

DRIVEAR

UADE

Trabajo Práctico Integrador

TURNO: NOCHE

GRUPO 02: *Drivear*

Integrantes del Grupo:

- L 1186773 – Luca Lioni
- L 1204795 – Lucas García
- L 1177995 - Ian Agustin Lionetti
- L 1126051 - Agustin Gazza
- L 1211804 - Jonathan Max Saravia Moreira

Profesor:

Ing. Franco Emanuel Salazar

Fecha de Entrega: 10/11/2025

 Ingeniería de Datos I	Grupo	Entrega	Fecha
	02	01	10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

Índice

1. Resumen Ejecutivo	3
2. Introducción	3
3. Objetivos del Sistema	3
4. Requisitos del Sistema	4
4.1. Requisitos Funcionales	4
4.2. Requisitos No Funcionales	14
5. Diseño de la Base de Datos	17
5.1. Modelo Conceptual	17
5.2. Modelo Lógico	17
5.3. Modelo Físico	18
5.4. Diccionario de Datos	19
5.5. Dependencias Funcionales	19
6. Implementación y Scripts SQL	21
6.1. Inserción de Datos	21
6.2. Consultas SQL	21
6.3. Triggers y Vistas	21
6.4. Funciones y Procedimientos	21
7. Interfaz de Usuario (Opcional)	21
8. Plan de Desarrollo y Metodología	22
9. Optimización y Rendimiento (Opcional)	22
10. Seguridad y Recuperación (Opcional)	23
11. Pruebas y Validación (Opcional)	23
12. Gestión de Riesgos (Opcional)	24
13. Costos y Presupuesto (Opcional)	24
14. Conclusiones y Recomendaciones	25
15. Referencias y Anexos	25

Grupo	Entrega	Fecha
02	01	10/11/2025
Trabajo Práctico		
Integrador Obligatorio		

1. Resumen Ejecutivo

Este proyecto tiene como objetivo crear una base de datos que garantice trazabilidad, seguridad, fiabilidad y escalabilidad de datos resultantes de las operaciones de Drivear; una plataforma que conecta choferes profesionales con propietarios de autos.

2. Introducción

Desde el equipo ejecutivo de *Drivear* hemos detectado la necesidad de personas y empresas propietarias de vehículos automotores que, en ocasiones, requieren de un chofer profesional.

Con el ajetreado horario de la actualidad, las personas no cuentan con suficiente tiempo para llevar su vehículo al taller mecánico, realizar controles o trámites cotidianos como la Verificación Técnica Vehicular Obligatoria (VTVO), rendir el exámen Práctico de Licencia de Conducir utilizando el vehículo propio. En otras ocasiones, desean ingerir bebidas alcohólicas pero se dan cuenta que no pueden regresar a sus hogares sin infringir las leyes locales.

Las empresas, por otro lado, a veces cuentan con flotas propias para sus ejecutivos de alto rango, pero los mismos no pueden recibir el beneficio por no contar con licencia de conducir vigente.

Drivear tiene por objetivo revolucionar el transporte ofreciendo un servicio de choferes profesionales selectos por una empresa que los respalda y asegura el cuidado de los vehículos y pasajeros.

El propósito de *Drivear* se basa en ayudar al modelo de negocio a crecer exponencialmente, de esta forma se planteó una solución base de datos relacional para poder guardar los datos respetando el modelo de negocio y permitiendo un fácil entendimiento de su estructura para poder usarlo con distintos propósitos.

Para esto se utilizará el potente y confiable motor de Microsoft SQL Server, alimentado por los datos de las operaciones diarias de la Aplicación *Drivear*.

3. Objetivos del Sistema

Los objetivos de este proyecto son:

- Diseñar un modelo de datos relacional.
- Diseñar un diagrama de entidad relación.
- Desarrollar una base de datos en el ecosistema Microsoft SQL Server.
- Implementar funcionalidades de CRUD para usuarios, vehículos, choferes, viajes y credenciales.
- Garantizar la seguridad, consistencia y accesibilidad de la información.

Grupo	Entrega	Fecha
02	01	10/11/2025
Trabajo Práctico		
Integrador Obligatorio		

4. Requisitos del Sistema

4.1. Requisitos Funcionales

RF1 – Alta de usuarios (pasajero/chofer)

El sistema debe permitir registrar usuarios con sus datos personales y el tipo de usuario .

Regla de negocio:

- tipo_usuario = chofer, pasajero,
- estado = activo, inactivo, suspendido,
- email y CUIT/CUIL = únicos.

```
CREATE TABLE usuarios (
    usuario_id INT IDENTITY(1,1) PRIMARY KEY,
    cuit_cuil BIGINT NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL,
    email VARCHAR(150) NOT NULL,
    telefono BIGINT,
    calle VARCHAR(150),
    altura INT,
    codigo_postal INT,
    tipo_usuario VARCHAR(50) CHECK (tipo_usuario IN ('chofer', 'pasajero')),
    estado VARCHAR(50) CHECK (estado IN ('activo', 'inactivo', 'suspendido')),
    contrasena VARCHAR(255) NOT NULL,
    creado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    actualizado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    CONSTRAINT uq_usuario_email UNIQUE (email),
    CONSTRAINT uq_usuario_cuit UNIQUE (cuit_cuil)
);
```

Evidencia: inserciones en usuarios

 Ingeniería de Datos I	Grupo	Entrega	Fecha
	02	01	10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

```
[INSERT INTO usuarios (cuit_cuil,nombre,apellido,email,telefono,calle,altura,codigo_postal,
tipo_usuario,estado,contrasena)
VALUES
(20345678902,'Ramon','Lucci','ramon@drivear.com',2222333344,'Av. Caseros',111,1100,'pasajero','activo','pass1'),
(20988765433,'Laura','Perez','laura@drivear.com',1133344455,'Belgrano',321,1000,'pasajero','activo','pass2'),
(20945678901,'Sofia','Gimenez','sofia@drivear.com',1177788899,'Corrientes',654,1700,'pasajero','activo','pass3'),
(20555222333,'Clara','Diaz','clara@drivear.com',1144466677,'Laprida',159,1640,'pasajero','activo','pass4'),
(2099911122,'Paula','Herrera','paula@drivear.com',1133300011,'Italia',888,1400,'pasajero','activo','pass5'),
(27333222119,'Carlos','Gomez','carlos@drivear.com',1167788990,'San Martín',456,1638,'chofer','activo','pass6'),
(27455678909,'Mariana','Lopez','mariana@drivear.com',1142233445,'Lavalle',789,1650,'chofer','activo','pass7'),
(27999887766,'Sergio','Martinez','sergio@drivear.com',1161122233,'Rivadavia',777,1800,'chofer','activo','pass8'),
(27888999112,'Andres','Silva','andres@drivear.com',1155566677,'Mitre',222,1900,'chofer','activo','pass9'),
(27876543211,'Leonardo','Mendez','leonardo@drivear.com',1177711122,'Rosales',970,1670,'chofer','activo','pass10'),
(27333222889,'Esteban','Fernandez','esteban@drivear.com',1167788990,'Av Gaona',456,1638,'chofer','activo','pass11'),
(27455678999,'Silvio','Leguizamón','silvio@drivear.com',1142233445,'Av. los Incas',789,1650,'chofer','activo','pass12'),
(27999880066,'Saul','Linares','saul@drivear.com',1161122233,'Rosario',777,1800,'chofer','activo','pass13'),
(27884499112,'Benito','Cáceres','benito@drivear.com',1155566677,'Emilio Mitre',222,1900,'chofer','activo','pass14'),
(27876588211,'Agustina','García','agustina@drivear.com',1177711122,'Cachimayo',970,1670,'chofer','activo','pass15');
```

