# AUTORENTA

***-***

**Eduar Damian Chanaga Gonzalez**

**P1**

**PEDRO FELIPE GÓMEZ BONILLA**

**CAMPUSLANDS**
**ARTEMIS**
**RUTA JAVA**
**FLORIDABLANCA**
**2024**

**Tabla de contenido**

# Introducción

Este documento detalla el diseño y la implementación de un sistema de información para AutoRental, una empresa dedicada al alquiler de vehículos con múltiples sucursales y planes de expansión. El sistema propuesto incluye una robusta estructura de base de datos y aplicativos de software diseñados para gestionar eficientemente las operaciones internas y mejorar la interacción con los clientes. A lo largo de este documento, se presentan los requerimientos detallados tanto para la base de datos como para las aplicaciones de software, junto con las consideraciones técnicas y prácticas para su implementación.

## Contenido del Documento

1. **Requerimientos de la Base de Datos**:
   - Se especifica la estructura necesaria para almacenar información crucial como sucursales, empleados, clientes, vehículos y alquileres, detallando los atributos y relaciones de cada entidad.

2. **Requerimientos de los Aplicativos de Software**:
   - Se describen las funcionalidades clave de los aplicativos tanto para la gestión interna como para la interacción con los clientes, incluyendo desde el registro de usuarios hasta la gestión de alquileres y consultas históricas.

3. **Instalación y Configuración del Sistema**:
   - Se detallan los pasos necesarios para la instalación del servidor de base de datos MySQL en diferentes plataformas, junto con los requisitos de hardware y software recomendados.

4. **Modelado y Diseño de la Base de Datos**:
   - Se presenta el modelo conceptual, lógico y físico de la base de datos, mostrando las entidades, atributos y relaciones implementadas.

5. **Implementación de la Lógica de Negocio**:
   - Se incluyen procedimientos almacenados y triggers que gestionan operaciones críticas como la gestión de alquileres, clientes y auditoría de datos.

6. **Planificación y Ejecución del Proyecto**:
   - Se ofrece una guía detallada sobre la planificación y ejecución del proyecto, desde la situación inicial hasta la implementación final del sistema de información.

7. **Consideraciones Adicionales**:
   - Se discuten temas como la seguridad, el rendimiento y la escalabilidad del sistema, asegurando una implementación sólida y efectiva para AutoRental.

Este documento proporciona una visión integral del proceso de diseño, implementación y despliegue del sistema de información para AutoRental, abordando todos los aspectos técnicos y prácticos necesarios para su éxito operacional.

# Caso de Estudio

**Situación problema:** La empresa Auto Rental se dedica al alquiler de vehículos y necesita un sistema de información robusto para gestionar sus operaciones. Con 5 sucursales actuales y planes de expansión, requieren una base de datos que pueda manejar eficientemente la información de sucursales, empleados, clientes, vehículos y alquileres. Además, necesitan aplicaciones de software que permitan a los empleados gestionar internamente y a los clientes interactuar con el servicio de alquiler de vehículos.

## Requerimientos de la base de datos:

1. **Sucursales:** Ciudad, dirección, teléfono fijo, celular, correo electrónico.
2. **Empleados:** Sucursal, cédula, nombres, apellidos, dirección, ciudad de residencia, celular, correo electrónico.
3. **Clientes:** Cédula, nombres, apellidos, dirección, ciudad de residencia, celular, correo electrónico.
4. **Vehículos:** Tipo de vehículo, placa, referencia, modelo, número de puertas, capacidad, sunroof (booleano), motor, color.
5. **Alquileres:** Vehículo alquilado, cliente, empleado que atendió, sucursal de salida y llegada, fecha de salida, fecha de llegada, fecha esperada de llegada, valor de alquiler por semana, valor de alquiler por día, porcentaje de descuento aplicado, valor cotizado, valor pagado.

Requerimientos de los aplicativos de software:

1. **Software de gestión:**
    - Gestión de sucursales, vehículos y empleados.
    - Gestión de alquileres.
    - Interfaz para empleados y atención al público.
2. **Aplicativo web para clientes:**
    - Registro de usuarios (signup).
    - Inicio de sesión (login).
    - Consulta de disponibilidad de vehículos por tipo, rango de precios y fechas.
    - Proceso de alquiler de vehículos.
    - Consulta de historial de alquileres.

Consideraciones adicionales:

- Implementación de la lógica de negocio para las operaciones clave como registro, inicio de sesión, consulta de disponibilidad de vehículos, alquileres y consulta de historial.
- Creación de cuentas de usuario para los aplicativos, gestionando permisos según el rol (cliente o empleado).

# Instalación General

**Sistema Operativo**
- Compatible con Windows, Ubuntu, y macOS.

**Hardware Recomendado**
- CPU: Mínimo 2 núcleos.
- RAM: Mínimo 4 GB.
- Espacio en Disco: Mínimo 10 GB.

**Software Necesario**
- MySQL Server 8.0 o superior.
- MySQL Workbench 8.0 (opcional, para gestión visual).

**Descargas**
**MySQL Server**
- Descargar desde: [MySQL Community Server](https://dev.mysql.com/downloads/mysql/)

**MySQL Workbench**
- Descargar desde: [MySQL Workbench](https://dev.mysql.com/downloads/workbench/)

**Instalación del Servidor de Base de Datos**
**Windows**
1. Ejecutar el instalador de MySQL.
2. Seguir las instrucciones del asistente de instalación.
3. Configurar la contraseña del usuario `root`.

