



**UNIVERSIDAD TECNOLÓGICA DE PUEBLA DIVISIÓN DE  
TECNOLOGÍAS DE LA INFORMACIÓN - INGENIERÍA EN  
DESARROLLO Y GESTIÓN DE SOFTWARE**

**MATERIA:**

**ADMINISTRACIÓN DE BASE DE DATOS**

**Producto I**

**Plan de Administración de una Base de Datos Relacional**

**DOCENTE:**

**JOSÉ FRANCISCO ESPINOSA GARITA**

**ALUMNOS:**

**ANDRADE REYES JUSTINO JUAN CARLOS - UTP0002150**

**DOMINGUEZ CASTILLO SALVADOR ESTEBAN - UTP0155077**

**8 ° A**

**FECHA: 10/02/2025**

# Índice de contenido

<b>Introducción</b>	<b>4</b>
<b>Contenido</b>	<b>5</b>
1. Modelo de la base de datos relacional. Normalizado a la 3FN	5
2. Esquema de la base de datos relacional	5
3. Carga de datos	7
1. Terminal con LOAD DATA	7
2. MYSQL WORKBENCH	7
4. Planeación de respaldos de la base de datos	8
5. Procedimiento de respaldo de los datos	11
1. Terminal con LOAD DATA	11
2. MYSQL WORKBENCH	12
6. Proceso de restauración de los respaldos realizados	13
7. Automatización de tareas	14
8. Lineamientos de seguridad en la base de datos relacional	17
9. Revocación de permisos de un usuario	21
10. Borrado de 1 de los usuarios	21
11. Reporte del rendimiento de una base de datos relacional con herramienta de monitoreo	21
1. Terminal (showprocesslist, mysqladmin)	21
2. MYSQL WORKBENCH (dashboard, Client Connection)	23
<b>Conclusión</b>	<b>26</b>
<b>Reflexiones</b>	<b>27</b>
<b>Referencias</b>	<b>34</b>

## Índice de imágenes

<b>1. Modelo E-R Normalizado</b>	<b>5</b>
<b>2.Ejecución de script de esquema de la bd en workbench</b>	<b>7</b>
<b>3.LOAD DATA TERMINAL</b>	<b>7</b>
<b>4.LOAD DATA WORKBENCH</b>	<b>8</b>
<b>5.MysqIDump Cmd</b>	<b>11</b>
<b>7.Restauración</b>	<b>13</b>
<b>8.Evento 1</b>	<b>14</b>
<b>9.Evento 2</b>	<b>15</b>
<b>10.Evento 3</b>	<b>15</b>
<b>11.Evento 4</b>	<b>16</b>
<b>12.Evento 5</b>	<b>17</b>
<b>13.Remove permisos</b>	<b>21</b>
<b>14.Borrar Usuario</b>	<b>21</b>
<b>15.Show Processlist</b>	<b>22</b>
<b>16.Show Status</b>	<b>22</b>
<b>17.Métricas clave</b>	<b>23</b>
<b>18.Métricas clave 2</b>	<b>23</b>

# Introducción

El presente reporte documenta el proceso de diseño, creación y configuración de una base de datos destinada a centralizar y gestionar la información asociada con un sistema de tickets. Este sistema tiene como objetivo facilitar la organización, seguimiento y resolución de incidencias o solicitudes de clientes mediante la interacción con representantes y administradores. El proyecto se desarrolla como parte de un enfoque práctico en la materia de administración de bases de datos, con el fin de aplicar lo aprendido en la gestión estructurada de datos y la elaboración de un plan integral de administración de bases de datos relacional.

La base de datos fue implementada utilizando MySQL 9.1, un sistema de gestión de bases de datos robusto y ampliamente utilizado, y se gestionó a través de MySQL Workbench, una herramienta gráfica que facilita el diseño, la administración y la visualización de esquemas y relaciones. Durante el proceso de desarrollo, se adoptaron principios de diseño relacional con el fin de garantizar un esquema eficiente, normalizado y escalable, capaz de manejar el crecimiento futuro de los datos de forma efectiva.

