



# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS



### BASES DE DATOS (TSDS)

ASIGNATURA:

Bases de Datos

PROFESOR:

Ing. Lorena Chulde

FECHA DE ENTREGA:

2025 - 02-05

PERÍODO ACADÉMICO:

2024-B

## TÍTULO

### PROYECTO FINAL:

**Base de Datos Relacional acerca de una terminal de autobuses**



### Estudiantes

**Joseph Changoluisa**

**Jhonn Lugmaña**

**Patricio Ponce**

### Resumen Ejecutivo

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. Para ello, se siguió un enfoque metodológico que incluyó la planificación, modelado, normalización y optimización de los datos. Como resultado, se obtuvo un sistema robusto que facilita la administración de rutas, conductores, tickets y usuarios.

## **Descripción de Cada Fase del Proyecto**

### **1. 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. Además, se establecieron los roles y responsabilidades dentro del equipo de trabajo, asegurando una distribución equitativa de las tareas.

### **2. Fase de Diseño:**

Luego se elaboró el Modelo Entidad-Relación (MER) para visualizar la estructura de la base de datos y sus relaciones. Posteriormente, se diseñó el modelo lógico y físico, asegurando una correcta normalización de los datos para evitar redundancias e inconsistencias.

### **3. 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. Se implementaron procedimientos almacenados, vistas y triggers para optimizar el rendimiento y facilitar la gestión de la base de datos.

### **4. Fase de Seguridad y Optimización:**

Se establecieron mecanismos de control de acceso y privilegios para proteger la información. Además, se realizaron ajustes en los índices y consultas para mejorar la eficiencia en la recuperación de datos.

### **5. 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. Se verificó la integridad de los datos y se realizaron ajustes según los resultados obtenidos.

### **6. Fase de Documentación y Presentación:**

Se elaboró un informe detallado con toda la información relevante del proyecto. Además, se llevó a cabo una presentación para exponer los resultados obtenidos y las mejoras implementadas

## **Requerimientos**

Como grupo hemos levantado los requerimientos que explican una base de datos de una terminal de buses: La base de datos debe incluir una tabla de Pagos que contenga los siguientes campos: un identificador único del pago (id\_pago), el identificador de la reserva asociada (id\_reserva), el monto del pago (monto) y el método utilizado para el pago (metodo\_pago).

En la tabla de Reservas, se deben registrar el identificador único de la reserva (id\_reserva), la fecha en que se realizó la reserva (fecha\_reserva), el número de asientos disponibles (asiento\_disponibles), el identificador del pasajero (id\_pasajero), el identificador del viaje (id\_viaje), el estado de la reserva (estado) y la fecha del pago de la reserva (fecha\_pago).

La tabla de Viajes debe incluir el identificador único del viaje (id\_viaje), la ciudad de salida (ciudad\_salida), la ciudad de destino (destino), la fecha de salida (fecha\_salida), la fecha de llegada (fecha\_llegada) y el identificador de la terminal (id\_terminal).

Para las Terminales, se deben registrar el identificador único de la terminal (id\_terminal), el nombre de la terminal (nombres), la ciudad donde se encuentra la terminal (ciudad), la dirección de la terminal (direccion) y el teléfono de contacto de la terminal (telefono).

En la tabla de Rutas, se deben incluir el identificador único de la ruta (id\_ruta), el nombre de la ruta (nombre\_ruta), la distancia de la ruta (distancia), el tiempo estimado del viaje (tiempo\_estimado), la tarifa estándar de la ruta (tarifa\_estandar) y el identificador del bus asignado a la ruta (id\_bus).

La tabla de Buses debe contener el identificador único del bus (id\_bus), la placa del bus (placa), el modelo del bus (modelo), la capacidad del bus (capacidad), la fecha del último mantenimiento (fecha\_mantenimiento) y la disponibilidad del bus (disponibilidad).

Para los Cobradores, se deben registrar el identificador único del cobrador (id\_cobrador), el turno de trabajo del cobrador (turno) y la fecha de contratación del cobrador (fecha\_contratacion).

En la tabla de Personas, se deben incluir el identificador único de la persona (id\_persona), los nombres de la persona (nombres), el documento de identidad (documento\_identidad) y el tipo de persona (tipo\_persona), que puede ser pasajero, conductor, etc.

La tabla de Pasajeros debe contener el identificador único del pasajero (id\_pasajero), el correo electrónico del pasajero (email) y la fecha de registro del pasajero (fecha\_registro).

Para los Conductores, se deben registrar el identificador único del conductor (id\_conductor), el número de licencia del conductor (licencia), la fecha de contratación del conductor (fecha\_contratacion), el teléfono de contacto del conductor (telefono) y los nombres del conductor (nombres).

Finalmente, la tabla de Tickets debe incluir el identificador único del ticket (id\_ticket), el identificador de la reserva asociada (id\_reserva), el monto del ticket (monto), el código

del ticket (codigo\_ticket), el estado del ticket (estado) y la fecha de emisión del ticket (fecha\_emision).

## Replicación de Bases de Datos:

**Replicación Maestro-Esclavo:** Un servidor maestro gestiona las escrituras y propaga los cambios a varios servidores de solo lectura (esclavos). Esto asegura alta disponibilidad y redundancia [1].

**Replicación Maestro-Maestro:** Varios servidores se encargan tanto de las lecturas como de las escrituras, proporcionando mayor disponibilidad y permitiendo la distribución de la carga de trabajo [1].

### **Escalabilidad Horizontal:**

Añadir más servidores para distribuir la carga, lo que permite manejar cargas de trabajo significativamente mayores mediante la difusión de los datos y las consultas a través de múltiples nodos [2].

### **Optimización de Consultas:**

Asegurarse de que las consultas SQL estén optimizadas para reducir el tiempo de respuesta y mejorar el rendimiento. Esto incluye el uso de índices adecuados y la reestructuración de consultas complejas [2].

### **Uso de Tecnologías NoSQL:**

Considerar el uso de bases de datos NoSQL como MongoDB o Cassandra para manejar grandes volúmenes de datos y proporcionar flexibilidad en la estructura de datos [3].

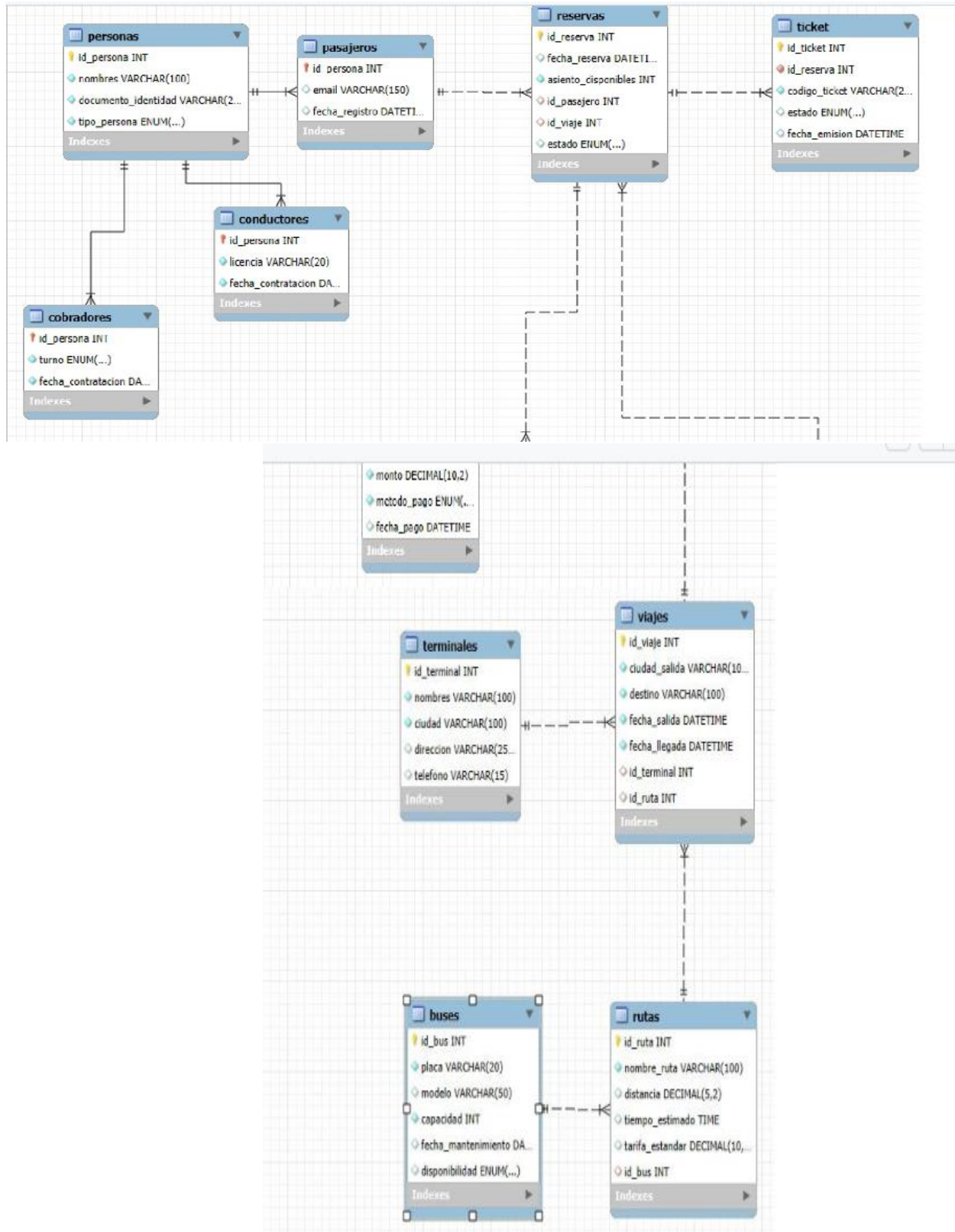
### **Monitoreo y Afinamiento del Rendimiento:**

Utilizar herramientas como Prometheus y Grafana para monitorear el rendimiento de la base de datos y realizar ajustes según sea necesario [2].

### **Estrategias de Caching:**

Implementar estrategias de caching con herramientas como Redis para reducir la carga en la base de datos y mejorar la velocidad de acceso a los datos [3].

**Importancia del Conocimiento:** Conocer cómo diseñar bases de datos adecuadas asegura la eficiencia y fácil mantenimiento de la aplicación.



Desarrollar un diccionario de datos detallado.

