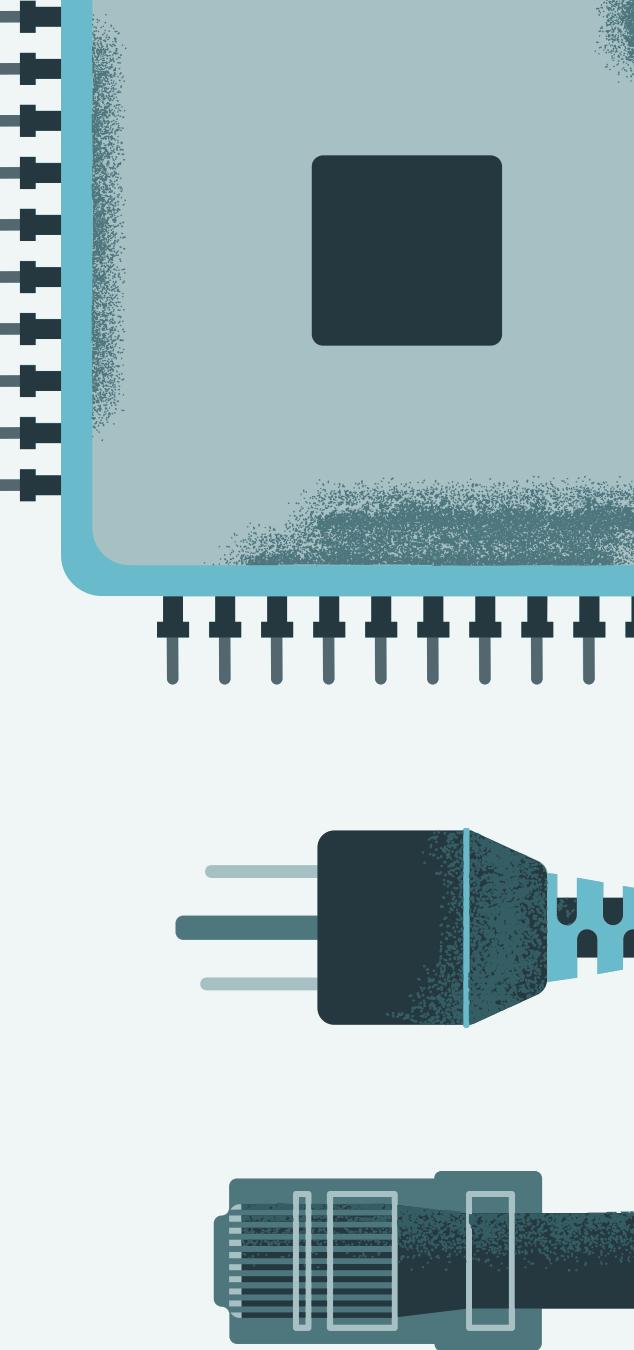
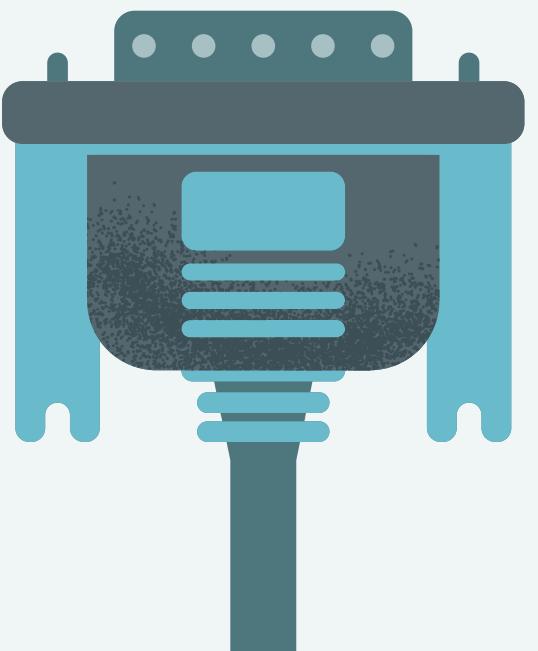
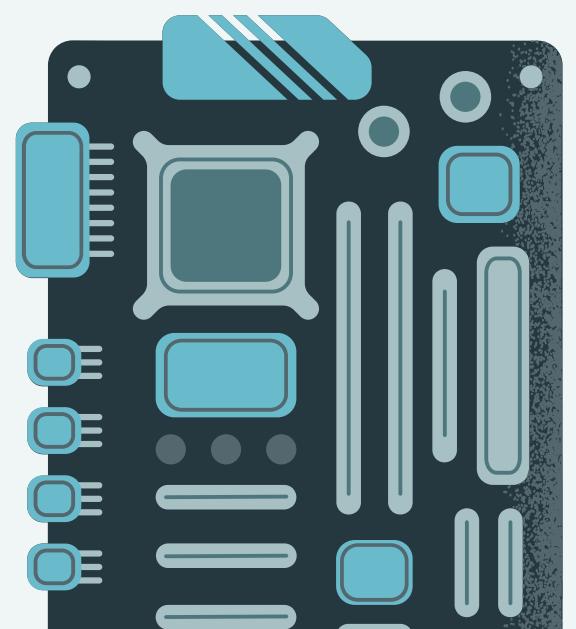
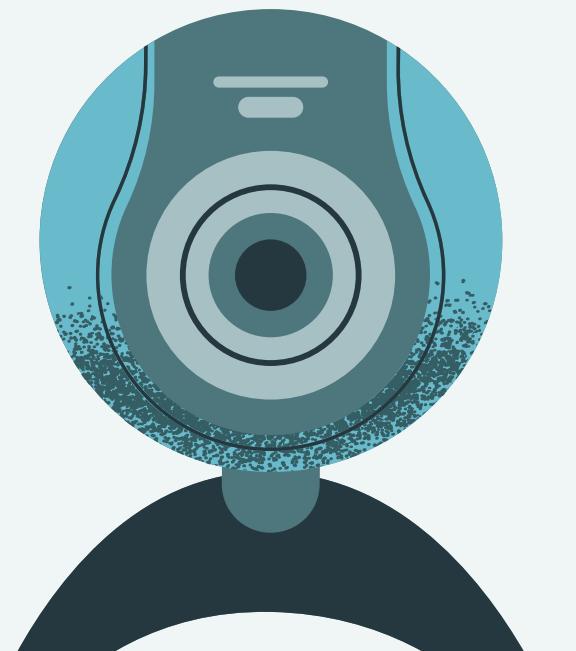
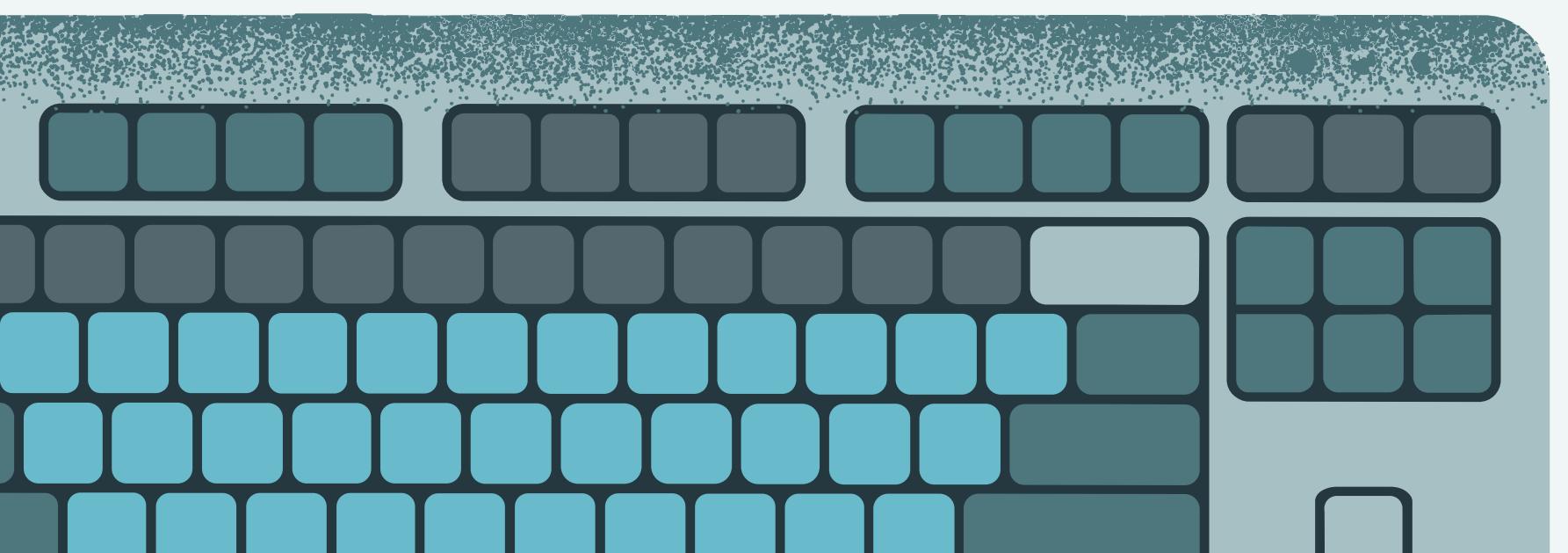