RF2 – Gestión de licencias de chofer

El sistema debe registrar licencias vinculadas a un chofer, validando que la fecha de vencimiento sea posterior a la de emisión.

Regla de negocio:

- fecha_vencimiento > fecha_emision.

```
[CREATE TABLE licencias (
    licencia_id INT IDENTITY(1,1) PRIMARY KEY,
    numero_licencia INT NOT NULL,
    categoria VARCHAR(50) NOT NULL,
    fecha_emision DATE NOT NULL,
    fecha_vencimiento DATE NOT NULL,
    usuario_id INT NOT NULL,
    creado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    actualizado_fecha DATETIMEOFFSET DEFAULT GETDATE(),

    CONSTRAINT uq_licencia_numero UNIQUE (numero_licencia),
    CONSTRAINT fk_licencia_usuario FOREIGN KEY (usuario_id) REFERENCES usuarios(usuario_id),
    CONSTRAINT chk_licencia_fechas CHECK (fecha_vencimiento > fecha_emision)
);]
```

Evidencia: inserción en licencias

```
[INSERT INTO licencias (numero_licencia,categoria,fecha_emision,fecha_vencimiento,usuario_id)
VALUES
(700001,'B1','2024-01-10','2028-01-10',6),
(700002,'C1','2023-05-12','2027-05-12',7),
(700003,'B1','2024-03-20','2028-03-20',8),
(700004,'C1','2023-09-01','2027-09-01',9),
(700005,'D1','2022-07-15','2026-07-15',10),
(700006,'B1','2024-01-10','2028-01-10',11),
(700007,'C1','2023-05-12','2027-05-12',12),
(700008,'B1','2024-03-20','2028-03-20',13),
(700009,'C1','2023-09-01','2027-09-01',14),
(700000,'D1','2022-07-15','2026-07-15',15);]
```

Grupo	Entrega	Fecha
02	01	10/11/2025
Trabajo Práctico		
Integrador Obligatorio		

RF3 – Gestión de vehículos del chofer

El sistema debe registrar vehículos asociados a un chofer, garantizando patente única.

Regla de negocio:

- patente única;
- FK a usuarios.

```
CREATE TABLE vehiculos (
    vehiculo_id INT IDENTITY(1,1) PRIMARY KEY,
    marca VARCHAR(100) NOT NULL,
    modelo VARCHAR(100) NOT NULL,
    color VARCHAR(50),
    patente VARCHAR(20) NOT NULL,
    anio INT NOT NULL,
    usuario_id INT NOT NULL,
    creado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    actualizado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    CONSTRAINT uq_vehiculo_patente UNIQUE (patente),
    CONSTRAINT fk_vehiculo_usuario FOREIGN KEY (usuario_id) REFERENCES usuarios(usuario_id)
);
```

Evidencia: inserción en vehículos

```
INSERT INTO vehiculos (marca,modelo,color,patente,anio,usuario_id)
VALUES
('Toyota', 'Corolla', 'Blanco', 'AA001AA', 2022, 1),
('Ford', 'Focus', 'Negro', 'AB002AB', 2021, 2),
('VW', 'Gol', 'Gris', 'AC003AC', 2020, 3),
('Chevrolet', 'Onix', 'Rojo', 'AD004AD', 2023, 4),
('Peugeot', '208', 'Azul', 'AE005AE', 2022, 5),
('Nissan', 'Versa', 'Negro', 'AF006AF', 2021, 1),
('Renault', 'Logan', 'Blanco', 'AG007AG', 2023, 2),
('Fiat', 'Cronos', 'Rojo', 'AH008AH', 2020, 3),
('Honda', 'Civic', 'Gris', 'AI009AI', 2022, 4),
('Chevrolet', 'Prisma', 'Azul', 'AJ010AJ', 2021, 5);
```

RF4 – Gestión de seguros del vehículo

El sistema debe registrar seguros vigentes asociados a un vehículo, controlando fechas de vencimiento.

Regla de negocio:

- fecha_vencimiento >= GETDATE().

 Ingeniería de Datos I	Grupo	Entrega	Fecha
	02	01	10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

```

]CREATE TABLE seguros (
    seguro_id INT IDENTITY(1,1) PRIMARY KEY,
    compania VARCHAR(100) NOT NULL,
    tipo_seguro VARCHAR(100) NOT NULL,
    numero_poliza INT NOT NULL,
    fecha_vencimiento DATETIMEOFFSET NOT NULL,
    cobertura_detalle VARCHAR(255),
    vehiculo_id INT NOT NULL,
    creado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    actualizado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    CONSTRAINT uq_seguro_poliza UNIQUE (numero_poliza),
    CONSTRAINT fk_seguro_vehiculo FOREIGN KEY (vehiculo_id) REFERENCES vehiculos(vehiculo_id),
    CONSTRAINT chk_seguro_vencimiento CHECK (fecha_vencimiento >= GETDATE())
);

```

Evidencia: inserción en seguros

```

]INSERT INTO seguros (compania,tipo_seguro,numero_poliza,fecha_vencimiento,cobertura_detalle,vehiculo_id)
VALUES
('Sancor Seguros', 'Total', 900001, DATEADD(MONTH, 12, GETDATE()), 'Cobertura completa', 1),
('Allianz', 'Terceros', 900002, DATEADD(MONTH, 18, GETDATE()), 'RC', 2),
('La Caja', 'Total', 900003, DATEADD(MONTH, 24, GETDATE()), 'Completa', 3),
('Provincia Seguros', 'Terceros', 900004, DATEADD(MONTH, 10, GETDATE()), 'RC', 4),
('San Cristobal', 'Total', 900005, DATEADD(MONTH, 30, GETDATE()), 'Total+Robo', 5),
('Federacion Patronal', 'Terceros', 900006, DATEADD(MONTH, 8, GETDATE()), 'Básica', 6),
('La Caja', 'Total', 900007, DATEADD(MONTH, 20, GETDATE()), 'Completa', 7),
('Sura', 'Terceros', 900008, DATEADD(MONTH, 14, GETDATE()), 'RC+Robo', 8),
('Mercantil Andina', 'Total', 900009, DATEADD(MONTH, 26, GETDATE()), 'Completa', 9),
('Mapfre', 'Terceros', 900010, DATEADD(MONTH, 16, GETDATE()), 'Básica+Incendio', 10);

```

RF5 – Gestión de medios de pago del pasajero

El sistema debe registrar tarjetas de pago de pasajeros y permitir su depuración (eliminar vencidas).