**Práctica:** Crear un diccionario que defina todas las tablas, sus campos, relaciones y restricciones.

2023-01-29

Alphabetic Index

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

buses

Column name	Data Type	PK	NN	UQ	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.

cobradores  
cobradores

Column name	Data Type	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	Data Type	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	Data Type	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.



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.		

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)		✓							Codigo 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		

Es un diccionario de datos para la base de datos "terminal\_buses". Describe las tablas, sus columnas y atributos en un formato tabular

**Investigación:** Investigar las mejores herramientas y métodos para generar diccionarios de datos y su uso en proyectos reales.

Entre las herramientas para generar diccionarios de datos podemos nombrar las siguientes:

MySQL Workbench: Esta herramienta permite generar automáticamente un diccionario de datos que resume la estructura de la base de datos. [5]

Oracle SQL Developer: Al gestionar diferentes bases de datos cuenta con funcionalidades específicas para generar diccionarios de datos [6].

ER/Studio Data Architect: Se trata de una herramienta de modelado de datos la cual permite generar el diseño, documentación y gestión de bases de datos [7].

Existen varios métodos para generar diccionarios de datos y son los siguientes:

Documentación Manual: Consiste en la creación manual de un diccionario de datos, se escriben todos los datos, uno por uno; de la base de datos para generar el diccionario [8].

Generación Automática: Con las herramientas previamente mencionadas, se puede generar automáticamente un diccionario de datos a partir del esquema de la base de datos [8].

Finalmente, estos diccionarios de datos son muy útiles en la vida real ya que, asegura que todas las personas que tengan acceso a una base de datos tengan la misma información. Además, proporciona una comunicación más fluida entre los administradores de las bases de datos y finalmente es importante para futuros procedimientos con las bases de datos como actualizaciones o migraciones.

**Importancia del Conocimiento:** Un diccionario de datos permite mantener la consistencia y facilita la colaboración entre desarrolladores.

**Definir las restricciones de integridad referencial.**

**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)  
);
```

---

**Investigación:** Investigar cómo la integridad referencial puede prevenir la pérdida de datos y mantener la consistencia.

La integridad referencial no es más que las relaciones entre tablas en una base de datos para que estas se mantengan coherentes y esto es posible gracias a las claves primarias y foráneas.

Por tanto, cuando se intenta eliminar o actualizar un registro en una tabla principal, la integridad referencial garantiza que no se creen registros huérfanos en las tablas secundarias.

De lo anterior mencionado podemos decir que las operaciones de inserción, actualización o eliminación no se podrá lograr. De esta manera, se evita que los datos se vuelvan inconsistentes o incorrectos [9].

**Importancia del Conocimiento:** Las restricciones de integridad aseguran que los datos no se corrompan.

## 2. Seguridad, Auditoría y Control de Acceso

**Objetivo:** Proteger los datos sensibles y controlar el acceso a la base de datos.

**Actividades:**

**Implementar políticas de acceso y seguridad.**

**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'@'%';
```

Creamos los roles y asignamos permisos a los usuarios. Este es el primer paso para asegurar que solo los usuarios adecuados tengan acceso a datos sensibles.

Este código se asegura de que solo las personas con los roles adecuados puedan interactuar con las tablas específicas. Esto es fundamental para evitar accesos no autorizados.

Este código cumple con la **política de acceso y seguridad**, ya que estamos restringiendo el acceso a las tablas basándonos en el rol del usuario. Solo los

administradores tienen acceso total, y los usuarios tienen acceso limitado a ciertas tablas.

Los usuarios deben ser creados antes de asignarles los roles.

Es necesario usar % como host en lugar de 'localhost' si se quiere permitir el acceso desde cualquier host.

**Investigación:** Investigar sobre los mejores enfoques para la seguridad en bases de datos en entornos de alta disponibilidad.

Entre los enfoques que se pudieron encontrar tenemos los siguientes

**Seguridad Física y Controles Administrativos:** El uso de controles de acceso, vigilancia y backup en caso de desastres naturales son ejemplos de la seguridad física de los servidores de bases de datos. Además, los controles administrativos, como políticas de seguridad junto con la capacitación del personal ayudan a tener un control administrativo eficiente [10].

**Encriptación de Datos:** El uso de TLS para conexiones de bases de datos y encriptación a nivel de columna para datos sensibles son técnicas para proteger los mismos cuando se encuentran en reposo o en tránsito [11].

**Autenticación y Autorización Fuertes:** La autenticación multifactor (MFA) es importante ya que asegura que solo usuarios autorizados puedan acceder a la base de datos. Además, los roles con permisos específicos o granulares hacen posible que el acceso a la información suficiente sin la necesidad de otorgar todos los permisos. [11].

**Monitoreo y Auditoría:** Es importante utilizar herramientas de detección de intrusos y análisis de comportamiento para tener un monitoreo constante de la base de datos. También, las auditorías regulares son de gran ayuda para identificar y reparar vulnerabilidades en la configuración de la base de datos [12].

**Copia de Seguridad y Recuperación ante Desastres:** es importante realizar copias de seguridad frecuentes y seguras de la base de datos para tener un respaldo en caso de algún desastre y poder realizar la recuperación de estas para minimizar el tiempo de inactividad [12].

**Importancia del Conocimiento:** El control adecuado de acceso previene fugas de información y mejora la seguridad general.

**Cifrado de datos sensibles.**

**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');

-- Descriptar 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';

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

```

Utilizamos un **gestor de claves** como **AWS KMS** para almacenar las claves de manera segura.

Ambas consultas están cifrando valores sensibles (el documento de identidad y el método de pago) utilizando la función AES\_ENCRYPT (). La clave de cifrado se usa para asegurar que la información esté protegida, de manera que incluso si alguien tiene acceso a la base de datos, no podrá leer los datos originales sin conocer la clave.

**Investigación:** Explorar algoritmos de cifrado y su impacto en el rendimiento de la base de datos.

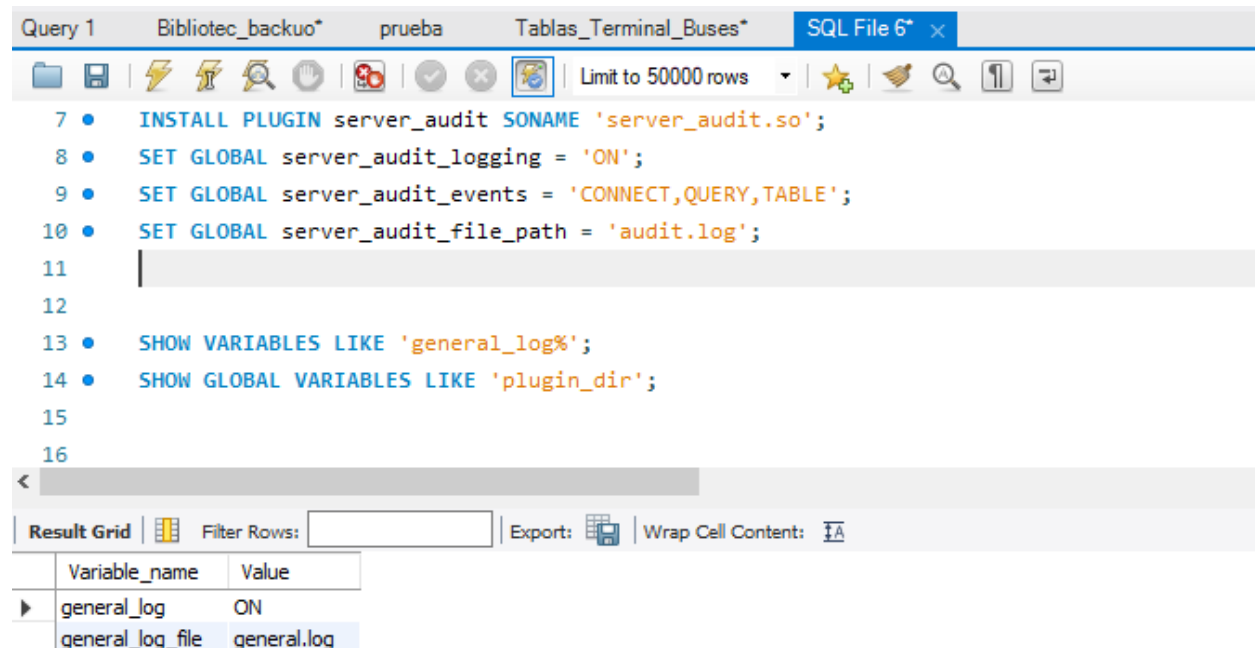
Entre los algoritmos de cifrado tenemos los siguientes: AES, RSA, 3DES, Salsa20. Cada uno de ellos tiene métricas diferentes para cifrar los datos de una base de datos específica, cada uno de ellos de diferencia por si son simétricos, asimétricos, rápidos, lentos, entre otros [13].

Por otra parte, el cifrado de datos tiene un impacto en el rendimiento de la base de datos. Este impacto depende de varios factores, incluyendo el algoritmo de cifrado utilizado, la implementación del cifrado, la latencia de E/S y la capacidad del hardware [14].

**Importancia del Conocimiento:** El cifrado es esencial para la protección de la información confidencial de los usuarios.

**Habilitar auditoría y registrar eventos de base de datos.**

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



```
Query 1  Bibliotec_backup*  prueba  Tablas_Terminal_Buses*  SQL File 6* x
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
```

Variable_name	Value
general_log	ON
general_log_file	general.log

**Investigación:** Buscar cómo configurar herramientas de auditoría en MySQL o PostgreSQL.

Entre las herramientas más utilizadas en PostgreSQL se debe seguir los siguientes pasos:

1. Instalar la extensión de PostgreSQL con el siguiente comando: *sudo apt-get install postgresql-contrib*
2. Luego, se crea esta extensión con el siguiente comando: *CREATE EXTENSION pgaudit;*
3. Además, es necesario configurar las opciones de auditoría en el archivo *postgresql.conf*: *pgaudit.log = 'read, write'*

Finalmente, ya se puede ocupar esta herramienta en PostgreSQL para realizar la auditoría en una base de datos específica [15].

**Importancia del Conocimiento:** La auditoría permite rastrear cambios en los datos y detectar actividades sospechosas.

### 3. RespalDOS y Recuperación de Datos

**Objetivo:** Asegurar la integridad y disponibilidad de los datos mediante técnicas de respaldo confiables.

**Actividades:**

**Crear respaldos completos (full backups).**

**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.sql"
```

Con mysqldump se puede realizar un respaldo de una base de datos teniendo en cuenta esta estructura:

Usuario, Contraseña (\* si la hay), Nombre\_basedato :

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

Dirección:

```
> "C:\Users\jhonc\OneDrive\Desktop\proyecto de base de datos\respaldo_buses_terminal.
```

**Investigación:** Buscar estrategias de respaldo para bases de datos de gran tamaño y la mejor manera de gestionarlas.

Entre las estrategias del respaldo de bases de datos tenemos las siguientes:

La copia de seguridad completa la cual se centra en copiar de manera completa de toda la base de datos en intervalos regulares. Además, va de la mano con la copia de seguridad incremental la cual se centra en respaldar los cambios desde la última fecha de respaldo de una base de datos. De lo anterior mencionado, la copia de seguridad diferencial es muy similar a la copia de seguridad incremental ya que tiene el mismo comportamiento, pero se considera un punto medio entre la completa y la incremental. Además, realizar copias de seguridad de registros de transacciones sirve para recuperar los datos hasta el último punto de transacción. Finalmente, replicar las bases de datos puede mejorar la sincronización de estas en diferentes lugares y es más fácil recuperarlas cuando existe algún desastre [16].