Proyecto final

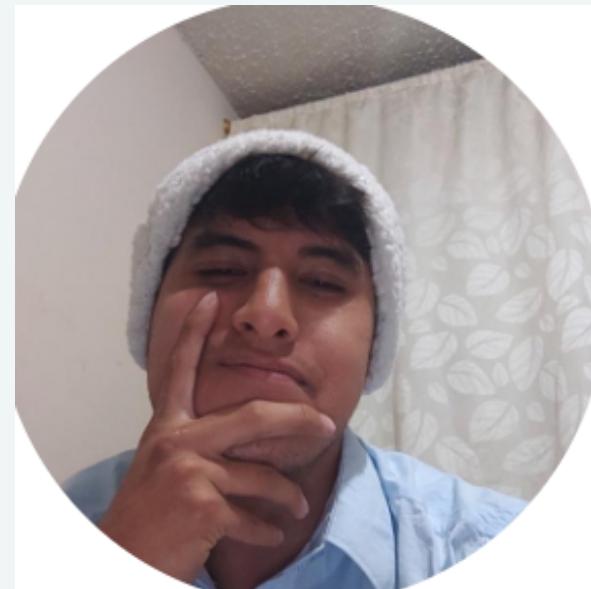
BASE DE DATOS RELACIONAL ACERCA DE UNA TERMINAL DE AUTOBUSES



INTEGRANTES



**JOSEPH
CHANGOLUIZA**



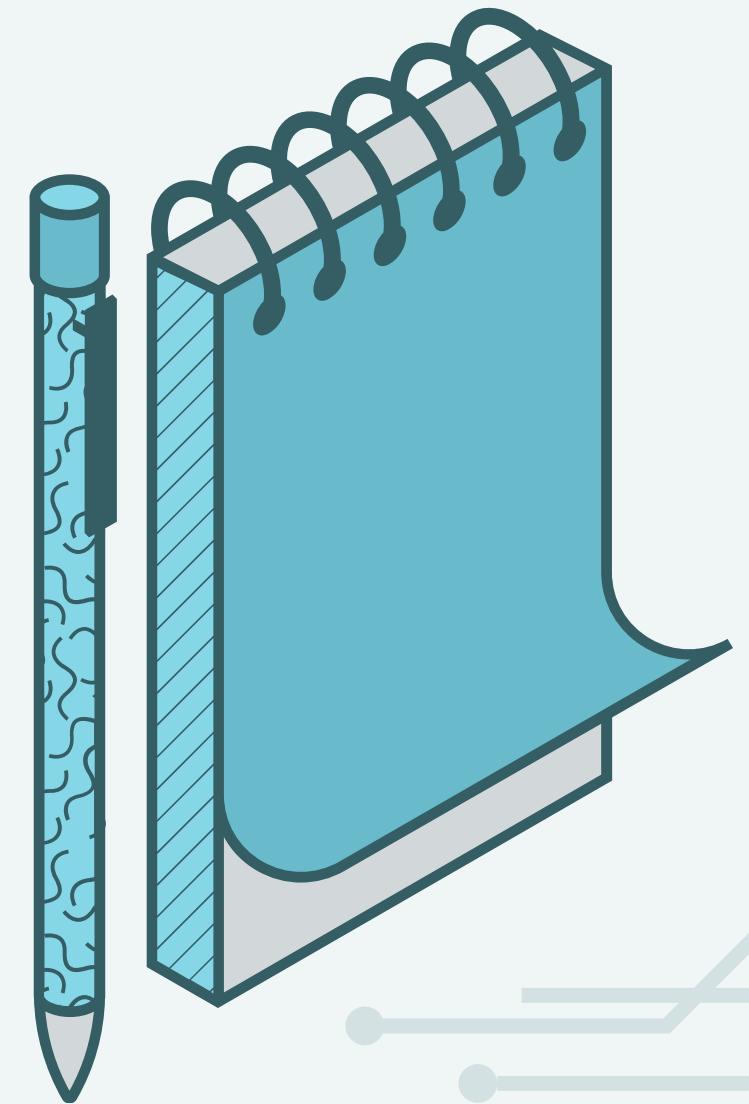
**JHON
LUGMAÑA**



**PATRICIO
PONCE**

INTRODUCCIÓN

El presente informe detalla el desarrollo e implementación de una base de datos para la gestión de una terminal de buses, empleando los conocimientos adquiridos a lo largo del semestre en el curso de Bases de Datos. El objetivo principal del proyecto fue diseñar y estructurar un sistema eficiente que permitiera almacenar, gestionar y consultar información de manera óptima, garantizando integridad, seguridad y rendimiento.

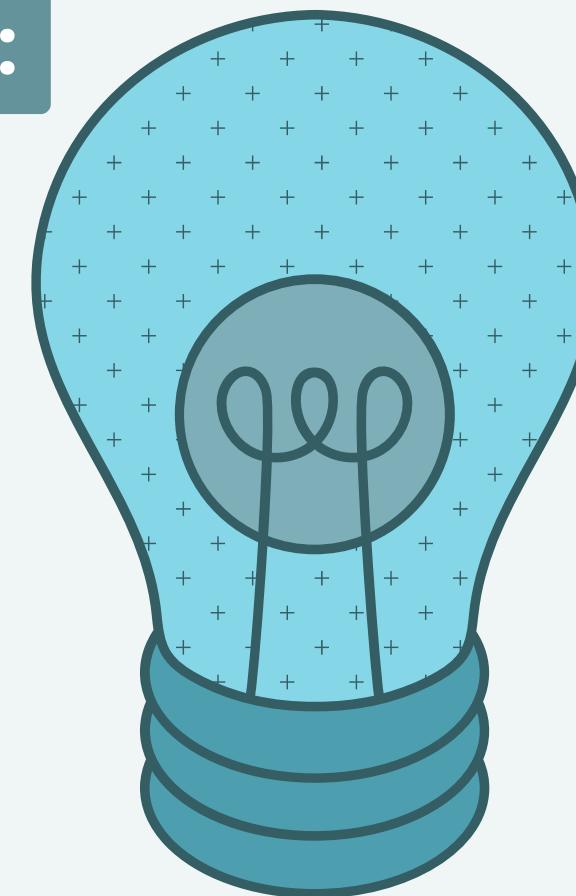


FASE DE PLANIFICACIÓN:

En esta etapa, se definió el alcance del proyecto, se definió el levantamiento de los requerimientos de nuestra base de datos con las principales entidades y sus relaciones.

FASE DE DISEÑO:

Luego se elaboró el Modelo Entidad-Relación (MER) para visualizar la estructura de la base de datos y sus relaciones



FASE DE SEGURIDAD Y OPTIMIZACIÓN:

Se establecieron mecanismos de control de acceso y privilegios para proteger la información.

FASE DE PRÁCTICAS

Se llevaron a cabo las prácticas que se verán a continuación de funcionalidad, rendimiento, automatización, seguridad, entre otras para validar el correcto funcionamiento del sistema.

FASE DE IMPLEMENTACIÓN:

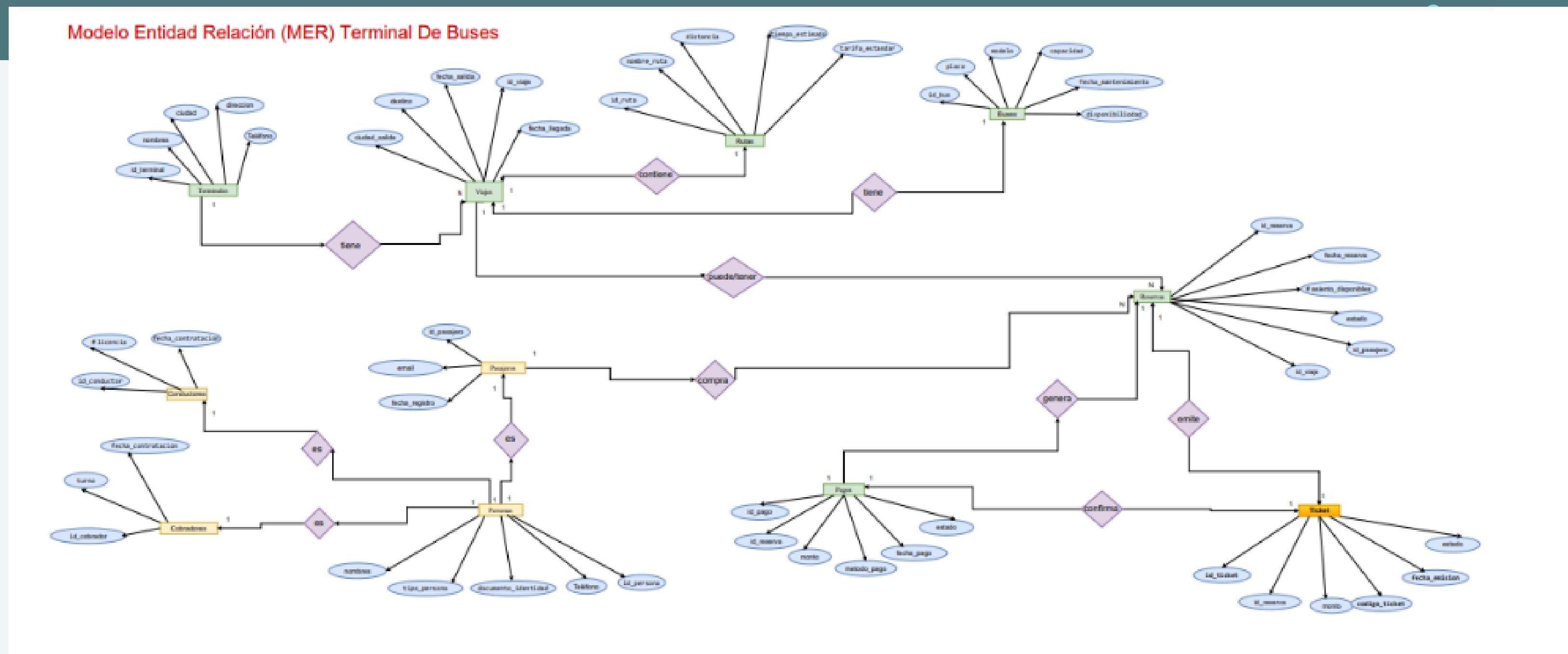
Con base en el diseño, se procedió a la creación de las tablas en MySQL, garantizando la integridad de los datos mediante el uso adecuado de claves primarias y foráneas.