**Ubuntu**
```bash
sudo apt update
sudo apt install mysql-server
sudo mysql_secure_installation
```

**macOS**
1. Usar Homebrew para instalar MySQL:
```bash
brew install mysql
```
2. Iniciar el servidor MySQL:
```bash
brew services start mysql
```

**Configuración Inicial**

**Crear Base de Datos y Tablas**
1. Iniciar sesión en MySQL como usuario `root`:

```bash
mysql -u root -p
```

2. Crear la base de datos y las tablas ejecutando los siguientes comandos SQL:

```sql
-- Creación de la base de datos
CREATE DATABASE alquiler;
USE alquiler;

-- Creación de la tabla sucursal
CREATE TABLE sucursal (
    id INT AUTO_INCREMENT PRIMARY KEY,
    ciudad VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    telefono_fijo VARCHAR(15),
    celular VARCHAR(15),
    correo_electronico VARCHAR(50)
);

-- Creación de la tabla empleado
CREATE TABLE Empleado (
    id INT AUTO_INCREMENT PRIMARY KEY,
    sucursal_id INT,
    cedula VARCHAR(20) UNIQUE NOT NULL,
    nombres VARCHAR(50) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    ciudad_residencia VARCHAR(50) NOT NULL,
    celular VARCHAR(15),
    correo_electronico VARCHAR(50),
    FOREIGN KEY (sucursal_id) REFERENCES sucursal(id)
);

-- Creación de la tabla cliente
CREATE TABLE Cliente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cedula VARCHAR(20) UNIQUE NOT NULL,
    nombres VARCHAR(50) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    ciudad_residencia VARCHAR(50) NOT NULL,
    celular VARCHAR(15),
    correo_electronico VARCHAR(50)
);

-- Creación de la tabla vehiculo
CREATE TABLE Vehiculo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    tipo_vehiculo VARCHAR(50) NOT NULL,
    placa VARCHAR(10) NOT NULL,
    referencia VARCHAR(50) NOT NULL,
    modelo YEAR NOT NULL,
    puertas INT NOT NULL,
    capacidad INT NOT NULL,
    sunroof TINYINT,
    motor VARCHAR(50),
    color VARCHAR(20)
);

-- Creación de la tabla alquiler
CREATE TABLE Alquiler (
    id INT AUTO_INCREMENT PRIMARY KEY,
    vehiculo_id INT,
    cliente_id INT,
    empleado_id INT,
    sucursal_salida_id INT,
    fecha_salida DATE NOT NULL,
    sucursal_llegada_id INT,
    fecha_llegada DATE NOT NULL,
    fecha_esperada_llegada DATE NOT NULL,
    valor_alquiler_semana DECIMAL(10,2) NOT NULL,
    valor_alquiler_dia DECIMAL(10,2) NOT NULL,
    porcentaje_descuento DECIMAL(5,2),
    valor_cotizado DECIMAL(10,2) NOT NULL,
    valor_pagado DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (vehiculo_id) REFERENCES Vehiculo(id),
    FOREIGN KEY (cliente_id) REFERENCES Cliente(id),
    FOREIGN KEY (empleado_id) REFERENCES Empleado(id),
    FOREIGN KEY (sucursal_salida_id) REFERENCES sucursal(id),
    FOREIGN KEY (sucursal_llegada_id) REFERENCES sucursal(id)
);
```

```sql
-- Creación de la tabla para llevar el registro de las modificaciones en la tabla alquiler
CREATE TABLE Alquiler_Log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    alquiler_id INT,
    fecha_modificacion TIMESTAMP,
    tipo_accion VARCHAR(10), -- Puede ser 'INSERT', 'UPDATE' o 'DELETE'
    FOREIGN KEY (alquiler_id) REFERENCES Alquiler(id)
);

-- Creación de la tabla para llevar el registro de las modificaciones en la tabla cliente
CREATE TABLE Cliente_Log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente INT,
    nombre_cliente VARCHAR(100),
    tipo_accion VARCHAR(10), -- Puede ser 'INSERT', 'UPDATE' o 'DELETE'
    fecha_modificacion TIMESTAMP
);

-- Creación de la tabla de respaldo para clientes
CREATE TABLE IF NOT EXISTS cliente_backup (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cedula VARCHAR(20) UNIQUE NOT NULL,
    nombres VARCHAR(50) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    ciudad_residencia VARCHAR(50) NOT NULL,
    celular VARCHAR(15),
    correo_electronico VARCHAR(50)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```sql
-- Creación de usuario 'empleado'
CREATE USER 'empleado'@'%' IDENTIFIED BY 'empleado123';

-- Conceder permisos al usuario 'empleado'
GRANT SELECT, UPDATE, INSERT ON alquiler.vehiculo TO 'empleado'@'%';
GRANT SELECT, UPDATE, INSERT ON alquiler.alquiler TO 'empleado'@'%';
GRANT SELECT, UPDATE, INSERT ON alquiler.cliente_log TO 'empleado'@'%';
GRANT EXECUTE ON PROCEDURE alquiler.modificar_alquiler TO 'empleado'@'%';
GRANT EXECUTE ON PROCEDURE alquiler.ver_registros_alquiler TO 'empleado'@'%';
GRANT EXECUTE ON PROCEDURE alquiler.Consultar_disponibilidad_vehiculos TO 'empleado'@'%'
GRANT EXECUTE ON PROCEDURE alquiler.Agregar_Alquiler_nuevo TO 'empleado'@'%';
GRANT EXECUTE ON PROCEDURE alquiler.agregar_cliente TO 'empleado'@'%';
GRANT EXECUTE ON PROCEDURE alquiler.modificar_cliente TO 'empleado'@'%';

-- Creación de usuario 'cliente'
CREATE USER 'cliente'@'%' IDENTIFIED BY 'cliente123';

-- Conceder permisos al usuario 'cliente'
GRANT SELECT ON alquiler.alquiler TO 'cliente'@'%';
GRANT EXECUTE ON PROCEDURE alquiler.ver_registros_alquiler TO 'cliente'@'%';
GRANT EXECUTE ON PROCEDURE alquiler.VerVehiculosDisponiblesPorTipo TO 'cliente'@'%';

-- Guardar cambios de permisos
FLUSH PRIVILEGES;
```

**Conexión a la Base de Datos**

**Usando MySQL Workbench**

1. Abrir MySQL Workbench.
2. Crear una nueva conexión con los siguientes detalles:
   - **Hostname**: localhost
   - **Port**: 3306
   - **Username**: root
   - **Password**: la contraseña configurada para root.
3. Probar la conexión y conectarse a la base de datos.

# Planificación

Ejecución del Proyecto: Sistema de Información para AutoRental

### Situación Problema

La empresa AutoRental ha contratado a su empresa para desarrollar un sistema de información que gestione su operación de alquiler de vehículos. AutoRental tiene cinco sucursales en diferentes ciudades y se proyecta a expandirse. La flota de vehículos incluye diferentes tipos y modelos. Los clientes pueden alquilar un vehículo en una sucursal y devolverlo en otra, y hay descuentos y tarifas variables dependiendo del tipo de vehículo y la duración del alquiler.