En cambio, es importante la gestión de grandes volúmenes de datos por lo cual se puede utilizar sistemas de cacheo como Redis o Memcached para reducir la carga en la base de datos [17].

Con lo anterior mencionado no se puede dejar a un lado la automatización y la escalabilidad de las bases de datos, por es importante considerar que lastareas rutinarias como copias de seguridad y actualizaciones se realicen de manera automática para mejorar la eficiencia operativa. De este modo, el monitoreo y el mantenimiento se pueden detectar desde una interfaz que notifique los cambios que se van realizando [18].

**Importancia del Conocimiento:** Los respaldos completos permiten restaurar toda la base de datos ante una falla.

**Configurar respaldos incrementales.**

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

```
C:\Users\jhonc>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25
Server version: 8.0.40 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

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

mysql>
```

Se debe de ingresar al sistema de mysql y verificar si está activo o no el registro binario.

Si devuelve ON, significa que el registro binario está habilitado. (Permite hacer **respaldos incrementales** sin copiar toda la base de datos)

Si devuelve OFF, significa que está deshabilitado y debes activarlo.

**Investigación:** Investigar cómo realizar respaldos incrementales y cuándo es más conveniente utilizarlos.

Los pasos que se deben seguir para realizar respaldos incrementales son: realizar una copia de seguridad completa inicial, identificar y respaldar cambios, actualizar el registro de copias de seguridad con la copia de seguridad incremental y finalmente para tener una eficiencia más alta en estas tareas se debe automatizar el proceso para gestionarlo a nuestro gusto [19].

Sin embargo, al realizar el anterior proceso debemos tener en cuenta que existen situaciones en las cuales podemos hacer uso de estos respaldos incrementales y estos son: cuando existen cambios constantes en la base de datos, cuando el almacenamiento de la base de datos es una preocupación, cuando se realizan planes de recuperación ante desastres y en ambientes críticos porque este tipo de copias de seguridad son mucho más veloces [20].

**Importancia del Conocimiento:** Los respaldos incrementales permiten optimizar los recursos y acelerar los tiempos de recuperación.

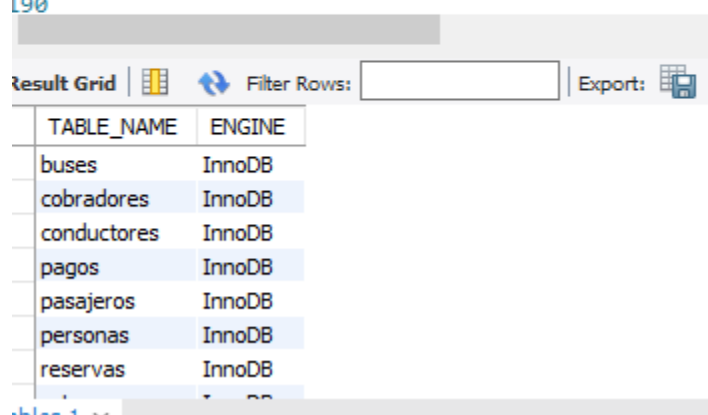
**Implementar respaldos en caliente (Hot Backups).**

**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.

1.- verificar

```
-- 3.- seguridad
SELECT table_name, engine
FROM information_schema.tables
WHERE table_schema = 'terminal_buses';
```



The screenshot shows a MySQL query result in a 'Result Grid' window. The query is: `SELECT table_name, engine FROM information_schema.tables WHERE table_schema = 'terminal_buses';`. The result is a table with two columns: 'TABLE\_NAME' and 'ENGINE'. The data rows are: buses (InnoDB), cobradores (InnoDB), conductores (InnoDB), pagos (InnoDB), pasajeros (InnoDB), personas (InnoDB), and reservas (InnoDB).

TABLE_NAME	ENGINE
buses	InnoDB
cobradores	InnoDB
conductores	InnoDB
pagos	InnoDB
pasajeros	InnoDB
personas	InnoDB
reservas	InnoDB

Si hay **MyISAM** en alguna tabla, se debe **convertir esa tabla a InnoDB** con:

**ALTER TABLE nombre\_de\_tabla ENGINE=InnoDB;**

Para hacer un respaldo en caliente es necesario activar el **log binario**. Caso contrario se debe de activar buscando y modificarlo :

- Abrir el archivo de configuración my.ini (generalmente en C:\ProgramData\MySQL\MySQL Server 8.0(o cualquier que se tu version)\my.ini).
- Agrega o modifica estas líneas:

[mysqld]

log-bin=mysql-bin

binlog-format=ROW

server-id=1

- Guardar y reiniciar el servicio.

Respaldo en caliente

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

Ejecuta en el Símbolo del Sistema (CMD) como administrador

ste equipo > Escritorio

Nombre	Estado
analisis de datos	✓
Office	✓
proyecto de base de datos	↻
proyecto_git_analisis	✓
MongoDBCompass	✓
movie_20m	✓
movies_with_ratings	✓
proyecto	✓
rating	✓
respaldo	✓
vie	Tipo: SQL Text File
Vis	Tamaño: 12,3 KB
Wd	Fecha de modificación: 2/2/2025 15:22
	Estado de disponibilidad: Disponible en este dispo:

```
1  -- MySQL dump 10.13  Distrib 8.0.40, for Win64 (x86_64)
2  --
3  -- Host: localhost    Database: terminal_buses
4  --
5  -- Server version  8.0.40
6
7  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
8  /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
9  /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
10 /*!50503 SET NAMES utf8mb4 */;
11 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
14 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
15 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
16 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
17
18 --
19 -- Table structure for table `buses`
```

**Investigación:** Investigar cómo hacer respaldos sin detener la base de datos.

Entre los métodos que se pueden utilizar para realizar respaldos sin detener la base de datos encontramos los siguientes:

Copia de Seguridad en Caliente (Hot Backup): Permite realizar copias de seguridad mientras la base de datos está en funcionamiento [21].

Replicación de Bases de Datos: Este método implica mantener una copia exacta de la base de datos en un servidor secundario, en este último se puede realizar una copia de seguridad del servidor secundario sin afectar al servidor principal [21].

*Snapshots*: Son una especie de *checkpoint* el cual realiza copias puntuales de la base de datos en un momento específico [21].

Backup Continuo: Este método implica realizar copias de seguridad incrementales de los cambios en la base de datos en tiempo real [21].

**Importancia del Conocimiento:** Los respaldos en caliente son esenciales para bases de datos de producción que no pueden permitirse inactividad.

## 4. Optimización y Rendimiento de Consultas

**Objetivo:** Mejorar la eficiencia en la recuperación de datos mediante la optimización de consultas y el uso adecuado de índices.

**Actividades:**

**Crear y gestionar índices.**


**Práctica:** Implementar índices en las columnas más consultadas, como VueloID, ClienteID, 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
180 -- 2 explain
181 • EXPLAIN SELECT * FROM Reservas r
182 JOIN Viajes v ON r.id_viaje = v.id_viaje
183 WHERE v.ciudad_salida = 'Ciudad A' AND r.estado = 'Confirmada';
184
// ver el resultado
38 • SHOW INDEX FROM Reservas;
39
40

```

result Grid

 Filter Rows:

 Export:

 Wrap Cell Content:




Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
reservas	0	PRIMARY	1	id_reserva	A	1	NULL
reservas	1	idx_pasajero	1	id_pasajero	A	1	NULL
reservas	1	idx_viaje	1	id_viaje	A	1	NULL
reservas	1	idx_id_pasajero	1	id_pasajero	A	1	NULL
reservas	1	idx_id_viaje	1	id_viaje	A	1	NULL
reservas	1	idx_id_pasajero_reservas	1	id_pasajero	A	1	NULL
reservas	1	idx_id_viaje_reservas	1	id_viaje	A	1	NULL

**Investigación:** Investigar sobre los tipos de índices más adecuados para bases de datos transaccionales y cómo afectan el rendimiento.

Entre los tipos de índices para las bases de datos tenemos las siguientes:

B-Tree: Son los más comunes en las bases de datos transaccionales [22].

Hash: Son muy eficaces para realizar búsquedas exactas, ya que permiten acceder rápidamente a los registros mediante una clave de búsqueda [22].

Índices Compuestos: Se crean sobre dos o más columnas de una tabla [22].

Índices Únicos: El valor de una columna (o conjunto de columnas) son únicos [22].

Índices Clustered: Organizan físicamente los datos en la tabla según el orden de las columnas indexadas [22].

Índices No Clustered: Almacenan los datos y los índices en espacios de almacenamiento separados [22].

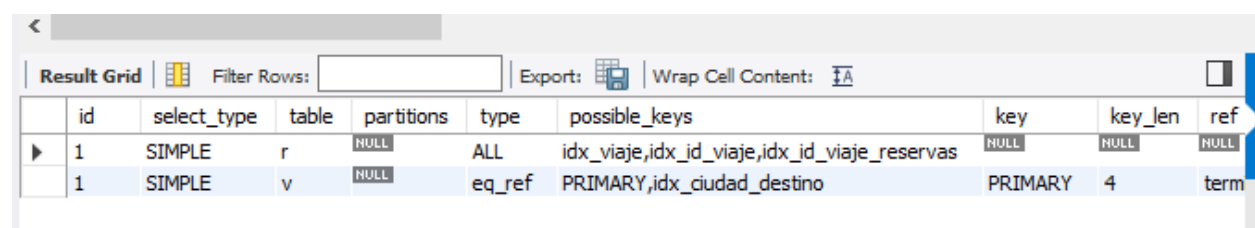
Cada uno de los índices mencionados tienen la misión de mejorar el rendimiento de las consultas y a mantener la integridad de las bases de datos transaccionales.

**Importancia del Conocimiento:** Los índices son cruciales para acelerar las consultas y mejorar el rendimiento general de la base de datos.

### Optimizar consultas SQL.

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