DESCRIPCIÓN DE CADA FASE DEL PROYECTO

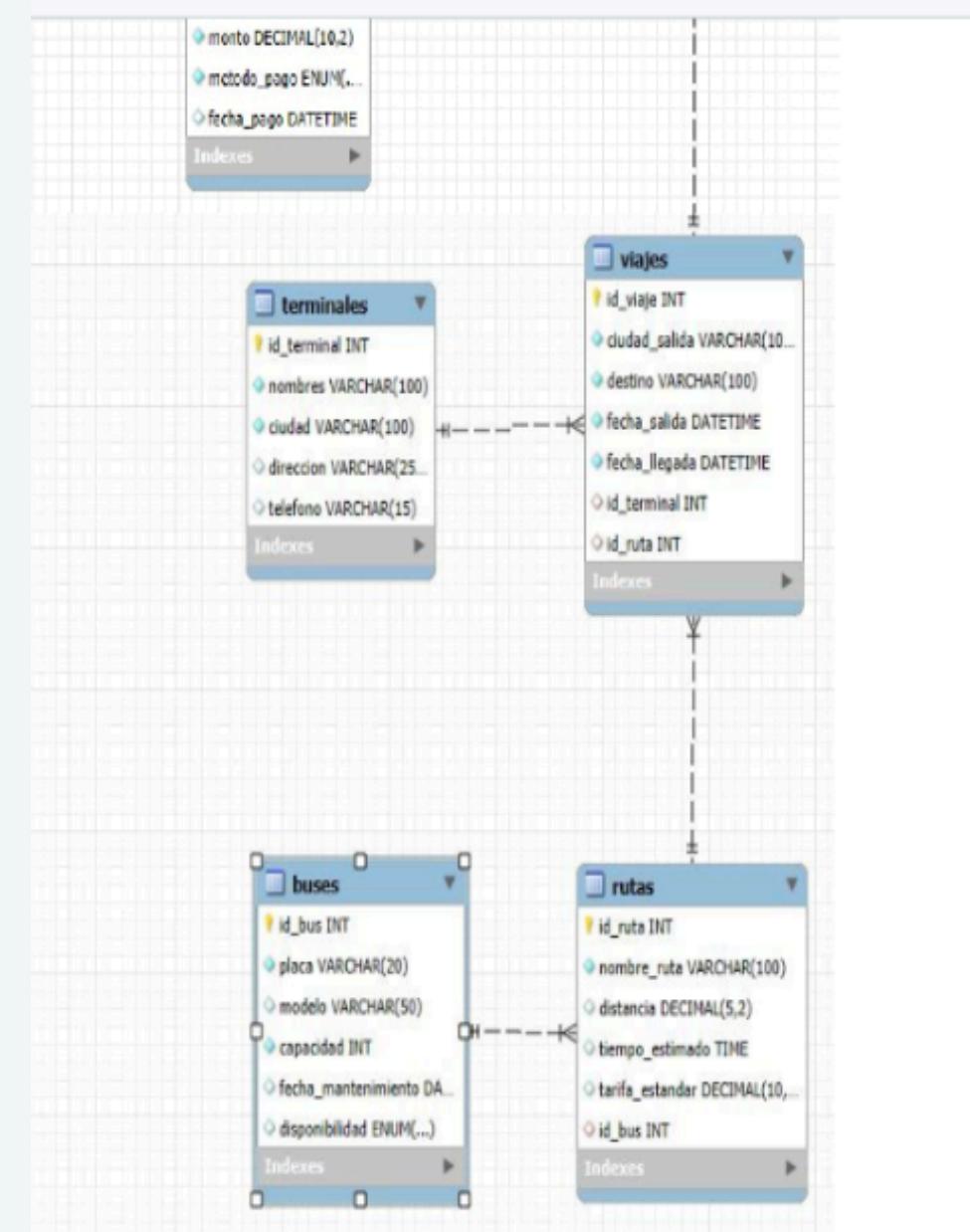
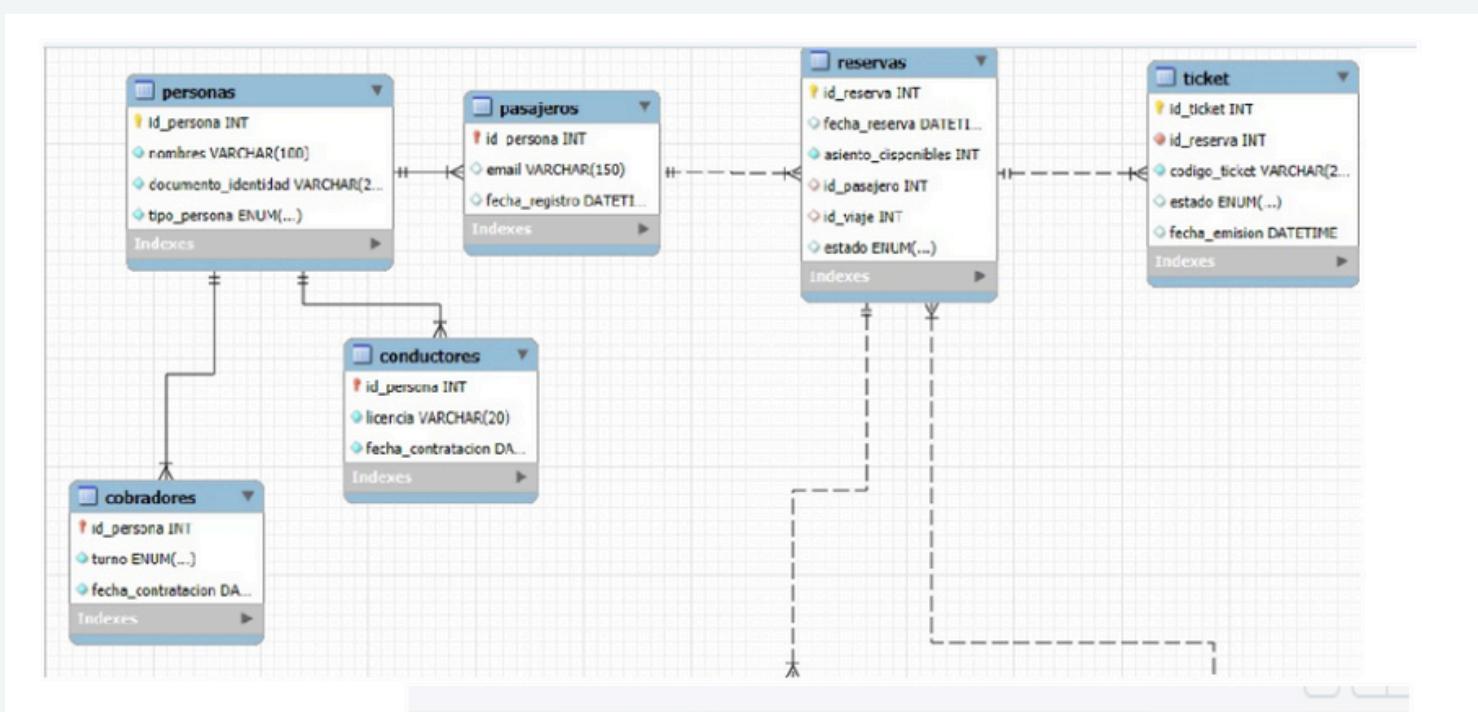
FASE DE DOCUMENTACIÓN Y PRESENTACIÓN:

Se elaboró un informe detallado con toda la información relevante del proyecto.

DISEÑAR EL MODELO CONCEPTUAL, LÓGICO Y FÍSICO.



DISEÑAR EL MODELO CONCEPTUAL, LÓGICO Y FÍSICO.



DESARROLLAR UN DICCIONARIO DE DATOS DETALLADO.

Alphabetic Index

- [buses](#)
- [cobradores](#)
- [conductores](#)
- [pagos](#)
- [pasajeros](#)
- [personas](#)
- [reservas](#)
- [rutas](#)
- [terminales](#)
- [ticket](#)
- [viajes](#)

buses

Column name	DataType	PK	NN	EQ	BIN	UN	ZF	AI	Default	Comment
id_bus	INT		✓					✓		Identificador unico del bus.
placa	VARCHAR(20)			✓						Placa del bus, debe ser unica.
modelo	VARCHAR(50)							NULL		Modelo del bus.
capacidad	INT			✓						Capacidad maxima de pasajeros del bus.
fecha_mantenimiento	DATE							NULL		Fecha del ultimo mantenimiento realizado.
disponibilidad	ENUM('Disponible', 'Mantenimiento')							'Disponible'		Estado del bus: Disponible o en mantenimiento.