### Requerimientos de Información

La base de datos debe almacenar información sobre sucursales, empleados, clientes, vehículos y alquileres. Además, debe soportar tres aplicaciones de software: una herramienta de gestión interna.

## 2. Construcción del Modelo Conceptual

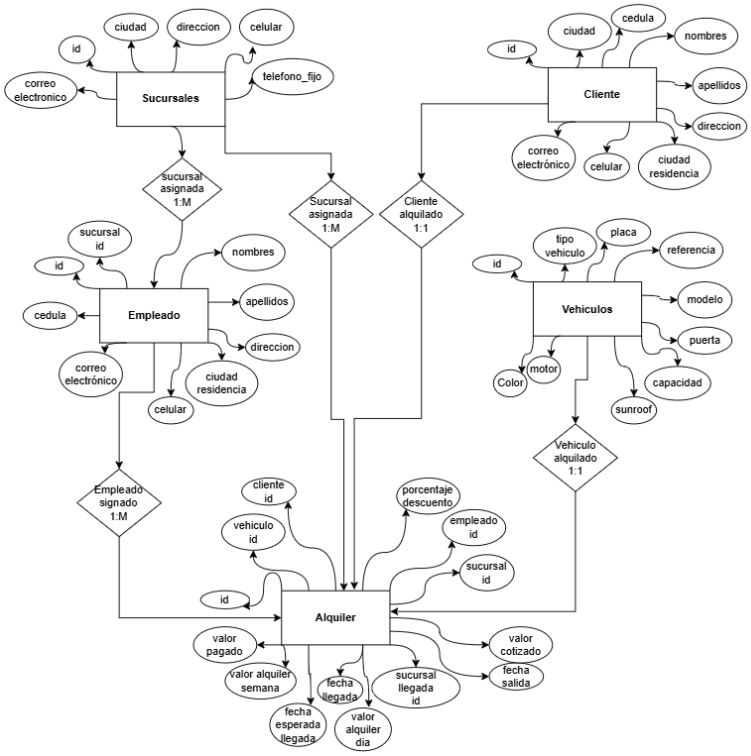
**Descripción**

El modelo conceptual identifica las entidades principales y sus relaciones para el sistema de AutoRental:

- **Sucursal:** Representa cada sucursal de AutoRental.
- **Empleado:** Representa a los empleados que trabajan en las sucursales.
- **Cliente:** Representa a los clientes que alquilan vehículos.
- **Vehículo:** Representa los vehículos disponibles para alquilar.
- **Alquiler:** Representa las transacciones de alquiler de vehículos.

**GRAFICA:**

**Gráfica**




**Descripción Técnica**

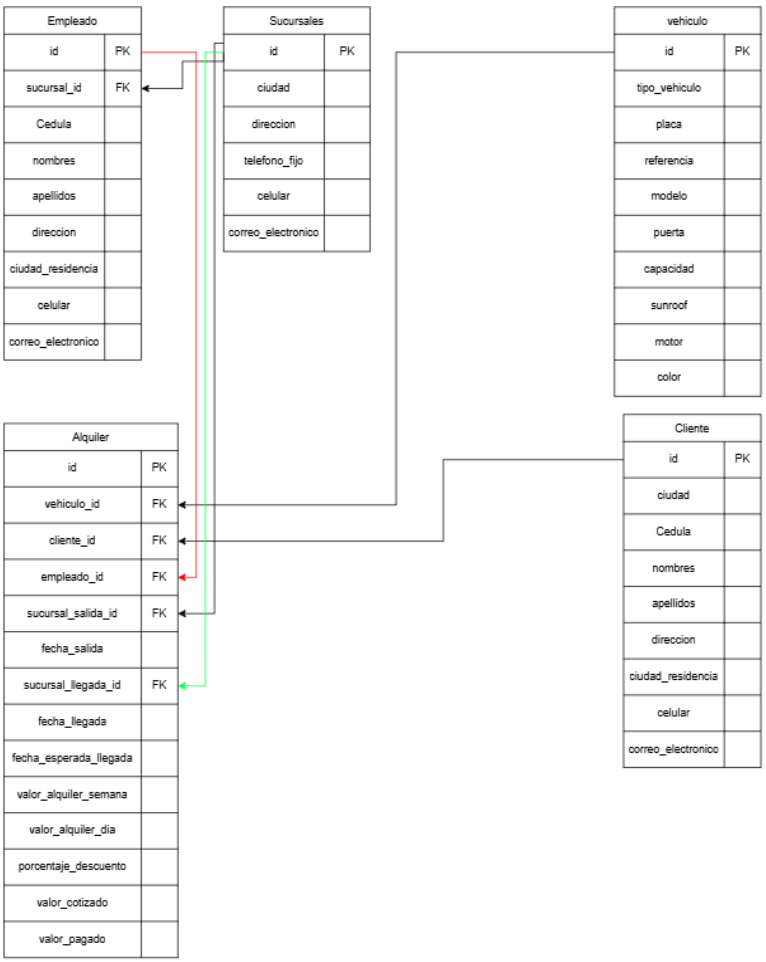
- **Entidad Sucursal:**
  - Atributos: id, ciudad, dirección, teléfono_fijo, celular, correo_electronico.
- **Entidad Empleado:**
  - Atributos: id, sucursal_id, cedula, nombres, apellidos, direccion, ciudad_residencia, celular, correo_electronico.
- **Entidad Cliente:**
  - Atributos: id, cedula, nombres, apellidos, direccion, ciudad_residencia, celular, correo_electronico.
- **Entidad Vehículo:**
  - Atributos: id, tipo_vehiculo, placa, referencia, modelo, puertas, capacidad, sunroof, motor, color.
- **Entidad Alquiler:**
  - Atributos: id, vehiculo_id, cliente_id, empleado_id, sucursal_salida_id, fecha_salida, sucursal_llegada_id, fecha_llegada, fecha_esperada_llegada, valor_semanal, valor_diario, descuento, valor_cotizado, valor_pagado.