```
l80      -- 2 explain
l81 •    EXPLAIN SELECT * FROM Reservas r
l82      JOIN Viajes v ON r.id_viaje = v.id_viaje
l83      WHERE v.ciudad_salida = 'Ciudad A' AND r.estado = 'Confirmada';
l84
```



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

**id:** Este número indica el orden de ejecución de las tablas involucradas en la consulta.

Si el EXPLAIN muestra que MySQL no está utilizando un índice, crea un índice en las columnas que se usan en los JOIN y en las cláusulas WHERE.

**Investigación:** Investigar cómo hacer uso eficiente de las uniones (JOIN), subconsultas, y optimizar las consultas complejas.

En primer lugar, para realizar tener un uso eficiente de las uniones se debe seleccionar el JOIN correcto para la consulta junto con las condiciones y alias correctos para evitar un error de lógica. Además, dentro de una base de datos el exceso de JOINs puede hacer más lento su procedimiento.

Además, las subconsultas tienen el mismo propósito de unir datos con operaciones que requieren un solo valor o un conjunto de valores. Las subconsultas correlacionadas, que dependen de la tabla externa, deben usarse con moderación debido a su impacto en el rendimiento.

Finalmente, para que los JOINS y las subconsultas estén optimizadas dentro de nuestra base de datos es necesario: utilizar CTEs y vistas para descomponer consultas complejas en partes más manejables y reutilizables, evitar JOINS innecesarios, utilizar herramientas que identifiquen cuellos de botella en el rendimiento y ajustar las consultas según sea necesario [23].

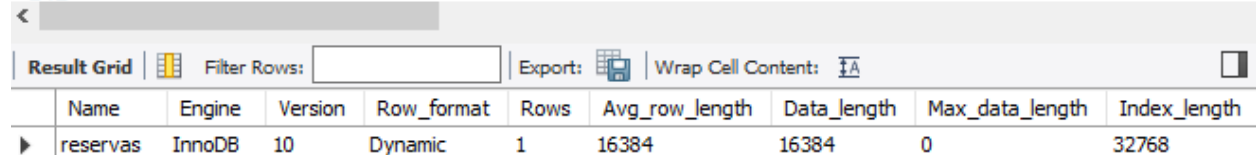
**Importancia del Conocimiento:** Las consultas optimizadas aseguran un sistema rápido y eficiente, especialmente en sistemas con alta demanda.

### Utilizar particionamiento de tablas.

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

Verifica ejecutando el siguiente comando:

```
240 -- reservas
241 • SHOW TABLE STATUS LIKE 'Reservas';
242
```



Name	Engine	Version	Row_format	Rows	Avg_row_length	Data_length	Max_data_length	Index_length
reservas	InnoDB	10	Dynamic	1	16384	16384	0	32768

crear una tabla para permitir particionamiento:

```
244 • CREATE TABLE Reservas_Particionada (
245     id_reserva INT AUTO_INCREMENT,
246     fecha_reserva DATETIME DEFAULT CURRENT_TIMESTAMP,
247     asiento_disponibles INT NOT NULL,
248     id_pasajero INT,
249     id_viaje INT,
250     estado ENUM('Confirmada', 'Cancelada') DEFAULT 'Confirmada',
251     PRIMARY KEY (id_reserva, fecha_reserva) -- Agregamos fecha_reserva a la clave primaria
252 )
253 PARTITION BY RANGE (YEAR(fecha_reserva)) (
254     PARTITION p_2021 VALUES LESS THAN (2022),
255     PARTITION p_2022 VALUES LESS THAN (2023),
256     PARTITION p_2023 VALUES LESS THAN (2024),
257     PARTITION p_2024 VALUES LESS THAN (2025)
258 );
259
260
261 • EXPLAIN SELECT * FROM Reservas_Particionada WHERE YEAR(fecha_reserva) = 2022;
262
```



SHOW TABLE STATUS LIKE 'Reservas\_Particionada';

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

Particionar la tabla Reservas puede mejorar el rendimiento y la gestión de los datos de reservas.

**Investigación:** Investigar sobre los beneficios del particionamiento y cómo implementarlo en sistemas de bases de datos grandes.

El particionamiento es dividir una tabla grande en particiones más pequeñas y manejables, entre los beneficios que nos ofrece el particionamiento tenemos:

- Optimización del Rendimiento
- Mantenimiento Simplificado
- Escalabilidad Mejorada
- Mayor Tolerancia a Fallos
- Flexibilidad Operativa [24]

Por tanto, al revisar los beneficios del particionamiento se puede implementar un método para bases de datos grandes:

- Particionamiento horizontal, el cual consiste en dividir una tabla en filas.
- Particionamiento vertical que ayuda a dividir la tabla en columnas.
- Particionamiento por lista, lo cual significa que cada partición tiene un conjunto específico de valores.
- Particionamiento por Hash, el cual consiste en distribuir uniformemente las filas entre las particiones. [24]

**Importancia del Conocimiento:** El particionamiento de tablas mejora la escalabilidad y el rendimiento en bases de datos con gran volumen de datos.

## 5. Procedimientos Almacenados, Vistas y Triggers

**Objetivo:** Mejorar la eficiencia y automatizar tareas mediante el uso de procedimientos almacenados, vistas y triggers.

**Actividades:**

**Crear procedimientos almacenados.**

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

```
use terminal_buses;
```

```
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)  
)
```

```
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 //
```

```
DELIMITER ;
```

```

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

```

	PrecioTotal	DescuentoTotal	CargoTotal
▶	10.00	0.00	0.00

**Investigación:** Explorar cómo los procedimientos almacenados pueden mejorar la reutilización de código y la eficiencia.

Un procedimiento almacenado es un conjunto de instrucciones SQL que se agrupan y se almacenan en el servidor de bases de datos para su reutilización y pueden mejorar la reutilización del código ya que: permiten encapsular la lógica del negocio en bloques de código, permiten el modularidad que reduce la duplicación de código para un mejor mantenimiento [25].

Además, utilizarlos en nuestra base de datos tiene beneficios en la eficiencia como: la reducción del tráfico de red minimiza la cantidad de datos transferidos entre el servidor y el cliente, al ser precompilados y optimizados reducen el tiempo de ejecución. Y, al automatizar tareas pueden ser utilizados para controlar el acceso a los datos y mantener la integridad de la base de datos [25].

**Importancia del Conocimiento:** Los procedimientos almacenados centralizan la lógica y pueden mejorar el rendimiento al ejecutarse directamente en el servidor.

**Crear vistas para simplificar consultas complejas.**

**Práctica:** Crear vistas que presenten información de varias tablas de manera unificada (por ejemplo, una vista que combine datos de Vuelos, Clientes y Reservas).

```
-- 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

```

	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 Martínez	2233445566	10	2i
	4	Quito	Ambato	2023-01-04 11:00:00	2023-01-04 13:00:00	10	Elena Díaz	8899001122	4	2i
	8	Guayaquil	Loja	2023-01-08 15:00:00	2023-01-08 23:00:00	10	Elena Díaz	8899001122	8	2i
	1	Quito	Guayaquil	2023-01-01 08:00:00	2023-01-01 14:00:00	1	Juan Pérez	1234567890	1	2i
	5	Quito	Riobamba	2023-01-05 12:00:00	2023-01-05 14:00:00	1	Juan Pérez	1234567890	5	2i

```

-- 2da Vista
• CREATE VIEW VistaBusConductorCobrador AS
SELECT
    b.id_bus,
    b.placa,
    b.modelo,
    b.capacidad,
    b.fecha_mantenimiento,
    b.disponibilidad,
    c.id_persona AS id_conductor,
    c.licencia,
    c.fecha_contratacion AS fecha_contratacion_conductor,
    co.id_persona AS id_cobrador,
    co.turno,
    co.fecha_contratacion AS fecha_contratacion_cobrador
FROM
    Buses b
JOIN
    Conductores c ON b.id_bus = c.id_persona
JOIN
    Cobradores co ON b.id_bus = co.id_persona;

58 • SELECT * FROM VistaBusConductorCobrador;
59

```

	id_bus	placa	modelo	capacidad	fecha_mantenimiento	disponibilidad	id_conductor	nombre_conductor	licencia	fecha_contratacion_c
▶	1	ABC1234	Mercedes	50	2023-01-10	Disponible	NULL	NULL	NULL	NULL
	2	DEF5678	Volvo	45	2023-02-15	Disponible	2	María Gómez	LIC123456	2022-01-15
	3	GHI9012	Scania	60	2023-03-20	Mantenimiento	NULL	NULL	NULL	NULL
	4	JKL3456	Mercedes	55	2023-04-25	Disponible	NULL	NULL	NULL	NULL
	5	MNO7890	Volvo	50	2023-05-30	Disponible	5	Luis Fernández	LIC654321	2022-02-20

VistaBusConductorCobrador 3 x

**Investigación:** Investigar las ventajas de usar vistas en lugar de consultas complejas repetitivas.

En primer lugar, las vistas tienen la lógica de las consultas en una sola operación mientras que las consultas son mucho más complejas y pueden llegar a ser repetitivas.

Entre las ventajas de utilizar las vistas en lugar de las consultas tenemos:

- Mantenimiento Simplificado: Las vistas permiten un mantenimiento más sencillo de las consultas.
- Mejora del Rendimiento: Se reduce el tiempo de procesamiento y se optimiza el acceso a los datos.
- Reutilización de Consultas: Las vistas facilitan la reutilización de consultas complejas
- Abstracción de la Complejidad: Las vistas al ser mucho más sencillas, ayudan a abstraer la complejidad de las consultas [26].

**Importancia del Conocimiento:** Las vistas ayudan a simplificar el acceso a datos complejos y pueden mejorar la seguridad al limitar el acceso directo a las tablas.

**Implementar triggers para auditoría y control de cambios.**

**Práctica:** 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  );
14
15  DELIMITER $$
16 • CREATE TRIGGER before_reservas_update
17   BEFORE UPDATE ON Reservas
18   FOR EACH ROW