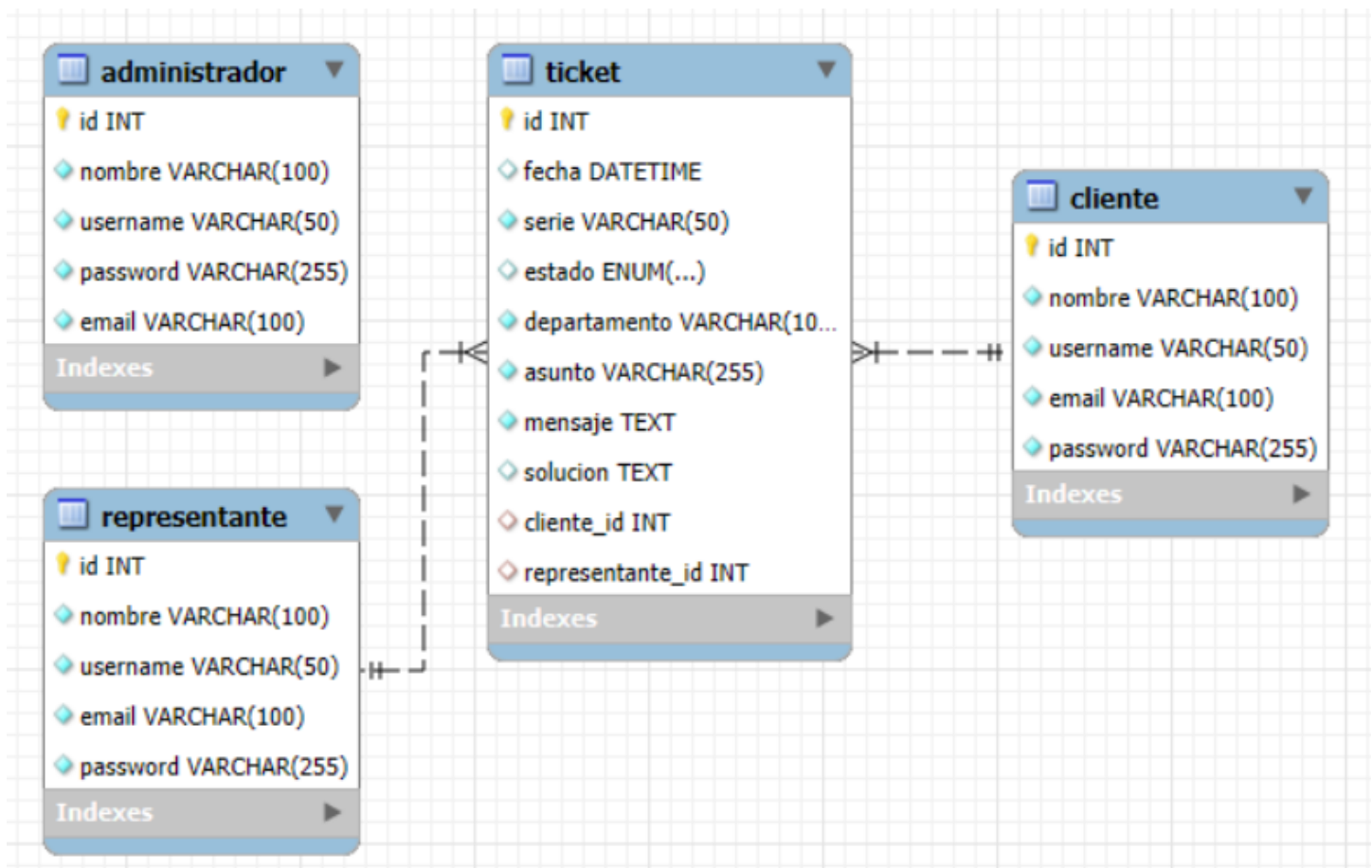
Con base en esta estructura, se ha diseñado un plan de administración de la base de datos relacional que abarca prácticas de seguridad, optimización de consultas, respaldo de datos, control de acceso y mantenimiento preventivo. Este plan asegura el rendimiento constante de la base de datos, minimizando el riesgo de pérdida de información y asegurando la integridad y disponibilidad de los datos en todo momento.