Reglas de negocio:

- tipo_tarjeta = debito, credito,
- cvv en rango válido, vencimiento >= GETDATE(),
- numero_tarjeta único.

```

CREATE TABLE tarjetas (
    tarjeta_id INT IDENTITY(1,1) PRIMARY KEY,
    numero_tarjeta BIGINT NOT NULL,
    nombre_titular VARCHAR(150) NOT NULL,
    tipo_tarjeta VARCHAR(50) CHECK (tipo_tarjeta IN ('debito', 'credito')),
    entidad_bancaria VARCHAR(100),
    red_pago VARCHAR(50),
    cvv INT CHECK (cvv BETWEEN 100 AND 9999),
    vencimiento DATETIMEOFFSET NOT NULL,
    usuario_id INT NOT NULL,

    creado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    actualizado_fecha DATETIMEOFFSET DEFAULT GETDATE(),

    CONSTRAINT uq_tarjeta_numero UNIQUE (numero_tarjeta),
    CONSTRAINT fk_tarjeta_usuario FOREIGN KEY (usuario_id) REFERENCES usuarios(usuario_id),
    CONSTRAINT chk_tarjeta_vencimiento CHECK (vencimiento >= GETDATE())
)

```

Evidencia:

Inserción en tarjetas

```

INSERT INTO tarjetas
(numero_tarjeta,nombre_titular,tipo_tarjeta,entidad_bancaria,red_pago,cvv,vencimiento,usuario_id)
VALUES
(4111111111111001,'Ramon Lucci','credito','Banco Nación','Visa',123,DATEADD(YEAR,2,GETDATE()),1),
(5555555555552002,'Laura Perez','debito','Santander','Mastercard',321,DATEADD(YEAR,1,GETDATE()),2),
(4111111111113003,'Sofia Gimenez','credito','Galicia','Visa',654,DATEADD(YEAR,2,GETDATE()),3),
(5555555555554004,'Clara Diaz','debito','BBVA','Mastercard',777,DATEADD(YEAR,3,GETDATE()),4),
(4111111111115005,'Paula Herrera','credito','ICBC','Visa',555,DATEADD(MONTH,18,GETDATE()),5),
(5555555555556006,'Ramon Lucci','debito','Santander','Maestro',999,DATEADD(YEAR,2,GETDATE()),6),
(4111111111117007,'Laura Perez','credito','Galicia','Visa',321,DATEADD(YEAR,3,GETDATE()),7),
(5555555555558008,'Sofia Gimenez','debito','ICBC','Mastercard',888,DATEADD(YEAR,2,GETDATE()),8),
(4111111111119009,'Clara Diaz','credito','Banco Nación','Visa',999,DATEADD(YEAR,3,GETDATE()),9),
(5555555555550010,'Paula Herrera','debito','BBVA','Maestro',123,DATEADD(YEAR,2,GETDATE()),10);

```

Borrado de vencidas

```

/* ----- 19) Eliminar tarjetas vencidas ----- */
DELETE FROM tarjetas
WHERE vencimiento < GETDATE();

```

RF6 – Alta de viaje

El sistema debe crear viajes con origen/destino, horarios, costo, distancia y estado inicial “pendiente”.

Regla de negocio:

- estado = pendiente, en curso, finalizado, cancelado,
- distancia_km ≥ 0,
- costo ≥ 0.

 Ingeniería de Datos I	Grupo	Entrega	Fecha
	02	01	10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

```

CREATE TABLE viajes (
    viaje_id INT IDENTITY(1,1) PRIMARY KEY,
    fecha_inicial DATETIMEOFFSET NOT NULL,
    fecha_final DATETIMEOFFSET,
    origen VARCHAR(150) NOT NULL,
    destino VARCHAR(150) NOT NULL,
    distancia_km FLOAT CHECK (distancia_km >= 0),
    costo FLOAT CHECK (costo >= 0),
    estado VARCHAR(50) CHECK (estado IN ('pendiente', 'en curso', 'finalizado', 'cancelado')),
    vehiculo_id INT NOT NULL,

    creado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    actualizado_fecha DATETIMEOFFSET DEFAULT GETDATE(),

    CONSTRAINT fk_viaje_vehiculo FOREIGN KEY (vehiculo_id) REFERENCES vehiculos(vehiculo_id)
);

```

Evidencia: inserción en viajes.

```

INSERT INTO viajes
(fecha_inicial, fecha_final, origen, destino, distancia_km, costo, estado, vehiculo_id)
VALUES
(SYSDATETIMEOFFSET(), DATEADD(MINUTE, 45, SYSDATETIMEOFFSET()), 'Buenos Aires', 'La Plata', 60.5, 5000, 'finalizado', 1),
(DATEADD(DAY, -1, SYSDATETIMEOFFSET()), DATEADD(MINUTE, 49, SYSDATETIMEOFFSET()), 'Quilmes', 'Avellaneda', 20.0, 1800, 'finalizado', 2),
(DATEADD(DAY, -2, SYSDATETIMEOFFSET()), DATEADD(MINUTE, 35, SYSDATETIMEOFFSET()), 'Lanús', 'Banfield', 15.2, 1500, 'finalizado', 3),
(DATEADD(DAY, -3, SYSDATETIMEOFFSET()), NULL, 'Morón', 'Haedo', 8.3, 950, 'pendiente', 4),
(DATEADD(DAY, -4, SYSDATETIMEOFFSET()), NULL, 'Lomas', 'Adrogué', 12.0, 1200, 'en curso', 5),
(DATEADD(DAY, -5, SYSDATETIMEOFFSET()), NULL, 'San Justo', 'Ciudadela', 10.1, 1100, 'pendiente', 6),
(DATEADD(DAY, -6, SYSDATETIMEOFFSET()), DATEADD(MINUTE, 20, SYSDATETIMEOFFSET()), 'La Plata', 'Berisso', 9.0, 900, 'finalizado', 7),
(DATEADD(DAY, -7, SYSDATETIMEOFFSET()), DATEADD(MINUTE, 25, SYSDATETIMEOFFSET()), 'Temperley', 'Lanús', 7.5, 850, 'finalizado', 8),
(DATEADD(DAY, -8, SYSDATETIMEOFFSET()), NULL, 'Moreno', 'Merlo', 11.0, 1000, 'pendiente', 9),
(DATEADD(DAY, -9, SYSDATETIMEOFFSET()), NULL, 'CABA', 'Tigre', 30.0, 3000, 'pendiente', 10);

```

RF7 – Asignación de chofer y pasajero a un viaje con medio de pago

El sistema debe relacionar chofer, pasajero, viaje y tarjeta en una relación única.