```

```

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 ;
25
26 DELIMITER $$
27 • CREATE TRIGGER before_reservas_delete
28 BEFORE DELETE ON Reservas
29 FOR EACH ROW
30 BEGIN
31     INSERT INTO log_reservas_pagos (tabla, operacion, id_registro, usuario, detalles)
32     VALUES ('Reservas', 'DELETE', OLD.id_reserva, USER(),
33             CONCAT('Estado: ', OLD.estado));
34 END$$
35 DELIMITER ;
36
37 DELIMITER $$
38 • CREATE TRIGGER before_pagos_update
39 BEFORE UPDATE ON Pagos
40 FOR EACH ROW
41 BEGIN
42     INSERT INTO log_reservas_pagos (tabla, operacion, id_registro, usuario, detalles)
43     VALUES ('Pagos', 'UPDATE', OLD.id_pago, USER(), CONCAT('Antes: ', OLD.monto, ', Después: ', NEW.monto));
44 END$$
45 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, Después: Cancelado');
59

```

Result Grid   Filter Rows:   Edit:   Export/Import:   Wrap Cell Content: IA							
id_auditoria	tabla	operacion	id_registro	fecha	usuario	detalles	
2	Pagos	DELETE	456	2025-02-03 16:51:37	usuario_prueba	Monto: 100.00	
* NULL	NULL	NULL	NULL	NULL	NULL	NULL	

```

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, Después: Activo'
65   WHERE id_auditoria = 1;

```

Result Grid   Filter Rows:   Edit:   Export/Import:   Wrap Cell Content: IA							
id_auditoria	tabla	operacion	id_registro	fecha	usuario	detalles	
2	Pagos	DELETE	456	2025-02-03 16:51:37	usuario_prueba	Monto: 100.00	
3	Pagos	DELETE	456	2025-02-03 16:53:14	usuario_prueba	Monto: 100.00	
4	Reservas	UPDATE	123	2025-02-03 16:53:25	usuario_prueba	Antes: Activo, Después: Cancelado	
* NULL	NULL	NULL	NULL	NULL	NULL	NULL	

**Investigación:** Investigar cómo utilizar triggers para mantener un historial de cambios en la base de datos.

Los triggers son procedimientos almacenados que se ejecutan automáticamente en respuesta a eventos específicos en una tabla, como inserciones, actualizaciones o eliminaciones [27].

Además, estos ayudan a mantener el historial de cambios de las tablas debido a su capacidad para registrar automáticamente las modificaciones, validar datos y realizar acciones de seguridad. Sin embargo, el rendimiento de la base de datos se ve afectado



por la utilización de triggers por lo cual no se debe introducir complejidad innecesaria [27].

**Importancia del Conocimiento:** Los triggers permiten automatizar tareas como la auditoría y validación de datos.

## 6. Monitoreo y Optimización de Recursos

**Objetivo:** Controlar el rendimiento de la base de datos, identificando y solucionando problemas de recursos.

**Actividades:**

**Monitorear el rendimiento de consultas.**

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

1.- se usa SHOW PROCESSLIST para detectar consultas

```
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	NULL	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 PROC

Revisar el plan de ejecución de la consulta:

```

4 • EXPLAIN SELECT * FROM Viajes WHERE ciudad_salida = 'Bogotá';
5
6
7
8 -- Monitoreo de índices existentes
9 • SHOW INDEX FROM Personas:

```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref
1	SIMPLE	Viajes	NULL	ref	idx_ciudad_destino,idx_ciudad_salida	idx_ciudad_destino	402	cor

## Revisar y mejorar los índices

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

## Ajustar los parámetros de MySQL

```

SHOW VARIABLES LIKE 'key_buffer_size';
SHOW VARIABLES LIKE 'query_cache_size';

```

SI ES NECESARIO AJUSTAR LOS PARAMENTROS DE MYSQL

## Monitoreo de Consultas Lentas (Slow Query Log)

Consultar el Slow Query Log:

- `SELECT * FROM mysql.slow_log WHERE start_time >= '2025-02-01';`

**Investigación:** Investigar las mejores prácticas para monitorear el rendimiento de las consultas en producción.

Entre las mejores prácticas para monitorear el rendimiento de las consultas en producción se encontraron las siguientes:

- Establecimiento de líneas base: Se recomienda establecer líneas base de rendimiento para las consultas.

- Uso de herramientas de monitoreo: New Relic, SolarWinds, o Dynatrace, son herramientas que sirven para el monitoreo de la base de datos.
- Análisis de planes de ejecución: Analizar planes de ejecución de las consultas puede ayudar a identificar cuellos de botella y optimizar el rendimiento.
- Monitoreo de recursos del sistema: Analizar los componentes de nuestro Hardware puede ser de gran ayuda para evaluar rendimiento de las consultas en nuestra base de datos.
- Implementación de índices adecuados: Implementar índices adecuados en las tablas puede ayudar al rendimiento de las consultas.
- Revisión: Se recomienda realizar revisiones periódicas del rendimiento de las consultas para corregir problemas. [28]

**Importancia del Conocimiento:** El monitoreo proactivo puede identificar cuellos de botella antes de que afecten el rendimiento del sistema.

### **Realizar pruebas de carga.**

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

Se necesita comprobar que java este instalado en windows

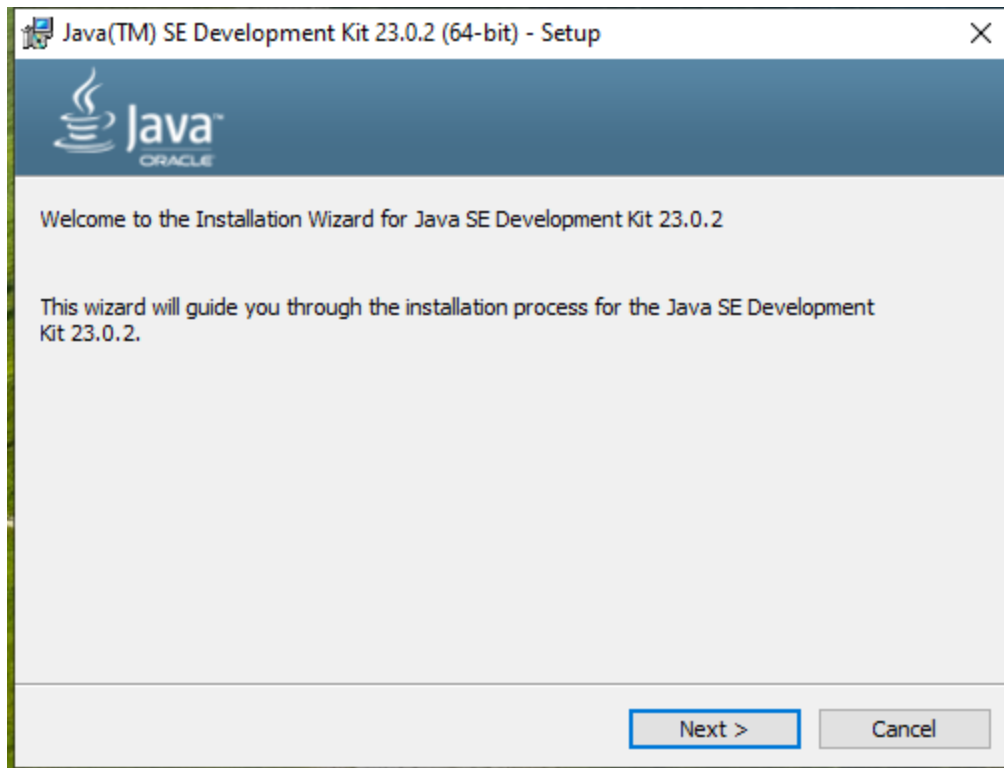
```
Microsoft Windows [Versión 10.0.19045.5371]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jhonc>java -version
java version "23.0.2" 2025-01-21
Java(TM) SE Runtime Environment (build 23.0.2+7-58)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.2+7-58, mixed mode, sharing)

C:\Users\jhonc>
```

Caso contrario instalar java desde este enlace:

[https://jmeter.apache.org/download\\_jmeter.cgi](https://jmeter.apache.org/download_jmeter.cgi).



## Ejecutar JMeter

- En la carpeta de instalación (C:\jmeter\bin), haz doble clic en **jmeter.bat** o ejecuta en CMD:

```
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
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a
release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a
release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a
release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a
release
=====
Don't use GUI mode for load testing !, only for Test creation and Test debugging.
For load testing, use CLI Mode (was NON GUI):
  jmeter -n -t [jmx file] -l [results file] -e -o [Path to web report folder]
& increase Java Heap to meet your test requirements:
  Modify current env variable HEAP="-Xms1g -Xmx1g -XX:MaxMetaspaceSize=256m" in the jmeter batch file
Check : https://jmeter.apache.org/usermanual/best-practices.html
=====
```

Descargar conector de mysql:

# MySQL Community Descargas

Conector/J

**Disponibilidad general (GA) Lanzaciones**

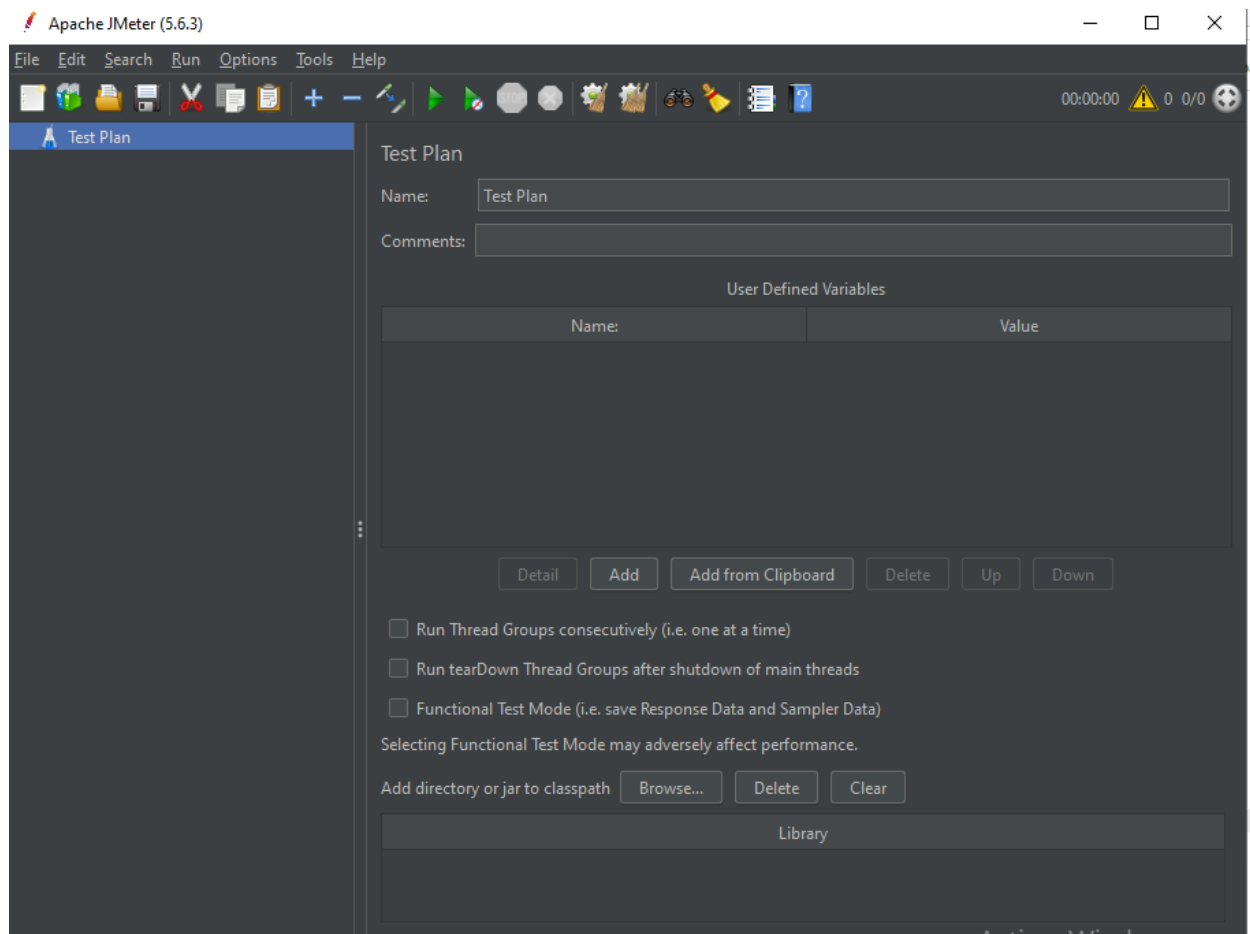
Archivos

**Conector/J 9.2.0**

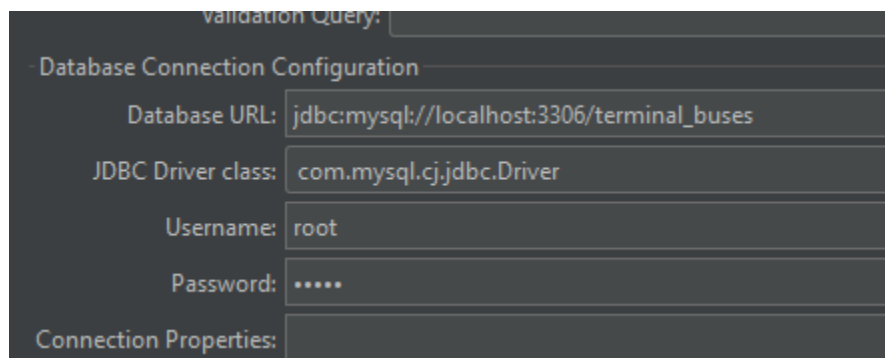
Seleccione sistema operativo:

Seleccione sistema operativo...

Ejecutar prueba en jmeter .



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



Agregar un Listener para Ver los Resultados:

The screenshot shows a web application interface. On the left, a sidebar contains three items: 'JDBC Request' (marked with a red X), 'JDBC Request' (marked with a red X), and 'JDBC Request' (marked with a green checkmark). The main content area displays a table with reservation data. The table has columns for reservation ID, reservation date, flight ID, route ID, available seats, departure city, destination, passenger ID, departure date, arrival date, and status. The data shows a reservation for a flight from Ciudad A to Ciudad B on 2025-02-01T17:03:30, with 40 available seats and a status of 'Confirmada'.

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

**Investigación:** Investigar cómo realizar pruebas de estrés y carga en bases de datos de alto rendimiento.

Para poder realizar pruebas en las bases de datos se necesita realizar los siguientes pasos:

**Definición de pruebas de estrés y carga:** Las pruebas de estrés se utilizan para evaluar el comportamiento de la base de datos bajo condiciones extremas, mientras que las pruebas de carga se centran en medir el rendimiento bajo condiciones normales de uso.

**Herramientas de Pruebas:** Se recomienda el uso de herramientas especializadas como Apache JMeter, LoadRunner.

**Configuración de escenarios de prueba:** Se sugiere definir escenarios de prueba realistas que reflejen el uso esperado de la base de datos.

**Análisis de Resultados:** Se recomienda analizar los resultados de las pruebas para identificar cuellos de botella y áreas de mejora.

**Optimización Continua:** Se concluyó que la optimización continua es esencial para mantener el rendimiento de la base de datos [29].

**Importancia del Conocimiento:** Las pruebas de carga aseguran que el sistema sea capaz de manejar tráfico alto y crecimiento de datos.

**Optimizar el uso de recursos y gestionar índices.**

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

1.- Identificamos cuales son los índices no utilizados:

```
1 • SELECT table_name, index_name, rows_selected, rows_inserted, rows_updated, rows_deleted
2 FROM sys.schema_index_statistics
3 WHERE table_schema = 'terminal_buses'
4 ORDER BY rows_selected ASC;
5
```

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

	table_name	index_name	rows_selected	rows_inserted	rows_updated	rows_deleted
•	personas	PRIMARY	0	0	0	0
	personas	documento_identidad	0	0	0	0
	personas	idx_documento_identidad	0	0	0	0
	pasajeros	PRIMARY	0	0	0	0
	pasajeros	email	0	0	0	0
	pasajeros	idx_email	0	0	0	0

Result Grid  
Form Editor

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;
```



```

-- Eliminar índices innecesarios (Si es necesario)
DROP INDEX idx_no_utilizado ON Personas;
DROP INDEX idx_no_utilizado ON Pagos;
DROP INDEX idx_no_utilizado ON Reservas;
DROP INDEX idx_no_utilizado ON Ticket;

-- Crear índices óptimos para mejorar consultas frecuentes
CREATE INDEX idx_documento_identidad ON Personas(documento_identidad);
CREATE INDEX idx_fecha_pago ON Pagos(fecha_pago);
CREATE INDEX idx_estado_reserva ON Reservas(estado);
CREATE INDEX idx_codigo_ticket ON Ticket(codigo_ticket);

```

Eliminar índices innecesarios puede mejorar el rendimiento de la base de datos, ya que reduce la sobrecarga al insertar, actualizar o eliminar registros.

**Investigación:** Investigar cómo ajustar el número de índices según el tipo de consulta (lectura/escritura) tenemos las siguientes:

Entre las mejores prácticas para ajustar el número de índices según el tipo de consulta (lectura/escritura):

Es importante mantener el equilibrio entre índices y rendimiento de las consultas. Los índices pueden mejorar significativamente las consultas de lectura.

Además, los índices para consultas de lectura intensiva se deben crear índices en las columnas que se utilizan frecuentemente en las cláusulas WHERE, JOIN y ORDER BY.

También, los índices para consultas de escritura (inserciones, actualizaciones, eliminaciones), se sugiere minimizar el número de índices para reducir la sobrecarga de mantenimiento.

Sin embargo, el monitoreo y ajuste continuo es importante para optimizar el rendimiento de los índices regularmente, por tanto; se debe ajustar según sea necesario.

Mientras tanto, los índices compuestos (que abarcan múltiples columnas) y funcionales (basados en expresiones o funciones) pueden ser útiles para optimizar consultas específicas. [29]

**Importancia del Conocimiento:** La optimización de los recursos asegura un uso eficiente del hardware y mejora la escalabilidad.

## 7. Git y Control de Versiones

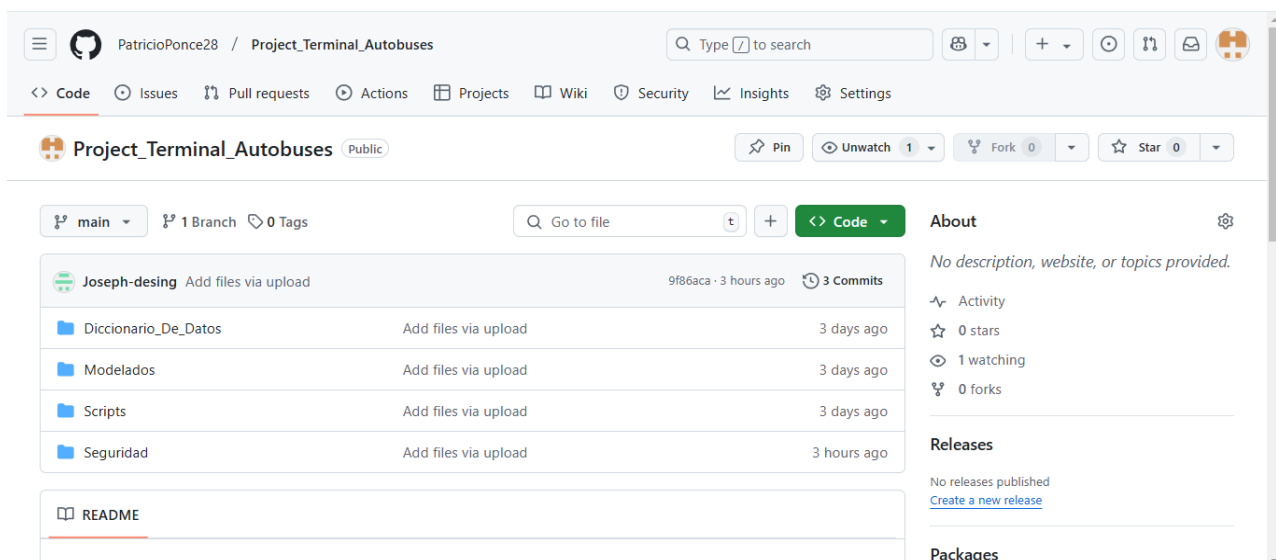
**Objetivo:** Asegurar que el código relacionado con la base de datos esté versionado y que el equipo pueda colaborar de manera eficiente.

**Actividades:**

**Configurar un repositorio de Git para el proyecto.**

**Práctica:** Inicializar un repositorio en Git y subir los archivos de definición de la base de datos, scripts de SQL y procedimientos almacenados.

