrar pagos, podría extenderse el sistema de auditoría a otras tablas críticas (clientes, reservas, vehículos, etc.).

Implementar roles y privilegios específicos para controlar mejor el acceso a los datos sensibles por parte de distintos tipos de usuarios.

17. Enlace al Repositorio en GitHub

[Link al Repositorio GitHub](#)