Regla de negocio:

- combinación (chofer, pasajero, viaje, tarjeta) única.

```

CREATE TABLE usuarios_viajes_tarjetas (
    uvt_id INT IDENTITY(1,1) PRIMARY KEY,
    usuario_chofer_id INT NOT NULL,
    usuario_pasajero_id INT NOT NULL,
    viaje_id INT NOT NULL,
    tarjeta_id INT NOT NULL,

    creado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
    actualizado_fecha DATETIMEOFFSET DEFAULT GETDATE(),

    CONSTRAINT uq_uvt UNIQUE (usuario_chofer_id, usuario_pasajero_id, viaje_id, tarjeta_id),
    CONSTRAINT fk_uvt_chofer FOREIGN KEY (usuario_chofer_id) REFERENCES usuarios(usuario_id),
    CONSTRAINT fk_uvt_pasajero FOREIGN KEY (usuario_pasajero_id) REFERENCES usuarios(usuario_id),
    CONSTRAINT fk_uvt_viaje FOREIGN KEY (viaje_id) REFERENCES viajes(viaje_id),
    CONSTRAINT fk_uvt_tarjeta FOREIGN KEY (tarjeta_id) REFERENCES tarjetas(tarjeta_id)
);

```

 Ingeniería de Datos I	Grupo 02	Entrega 01	Fecha 10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

Evidencia: inserción en usuarios_viajes_tarjetas.

```

| INSERT INTO usuarios_viajes_tarjetas (usuario_chofer_id, usuario_pasajero_id, viaje_id, tarjeta_id)
| VALUES
| (6,1,1,1),
| (7,2,2,2),
| (8,3,3,3),
| (9,4,4,4),
| (10,5,5,5),
| (11,1,6,1),
| (12,2,7,2),
| (13,3,8,3),
| (14,4,9,4),
| (15,5,10,5);

```

RF8 – Actualización del estado del viaje

El sistema debe permitir cambiar el estado del viaje a “en curso” y luego a “finalizado” (incluyendo hora final).

Evidencia: updates de estado.

RF9 – Calificación de viajes

El sistema debe permitir calificar viajes finalizados con puntaje y comentario, registrando trazabilidad.

Regla de negocio:

- Puntuacion entre 1 y 5.

```

| CREATE TABLE calificaciones (
|   calificacion_id INT IDENTITY(1,1) PRIMARY KEY,
|   puntuacion INT CHECK (puntuacion BETWEEN 1 AND 5),
|   comentario VARCHAR(255),
|   fecha DATETIMEOFFSET DEFAULT (GETDATE()),
|   viaje_id INT NOT NULL,
|
|   creado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
|   actualizado_fecha DATETIMEOFFSET DEFAULT GETDATE(),
|
|   CONSTRAINT fk_calificacion_viaje FOREIGN KEY (viaje_id) REFERENCES viajes(viaje_id)
| );

```

Evidencia: inserción en calificaciones.

Grupo	Entrega	Fecha
02	01	10/11/2025
Trabajo Práctico		
Integrador Obligatorio		

```
|INSERT INTO calificaciones (puntuacion, comentario, viaje_id)
VALUES
(5, 'Excelente servicio', 1),
(4, 'Buen viaje, puntual', 2),
(5, 'Muy cómodo y limpio', 3),
(3, 'Demora en la salida', 4),
(4, 'Correcto, sin inconvenientes', 5),
(5, 'Excelente atención', 6),
(4, 'Todo bien', 7),
(5, 'Muy amable el chofer', 8),
(3, 'Un poco de tráfico, todo ok', 9),
(5, 'Rápido y seguro', 10);
```

RF10 – Reporte de puntuación por chofer

El sistema debe ofrecer el promedio de calificaciones por chofer.

Evidencia: consulta de promedio

```
/* ----- 12) Promedio de calificaciones por chofer ----- */
SELECT uvt.usuario_chofer_id AS id_chofer,
       CONCAT(u.nombre, ' ', u.apellido) nombre_completo,
       AVG(calif.puntuacion) puntuacion_promedio
  FROM usuarios_viajes_tarjetas uvt
 LEFT JOIN calificaciones calif
    ON uvt.viaje_id = calif.viaje_id
 LEFT JOIN usuarios AS u
    ON uvt.usuario_chofer_id = u.usuario_id
 GROUP BY uvt.usuario_chofer_id,
          CONCAT(u.nombre, ' ', u.apellido)
 ORDER BY AVG(calif.puntuacion) DESC;
```

 Ingeniería de Datos I	Grupo 02	Entrega 01	Fecha 10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

RF11 – Historial de viajes del pasajero

El sistema debe listar los viajes de un pasajero con sus datos relevantes.

Evidencia: consulta de historial

```
/* ----- 13) Historial de viajes de un pasajero ----- */
SELECT v_viaje_id, v_origen, v_destino, v_costo, v_estado
FROM usuarios_viajes_tarjetas uvt
JOIN viajes v ON uvt.viaje_id = v.viaje_id
WHERE uvt.usuario_pasajero_id = 1;
```

RF12 – Identificación del chofer asignado a un viaje

El sistema debe permitir consultar el chofer asignado - datos de contacto.

Evidencia: consulta de chofer por viaje

```
/* ----- 14) Chofer asignado a un viaje ----- */
SELECT u.nombre, u.apellido, u.email, u.telefono
FROM usuarios_viajes_tarjetas uvt
JOIN usuarios u ON u.usuario_id = uvt.usuario_chofer_id
WHERE uvt.viaje_id = 1;
```

RF13 – Control de seguros vigentes

El sistema debe listar vehículos con seguros vigentes al día de la consulta.

Evidencia: consulta de vigencia

```
/* ----- 15) Vehículos con seguros vigentes ----- */
SELECT v.marca, v.modelo, v.patente,
       s.compania, s.fecha_vencimiento
FROM vehiculos v
JOIN seguros s ON v.vehiculo_id = s.vehiculo_id
WHERE s.fecha_vencimiento >= GETDATE();
```

RF14 – Análisis de gasto por pasajero

El sistema debe calcular el total gastado por pasajero considerando viajes finalizados.