El repositorio que hemos creado como grupo es el siguiente:



**Investigación:** Investigar buenas prácticas de flujo de trabajo en Git (por ejemplo, uso de ramas, git merge).

Entre las prácticas para tener un flujo de trabajo en Git encontramos las siguientes:

**Nomenclatura Clara y Consistente:** Se recomienda utilizar nombres de ramas descriptivos y fáciles de entender [30].

**Uso de Ramas de Características:** Cada nueva característica, corrección de errores o mejora debe desarrollarse en su propia rama derivada de la rama principal [30].

**Fusión Frecuente:** Se sugiere integrar ramas de características con la rama principal de manera frecuente [30].

**Revisiones de Código:** Antes de fusionar una rama de característica en la rama principal, se recomienda someter los cambios a una revisión de código. Utilizar Pull Requests (PR) o Merge Requests (MR) facilita la revisión y discusión de los cambios [30].

Mantener un Historial de Commits Limpio: Antes de fusionar una rama, se sugiere usar rebase interactivo para limpiar el historial de commits [30].

Usar Fusiones No Fast-Forward cuando Correspondan: Aunque las fusiones Fast-Forward mantienen un historial lineal, a veces es útil forzar un commit de fusión para que el historial sea más comprensible [30].

Resolución de Conflictos: Se recomienda abordar los conflictos de fusión de manera proactiva, comunicándose con otros miembros del equipo y documentando cualquier decisión tomada durante la resolución de estos conflictos [30].

Eliminar Ramas: Una vez que una rama ha sido fusionada y ya no es necesaria, se sugiere eliminarla para mantener el repositorio limpio [30].

**Importancia del Conocimiento:** Git permite la colaboración y el manejo eficiente de cambios en el código, especialmente cuando se trabaja en equipo.

**Realizar commits frecuentes y con mensajes claros.**

**Práctica:** Hacer commits regularmente, describiendo claramente los cambios realizados en los scripts SQL y la estructura de la base de datos.

PatricioPonce28 Commit Workflow		1103ed5 · now	🕒 10 Commits
📁 .github/workflows	Commit Workflow	now	
📁 Diccionario_De_Datos	Add files via upload	5 days ago	

**Investigación:** Investigar cómo utilizar git rebase y git pull para evitar conflictos.

Para utilizar git rebase y git pull con el fin de evitar conflictos, y se identificaron varios enfoques clave:

Uso de git pull --rebase: Se recomienda utilizar el comando git pull --rebase en lugar de git pull estándar. Este comando combina git fetch y git rebase, lo que ayuda a mantener un historial de commits más limpio y lineal, reduciendo la probabilidad de conflictos. [31]

Rebase frecuente: Esto implica ejecutar git fetch seguido de git rebase origin/main (o la rama correspondiente) para aplicar los commits locales sobre los cambios más recientes de la rama principal [32].

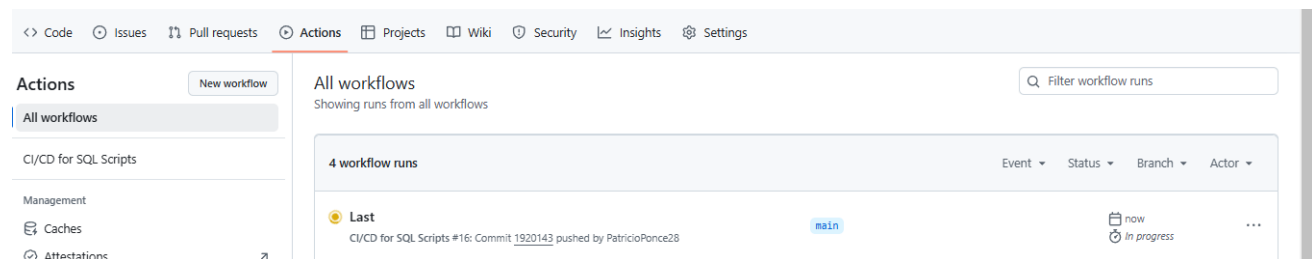
Resolución de conflictos durante el rebase: Durante el proceso de rebase, si se encuentran conflictos, Git pausará y permitirá resolverlos manualmente. Una vez resueltos, se debe ejecutar git rebase --continue para completar el rebase. Si se desea abortar el rebase, se puede usar git rebase --abort [32].

**Evitar Rebase en Ramas Compartidas:** Se determinó que es mejor evitar el uso de git rebase en ramas que ya han sido compartidas o publicadas en un repositorio remoto, ya que esto puede causar problemas de divergencia en el historial y confusión entre los colaboradores [32].

**Importancia del Conocimiento:** Un flujo de trabajo claro en Git mejora la colaboración y la gestión de versiones.

### Automatizar pruebas con GitHub Actions.

**Práctica:** Crear flujos de trabajo de CI/CD que automaticen las pruebas de las consultas SQL y otros scripts relacionados con la base de datos.



**Investigación:** Investigar sobre integración continua y cómo aplicarla en bases de datos con GitHub Actions.

La integración continua es una práctica de desarrollo de software que implica la integración frecuente de cambios de código en un repositorio compartido [33].

**Configuración de GitHub Actions:** GitHub Actions permite automatizar flujos de trabajo de CI directamente en un repositorio de GitHub. Se pueden crear flujos de trabajo que compilen el código, ejecuten pruebas y desplieguen aplicaciones [33].

**Automatización de Pruebas y Despliegue:** Con GitHub Actions, es posible configurar flujos de trabajo que se ejecuten automáticamente cuando se realicen cambios en el repositorio [33]

**Importancia del Conocimiento:** Las pruebas automáticas aseguran que las bases de datos se mantengan consistentes y funcionales a lo largo del tiempo.

### Resultados Obtenidos

- Se logró diseñar y desarrollar una base de datos funcional y optimizada para la administración de una terminal de buses.
- Se implementaron mecanismos de integridad de datos que garantizan la consistencia de la información almacenada.
- Se mejoró la eficiencia de las consultas mediante el uso de índices y optimización de queries.

- Se establecieron medidas de seguridad, automatización, respaldos, entre otras funcionalidades dentro de nuestra base de datos para tener una mejor desempeño y escalabilidad.
- Se realizó una capacitación a los compañeros del equipo para asegurar el correcto uso y mantenimiento del sistema.

## Conclusiones

El desarrollo de este proyecto permitió consolidar los conocimientos adquiridos en el semestre, aplicándolos en un caso práctico que simula una situación real. A través de este proceso, se comprendió la importancia de una correcta planificación, modelado y optimización en la gestión de bases de datos.

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.

## Referencias

- [1] B. Barua y M. S. Kaiser, "Enhancing Resilience and Scalability in Travel Booking Systems: A Microservices Approach to Fault Tolerance, Load Balancing, and Service Discovery," *arXiv preprint arXiv:2410.19701*, 2024. [En línea]. Disponible en: <https://arxiv.org/abs/2410.19701>.
- [2] "Diseño de bases de datos escalables para apoyar el crecimiento empresarial," Isita. [En línea]. Disponible en: <https://isitatech.com/es/disenando-de-bases-de-datos-escalables/>.
- [3] "Escalabilidad de Datos ¿Qué es y cómo se relaciona con las bases de datos?," GTD Perú. [En línea]. Disponible en: [https://www.gtdperu.com/es/w/novedades/escalabilidad\\_de\\_datos\\_que\\_es\\_y\\_como\\_se relaciona con las bases de datos](https://www.gtdperu.com/es/w/novedades/escalabilidad_de_datos_que_es_y_como_se relaciona con las bases de datos).
- [4] "Diseñando arquitecturas escalables: Recomendaciones," Rootstack. [En línea]. Disponible en: <https://rootstack.com/es/blog/disenando-arquitecturas-escalables-recomendaciones>.
- [5] "Genera tu diccionario de datos MySQL Workbench fácilmente," MySQL YA. [En línea]. Disponible en: <https://mysqlya.com.ar/tecnologia/generar-diccionario-de-datos-mysql-workbench/>.
- [6] "Guía completa para crear un diccionario de datos con ejemplos prácticos," MDP Ajedrez. [En línea]. Disponible en: <https://mdpajedrez.com.ar/guia-completa-para-crear-un-diccionario-de-datos-con-ejemplos-practicos/>.
- [7] "20 herramientas de modelado de datos que debe conocer en 2025," Carmatec. [En línea]. Disponible en: [https://www.carmatec.com/es\\_mx/blog/20-herramientas-de-modelado-de-datos-que-debe-conocer/](https://www.carmatec.com/es_mx/blog/20-herramientas-de-modelado-de-datos-que-debe-conocer/).

- [8] "Diccionario de Datos," Universidad Distrital Francisco José de Caldas. [En línea]. Disponible en: <https://repository.udistrital.edu.co/bitstreams/dcd3b890-1c87-4496-bbb4-e8428fee4fed/download>.
- [9] "Que es Integridad Referencial en Base de Datos," codigosql.top. [En línea]. Disponible en: <https://codigosql.top/bases-de-datos/integridad-referencial/>.
- [10] "Seguridad de la base de datos: qué es y cómo protegerla," Delta Protect. [En línea]. Disponible en: <https://www.deltaprotect.com/blog/seguridad-para-base-de-datos>.
- [11] "Mejores Prácticas de Seguridad de Bases de Datos," DataSunrise. [En línea]. Disponible en: <https://www.datasunrise.com/es/centro-de-conocimiento/mejores-practicas-de-seguridad-en-bases-de-datos/>.
- [12] "Seguridad de las bases de datos: guía básica," IBM. [En línea]. Disponible en: <https://www.ibm.com/es-es/topics/database-security>.
- [13] "El impacto del cifrado en el rendimiento," IBM. [En línea]. Disponible en: <https://www.ibm.com/docs/es/db2/11.1?topic=considerations-impact-encryption-performance>.
- [14] "Descubre los Algoritmos de Criptografía Más Populares Hoy en Día," Experto Seguridad. [En línea]. Disponible en: <https://experto-seguridad.es/ciberseguridad/los-algoritmos-de-criptografia-mas-utilizados-en-la-actualidad/>.
- [15] "Auditoría en PostgreSQL con PgAudit," GPSOS. [En línea]. Disponible en: <https://www.gpsos.es/2022/03/auditoria-en-postgresql/>.
- [16] "Estrategias de copia de seguridad y migración de datos para bases de datos relacionales," AppMaster. [En línea]. Disponible en: <https://appmaster.io/es/blog/estrategias-de-migracion-y-copia-de-seguridad-de-datos-para-bases-de-datos-relacionales>.
- [17] "Estrategias de copia de seguridad y migración de datos para bases de datos relacionales," AppMaster. [En línea]. Disponible en: <https://appmaster.io/es/blog/estrategias-de-migracion-y-copia-de-seguridad-de-datos-para-bases-de-datos-relacionales>.
- [18] "Big Data: cómo manejar y analizar grandes volúmenes de información," London Consulting Group. [En línea]. Disponible en: <https://londoncg.com/blog/big-data-como-manejar-y-analizar-grandes-volumenes-de-informacion>.
- [19] "Backup Incremental: La Mejor Estrategia para Proteger tus Datos," Datos 101. [En línea]. Disponible en: <https://www.datos101.com/blog/backup-incremental-estrategia-datos/>.
- [20] "Copia de seguridad incremental versus diferencial: todo lo que necesita saber," Trilio. [En línea]. Disponible en: <https://trilio.io/es/digitales/copia-de-seguridad-incremental-vs-diferencial/>.