# 3. Construcción del Modelo Lógico

## Descripción

El modelo lógico traduce el modelo conceptual a una estructura de base de datos relacional con tablas y relaciones.

## Gráfica




## Descripción Técnica

**- Tabla Sucursal:**
  - Columnas: id (PK), ciudad, dirección, teléfono_fijo, celular, correo_electronico.
- **Tabla Empleado:**
  - Columnas: id (PK), sucursal_id (FK), cedula (UNIQUE), nombres, apellidos, direccion, ciudad_residencia, celular, correo_electronico.
**- Tabla Cliente:**
  - Columnas: id (PK), cedula (UNIQUE), nombres, apellidos, direccion, ciudad_residencia, celular, correo_electronico.
**- Tabla Vehículo:**

- Columnas: id (PK), tipo_vehiculo, placa (UNIQUE), referencia, modelo, puertas, capacidad, sunroof, motor, color.

**- Tabla Alquiler:**

- Columnas: id (PK), vehiculo_id (FK), cliente_id (FK), empleado_id (FK), sucursal_salida_id (FK), fecha_salida, sucursal_llegada_id (FK), fecha_llegada, fecha_esperada_llegada, valor_semanal, valor_diario, descuento, valor_cotizado, valor_pagado.

## 4. Normalización del Modelo Lógico

### Primera Forma Normal (1FN)

- **Descripción:**

  Asegurar que cada campo contiene solo valores atómicos.

- **Gráfica:**


### Segunda Forma Normal (2FN)

**- Descripción:**

Eliminar dependencias parciales.

- **Gráfica:**

**Tercera Forma Normal (3FN)**

- **Descripción:**
  Eliminar dependencias transitivas.
- **Gráfica:**

| Sucursal | | | | | |
|---|---|---|---|---|---|
| id | ciudad | direccion | telefono_fijo | celular | correo_electronico |
| | | | | | |

| Empleado | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| id | sucursal_id | cedula | nombres | apellidos | direccion | ciudad_residencia | celular | correo_electronico |
| | | | | | | | | |

| Cliente | | | | | | | |
|---|---|---|---|---|---|---|---|
| id | cedula | nombres | apellidos | direccion | ciudad_residencia | celular | correo_electronico |
| | | | | | | | |

| Vehiculo | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| id | tipo_vehiculo_id | placa | referencia | modelo | puertas | capacidad | sunroof | motor | color |
| | | | | | | | | | |

| Alquiler | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| id | vehiculo_id | cliente_id | empleado_id | sucursal_salida_id | fecha_salida | sucursal_llegada_id | fecha_llegada | fecha_esperada_llegada | valor_cotizado | valor_pagado |
| | | | | | | | | | | |

| Tipo_Vehiculo | | | |
|---|---|---|---|
| id | descripcion | valor_semana | valor_dia |
| | | | |

| Descuento | | | | |
|---|---|---|---|---|
| id | tipo_vehiculo_id | porcentaje_descuento | fecha_inicio | fecha_fin |
| | | | | |

## 5. Construcción del Modelo Físico

**Descripción**

Implementación de la base de datos en un SGBD, creando tablas, índices y restricciones.

**Código**

```sql
-- creacion de base de datos
create database alquiler;
use alquiler;
-- creacion de la tabla sucursal
CREATE TABLE sucursal (
    id INT AUTO_INCREMENT PRIMARY KEY,
    ciudad VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    telefono_fijo VARCHAR(15),
    celular VARCHAR(15),
    correo_electronico VARCHAR(50)
);
-- creacion de la tabla empleado
CREATE TABLE Empleado (
    id INT AUTO_INCREMENT PRIMARY KEY,
    sucursal_id INT,
    cedula VARCHAR(20) UNIQUE NOT NULL,
    nombres VARCHAR(50) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    ciudad_residencia VARCHAR(50) NOT NULL,
    celular VARCHAR(15),
    correo_electronico VARCHAR(50),
    FOREIGN KEY (sucursal_id) REFERENCES sucursal(id)
);
-- creacion de la tabla cliente
CREATE TABLE Cliente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cedula VARCHAR(20) UNIQUE NOT NULL,
    nombres VARCHAR(50) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    ciudad_residencia VARCHAR(50) NOT NULL,
    celular VARCHAR(15),
    correo_electronico VARCHAR(50)
);
```