Evidencia: consulta de sumatoria

```

/* ----- 16) Total gastado por pasajero ----- */
SELECT u.usuario_id,
       CONCAT(u.nombre, ' ', u.apellido) nombre_completo,
       SUM(v.costo) AS total_gastado
FROM usuarios u
JOIN usuarios_viajes_tarjetas uvt
  ON u.usuario_id = uvt.usuario_pasajero_id
JOIN viajes v
  ON uvt.viaje_id = v.viaje_id
WHERE v.estado = 'finalizado'
GROUP BY u.usuario_id,
         CONCAT(u.nombre, ' ', u.apellido)
ORDER BY total_gastado DESC;

```

RF15 – Bandeja de viajes pendientes por chofer

El sistema debe listar los viajes en estado “pendiente” asignados a un chofer.

Evidencia: consulta de pendientes (Query #17).

```

/* 20) Reporte de viajes (chofer, pasajero, vehículo, costo, estado) */
SELECT
    v.viaje_id,
    v.origen,
    v.destino,
    chofer.nombre + ' ' + chofer.apellido AS chofer,
    pasajero.nombre + ' ' + pasajero.apellido AS pasajero,
    v.costo,
    v.estado,
    vh.patente
FROM usuarios_viajes_tarjetas uvt
JOIN viajes v ON uvt.viaje_id = v.viaje_id
JOIN usuarios chofer ON uvt.usuario_chofer_id = chofer.usuario_id
JOIN usuarios pasajero ON uvt.usuario_pasajero_id = pasajero.usuario_id
JOIN vehiculos vh ON v.vehiculo_id = vh.vehiculo_id;

```

RF16 – Administración del estado de usuario

El sistema debe permitir suspender usuarios con registro de auditoría temporal.

Evidencia: update a “suspendido”

```
/* ----- 18) Suspender un usuario ----- */
UPDATE usuarios
SET estado = 'suspendido',
actualizado_fecha = GETDATE()
WHERE usuario_id = 1;
```

RF17 – Reporte operativo integral de viajes

El sistema debe emitir un reporte consolidado por viaje con chofer, pasajero, costo, estado y patente.

Evidencia: reporte completo (Query #20).

```
111 -- 20. Reporte completo de viajes (chofer, pasajero, vehículo, costo, estado)
112 SELECT
113     v_viaje_id,
114     v_fecha,
115     v_origen,
116     v_destino,
117     chofer.nombre + ' ' + chofer.apellido AS chofer,
118     pasajero.nombre + ' ' + pasajero.apellido AS pasajero,
119     v_costo,
120     v_estado,
121     vh_patente
122 FROM usuarios_viajes_tarjetas uvt
123 JOIN viajes v ON uvt.viaje_id = v.viaje_id
124 JOIN usuarios chofer ON uvt.usuario_chofer_id = chofer.usuario_id
125 JOIN usuarios pasajero ON uvt.usuario_pasajero_id = pasajero.usuario_id
126 JOIN vehiculos vh ON v.vehiculo_id = vh.vehiculo_id;
127
```

4.2. Requisitos No Funcionales

RNF1 – Rendimiento

El sistema debe procesar consultas comunes (inserciones, actualizaciones y reportes) en un tiempo menor a 1 segundo bajo carga normal (hasta 10 usuarios concurrentes).

RNF2 – Usabilidad

El sistema debe permitir una interacción simple con las tablas y vistas mediante un entorno SQL estándar.

Las vistas se diseñaron para ofrecer información consolidada y fácilmente interpretable, mejorando la legibilidad de los datos.

RNF4 – Escalabilidad

El sistema debe poder ampliarse para manejar un mayor volumen de usuarios, viajes y vehículos sin afectar el rendimiento, mediante:

Posibilidad de migrar a Azure SQL o SQL Server completo sin cambios estructurales.

 Ingeniería de Datos I	Grupo 02	Entrega 01	Fecha 10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

RNF5 – Disponibilidad

El servicio de base de datos debe reiniciarse automáticamente en caso de fallo.

RNF6 – Mantenibilidad

El código SQL está modularizado:

Scripts separados para creación de tablas, inserciones, y consultas.

Uso de views para centralizar reportes y evitar duplicación de código.

Esto facilita la actualización y depuración futura del sistema.

RNF7 – Integridad de datos

Se implementan restricciones y claves foráneas para garantizar consistencia:

Restricciones CHECK para validación (por ejemplo, vencimiento de tarjetas).

FOREIGN KEY para mantener relaciones entre usuarios, viajes, vehículos y seguros.

Eliminación controlada (por ejemplo, solo tarjetas vencidas).

RNF8 – Seguridad temporal y trazabilidad

Toda actualización registra la fecha y el usuario responsable mediante los campos actualizado_por y actualizado_fecha.

Esto asegura una trazabilidad completa de los cambios y soporte a auditorías.