La base de datos incluye varias tablas que almacenan información clave sobre los usuarios del sistema (administradores, clientes y representantes), así como los tickets generados para la gestión de solicitudes. Las tablas y relaciones están diseñadas para optimizar el rendimiento y la integridad de los datos, permitiendo consultas eficientes, generación de reportes y análisis detallados. Además, se ha implementado un sistema de control de acceso mediante autenticación de usuarios, donde los administradores tienen acceso completo a la gestión del sistema, mientras que los clientes y representantes tienen permisos específicos de acuerdo a su rol.

Este reporte detalla las decisiones técnicas tomadas durante la creación del esquema, los procesos de carga de datos y la implementación de relaciones entre tablas. También se abordan los retos enfrentados durante el desarrollo, así como las soluciones aplicadas. El resultado es una base de datos confiable y funcional que respalda el funcionamiento del sistema de tickets, proporcionando una herramienta robusta para la gestión eficiente de incidencias y la mejora de la atención al cliente, todo ello dentro de un marco de administración adecuado y sustentable.

# Contenido

## 1. Modelo de la base de datos relacional. Normalizado a la 3FN



### 1. Modelo E-R Normalizado

## 2. Esquema de la base de datos relacional

```
CREATE DATABASE TicketSystem;
```

```
USE TicketSystem;
```

```
-- Tabla Administrador
```

```
CREATE TABLE Administrador (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    username VARCHAR(50) UNIQUE NOT NULL,  
    password VARCHAR(255) NOT NULL,  
    email VARCHAR(100) UNIQUE NOT NULL
```

```
);
```

```
-- Tabla Cliente
```

CREATE TABLE Cliente (

id INT AUTO\_INCREMENT PRIMARY KEY,  
nombre VARCHAR(100) NOT NULL,  
username VARCHAR(50) UNIQUE NOT NULL,  
email VARCHAR(100) UNIQUE NOT NULL,  
password VARCHAR(255) NOT NULL

);

-- Tabla Representante

CREATE TABLE Representante (

id INT AUTO\_INCREMENT PRIMARY KEY,  
nombre VARCHAR(100) NOT NULL,  
username VARCHAR(50) UNIQUE NOT NULL,  
email VARCHAR(100) UNIQUE NOT NULL,  
password VARCHAR(255) NOT NULL

);

-- Tabla Ticket

CREATE TABLE Ticket (

id INT AUTO\_INCREMENT PRIMARY KEY,  
fecha DATETIME DEFAULT CURRENT\_TIMESTAMP,  
serie VARCHAR(50) UNIQUE NOT NULL,  
estado ENUM('Abierto', 'En Proceso', 'Cerrado') DEFAULT 'Abierto',  
departamento VARCHAR(100) NOT NULL,  
asunto VARCHAR(255) NOT NULL,  
mensaje TEXT NOT NULL,  
solucion TEXT,  
cliente\_id INT, -- Relación con el Cliente  
representante\_id INT, -- Relación con el Representante  
FOREIGN KEY (cliente\_id) REFERENCES Cliente(id) ON DELETE CASCADE,  
FOREIGN KEY (representante\_id) REFERENCES Representante(id) ON DELETE SET NULL

);

Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
8	20:58:10	USE TicketSystem	0 row(s) affected	0.000 sec
9	20:58:10	CREATE TABLE Administrador ( id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(100) N...	0 row(s) affected	0.094 sec
10	20:58:10	CREATE TABLE Cliente ( id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(100) NOT NU...	0 row(s) affected	0.235 sec
11	20:58:10	CREATE TABLE Ticket ( id INT AUTO_INCREMENT PRIMARY KEY, fecha DATETIME DEFAULT CU...	0 row(s) affected	0.140 sec
12	20:58:11	CREATE TABLE Representante ( id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(100) ...	0 row(s) affected	0.172 sec
13	21:02:54	DROP DATABASE 'ticketsystem'	4 row(s) affected	0.234 sec
14	21:02:56	CREATE DATABASE TicketSystem	1 row(s) affected	0.016 sec
15	21:02:56	USE TicketSystem	0 row(s) affected	0.016 sec
16	21:02:56	CREATE TABLE Administrador ( id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(100) N...	0 row(s) affected	0.188 sec
17	21:02:57	CREATE TABLE Cliente ( id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(100) NOT NU...	0 row(s) affected	0.187 sec
18	21:02:57	CREATE TABLE Representante ( id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(100) ...	0 row(s) affected	0.062 sec
19	21:02:57	CREATE TABLE Ticket ( id INT AUTO_INCREMENT PRIMARY KEY, fecha DATETIME DEFAULT CU...	0 row(s) affected	0.157 sec