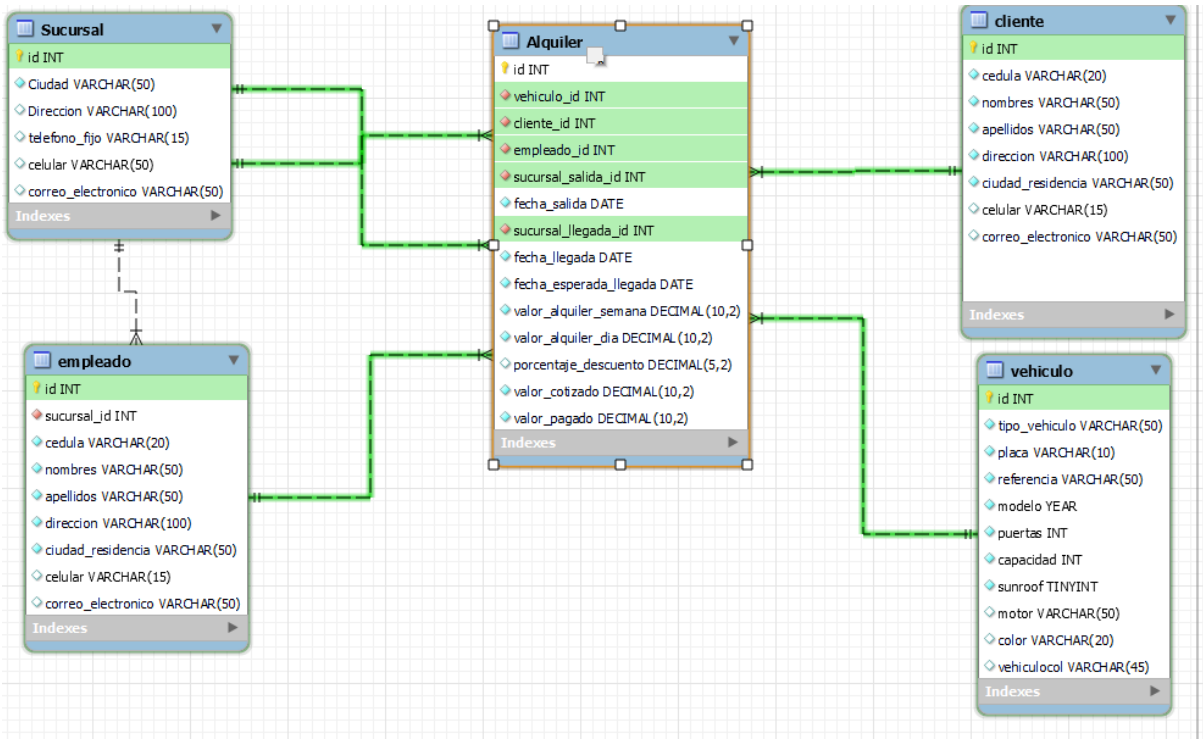
```sql
-- creacion de la tabla vehiculo
CREATE TABLE Vehiculo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    tipo_vehiculo VARCHAR(50) NOT NULL,
    placa VARCHAR(10) NOT NULL,
    referencia VARCHAR(50) NOT NULL,
    modelo YEAR NOT NULL,
    puertas INT NOT NULL,
    capacidad INT NOT NULL,
    sunroof TINYINT,
    motor VARCHAR(50),
    color VARCHAR(20)
);
-- creacion de la tabla alquiler
CREATE TABLE Alquiler (
    id INT AUTO_INCREMENT PRIMARY KEY,
    vehiculo_id INT,
    cliente_id INT,
    empleado_id INT,
    sucursal_salida_id INT,
    fecha_salida DATE NOT NULL,
    sucursal_llegada_id INT,
    fecha_llegada DATE NOT NULL,
    fecha_esperada_llegada DATE NOT NULL,
    valor_alquiler_semana DECIMAL(10,2) NOT NULL,
    valor_alquiler_dia DECIMAL(10,2) NOT NULL,
    porcentaje_descuento DECIMAL(5,2),
    valor_cotizado DECIMAL(10,2) NOT NULL,
    valor_pagado DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (vehiculo_id) REFERENCES Vehiculo(id),
    FOREIGN KEY (cliente_id) REFERENCES Cliente(id),
    FOREIGN KEY (empleado_id) REFERENCES Empleado(id),
    FOREIGN KEY (sucursal_salida_id) REFERENCES sucursal(id),
    FOREIGN KEY (sucursal_llegada_id) REFERENCES sucursal(id)
);
```

```sql
-- creacion de tabla para llevar el registro de las modificaciones echas en la tabla alquiler

CREATE TABLE Alquiler_Log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    alquiler_id INT,
    fecha_modificacion TIMESTAMP,
    tipo_accion VARCHAR(10), -- Puede ser 'INSERT', 'UPDATE' o 'DELETE'
    FOREIGN KEY (alquiler_id) REFERENCES Alquiler(id)
);


-- creacion de tabla para llevar el registro de las modificaciones echas en la tabla cliente

CREATE TABLE Cliente_Log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente INT,
    nombre_cliente VARCHAR(100),
    tipo_accion VARCHAR(10), -- Puede ser 'INSERT', 'UPDATE' o 'DELETE'
    fecha_modificacion TIMESTAMP
);
```

6. Diagrama E-R

**Descripción**
El diagrama E-R muestra las entidades y sus relaciones.

**Gráfica**



7. Inserción de Datos

**Descripción**
Cómo se insertan los datos en las tablas.

**Gráfica**

| id | vehiculo_id | cliente_id | empleado_id | sucursal_salida_id | fecha_salida | sucursal_llegada_id | fecha_llegada | fecha_esperada_llegada | valor_alquiler_semana | valor_alquiler_dia | porcentaje_descuento | valor_cotizado | valor_pagado |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 10 | 10 | 5 | 2024-06-25 | 5 | 2024-06-28 | 2024-06-30 | 400.00 | 50.00 | 10.00 | 350.00 | 300.00 |
| 2 | 87 | 25 | 8 | 3 | 2024-06-26 | 1 | 2024-06-28 | 2024-06-29 | 250.00 | 40.00 | 5.00 | 270.00 | 270.00 |
| 3 | 102 | 10 | 5 | 4 | 2024-06-27 | 5 | 2024-06-29 | 2024-06-30 | 350.00 | 60.00 | 15.00 | 380.00 | 380.00 |
| 4 | 45 | 30 | 2 | 1 | 2024-06-28 | 3 | 2024-06-30 | 2024-07-01 | 280.00 | 45.00 | 8.00 | 300.00 | 300.00 |
| 5 | 65 | 15 | 6 | 5 | 2024-06-29 | 2 | 2024-07-01 | 2024-07-02 | 320.00 | 55.00 | 12.00 | 350.00 | 350.00 |
| 6 | 120 | 40 | 9 | 3 | 2024-06-30 | 4 | 2024-07-02 | 2024-07-03 | 300.00 | 50.00 | 10.00 | 330.00 | 330.00 |
| 7 | 25 | 5 | 4 | 4 | 2024-07-01 | 5 | 2024-07-03 | 2024-07-04 | 350.00 | 60.00 | 15.00 | 380.00 | 380.00 |

**Descripción Técnica**
Ejemplos de sentencias SQL para la inserción de datos.

```sql
INSERT INTO sucursal (ciudad, direccion, telefono_fijo, celular, correo_electronico)
VALUES ('Ciudad Ejemplo', 'Dirección Ejemplo', '123456', '789012', 'ejemplo@correo.com');

INSERT INTO empleado (sucursal_id, cedula, nombres, apellidos, direccion, ciudad_residencia, celular, correo_electronico)
VALUES (1, '123456789', 'Juan', 'Pérez', 'Dirección Ejemplo', 'Ciudad Ejemplo', '789012', 'juan@correo.com');

INSERT INTO cliente (cedula, nombres, apellidos, direccion, ciudad_residencia, celular, correo_electronico)
VALUES ('987654321', 'María', 'Gómez', 'Dirección Ejemplo', 'Ciudad Ejemplo', '654321', 'maria@correo.com');

INSERT INTO vehiculo (tipo_vehiculo, placa, referencia, modelo, puertas, capacidad, sunroof, motor, color)
VALUES ('Sedán', 'ABC123', 'Toyota Corolla', 2022, 4, 5, TRUE, '2.0L', 'Azul');

INSERT INTO alquiler (vehiculo_id, cliente_id, empleado_id, sucursal_salida_id, fecha_salida, sucursal_llegada_id, fecha_llegada,
 fecha_esperada_llegada, valor_semanal, valor_diario, descuento, valor_cotizado, valor_pagado)
VALUES (1, 1, 1, 1, '2023-07-01', 2, '2023-07-08', '2023-07-07', 300.00, 50.00, 10.00, 270.00, 270.00);
```