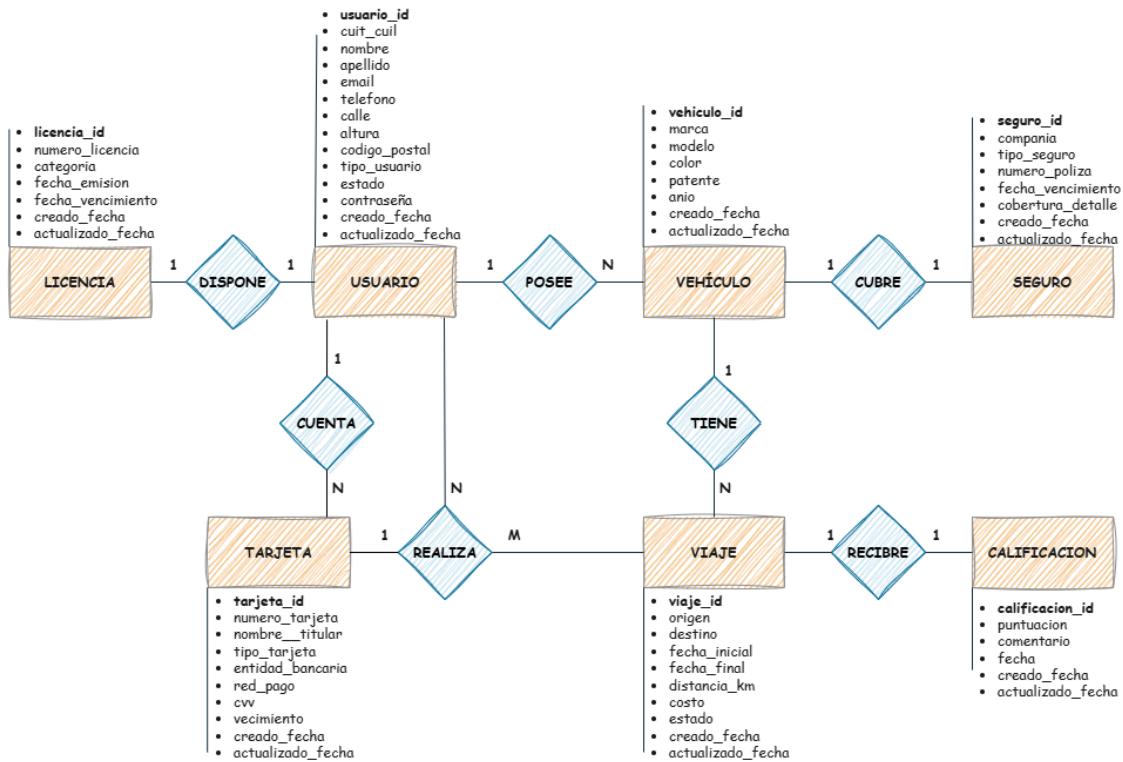
5. Diseño de la Base de Datos

5.1. Modelo Conceptual

Se entrega Diagrama Entidad-Relación en archivo drivear_DER.drawio

5.2. Modelo Lógico

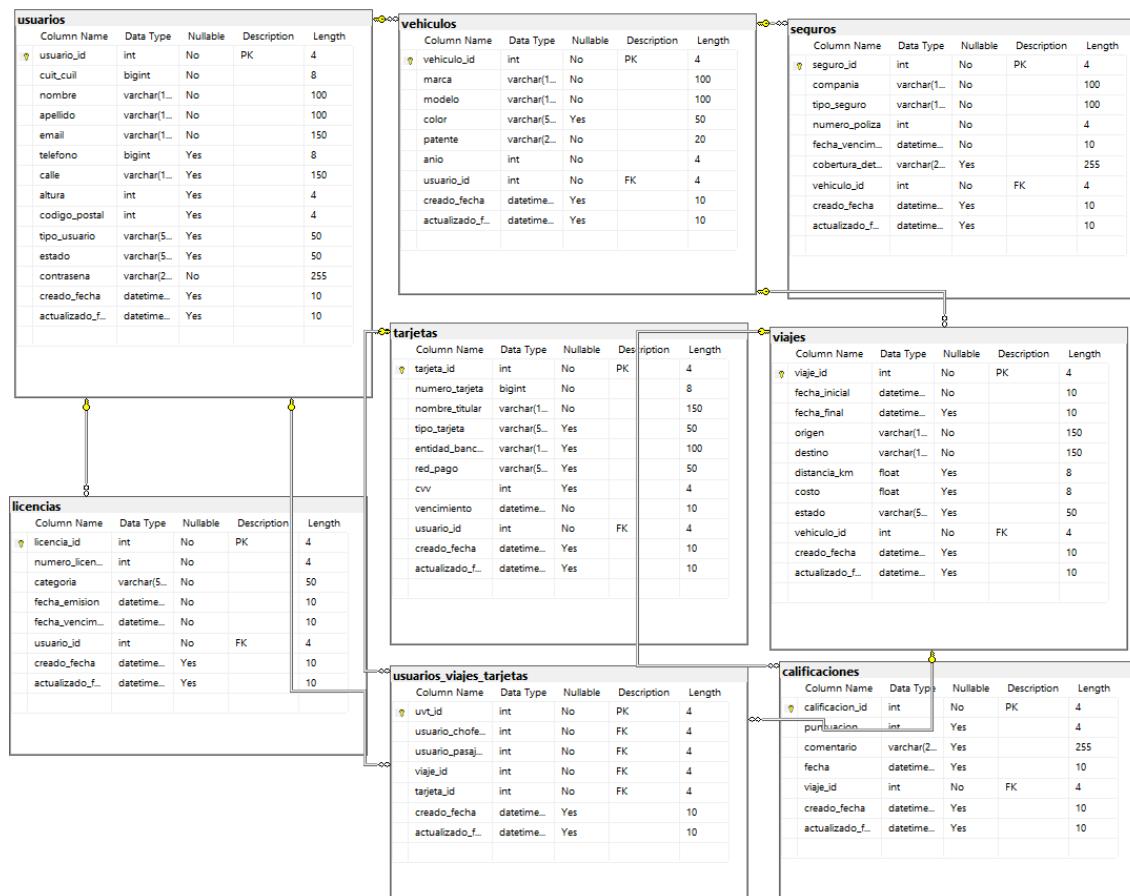
Se entrega detalle de tablas, atributos, claves primarias y foráneas en archivo drivear_MR.drawio





5.3. Modelo Físico

Se detalla el Esquema SQL (tablas con sus restricciones) en el archivo diccionario_de_datos_drivear.xlsx.



5.4. Diccionario de Datos

Se entrega por separado un documento `diccionario_de_datos_drivear.xlsx` que describe las tablas y sus atributos.

5.5. Dependencias Funcionales

Tabla: usuarios

Clave primaria: `usuario_id`

Dependencia funcional:

`usuario_id` → `cuit_cuil, nombre, apellido, email, tipo_usuario, estado`

- 1FN: Cumple. Todos los campos son atómicos (no hay listas o valores múltiples).
- 2FN: Cumple. Todos los atributos dependen completamente de la clave primaria `usuario_id`.
- 3FN: Cumple. No existen dependencias transitivas entre los atributos no clave.

Tabla: licencias

Clave primaria: `licencia_id`

Dependencia funcional:

 Ingeniería de Datos I	Grupo 02	Entrega 01	Fecha 10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

numero_licencia → categoria, fecha_emision, fecha_vencimiento, usuario_id

- 1FN: Cumple. Los datos son atómicos.
- 2FN: Cumple. Todos los atributos dependen completamente de licencia_id o del número de licencia, que es único.
- 3FN: Cumple. No existen dependencias entre los atributos no clave.

Tabla: vehiculos

Clave primaria: vehiculo_id

Dependencia funcional: vehiculo_id → marca, modelo, patente, anio, usuario_id

- 1FN: Cumple. Los valores son indivisibles.
- 2FN: Cumple. Todos los atributos dependen solo del identificador del vehículo.
- 3FN: Cumple. No hay dependencias entre atributos no clave (por ejemplo, la marca no determina el modelo).

Tabla: seguros

Clave primaria: seguro_id

Dependencia funcional: numero_poliza → compania, tipo_seguro, vehiculo_id

- 1FN: Cumple. No hay grupos repetidos ni atributos multivaluados.
- 2FN: Cumple. Los atributos dependen solo de la clave primaria o del número de póliza, que es único.
- 3FN: Cumple. No existen dependencias transitivas entre los atributos.

Tabla: tarjetas

Clave primaria: tarjeta_id

Dependencia funcional:

numero_tarjeta → nombre_titular, tipo_tarjeta, vencimiento, usuario_id

- 1FN: Cumple. Todos los valores son atómicos.
- 2FN: Cumple. Todos los campos dependen completamente de la clave primaria o del número de tarjeta.
- 3FN: Cumple. No hay dependencias transitivas.