## 2.Ejecución de script de esquema de la bd en workbench

## 3. Carga de datos

### 1. Terminal con LOAD DATA

**LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 9.1\\Uploads\\tickets.csv'**

**INTO TABLE ticket**

**FIELDS TERMINATED BY ',' ENCLOSED BY ''''**

**LINES TERMINATED BY '\n'**

**IGNORE 1 LINES**

**(FECHA, SERIE, ESTADO, DEPARTAMENTO, ASUNTO, MENSAJE, SOLUCION, CLIENTE\_ID, REPRESENTANTE\_ID)**

**SET FECHA = STR\_TO\_DATE(FECHA, '%Y-%m-%d %H:%i:%s');**

```

MySQL 9.1 Command Line C...
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 9.1.0 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> use ticketsystem
Database changed
mysql> LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 9.1\\Uploads\\tickets.csv'
-> INTO TABLE ticket
-> FIELDS TERMINATED BY ',' ENCLOSED BY ''''
-> LINES TERMINATED BY '\n'
-> IGNORE 1 LINES
-> (FECHA, SERIE, ESTADO, DEPARTAMENTO, ASUNTO, MENSAJE, SOLUCION, CLIENTE_ID, REPRESENTANTE_ID)
-> SET FECHA = STR_TO_DATE(FECHA, '%Y-%m-%d %H:%i:%s');
Query OK, 1000 rows affected (0.09 sec)
Records: 1000 Deleted: 0 Skipped: 0 Warnings: 0

mysql>

```

### 3.LOAD DATA TERMINAL

### 2. MYSQL WORKBENCH

**LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 9.1\\Uploads\\tickets.csv'**

**INTO TABLE ticket**

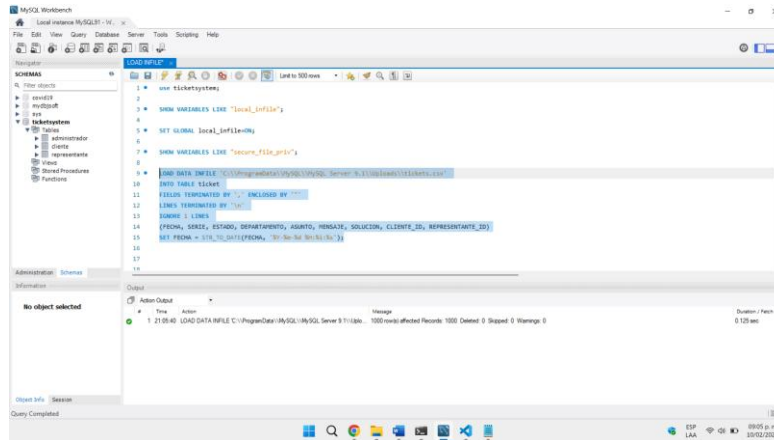
**FIELDS TERMINATED BY ',' ENCLOSED BY ''''**

**LINES TERMINATED BY '\n'**

**IGNORE 1 LINES**

**(FECHA, SERIE, ESTADO, DEPARTAMENTO, ASUNTO, MENSAJE, SOLUCION, CLIENTE\_ID, REPRESENTANTE\_ID)**

**SET FECHA = STR\_TO\_DATE(FECHA, '%Y-%m-%d %H:%i:%s');**



#### 4.LOAD DATA WORKBENCH

### 4. Planeación de respaldos de la base de datos

#### Planeación de Respaldos - TicketSystem

##### Objetivo:

Realizar respaldos regulares de la base de datos para garantizar la disponibilidad y protección de la información crítica, incluyendo usuarios, tickets, contraseñas y otras configuraciones del sistema.