DESARROLLAR UN DICCIONARIO DE DATOS DETALLADO.

cobradores										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_persona	INT	✓	✓							Identificador unico del cobrador, referencia a Personas.
turno	ENUM('Mañana', 'Tarde', 'Noche')		✓							Turno en el que trabaja el cobrador.
fecha_contratacion	DATE		✓							Fecha en la que fue contratado el cobrador.

conductores										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_persona	INT	✓	✓							Identificador unico del conductor, referencia a Personas.
licencia	VARCHAR(20)		✓							Numero de licencia de conducir.
fecha_contratacion	DATE		✓							Fecha en la que fue contratado el conductor.

pagos										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_pago	INT	✓	✓				✓			Identificador unico del pago.
id_reserva	INT		✓							Referencia a la reserva pagada.
monto	DECIMAL(10,2)		✓							Monto total del pago.
metodo_pago	ENUM('Efectivo', 'Tarjeta', 'Transferencia')		✓							Metodo de pago utilizado.
fecha_pago	DATETIME							CURRENT_TIMESTAMP		Fecha y hora en que se realizo el pago.

DESARROLLAR UN DICCIONARIO DE DATOS DETALLADO.

pasajeros										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_persona	INT	✓	✓							Identificador unico del pasajero, referencia a Personas.
email	VARCHAR(150)								NULL	Correo electronico del pasajero.
fecha_registro	DATETIME								CURRENT_TIMESTAMP	Fecha de registro del pasajero en el sistema.

personas										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_persona	INT	✓	✓						✓	Identificador unico de la persona.
nombres	VARCHAR(100)			✓						Nombre completo de la persona.
documento_identidad	VARCHAR(20)			✓						Numero de documento unico de identidad.
tipo_persona	ENUM('Pasajero', 'Conductor', 'Controlador')		✓							Tipo de persona dentro del sistema.

reservas										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_reserva	INT	✓	✓					✓		Identificador unico de la reserva.
fecha_reserva	DATETIME								CURRENT_TIMESTAMP	Fecha en la que se realizo la reserva.
asiento_disponibles	INT			✓						Numero de asientos disponibles en la reserva.
id_pasajero	INT								NULL	Identificador del pasajero que realiza la reserva.
id_viaje	INT								NULL	Identificador del viaje reservado.
estado	ENUM('Confirmada', 'Cancelada')								'Confirmada'	Estado de la reserva.

DESARROLLAR UN DICCIONARIO DE DATOS DETALLADO.

rutas										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_ruta	INT	✓	✓						✓	Identificador unico de la ruta.
nombre_ruta	VARCHAR(100)			✓						Nombre de la ruta.
distancia	DECIMAL(5,2)								NULL	Distancia en kilometros.
tiempo_estimado	TIME								NULL	Tiempo estimado del recorrido.
tarifa_estandar	DECIMAL(10,2)								NULL	Tarifa estandar de la ruta.
id_bus	INT								NULL	Identificador del bus asignado a la ruta.

terminales										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_terminal	INT	✓	✓						✓	Identificador unico del terminal.
nombres	VARCHAR(100)			✓						Nombre del terminal.
ciudad	VARCHAR(100)			✓						Ciudad donde se ubica el terminal.
direccion	VARCHAR(255)								NULL	Direccion exacta del terminal.
telefono	VARCHAR(15)								NULL	Numero de telefono del terminal.

ticket										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_ticket	INT	✓	✓						✓	Identificador unico del ticket.
id_reserva	INT			✓						Referencia a la reserva asociada.
codigo_ticket	VARCHAR(20)			✓						Código unico del ticket.
estado	ENUM('Emitido', 'Cancelado')							'Emitido'		Estado del ticket.
fecha_emision	DATETIME								CURRENT_TIMESTAMP	Fecha y hora de emision del ticket.

viajes										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id_viaje	INT	✓	✓						✓	Identificador unico del viaje.
ciudad_salida	VARCHAR(100)			✓						Ciudad de salida del viaje.
destino	VARCHAR(100)			✓						Destino del viaje.
fecha_salida	DATETIME			✓						Fecha y hora de salida.
fecha_llegada	DATETIME			✓						Fecha y hora de llegada.
id_terminal	INT								NULL	Identificador del terminal de salida.
id_ruta	INT								NULL	Identificador de la ruta asignada

PRÁCTICA: ESTABLECER CLAVES PRIMARIAS Y FORÁNEAS ENTRE LAS TABLAS, ASEGURANDO LA COHERENCIA DE LOS DATOS (POR EJEMPLO, CLIENTEID DEBE ESTAR PRESENTE EN LAS TABLAS RELACIONADAS).

-- Tabla Personas (Padre)

```
CREATE TABLE Personas (
    id_persona INT AUTO_INCREMENT PRIMARY KEY,
    nombres VARCHAR(100) NOT NULL,
    documento_identidad VARCHAR(20) UNIQUE NOT NULL,
    tipo_persona ENUM('Pasajero', 'Conductor', 'Controlador') NOT NULL
);
```

-- Tabla Pasajeros

```
CREATE TABLE Pasajeros (
    id_persona INT PRIMARY KEY,
    email VARCHAR(150) UNIQUE,
    fecha_registro DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_persona) REFERENCES Personas(id_persona)
);
```

-- Tabla Conductores

```
CREATE TABLE Conductores (
    id_persona INT PRIMARY KEY,
    licencia VARCHAR(20) NOT NULL,
    fecha_contratacion DATE NOT NULL,
    FOREIGN KEY (id_persona) REFERENCES Personas(id_persona)
);
```

-- Tabla Cobradores

```
CREATE TABLE Cobradores (
    id_persona INT PRIMARY KEY,
    turno ENUM('Mañana', 'Tarde', 'Noche') NOT NULL,
    fecha_contratacion DATE NOT NULL,
    FOREIGN KEY (id_persona) REFERENCES Personas(id_persona)
);
```

-- Tabla Terminales

```
CREATE TABLE Terminales (
    id_terminal INT AUTO_INCREMENT PRIMARY KEY,
    nombres VARCHAR(100) NOT NULL,
    ciudad VARCHAR(100) NOT NULL,
    direccion VARCHAR(255),
    telefono VARCHAR(15)
);
```

-- Tabla Buses

```
CREATE TABLE Buses (
    id_bus INT AUTO_INCREMENT PRIMARY KEY,
    placa VARCHAR(20) UNIQUE NOT NULL,
    modelo VARCHAR(50),
    capacidad INT NOT NULL,
    fecha_mantenimiento DATE,
    disponibilidad ENUM('Disponible', 'Mantenimiento') DEFAULT 'Disponible'
);
```

-- Tabla Conductores

```
CREATE TABLE Conductores (
    id_persona INT PRIMARY KEY,
    licencia VARCHAR(20) NOT NULL,
    fecha_contratacion DATE NOT NULL,
    FOREIGN KEY (id_persona) REFERENCES Personas(id_persona)
);
```

-- Tabla Cobradores

```
CREATE TABLE Cobradores (
    id_persona INT PRIMARY KEY,
    turno ENUM('Mañana', 'Tarde', 'Noche') NOT NULL,
    fecha_contratacion DATE NOT NULL,
    FOREIGN KEY (id_persona) REFERENCES Personas(id_persona)
);
```

-- Tabla Rutas

```
CREATE TABLE Rutas (
    id_ruta INT AUTO_INCREMENT PRIMARY KEY,
    nombre_ruta VARCHAR(100) NOT NULL,
    distancia DECIMAL(5, 2),
    tiempo_estimado TIME,
    tarifa_estandar DECIMAL(10, 2),
    id_bus INT,
    FOREIGN KEY (id_bus) REFERENCES Buses(id_bus)
);
```

-- Tabla Viajes

```
CREATE TABLE Viajes (
    id_viaje INT AUTO_INCREMENT PRIMARY KEY,
    ciudad_salida VARCHAR(100) NOT NULL,
    destino VARCHAR(100) NOT NULL,
    fecha_salida DATETIME NOT NULL,
    fecha_llegada DATETIME NOT NULL,
    id_terminal INT,
    id_ruta INT,
    FOREIGN KEY (id_terminal) REFERENCES Terminales(id_terminal),
    FOREIGN KEY (id_ruta) REFERENCES Rutas(id_ruta)
);
```

-- Tabla Reservas

```
CREATE TABLE Reservas (
    id_reserva INT AUTO_INCREMENT PRIMARY KEY,
    fecha_reserva DATETIME DEFAULT CURRENT_TIMESTAMP,
    asiento_disponibles INT NOT NULL,
    id_pasajero INT,
    id_viaje INT,
    estado ENUM('Confirmada', 'Cancelada') DEFAULT 'Confirmada',
    FOREIGN KEY (id_pasajero) REFERENCES Pasajeros(id_persona),
    FOREIGN KEY (id_viaje) REFERENCES Viajes(id_viaje)
);
```