Tabla: viajes

Clave primaria: viaje_id

Dependencia funcional:

viaje_id → fecha, origen, destino, hora_inicial, estado, vehiculo_id

- 1FN: Cumple. No hay atributos multivaluados.
- 2FN: Cumple. Los campos dependen totalmente de la clave viaje_id.
- 3FN: Cumple. No hay dependencias entre atributos no clave.

 Ingeniería de Datos I	Grupo	Entrega	Fecha
	02	01	10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

Tabla: calificaciones

Clave primaria: calificacion_id

Dependencia funcional: calificacion_id → puntuacion, comentario, fecha, viaje_id

- 1FN: Cumple. Todos los valores son atómicos.
- 2FN: Cumple. Los atributos dependen totalmente de la PK.
- 3FN: Cumple. No hay dependencias transitivas entre atributos.

Tabla: usuarios_viajes_tarjetas

Clave primaria: uvt_id

Dependencia funcional:

(usuario_chofer_id, usuario_pasajero_id, viaje_id, tarjeta_id) → uvt_id

- 1FN: Cumple. Cada fila representa una relación única entre chofer, pasajero, viaje y tarjeta.
- 2FN: Cumple. La clave compuesta determina completamente el registro.
- 3FN: Cumple. No existen dependencias entre los atributos no clave.

6. Implementación y Scripts SQL

6.1. Inserción de Datos

Lista de inserciones:

- usuarios – 15 registros (pasajeros y choferes)
- licencias – 10 registros (una por chofer, usuario_id 6 a 15)
- vehiculos – 10 registros (asignados a pasajeros, usuario_id 1 a 5)
- seguros – 10 registros (uno por vehículo, vehiculo_id 1 a 10)
- tarjetas – 10 registros (asociadas a pasajeros/usuarios, usuario_id 1 a 10)
- viajes – 10 registros (referencian vehículos, vehiculo_id 1 a 10)
- usuarios_viajes_tarjetas – 10 registros (relacionan chofer, pasajero, viaje y tarjeta)
- calificaciones – 10 registros (una por viaje)

6.2. Consultas SQL

Lista de consultas:

- Promedio de calificaciones por chofer
- Historial de viajes de un pasajero
- Chofer asignado a un viaje
- Vehículos con seguros vigentes
- Total gastado por pasajero
- Reporte de viajes (chofer, pasajero, vehículo, costo, estado)

6.3. Triggers y Vistas

trg_BloquearTarjetaVencida: se ejecuta después de insertar datos en la tabla usuarios_viajes_tarjetas y verifica si alguna de las tarjetas asociadas está vencida

 Ingeniería de Datos I	Grupo 02	Entrega 01	Fecha 10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

comparando su fecha de vencimiento con la fecha actual. Si detecta una tarjeta vencida, muestra un error y revierte la transacción para impedir que se registren viajes con tarjetas no válidas.

trg_Viajes_Estado: se ejecuta en lugar de una actualización (INSTEAD OF UPDATE) sobre la tabla viajes y controla los cambios de estado de un viaje. Impide modificar viajes ya finalizados y define reglas válidas de transición entre estados: solo se puede pasar de “pendiente” a “en curso”, de “en curso” a “finalizado”, y volver a “pendiente” únicamente desde “en curso”. Si se intenta una transición no permitida, lanza un error y revierte la operación. Finalmente, realiza el UPDATE real ajustando fechas y campos según el nuevo estado.

vw_ViajesDetalle: muestra una vista consolidada de los viajes con información completa: datos del viaje (fechas, origen, destino, costo, estado), del vehículo, del chofer, del pasajero, de la tarjeta usada y del seguro más reciente del vehículo. Además, calcula si la tarjeta o el seguro están vencidos y la duración del viaje en minutos.

vw_MetricasChofer: métricas de cada chofer, mostrando su nombre, la cantidad total de viajes realizados, los ingresos acumulados de los viajes finalizados y el promedio de puntuación recibida por los pasajeros.

6.4. Funciones y Procedimientos

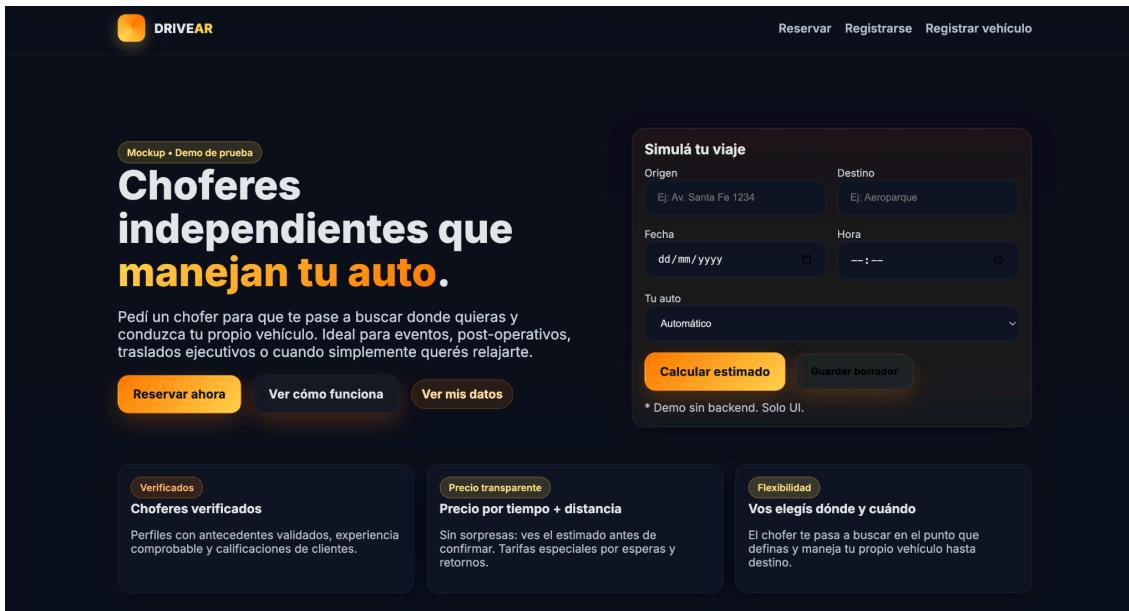
Incluya al menos 1 función y 1 procedimiento.