##### Febrero

#### 1. Primer Respaldo

- **Fecha:** Lunes, 12 de febrero
- **Hora:** 02:00 AM
- **Responsable:** Administrador del sistema
- **Datos a respaldar:**
  - **Base de Datos Completa:** Incluye las tablas Administrador, Cliente, Representante y Ticket.
- **Justificación:** Primer respaldo del mes, realizado temprano para evitar interferencias con la carga de trabajo del sistema. Se realiza un respaldo completo para garantizar que todos los datos estén protegidos.

#### 2. Segundo Respaldo



- **Fecha:** Lunes, 26 de febrero
  - **Hora:** 02:00 AM
  - **Responsable:** Administrador del sistema
  - **Datos a respaldar:**
    - **Base de Datos Completa**
  - **Justificación:** Segundo respaldo del mes, realizado de nuevo fuera de las horas de operación para evitar impacto en el sistema. El respaldo cubre todo el conjunto de datos.
- 

## Marzo

### 1. Primer Respaldo

- **Fecha:** Lunes, 5 de marzo
- **Hora:** 02:00 AM
- **Responsable:** Administrador del sistema
- **Datos a respaldar:**
  - **Base de Datos Completa**
- **Justificación:** Se realiza el primer respaldo mensual después de verificar las actividades recientes de los usuarios, incluyendo nuevas cuentas y tickets generados.

### 2. Segundo Respaldo

- **Fecha:** Lunes, 19 de marzo
  - **Hora:** 02:00 AM
  - **Responsable:** Administrador del sistema
  - **Datos a respaldar:**
    - **Base de Datos Completa**
  - **Justificación:** Segundo respaldo del mes para mantener la seguridad y la disponibilidad. Asegura que los cambios recientes sean capturados de manera adecuada.
- 

## Abril

### 1. Primer Respaldo

- **Fecha:** Lunes, 2 de abril
- **Hora:** 02:00 AM
- **Responsable:** Administrador del sistema
- **Datos a respaldar:**
  - **Base de Datos Completa**
- **Justificación:** Inicio del mes, el primer respaldo garantiza que cualquier cambio desde el mes anterior esté respaldado. Es importante asegurarse de que la base de datos está en buen estado antes de la carga de trabajo del mes.

## 2. Segundo Respaldo

- **Fecha:** Lunes, 16 de abril
- **Hora:** 02:00 AM
- **Responsable:** Administrador del sistema
- **Datos a respaldar:**
  - **Base de Datos Completa**
- **Justificación:** Segundo respaldo de abril, con enfoque en las semanas previas al cambio de mes. Se asegura la integridad y disponibilidad de la base de datos con el fin de estar listos para cualquier contingencia.

## Justificación del Plan de Respaldos:

### 1. Frecuencia Bimensual:

El respaldo cada 15 días (aproximadamente) asegura que los datos estén protegidos sin generar sobrecarga en el sistema. Dado que la base de datos no debería experimentar cambios extremos de un día para otro, un respaldo cada dos semanas es suficiente para la mayoría de los casos.

### 2. Hora y Día:

Se eligió realizar los respaldos en las primeras horas de la mañana (02:00 AM), ya que es un horario de baja actividad en el sistema, lo que minimiza el impacto en el rendimiento y las operaciones.

### 3. Responsable:

El responsable de realizar los respaldos es el Administrador del sistema, ya que es quien tiene acceso total y conocimiento para ejecutar y verificar que los respaldos se realicen correctamente.

### 4. Cobertura Completa de Datos:

Los respaldos son completos, cubriendo todas las tablas importantes, como Administrador,

Cliente, Representante y Ticket. Esto asegura que toda la información, incluyendo datos personales y tickets abiertos, esté protegida.