-- Tabla Pagos

```
CREATE TABLE Pagos (
    id_pago INT AUTO_INCREMENT PRIMARY KEY,
    id_reserva INT NOT NULL,
    monto DECIMAL(10, 2) NOT NULL,
    metodo_pago ENUM('Efectivo', 'Tarjeta', 'Transferencia') NOT NULL,
    fecha_pago DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_reserva) REFERENCES Reservas(id_reserva)
);
```

-- Tabla Ticket

```
CREATE TABLE Ticket (
    id_ticket INT AUTO_INCREMENT PRIMARY KEY,
    id_reserva INT NOT NULL,
    codigo_ticket VARCHAR(20) UNIQUE NOT NULL,
    estado ENUM('Emitido', 'Cancelado') DEFAULT 'Emitido',
    fecha_emision DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_reserva) REFERENCES Reservas(id_reserva)
);
```

SEGURIDAD, AUDITORÍA Y CONTROL DE ACCESO

Práctica: Crear roles y permisos de usuario (por ejemplo, roles de Administrador, Usuario, Auditor) para controlar el acceso a las tablas y vistas.

```
-- Crear los roles
CREATE ROLE 'Administrador';
CREATE ROLE 'Usuario';

-- Asignar permisos al rol 'Administrador' (acceso total)
GRANT ALL PRIVILEGES ON terminal_buses.* TO 'Administrador'@'%';

-- Asignar permisos al rol 'Usuario' (acceso limitado a Reservas, Pagos y Ticket)
GRANT SELECT, INSERT, UPDATE ON terminal_buses.Reservas TO 'Usuario'@'%';
GRANT SELECT, INSERT ON terminal_buses.Pagos TO 'Usuario'@'%';
GRANT SELECT, UPDATE ON terminal_buses.Ticket TO 'Usuario'@'%';

-- Crear usuarios y asignarles roles
-- Administrador
CREATE USER 'admin_user'@'%' IDENTIFIED BY '1222';
GRANT 'Administrador' TO 'admin_user'@'%';

-- Usuario
CREATE USER 'user'@'%' IDENTIFIED BY '14432';
GRANT 'Usuario' TO 'user'@'%';
```

SEGURIDAD, AUDITORÍA Y CONTROL DE ACCESO

Práctica: Cifrar información sensible como contraseñas y detalles de pago (por ejemplo, usando AES_ENCRYPT en MySQL).

```
-- Cifrar el documento de identidad de los pasajeros usando AES_ENCRYPT
SET SQL_SAFE_UPDATES = 0;

UPDATE Personas
SET documento_identidad = AES_ENCRYPT(documento_identidad, '1234567')
WHERE tipo_persona = 'Pasajero';
SET SQL_SAFE_UPDATES = 1;

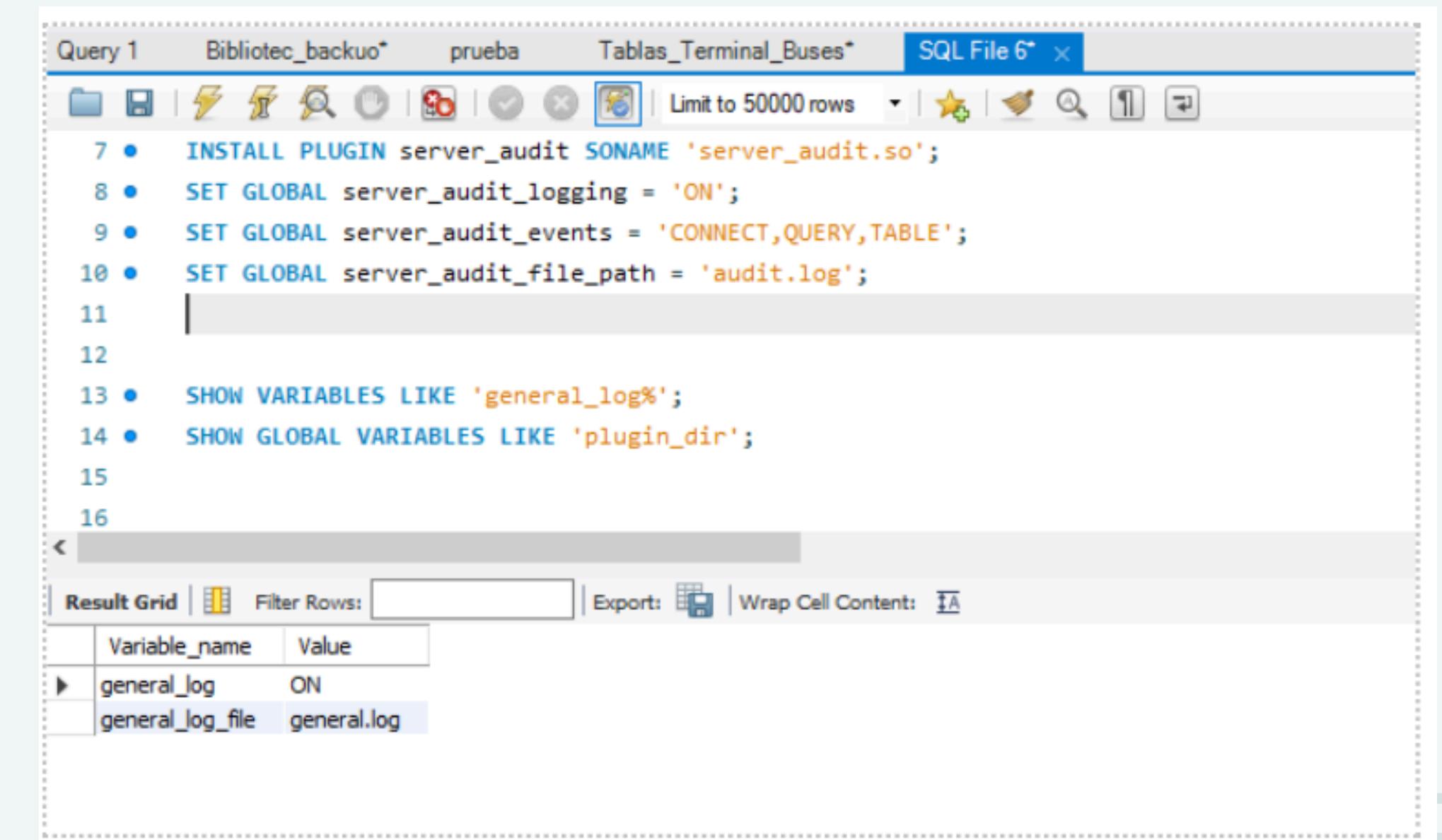
-- Cifrar detalles de pago, como el método de pago (por ejemplo, 'Efectivo', 'Tarjeta', 'Transferencia')
UPDATE Pagos
SET metodo_pago = AES_ENCRYPT(metodo_pago, '1233');

-- Desencriptar el documento de identidad del pasajero cuando sea necesario
SELECT id_persona, AES_DECRYPT(documento_identidad, 'clave_secreta') AS documento_identidad
FROM Personas
WHERE tipo_persona = 'Pasajero';

-- Desencriptar el método de pago si es necesario
SELECT id_pago, AES_DECRYPT(metodo_pago, 'clave_secreta') AS metodo_pago
FROM Pagos;
```

SEGURIDAD, AUDITORÍA Y CONTROL DE ACCESO

Práctica: Activar los logs de acceso y auditoría para monitorear las actividades de los usuarios (por ejemplo, registrar quién accedió a qué datos).



The screenshot shows a MySQL Workbench interface with several tabs at the top: Query 1, Bibliotec_backup*, prueba, Tablas_Terminal_Buses*, and SQL File 6*. Below the tabs is a toolbar with various icons. The main area contains the following SQL code:

```
7 • INSTALL PLUGIN server_audit SONAME 'server_audit.so';
8 • SET GLOBAL server_audit_logging = 'ON';
9 • SET GLOBAL server_audit_events = 'CONNECT,QUERY,TABLE';
10 • SET GLOBAL server_audit_file_path = 'audit.log';
11
12
13 • SHOW VARIABLES LIKE 'general_log%';
14 • SHOW GLOBAL VARIABLES LIKE 'plugin_dir';
15
16
```

Below the code is a Result Grid showing the configuration of the general log:

Variable_name	Value
general_log	ON
general_log_file	general.log

RESPALDOS Y RECUPERACIÓN DE DATOS

Práctica: Utilizar mysqldump o herramientas similares para hacer respaldos completos de la base de datos

```
Microsoft Windows [Versión 10.0.19045.5371]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jhonc>mysqldump -u root -p12345 terminal_buses
```

```
-u root -p12345 terminal_buses
```

RESPALDOS Y RECUPERACIÓN DE DATOS

Práctica: Realizar respaldos incrementales para reducir el tiempo y espacio de almacenamiento.

```
mysql> SHOW VARIABLES LIKE 'log_bin';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| log_bin       | ON    |
+-----+-----+
1 row in set, 1 warning (1.00 sec)

mysql>
```

RESPALDOS Y RECUPERACIÓN DE DATOS

Práctica: Hacer respaldos sin interrumpir el servicio (por ejemplo, usando Percona XtraBackup).