# Procedimientos

Estos procedimientos proporcionan una estructura organizada y eficiente para manejar las operaciones básicas de gestión de alquileres y clientes en tu aplicación, mejorando la seguridad, rendimiento y mantenibilidad del sistema.

Aquí tienes la explicación detallada para cada uno de los procedimientos que mencionaste:

1. `modificar_alquiler()`

**Descripción:**
Este procedimiento actualiza los datos de un alquiler específico en la tabla `Alquiler`.

**Sintaxis:**

```
-- Modificar datos de alquiler
CREATE PROCEDURE modificar_alquiler(
    IN p_vehiculo_id INT,
    IN p_cliente_id INT,
    IN p_empleado_id INT,
    IN p_sucursal_salida_id INT,
    IN p_fecha_salida DATE,
    IN p_sucursal_llegada_id INT,
    IN p_fecha_llegada DATE,
    IN p_fecha_esperada_llegada DATE,
    IN p_valor_alquiler_semana DECIMAL(10,2),
    IN p_valor_alquiler_dia DECIMAL(10,2),
    IN p_porcentaje_descuento DECIMAL(5,2),
    IN p_valor_pagado DECIMAL(10,2),
    IN p_id INT
)
BEGIN
    UPDATE Alquiler
    SET
        vehiculo_id = p_vehiculo_id,
        cliente_id = p_cliente_id,
        empleado_id = p_empleado_id,
        sucursal_salida_id = p_sucursal_salida_id,
        fecha_salida = p_fecha_salida,
        sucursal_llegada_id = p_sucursal_llegada_id,
        fecha_llegada = p_fecha_llegada,
        fecha_esperada_llegada = p_fecha_esperada_llegada,
        valor_alquiler_semana = p_valor_alquiler_semana,
        valor_alquiler_dia = p_valor_alquiler_dia,
        porcentaje_descuento = p_porcentaje_descuento,
        valor_pagado = p_valor_pagado
    WHERE id = p_id;
END //
```

**Parámetros:**
- `p_vehiculo_id`: ID del vehículo involucrado en el alquiler.
- `p_cliente_id`: ID del cliente que realiza el alquiler.
- `p_empleado_id`: ID del empleado que gestiona el alquiler.
- `p_sucursal_salida_id`: ID de la sucursal desde donde se realiza el alquiler.
- `p_fecha_salida`: Fecha de inicio del alquiler.
- `p_sucursal_llegada_id`: ID de la sucursal de destino del alquiler.
- `p_fecha_llegada`: Fecha de fin del alquiler.
- `p_fecha_esperada_llegada`: Fecha esperada de llegada (planificada).
- `p_valor_alquiler_semana`: Valor del alquiler por semana.
- `p_valor_alquiler_dia`: Valor del alquiler por día.
- `p_porcentaje_descuento`: Porcentaje de descuento aplicado al alquiler.
- `p_valor_pagado`: Valor total pagado por el alquiler.
- `p_id`: ID único del registro de alquiler que se desea modificar.

**Salida/Retorno:**
No hay retorno explícito ya que es un procedimiento de actualización (`UPDATE`).

**Ejemplo de Implementación:**

```
CALL modificar_alquiler(1, 10, 3, 2, '2024-07-01', 4, '2024-07-10', '2024-07-08', 500.00, 80.00, 10.00, 600.00, 25);
```

2. `ver_registros_alquiler()`

**Descripción:**
Este procedimiento devuelve todos los registros de la tabla `Alquiler`.

```
-- Ver datos de alquiler
create procedure ver_registros_alquiler()
begin
    select *
    from Alquiler;
end //
```

**Parámetros:**
Ninguno.

**Salida/Retorno:**
Resultados de la consulta que muestra todos los registros de alquiler.

**Ejemplo de Implementación:**

```
CALL ver_registros_alquiler();
```

3. `Consultar_disponibilidad_vehiculos()`

**Descripción:**
Este procedimiento consulta la disponibilidad de vehículos de un tipo específico en un rango de fechas dado.

**Sintaxis:**

```
-- Consulta de disponibilidad de vehiculo
create procedure Consultar_disponibilidad_vehiculos(
    IN tipo_vehiculo_param VARCHAR(50),
    IN precio_min DECIMAL(10,2),
    IN precio_max DECIMAL(10,2),
    IN fecha_inicio DATE,
    IN fecha_fin DATE
)
BEGIN
    SELECT
        V.id AS vehiculo_id,
        V.tipo_vehiculo,
        V.referencia,
        V.modelo,
        V.puertas,
        V.capacidad,
        V.sunroof,
        V.motor,
        V.color
    FROM Vehiculo V
    right join Alquiler A on vehiculo_id = V.id
    where V.tipo_vehiculo = tipo_vehiculo_param and
            A.fecha_salida = fecha_inicio and
            A.fecha_llegada = fecha_fin;

END //
```

**Parámetros:**
- `tipo_vehiculo_param`: Tipo de vehículo a consultar.
- `precio_min`: Precio mínimo del vehículo.
- `precio_max`: Precio máximo del vehículo.
- `fecha_inicio`: Fecha de inicio del rango de disponibilidad.
- `fecha_fin`: Fecha de fin del rango de disponibilidad.