7. Interfaz de Usuario (Opcional)

Se generó un mockup a modo de ejemplo para demostrar cómo el usuario podrá navegar por el sitio, solicitar un servicio, calificar al conductor, entre otras funcionalidades.

link: <https://agusgazza.github.io/mockupbd1/index.html>

 Ingeniería de Datos I	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="padding: 5px;">Grupo</th><th style="padding: 5px;">Entrega</th><th style="padding: 5px;">Fecha</th></tr> </thead> <tbody> <tr> <td style="padding: 5px; text-align: center;">02</td><td style="padding: 5px; text-align: center;">01</td><td style="padding: 5px; text-align: center;">10/11/2025</td></tr> <tr> <td colspan="3" style="text-align: center; padding: 5px;">Trabajo Práctico</td></tr> <tr> <td colspan="3" style="text-align: center; padding: 5px;">Integrador Obligatorio</td></tr> </tbody> </table>	Grupo	Entrega	Fecha	02	01	10/11/2025	Trabajo Práctico			Integrador Obligatorio		
Grupo	Entrega	Fecha											
02	01	10/11/2025											
Trabajo Práctico													
Integrador Obligatorio													



8. Plan de Desarrollo y Metodología

Se utilizará la metodología **SCRUM** para organizar el desarrollo. El proyecto se dividirá en 4 sprints de 7 días, donde cada sprint entregará avances incrementales del sistema.

- Sprint 1:** relevamiento, definición de requerimientos y diseño de la base de datos.
- Sprint 2:** desarrollo de las funcionalidades principales e integración con la base de datos.
- Sprint 3:** pruebas, correcciones, documentación y despliegue.
- Sprint 4:** preparación de la presentación final de la aplicación (diapositivas, demo funcional y discurso).

9. Optimización y Rendimiento (Opcional)

Con el objetivo de mejorar la velocidad de búsqueda y acceso a los datos, se aplicaron índices en columnas que se utilizan con frecuencia en consultas y validaciones dentro del sistema. En particular, se crearon índices sobre los siguientes campos: **numero_licencia** en la tabla licencias, **patente** en vehiculos, **numero_tarjeta** en tarjetas y **numero_poliza** en seguros.

Estos campos se utilizan habitualmente para identificar registros específicos, por ejemplo, al validar una licencia, buscar un vehículo por su patente o comprobar la vigencia de una póliza o tarjeta. Gracias a los índices, la base de datos puede acceder directamente al registro correspondiente sin necesidad de recorrer todas las filas de la tabla, lo que reduce el tiempo de respuesta y mejora la eficiencia general del sistema, especialmente a medida que el volumen de datos aumenta.

 Ingeniería de Datos I	Grupo 02	Entrega 01	Fecha 10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

10. Seguridad y Recuperación (Opcional)

Para garantizar la protección de la información y evitar accesos no autorizados, se definieron políticas de seguridad a nivel de aplicación. Se estableció un control de acceso mediante usuario y contraseña, diferenciando permisos según el rol: el **Administrador** puede gestionar usuarios, vehículos y viajes, mientras que los usuarios **Chofer** y **Pasajero** solo pueden acceder a la información y funciones correspondientes a su perfil. Además, se recomienda el uso de contraseñas encriptadas y la ejecución de copias de seguridad periódicas de la base de datos para asegurar la recuperación ante fallas o pérdida de datos.

11. Pruebas y Validación (Opcional)

Caso 1: Insertar un usuario con un valor no válido en tipo_usuario.

Resultado esperado: La inserción debe fallar porque tipo usuario solo admite “chofer” o “pasajero”

Query: `INSERT INTO usuarios (cuit_cuil, nombre, apellido, email, tipo_usuario, contrasena)
VALUES (20333444456, 'Juan', 'Perez', 'juan@example.com', 'piloto', '1234');`

Resultado obtenido: The INSERT statement conflicted with the CHECK constraint "CK_usuarios_tipo_u_398D8EEE". The conflict occurred in database "sistema_transporte", table "dbo.usuarios", column 'tipo_usuario'.

Captura:

```

10 %
Messages
Msg 547, Level 16, State 0, Line 18
The INSERT statement conflicted with the CHECK constraint "CK_usuarios_tipo_u_398D8EEE". The conflict occurred in database "sistema_transporte", table "dbo.usuarios", column 'tipo_usuario'.
The statement has been terminated.

```

Caso 2: Insertar una CVV fuera del rango permitido (3-4 dígitos)

Resultado esperado: La inserción debe rechazarse porque CVV debe estar entre 100 y 9999.

Query: `INSERT INTO tarjetas (numero_tarjeta, nombre_titular, tipo_tarjeta, cvv, vencimiento, usuario_id)
VALUES (44443332221111, 'Carlos Lopez', 'debito', 12, '2026-05-01', 1);`

 Ingeniería de Datos I	Grupo 02	Entrega 01	Fecha 10/11/2025
Trabajo Práctico			
Integrador Obligatorio			

Resultado obtenido: The INSERT statement conflicted with the CHECK constraint "CK_tarjetas_cvv_5BE2A6F2". The conflict occurred in database "sistema_transporte", table "dbo.tarjetas", column 'cvv'.

Captura:

```

INSERT INTO tarjetas (numero_tarjeta, nombre_titular, tipo_tarjeta, cvv, vencimiento, usuario_id)
VALUES (4444333322221111, 'Carlos Lopez', 'debito', 12, '2026-05-01', 1);

Msg 547, Level 16, State 0, Line 191
The INSERT statement conflicted with the CHECK constraint "CK_tarjetas_cvv_5BE2A6F2". The conflict occurred in database "sistema_transporte", table "dbo.tarjetas", column 'cvv'.
The statement has been terminated.

```

12. Costos y Presupuesto (Opcional)

item	Descripción	Costo unitario	Cantidad	Subtotal
01	Horas Desarrollo	USD 60	120	USD 7200
02	Servidor - Puesta en marcha	USD 200	1	USD 200
03	Proyecto	USD 100	30	USD 3000
			Total	USD 10400

13. Conclusiones y Recomendaciones

A lo largo del desarrollo de este trabajo práctico se logró diseñar e implementar un sistema orientado a la gestión de la APP DriveAR. Se trabajó con restricciones de integridad, relaciones entre tablas e índices para mejorar el rendimiento, lo que permitió comprender la importancia de asegurar la consistencia y eficiencia en el almacenamiento de la información.

Además, se utilizaron conceptos de **metodologías ágiles**, planificación por sprints y control de versiones con Git, lo que permitió organizar el trabajo en etapas y mantener trazabilidad de los cambios realizados.

Grupo	Entrega	Fecha
02	01	10/11/2025
Trabajo Práctico		
Integrador Obligatorio		

En conclusión, este proyecto permitió **integrar teoría y práctica**, abordando desde el análisis funcional hasta la implementación técnica y la optimización en sistemas basados en bases de datos.

14. Referencias y Anexos

Bibliografía, enlaces, diagramas extra y documentos de apoyo.

Bibliografía:

- SQL Server Documentation,
- <https://docs.microsoft.com/sql>.