En Windows, Percona XtraBackup no tiene soporte oficial, pero puedes hacer un respaldo en caliente con las herramientas nativas de MySQL.

```
C:\Windows\system32>mysqldump -u root -p12345 --single-transaction --quick --lock-tables=false terminal_buses > "C:\Users\jhonc\OneDrive\Desktop\respaldo.sql"
```

```
mysqldump --single-transaction --quick --routines --triggers --events -u root -p mydatabase > backup.sql
```

UN RESPALDO EN CALIENTE ES NECESARIO ACTIVAR
EL LOG BINARIO

OPTIMIZACIÓN Y RENDIMIENTO DE CONSULTAS

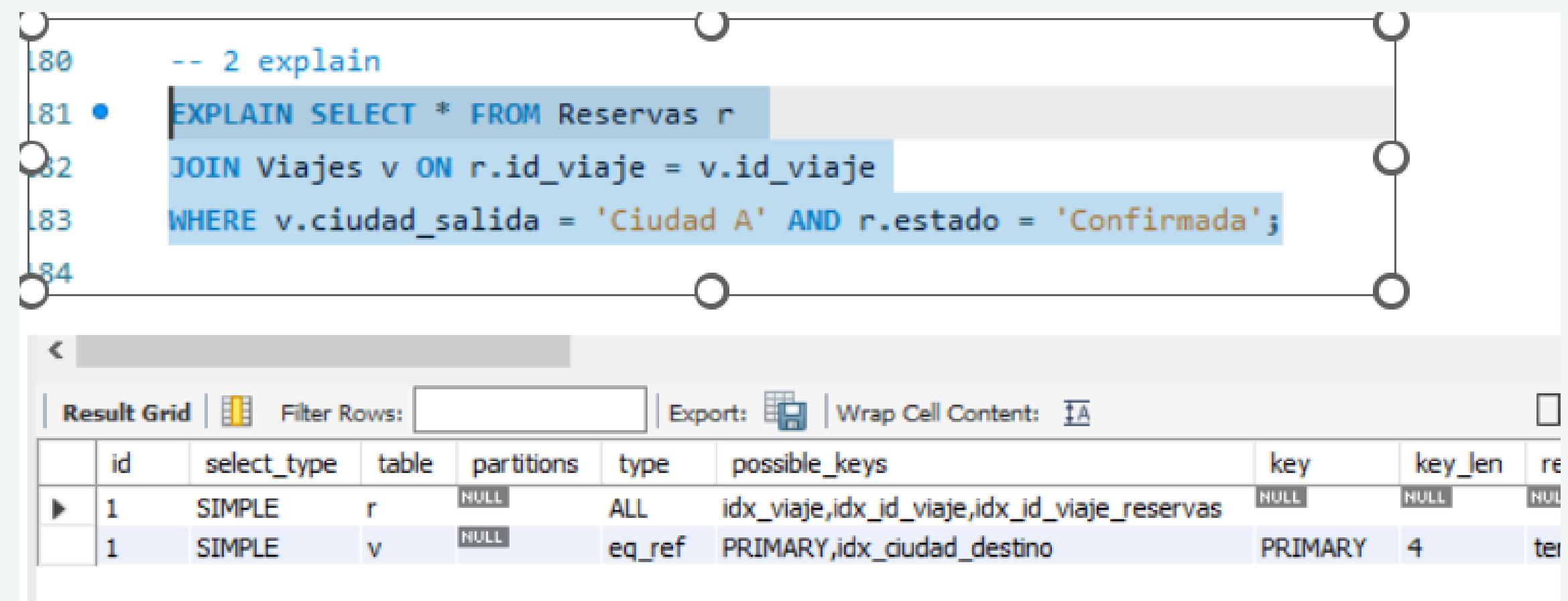
Práctica:

Implementar índices en las columnas más consultadas, como VueloID, ClientelID, etc.

```
170      -- crear index
171 •  CREATE INDEX idx_pasajero ON Reservas(id_pasajero);
172 •  CREATE INDEX idx_viaje ON Reservas(id_viaje);
173
174 •  CREATE INDEX idx_ciudad_destino ON Viajes(ciudad_salida, destino);
175
176 •  CREATE INDEX idx_placa ON Buses(placa);
177
178 •  CREATE INDEX idx_reserva_pago ON Pagos(id_reserva);
179
```

OPTIMIZACIÓN Y RENDIMIENTO DE CONSULTAS

Práctica: Utilizar herramientas como EXPLAIN para identificar cuellos de botella en las consultas y optimizarlas.



The screenshot shows a MySQL Workbench interface. In the top-left pane, there is a code editor with the following SQL query:

```
-- 2 explain
EXPLAIN SELECT * FROM Reservas r
JOIN Viajes v ON r.id_viaje = v.id_viaje
WHERE v.ciudad_salida = 'Ciudad A' AND r.estado = 'Confirmada';
```

In the bottom-right pane, there is a result grid showing the execution plan. The columns are: id, select_type, table, partitions, type, possible_keys, key, key_len, and ref. The data is as follows:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref
▶	1	SIMPLE	r	HULL	ALL	idx_viaje, idx_id_viaje, idx_id_viaje_reservas	NULL	NULL	NULL
	1	SIMPLE	v	HULL	eq_ref	PRIMARY, idx_ciudad_destino	PRIMARY	4	ter

OPTIMIZACIÓN Y RENDIMIENTO DE CONSULTAS

Práctica: Dividir tablas grandes, como Reservas, en particiones según una clave (por ejemplo, por fecha).

```
44 • CREATE TABLE Reservas_Particionada (
45   id_reserva INT AUTO_INCREMENT,
46   fecha_reserva DATETIME DEFAULT CURRENT_TIMESTAMP,
47   asiento_disponibles INT NOT NULL,
48   id_pasajero INT,
49   id_viaje INT,
50   estado ENUM('Confirmada', 'Cancelada') DEFAULT 'Confirmada',
51   PRIMARY KEY (id_reserva, fecha_reserva) -- Agregamos fecha_reserva a la clave primaria
52 )
53 PARTITION BY RANGE (YEAR(fecha_reserva)) (
54   PARTITION p_2021 VALUES LESS THAN (2022),
55   PARTITION p_2022 VALUES LESS THAN (2023),
56   PARTITION p_2023 VALUES LESS THAN (2024),
57   PARTITION p_2024 VALUES LESS THAN (2025)
58 );
59 ...
60
61 • EXPLAIN SELECT * FROM Reservas_Particionada WHERE YEAR(fecha_reserva) = 2022;
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref
1	1	SIMPLE	Reservas_Particionada	p_2021,p_2022,p_2023,p_2024	ALL	HULL	HULL	NULL	

PROCEDIMIENTOS ALMACENADOS, VISTAS Y TRIGGERS

Práctica: Crear un procedimiento para calcular el precio total de una reserva, aplicando descuentos y cargos adicionales.

```
CREATE TABLE DescuentosCargos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    tipo ENUM('Descuento', 'Cargo') NOT NULL,
    descripcion VARCHAR(255),
    porcentaje DECIMAL(5, 2) NOT NULL
);

DELIMITER //

CREATE PROCEDURE CalcularPrecioTotalReserva(
    IN id_reserva INT,
    OUT precio_total DECIMAL(10, 2),
    OUT descuento_total DECIMAL(10, 2),
    OUT cargo_total DECIMAL(10, 2)
);
```

PROCEDIMIENTOS ALMACENADOS, VISTAS Y TRIGGERS

```
DELIMITER //

CREATE PROCEDURE CalcularPrecioTotalReserva(
    IN id_reserva INT,
    OUT precio_total DECIMAL(10, 2),
    OUT descuento_total DECIMAL(10, 2),
    OUT cargo_total DECIMAL(10, 2)
)

BEGIN
    DECLARE precio_base DECIMAL(10, 2);

    SELECT monto INTO precio_base
    FROM Pagos
    WHERE Id_reserva = id_reserva
    LIMIT 1;

    SELECT IFNULL(SUM(porcentaje), 0) INTO descuento_total
    FROM DescuentosCargos
    WHERE tipo = 'Descuento';

    SELECT IFNULL(SUM(porcentaje), 0) INTO cargo_total
    FROM DescuentosCargos
    WHERE tipo = 'Cargo';

    SET precio_total = precio_base - (precio_base * (descuento_total / 100)) + (precio_base * (cargo_total / 100));
END //
```

PROCEDIMIENTOS ALMACENADOS, VISTAS Y TRIGGERS

```
39 •     CALL CalcularPrecioTotalReserva(5, @precio_total, @descuento_total, @cargo_total);
40 •     SELECT @precio_total AS PrecioTotal, @descuento_total AS DescuentoTotal, @cargo_total AS CargoTotal;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	PrecioTotal	DescuentoTotal	CargoTotal
▶	10.00	0.00	0.00

PROCEDIMIENTOS ALMACENADOS, VISTAS Y TRIGGERS

Crear vistas que presenten información de varias tablas de manera unificada

```
-- 1era Vista
CREATE VIEW VistaViajesPasajerosReservas AS
SELECT
    v.id_viaje,
    v.ciudad_salida,
    v.destino,
    v.fecha_salida,
    v.fecha_llegada,
    p.id_persona AS id_pasajero,
    p.nombres AS nombre_pasajero,
    p.documento_identidad,
    r.id_reserva,
    r.fecha_reserva,
    r.asiento_disponibles,
    r.estado
FROM
    Viajes v
JOIN
    Reservas r ON v.id_viaje = r.id_viaje
JOIN
    Pasajeros pa ON r.id_pasajero = pa.id_persona
JOIN
    Personas p ON pa.id_persona = p.id_persona;
```

```
27 •    SELECT * FROM VistaViajesPasajerosReservas;
```

```
28
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	id_viaje	ciudad_salida	destino	fecha_salida	fecha_llegada	id_pasajero	nombre_pasajero	documento_identidad	id_reserva	f
	10	Guayaquil	Riobamba	2023-01-10 17:00:00	2023-01-10 19:00:00	4	Ana Martinez	2233445566	10	20
	4	Quito	Ambato	2023-01-04 11:00:00	2023-01-04 13:00:00	10	Elena Diaz	8899001122	4	20
	8	Guayaquil	Loja	2023-01-08 15:00:00	2023-01-08 23:00:00	10	Elena Diaz	8899001122	8	20
	1	Quito	Guayaquil	2023-01-01 08:00:00	2023-01-01 14:00:00	1	Juan Perez	1234567890	1	20
	5	Quito	Riobamba	2023-01-05 12:00:00	2023-01-05 14:00:00	1	Juan Perez	1234567890	5	20
	6	Quito	Quito	2023-01-06 08:00:00	2023-01-06 10:00:00	1	Juan Perez	1234567890	6	20

PROCEDIMIENTOS ALMACENADOS, VISTAS Y TRIGGERS

Crear triggers que registren cambios en las tablas de Reservas y Pagos cada vez que un registro se actualiza o elimina.