- [21] E. López, "Esquema de Respaldos (Backups), cómo elegir el adecuado," *YouTube*, 1 de marzo de 2017. [En línea]. Disponible en: [https://www.youtube.com/watch?v=KIBewK7\\_Ha4](https://www.youtube.com/watch?v=KIBewK7_Ha4).
- [22] Velneo, "Base de datos modelo real: tipos de índices," *Velneo Blog*, 1 de marzo de 2017. [En línea]. Disponible en: <https://www.velneo.com/blog/base-de-datos-modelo-real-tipos-de-indices>.
- [23] E. Sánchez Contreras, "Optimización de consultas en MySQL," *Adictos al Trabajo*, 24 de octubre de 2016. [En línea]. Disponible en: <https://adictosaltrabajo.com/2016/10/24/optimizacion-de-consultas-en-mysql/>.
- [24] M. Medic, "Particionamiento de tablas de bases de datos en SQL Server," *SQL Shack*, 4 de diciembre de 2015. [En línea]. Disponible en: <https://www.sqlshack.com/es/particionamiento-de-tablas-de-bases-de-datos-en-sql-server/>.
- [25] EntreData, "Automatización de Tareas en SQL: Procedimientos Almacenados," *EntreData*, 24 de octubre de 2016. [En línea]. Disponible en: <https://www.entredata.org/fundamentos-de-sql/automatizacion-de-tareas-en-sql-procedimientos-almacenados>.
- [26] DBA Experts, "Aumenta el rendimiento con vistas SQL," [Online]. Available: <https://dbaexperts.tech/wp/sql/aumenta-el-rendimiento-con-vistas-sql/>.
- [27] DataCamp, "SQL Triggers," [Online]. Available: <https://www.datacamp.com/es/tutorial/sql-triggers>.
- [28] T. Hamilton, "Data Testing," *Guru99*, [Online]. Available: <https://www.guru99.com/es/data-testing.html>.
- [29] "Índices en bases de datos relacionales," *Codigonautas*, [Online]. Available: <https://codigonautas.com/indices-base-datos-relacional/>.
- [30] "Buenas prácticas de branching y merging," *Chuck's Academy*, [Online]. Available: <https://www.chucksacademy.com/es/topic/git-branching/buenas-practicas-branching-merging>.
- [31] "Buenas prácticas de branching y merging," *Chuck's Academy*, [Online]. Available: <https://www.chucksacademy.com/es/topic/git-branching/buenas-practicas-branching-merging>.
- [32] "Comandos de Git," *Hostinger*, [Online]. Available: <https://www.hostinger.es/tutoriales/comandos-de-git>.
- [33] "Integración continua," *Atlassian*, [Online]. Available: <https://www.atlassian.com/es/continuous-delivery/continuous-integration>.

## CONSIDERACIONES

**Sugerencia para mejorar el trabajo en equipo y habilidades blandas:**

Para optimizar la colaboración, sugiero crear una **tabla de responsabilidades y capacitación**. Esta tabla permitirá monitorear quién es responsable de cada tema, qué actividades se han realizado para capacitar a los compañeros y cuándo se realizaron. Esto fomenta la responsabilidad individual y la transparencia en el equipo.

Ejemplo de tabla de seguimiento:

Responsable	Tema Asignado	Fecha de asignación	Fecha de culminación	Fecha de Capacitación	Capacitación a Compañeros	Observaciones
Guissela Franco	Consultas			01/12/2025	Índices y optimización	Uso de EXPLAIN
Danna López	Seguridad			03/12/2025	Cifrado y control de acceso	Implementación

### Mejoras en habilidades blandas:

**Comunicación efectiva:** Promover reuniones de retroalimentación para que todos los miembros intercambien ideas y soluciones. Fomentar la participación activa en las discusiones y evitar el trabajo mecánico.

**Investigación y curiosidad:** Incentivar a los miembros a investigar profundamente sobre los temas, identificando problemas no documentados y buscando soluciones innovadoras.

**Colaboración activa:** Fomentar un ambiente de colaboración, promoviendo sesiones de brainstorming y revisiones entre compañeros, y asegurando que todos estén alineados con el progreso del proyecto.

**Responsabilidad colectiva:** Asegurar que los miembros del equipo no solo sean responsables de sus tareas individuales, sino también del éxito global del proyecto. Esto incluye apoyar a los compañeros en su aprendizaje.

### Temáticas Disponibles

- El grupo es libre de elegir una temática sin repetirse con los demás grupos, seguir el ejemplo indicado

### Entregables



```
/Project-AEROLINEAS
/Presentaciones
  Proyecto.pptx
/Informe
  Informe_Proyecto.pdf
/Modelados
  Modelo_ER_Conceptual.png
  Modelo_ER_Logico.png
  Modelo_ER_Fisico.png
/Diccionario_De_Datos
  diccionario_datos.xlsx
/Responsabilidades
  responsabilidades_equipo.xlsx
/Scripts
  /Modelado
    crear_tablas.sql
    relaciones_integridad.sql
  /Seguridad
    crear_rols.sql
    cifrado_datos.sql
  /Auditoria
    activar_auditoria.sql
  /Optimización
    crear_indices.sql
    optimizar_consultas.sql
README.md
```

## EXPLICACION

### **Presentación (PPT):**

Crear una carpeta llamada Presentaciones donde se suba el archivo .ppt o .pptx correspondiente a la explicación del proyecto, los objetivos, las actividades, y los resultados alcanzados.

### **Documento Informe:**

- Subir el informe detallado del proyecto en formato .docx o .pdf, incluyendo:
  - Resumen ejecutivo
  - Descripción de cada fase del proyecto
  - Resultados obtenidos
  - Conclusiones

### **Modelados (ER):**

Crear una carpeta llamada Modelados para almacenar los diagramas de modelado ER. Estos pueden estar en formatos como .png, .jpg, .pdf.

- Incluir versiones del modelo conceptual, lógico y físico.

### **Diccionario de Datos:**

Subir un archivo en formato .xlsx o .csv que contenga el diccionario de datos. Este debe incluir detalles como:

- Nombre de la tabla
- Descripción de la tabla
- Campos (nombre, tipo de datos, restricciones, etc.)
- Relación con otras tablas

### **Responsabilidades:**

Subir un archivo que detalle las responsabilidades de cada miembro del equipo, indicando qué tareas corresponden a cada uno. Este archivo puede ser una tabla en formato .xlsx o .docx.

**Script Actividades a Realizar:**

Subir los scripts de las actividades, como la creación de la base de datos, la implementación de procedimientos almacenados, vistas, triggers, etc. Estos archivos pueden ser .sql o .sh (si son scripts de shell para automatizar tareas). Estos scripts deben estar organizados en carpetas según la actividad, por ejemplo, Scripts/Modelado, Scripts/Seguridad, Scripts/Auditoría, etc.

**RUBRICA**

Criterio	Descripción	Puntos
<b>1. Modelado de Base de Datos y Diccionario de Datos</b>		<b>8</b>
<b>Diseño del Modelo Conceptual, Lógico y Físico</b>	El modelo ER refleja las entidades y relaciones correctamente.	4
<b>Diccionario de Datos</b>	El diccionario de datos es detallado, claro y bien estructurado, incluye tablas, campos, relaciones y restricciones.	2
<b>Restricciones de Integridad Referencial</b>	Se definen correctamente las claves primarias y foráneas entre las tablas.	1
<b>Escalabilidad y Mejores Prácticas</b>	El modelo incluye prácticas recomendadas para la escalabilidad y la integración de sistemas de reservas.	1
<b>2. Seguridad, Auditoría y Control de Acceso</b>		<b>8</b>
<b>Políticas de Acceso y Seguridad</b>	Roles y permisos bien definidos para controlar el acceso a las tablas y vistas.	3
<b>Cifrado de Datos Sensibles</b>	Se implementa correctamente el cifrado de datos sensibles, como contraseñas y detalles de pago.	2

<b>Auditoría y Registro de Eventos</b>	Se habilitan herramientas de auditoría para monitorear el acceso y las actividades de los usuarios en la base de datos.	3
<b>3. Respaldos y Recuperación de Datos</b>		<b>5</b>
<b>Respaldos Completos e Incrementales</b>	Los respaldos completos e incrementales están implementados y explicados correctamente.	3
<b>Respaldos en Caliente</b>	Se implementa correctamente el respaldo sin interrumpir el servicio (hot backups).	2
<b>4. Optimización y Rendimiento de Consultas</b>		<b>5</b>
<b>Índices y Optimización de Consultas</b>	Se implementan índices apropiados y se optimizan consultas SQL con herramientas como EXPLAIN.	3
<b>Particionamiento de Tablas</b>	El particionamiento de tablas está correctamente implementado y explicado.	2
<b>5. Procedimientos Almacenados, Vistas y Triggers</b>		<b>5</b>
<b>Procedimientos Almacenados</b>	Se crean procedimientos almacenados para cálculos y tareas recurrentes.	2
<b>Vistas y Simplificación de Consultas</b>	Se crean vistas para simplificar consultas complejas y mejorar el acceso a datos.	2
<b>Triggers para Auditoría y Control de Cambios</b>	Se implementan triggers para mantener un historial de cambios y automatizar tareas.	1
<b>6. Monitoreo y Optimización de Recursos</b>		<b>2</b>

<b>7. Git y Control de Versiones</b>		<b>2</b>
<b>Uso de Git y Control de Versiones</b>	El repositorio está correctamente configurado, con commits claros y frecuentes.	1
<b>Colaboración y Flujo de Trabajo en Git</b>	Se siguen buenas prácticas en el flujo de trabajo (uso de ramas, fusión, etc.).	1
<b>Total</b>		<b>35</b>