**Salida/Retorno:**
Resultados de la consulta que muestra los vehículos disponibles según los parámetros dados.

**Ejemplo de Implementación:**

```
CALL Consultar_disponibilidad_vehiculos('SUV', 100.00, 500.00, '2024-07-01', '2024-07-15');
```

4. `Agregar_Alquiler_nuevo()`

**Descripción:**
Este procedimiento añade un nuevo registro de alquiler a la tabla `Alquiler`.

**Sintaxis:**

```
-- Creacion de datos de alquiler
create procedure Agregar_Alquiler_nuevo(
in vehiculo_id int,
in cliente_id int,
in empleado_id int,
in sucursal_salida_id int,
in fecha_salida date,
in sucursal_llegada_id int,
in fecha_llegada date,
in fecha_esperada_llegada date,
in valor_alquiler_semana DECIMAL(10,2),
in valor_alquiler_dia decimal(10,2),
in porcentaje_descuento decimal(5,2),
in valor_cotizado decimal (10,2),
in valor_pagado decimal (10,2)
)
begin
    INSERT INTO Alquiler (vehiculo_id, cliente_id, empleado_id, sucursal_salida_id, fecha_salida, sucursal_llegada_id, fecha_llegada, fecha_esperada_llegada,
    valor_alquiler_semana, valor_alquiler_dia, porcentaje_descuento, valor_cotizado, valor_pagado) VALUES
    (vehiculo_id, cliente_id, empleado_id, sucursal_salida_id, fecha_salida, sucursal_llegada_id, fecha_llegada, fecha_esperada_llegada, valor_alquiler_semana, valor_alquiler_dia,
    porcentaje_descuento, valor_cotizado, valor_pagado);
end //
```

**Parámetros:**
- Detallados en la declaración del procedimiento.

**Salida/Retorno:**
No hay retorno explícito ya que es un procedimiento de inserción (`INSERT`).

**Ejemplo de Implementación:**

```
CALL Agregar_Alquiler_nuevo(1, 10, 3, 2, '2024-07-01', 4, '2024-07-10', '2024-07-08', 500.00, 80.00, 10.00, 550.00, 500.00);
```

5. `VerVehiculosDisponiblesPorTipo()`

**Descripción:**
Este procedimiento consulta y devuelve los vehículos disponibles de un tipo específico que no están actualmente alquilados.

**Sintaxis:**

```
-- ver vehiculos no alquilados

CREATE PROCEDURE VerVehiculosDisponiblesPorTipo(IN tipo_vehiculo_param VARCHAR(50))
BEGIN
    SELECT v.*
    FROM Vehiculo v
    LEFT JOIN Alquiler a ON v.id = a.vehiculo_id
    WHERE a.vehiculo_id IS NULL
    AND v.tipo_vehiculo = tipo_vehiculo_param;
END //
```

**Parámetros:**
- `tipo_vehiculo_param`: Tipo de vehículo para filtrar la consulta.

**Salida/Retorno:**
Resultados de la consulta que muestra los vehículos disponibles según el tipo especificado.
**Ejemplo de Implementación:**

```sql
CALL VerVehiculosDisponiblesPorTipo('Compacto');
```

6. `agregar_cliente()`

**Descripción:**
Este procedimiento añade un nuevo cliente a la tabla `Cliente`.

**Sintaxis:**

```sql
-- Agregar cliente nuevo
CREATE PROCEDURE agregar_cliente(
    IN p_cedula VARCHAR(20),
    IN p_nombres VARCHAR(50),
    IN p_apellidos VARCHAR(50),
    IN p_direccion VARCHAR(100),
    IN p_ciudad_residencia VARCHAR(50),
    IN p_celular VARCHAR(15),
    IN p_correo_electronico VARCHAR(50)
)
BEGIN
    -- Variable para guardar el ID del nuevo cliente
    DECLARE cliente_id INT;

    -- Insertar el cliente en la tabla Cliente
    INSERT INTO Cliente (cedula, nombres, apellidos, direccion, ciudad_residencia, celular, correo_electronico)
    VALUES (p_cedula, p_nombres, p_apellidos, p_direccion, p_ciudad_residencia, p_celular, p_correo_electronico);
END //
```

**Parámetros:**
- Detallados en la declaración del procedimiento.

**Salida/Retorno:**
No hay retorno explícito ya que es un procedimiento de inserción (`INSERT`).

**Ejemplo de Implementación:**

```sql
CALL agregar_cliente('1234567890', 'Juan', 'Pérez', 'Calle 123', 'Ciudad X', '3212345678', 'juan@example.com');
```

7. `modificar_cliente()`

**Descripción:**
Este procedimiento modifica los datos de un cliente existente en la tabla `Cliente`.

**Sintaxis:**

```sql
-- modificar cliente

CREATE PROCEDURE modificar_cliente(
    IN cliente_id INT,
    IN nuevo_nombres VARCHAR(50),
    IN nuevo_apellidos VARCHAR(50),
    IN nueva_direccion VARCHAR(100),
    IN nueva_ciudad_residencia VARCHAR(50),
    IN nuevo_celular VARCHAR(15),
    IN nuevo_correo_electronico VARCHAR(50)
)
BEGIN
    UPDATE Cliente
    SET
        nombres = nuevo_nombres,
        apellidos = nuevo_apellidos,
        direccion = nueva_direccion,
        ciudad_residencia = nueva_ciudad_residencia,
        celular = nuevo_celular,
        correo_electronico = nuevo_correo_electronico
    WHERE id = cliente_id;
END //
```

**Parámetros:**
- `cliente_id`: ID del cliente que se desea modificar.
- Resto de parámetros: Nuevos valores para los campos del cliente.

**Salida/Retorno:**
No hay retorno explícito ya que es un procedimiento de actualización (`UPDATE`).

**Ejemplo de Implementación:**

```sql
CALL modificar_cliente(10, 'Carlos', 'Gómez', 'Avenida 456', 'Ciudad Y', '333555777', 'carlos@example.com');
```

## Triggers

Estos triggers son esenciales para el mantenimiento de registros de auditoría y cumplimiento, proporcionando transparencia y seguimiento sobre las acciones críticas realizadas en las tablas `Alquiler` y `Cliente` de tu base de datos.

Trigger `trg_log_alquiler_insert`

**Descripción:** Registra la inserción de nuevos registros en la tabla `Alquiler` en la tabla de registro `Alquiler_Log`.