```
1 • use terminal_buses;
2   -- Triggers
3
4 • drop trigger before_reservas_update;
5 • CREATE TABLE log_reservas_pagos (
6   id_auditoria INT AUTO_INCREMENT PRIMARY KEY,
7   tabla VARCHAR(50),
8   operacion ENUM('UPDATE', 'DELETE'),
9   id_registro INT,
10  fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
11  usuario VARCHAR(50),
12  detalles TEXT
13 );
```

```
15    DELIMITER $$  
16 • CREATE TRIGGER before_reservas_update  
17     BEFORE UPDATE ON Reservas  
18     FOR EACH ROW  
19     BEGIN  
20         INSERT INTO log_reservas_pagos (tabla, operacion, id_registro, usuario, detalles)  
21             VALUES ('Reservas', 'UPDATE', OLD.id_reserva, USER(),  
22                         CONCAT('Antes: ', OLD.estado, ', Después: ', NEW.estado));  
23     END$$  
24     DELIMITER ;
```

```
DELIMITER $$  
CREATE TRIGGER before_reservas_delete  
BEFORE DELETE ON Reservas  
FOR EACH ROW  
BEGIN  
    INSERT INTO log_reservas_pagos (tabla, operacion, id_registro, usuario, detalles)  
        VALUES ('Reservas', 'DELETE', OLD.id_reserva, USER(),  
                CONCAT('Estado: ', OLD.estado));  
END$$  
DELIMITER ;
```

```
DELIMITER $$  
• CREATE TRIGGER before_pagos_update  
  BEFORE UPDATE ON Pagos  
  FOR EACH ROW  
  BEGIN  
    INSERT INTO log_reservas_pagos (tabla, operacion, id_registro, usuario, detalles)  
    VALUES ('Pagos', 'UPDATE', OLD.id_pago, USER(), CONCAT('Antes: ', OLD.monto, ', Después: ', NEW.monto));  
  END $$  
DELIMITER ;
```

```
DELIMITER $$  
• CREATE TRIGGER before_pagos_delete  
  BEFORE DELETE ON Pagos  
  FOR EACH ROW  
  BEGIN  
    INSERT INTO log_reservas_pagos (tabla, operacion, id_registro, usuario, detalles)  
    VALUES ('Pagos', 'DELETE', OLD.id_pago, USER(), CONCAT('Monto: ', OLD.monto));  
  END $$
```

```
57 •     INSERT INTO log_reservas_pagos (tabla, operacion, id_registro, usuario, detalles)
58     VALUES ('Reservas', 'UPDATE', 123, 'usuario_prueba', 'Antes: Activo, Despu s: Cancelado');
59
```

```
60 • INSERT INTO log_reservas_pagos (tabla, operacion, id_registro, usuario, detalles)
61     VALUES ('Pagos', 'DELETE', 456, 'usuario_prueba', 'Monto: 100.00');
62
63 • UPDATE log_reservas_pagos
64     SET detalles = 'Antes: Inactivo, Despu s: Activo'
65     WHERE id_auditoria = 1;
```

MONITOREO Y OPTIMIZACIÓN DE RECURSOS

Práctica: Usar herramientas como SHOW PROCESSLIST para detectar consultas lentas y optimizarlas.

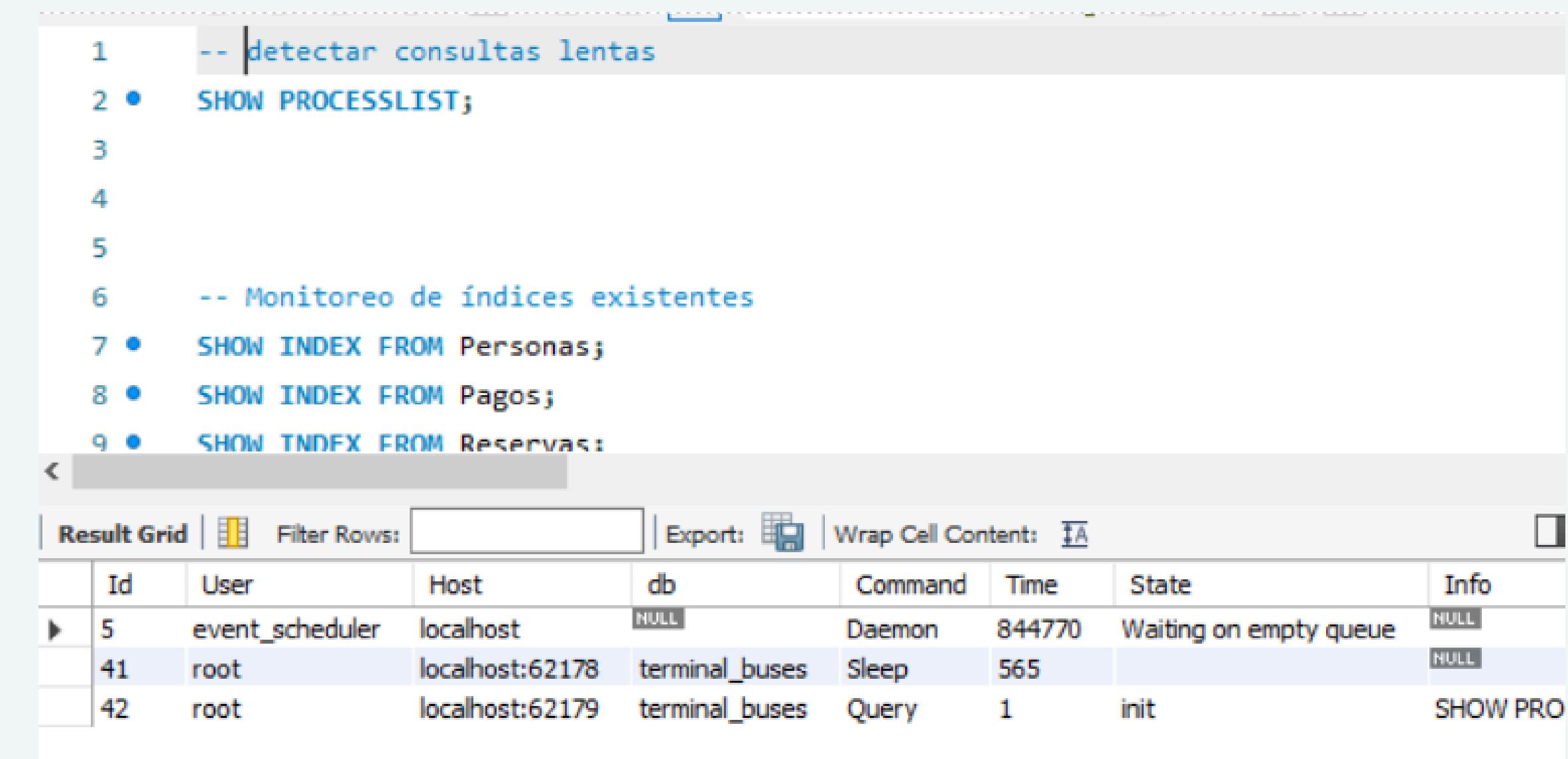
```
1 -- detectar consultas lentas
2 • SHOW PROCESSLIST;
3
4
5
6 -- Monitoreo de índices existentes
7 • SHOW INDEX FROM Personas;
8 • SHOW INDEX FROM Pagos;
9 • SHOW INDEX FROM Reservas;
```

Result Grid | Filter Rows: Export: Wrap Cell Content: □

	ID	User	Host	db	Command	Time	State	Info
▶	5	event_scheduler	localhost	NUL	Daemon	844770	Waiting on empty queue	NULL
	41	root	localhost:62178	terminal_buses	Sleep	565		NULL
	42	root	localhost:62179	terminal_buses	Query	1	init	SHOW PRO

MONITOREO Y OPTIMIZACIÓN DE RECURSOS

Práctica: Usar herramientas como SHOW PROCESSLIST para detectar consultas lentas y optimizarlas.



The screenshot shows a MySQL Workbench interface. The query editor contains the following SQL code:

```
1 -- detectar consultas lentas
2 • SHOW PROCESSLIST;
3
4
5
6 -- Monitoreo de índices existentes
7 • SHOW INDEX FROM Personas;
8 • SHOW INDEX FROM Pagos;
9 • SHOW INDEX FROM Reservas;
```

The result grid displays the following process list:

	Id	User	Host	db	Command	Time	State	Info
▶	5	event_scheduler	localhost	HULL	Daemon	844770	Waiting on empty queue	NULL
	41	root	localhost:62178	terminal_buses	Sleep	565		NULL
	42	root	localhost:62179	terminal_buses	Query	1	init	SHOW PRO

- ▶ `ALTER TABLE Viajes DROP INDEX idx_ciudad_salida ;`
- ▶ `ALTER TABLE Viajes ADD INDEX idx_ciudad_salida (ciudad_salida);`

MONITOREO Y OPTIMIZACIÓN DE RECURSOS

Práctica: Simular múltiples usuarios concurrentes usando herramientas como Apache JMeter para ver cómo responde la base de datos bajo alta carga.

JAVA ESTE INSTALADO EN WINDOWS

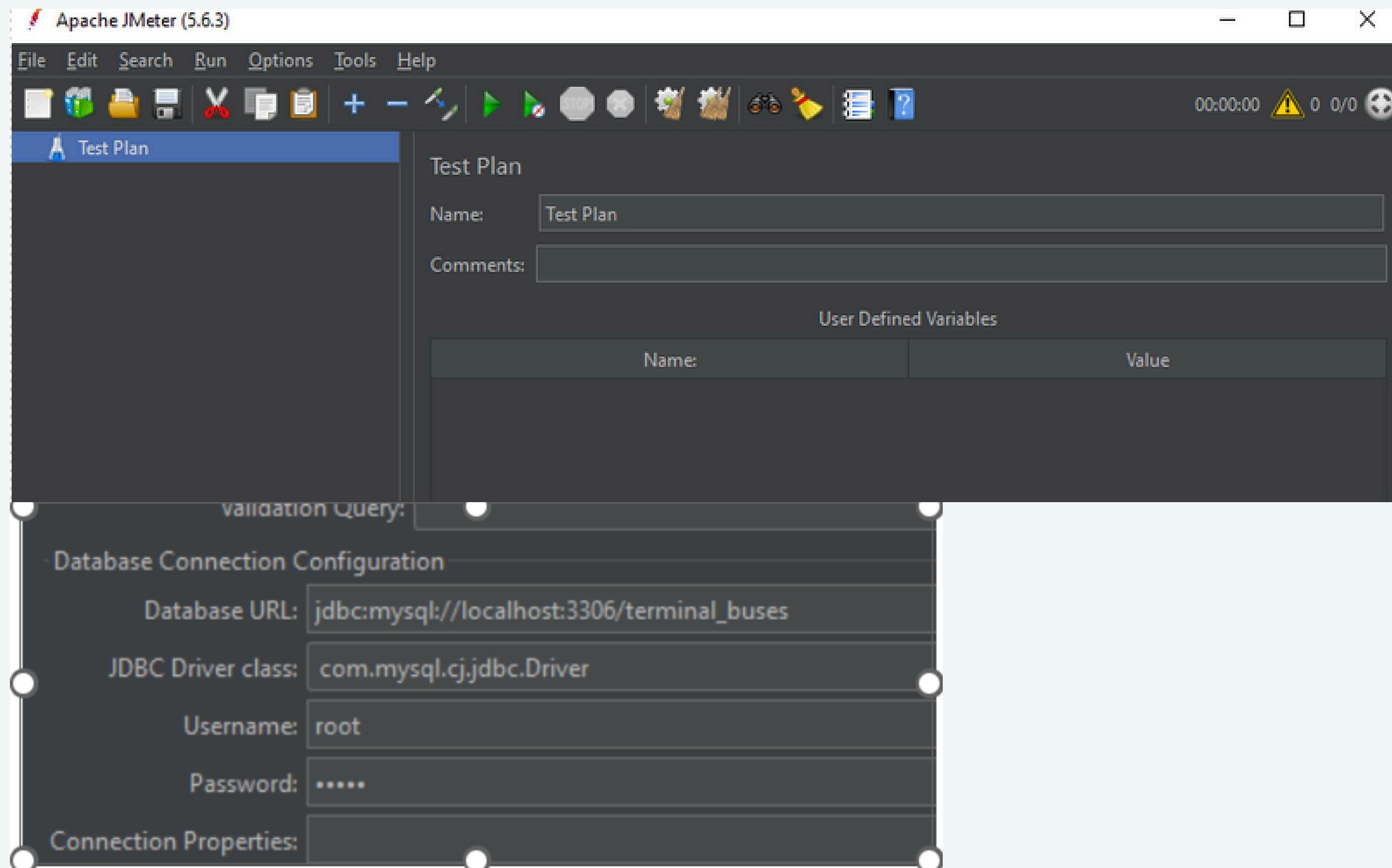
```
C:\Users\jhonc>cd C:\Users\jhonc\OneDrive\Desktop\proyecto de base de datos\apache-jmeter-5.6.3\bin  
C:\Users\jhonc\OneDrive\Desktop\proyecto de base de datos\apache-jmeter-5.6.3\bin>jmeter.bat
```

Conector/J 9.2.0

Seleccione sistema operativo:

Seleccione sistema operativo..

MONITOREO Y OPTIMIZACIÓN DE RECURSOS



Crear un "Test Plan"---
Agregar la Configuración de Conexión JDBC--- Agregar el "JDBC Request"

MONITOREO Y OPTIMIZACIÓN DE RECURSOS

Text

JDBC Request

JDBC Request

JDBC Request

Response data						
Response Body				Response headers		
id_reserva	fecha_reserva	asiento_disponibles	id_viaje	id_pasajero	id_viaje	estado
id_viaje	ciudad_salida	destino		fecha_salida	fecha_llegada	id_terminal
id_ruta						
1	2025-02-01T17:03:30	40	2	1	Confirmada	
	1	Ciudad A	Ciudad B	2023-02-10T10:00	2023-02-10T1	
3:00	1	1				

Agregar un Listener para Ver los Resultados:

MONITOREO Y OPTIMIZACIÓN DE RECURSOS

Práctica: Identificar índices no utilizados y eliminarlos para liberar recursos y mejorar la velocidad de las operaciones de escritura.

Verificar los índices creados en cada tabla:

```
5  
6 • SHOW INDEX FROM Reservas;  
7 • SHOW INDEX FROM Viajes;  
8 • SHOW INDEX FROM Buses;  
9 • SHOW INDEX FROM Pagos;
```

10

Eliminar Índices Innecesarios

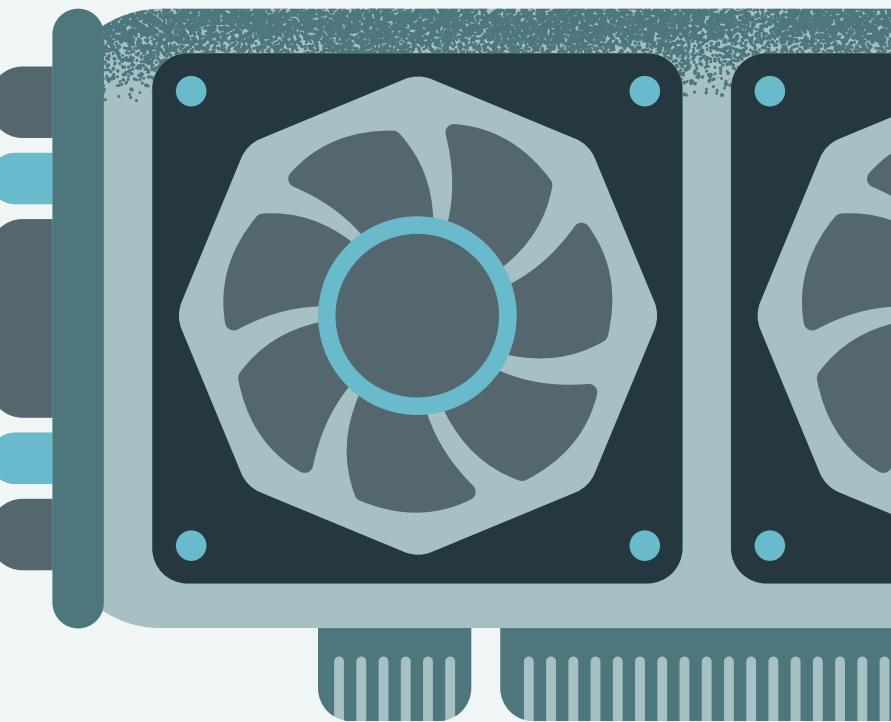
Si identificas un índice que **no se usa**, elimínalo con:

```
• DROP INDEX idx_pasajero ON Reservas;
```

CONCLUSIONES

El desarrollo de este proyecto permitió consolidar los conocimientos adquiridos en el semestre, aplicándolos en un caso práctico que simula una situación real.

Además, se evidenció el impacto de la seguridad y la eficiencia en la administración de la información. Finalmente, se concluye que el trabajo en equipo y la asignación de responsabilidades fueron clave para el éxito del proyecto.



**Y COMO DIJO MI EX
(TERMINAMOS)**