**Sintaxis:**

```
-- Trigger guardar registro de inserts en la tabla de alquiler
DELIMITER //
CREATE TRIGGER trg_log_alquiler_insert
AFTER INSERT ON Alquiler
FOR EACH ROW
BEGIN
    INSERT INTO Alquiler_Log (alquiler_id, tipo_accion, fecha_modificacion)
    VALUES (NEW.id, 'INSERT', CURRENT_TIMESTAMP);
END;
//
DELIMITER ;
```

**Parámetros:** No aplica explícitamente.

**Salida/Retorno:** No aplica explícitamente.

Trigger `trg_log_alquiler_update`

**Descripción:** Registra las actualizaciones de registros en la tabla `Alquiler` en la tabla de registro `Alquiler_Log`.

**Sintaxis:**

```
-- Trigger para guardar registro de update de la tabla de alquiler
DELIMITER //
CREATE TRIGGER trg_log_alquiler_update
AFTER UPDATE ON Alquiler
FOR EACH ROW
BEGIN
    INSERT INTO Alquiler_Log (alquiler_id, tipo_accion, fecha_modificacion)
    VALUES (OLD.id, 'UPDATE', CURRENT_TIMESTAMP);
END;
//
DELIMITER ;
```

**Parámetros:** No aplica explícitamente.

**Salida/Retorno:** No aplica explícitamente.

Trigger `trg_log_alquiler_delete`

**Descripción:** Registra las eliminaciones de registros en la tabla `Alquiler` en la tabla de registro `Alquiler_Log`.

**Sintaxis:**

```
-- Trigger para guardar registro de delete de la tabla de alquiler
DELIMITER //
CREATE TRIGGER trg_log_alquiler_delete
AFTER DELETE ON Alquiler
FOR EACH ROW
BEGIN
    INSERT INTO Alquiler_Log (alquiler_id, tipo_accion, fecha_modificacion)
    VALUES (OLD.id, 'DELETE', CURRENT_TIMESTAMP);
END;
//
DELIMITER ;
```

**Parámetros:** No aplica explícitamente.

**Salida/Retorno:** No aplica explícitamente.

Trigger `trg_log_cliente_insert`

**Descripción:** Registra la inserción de nuevos registros en la tabla `Cliente` en la tabla de registro `Cliente_Log`.

**Sintaxis:**

```
-- Trigger guardar registro de inserts en la tabla de alquiler
DELIMITER //
CREATE TRIGGER trg_log_alquiler_insert
AFTER INSERT ON Alquiler
FOR EACH ROW
BEGIN
    INSERT INTO Alquiler_Log (alquiler_id, tipo_accion, fecha_modificacion)
    VALUES (NEW.id, 'INSERT', CURRENT_TIMESTAMP);
END;
//
DELIMITER ;
```

**Parámetros:** No aplica explícitamente.

**Salida/Retorno:** No aplica explícitamente.

Trigger `trg_log_cliente_update`

**Descripción:** Registra las actualizaciones de registros en la tabla `Cliente` en la tabla de registro `Cliente_Log`.

**Sintaxis:**

```
-- Trigger para guardar registro de update de la tabla de Cliente
DELIMITER //
CREATE TRIGGER trg_log_cliente_update
AFTER UPDATE ON Cliente
FOR EACH ROW
BEGIN
    INSERT INTO Cliente_Log (id_cliente, nombre_cliente, tipo_accion, fecha_modificacion)
    VALUES (OLD.id, CONCAT(NEW.nombres, ' ', NEW.apellidos), 'UPDATE', CURRENT_TIMESTAMP);
END;
//
DELIMITER ;
```

**Parámetros:** No aplica explícitamente.

**Salida/Retorno:** No aplica explícitamente.

Trigger `trg_log_cliente_delete`

**Descripción:** Registra las eliminaciones de registros en la tabla `Cliente` en la tabla de registro `Cliente_Log`.

```
-- Trigger para guardar registro de delete de la tabla de Cliente
DELIMITER //
CREATE TRIGGER trg_log_cliente_delete
AFTER DELETE ON Cliente
FOR EACH ROW
BEGIN
    INSERT INTO Cliente_Log (id_cliente, nombre_cliente, tipo_accion, fecha_modificacion)
    VALUES (OLD.id, CONCAT(OLD.nombres, ' ', OLD.apellidos), 'DELETE', CURRENT_TIMESTAMP);
END;
//
DELIMITER ;
```

**Parámetros**: No aplica explícitamente.

**Salida/Retorno:** No aplica explícitamente.

# Eventos

Este evento asegura que la tabla `cliente_backup` se actualice diariamente con los datos actuales de la tabla `Cliente`, proporcionando así un mecanismo automático para respaldar la información regularmente.

weekly_cliente_backup

**Descripción:**
Este evento MySQL está diseñado para realizar un backup de la tabla `Cliente` diariamente.

**Sintaxis:**

```
-- evento para realizar un backup de la tabla alquiler cada 1 hora
DELIMITER //
create event if not exists weekly_cliente_backup
on schedule every 1 day
do
begin
    truncate table cliente_backup;
    insert into cliente_backup(id,cedula,nombres,apellidos,direccion,ciudad_residencia, celular, correo_electronico)
    select id,cedula,nombres,apellidos,direccion,ciudad_residencia, celular, correo_electronico from Cliente;
end //
```

**Parámetros:**
- No aplica explícitamente, pero el evento está programado para ejecutarse diariamente (cada 1 día).

**Salida/Retorno:**
- No aplica explícitamente, ya que el evento realiza acciones en la base de datos sin un retorno de datos directo.

# Usuarios y Accesos

## Tabla de Usuarios

| Usuario | Acceso | Descripción | Comandos | Procedimientos |
|---|---|---|---|---|
| cliente | % | Usuario para todos los clientes | SELECT | ver_registros_alquiler()<br>VerVehiculosDisponiblesPorTipo() |
| empleado | % | Usuario para todos los empleados | SELECT<br>UPDATE<br>INSERT | modificar_alquiler()<br>ver_registros_alquiler()<br>Consultar_disponibilidad_vehiculos()<br>Agregar_Alquiler_nuevo()<br>agregar_cliente()<br>modificar_cliente() |