## 5. Procedimiento de respaldo de los datos

### 1. Terminal con LOAD DATA

FEBRERO

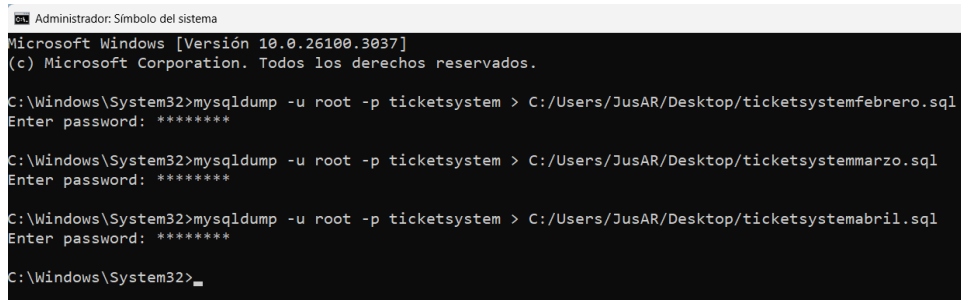
```
mysqldump -u root -p ticketsystem > C:/Users/JusAR/Desktop/ticketsystemfebrero.sql
```

MARZO

```
mysqldump -u root -p ticketsystem > C:/Users/JusAR/Desktop/ticketsystemmarzo.sql
```

ABRIL

```
mysqldump -u root -p ticketsystem > C:/Users/JusAR/Desktop/ticketsystemabril.sql
```



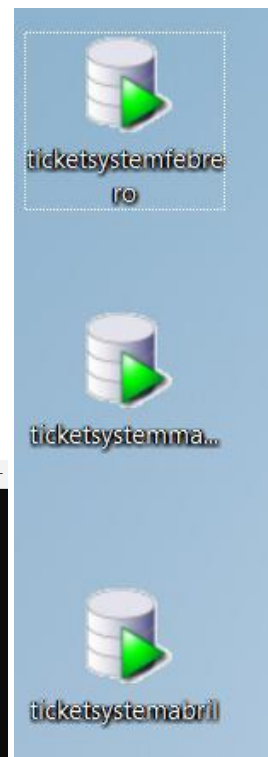
```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.26100.3037]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\System32>mysqldump -u root -p ticketsystem > C:/Users/JusAR/Desktop/ticketsystemfebrero.sql
Enter password: *****

C:\Windows\System32>mysqldump -u root -p ticketsystem > C:/Users/JusAR/Desktop/ticketsystemmarzo.sql
Enter password: *****

C:\Windows\System32>mysqldump -u root -p ticketsystem > C:/Users/JusAR/Desktop/ticketsystemabril.sql
Enter password: *****

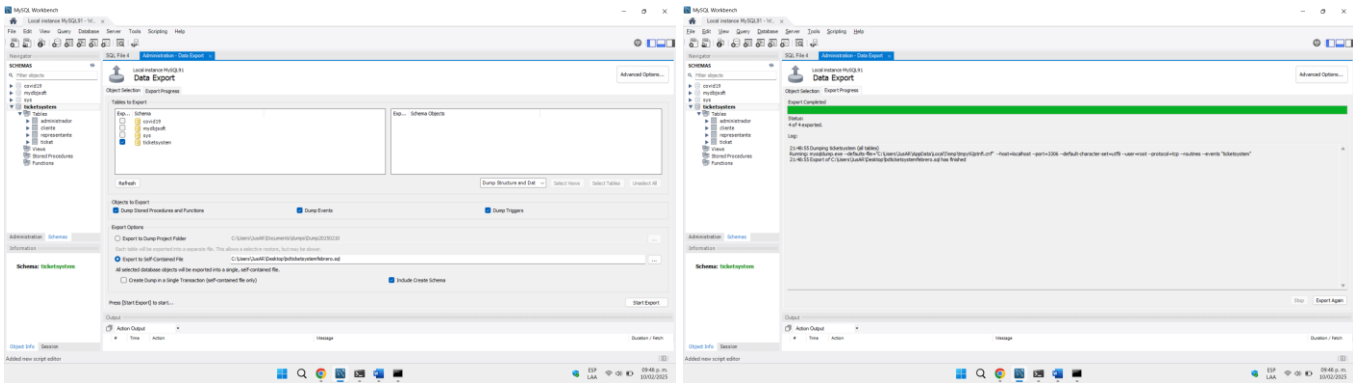
C:\Windows\System32>
```



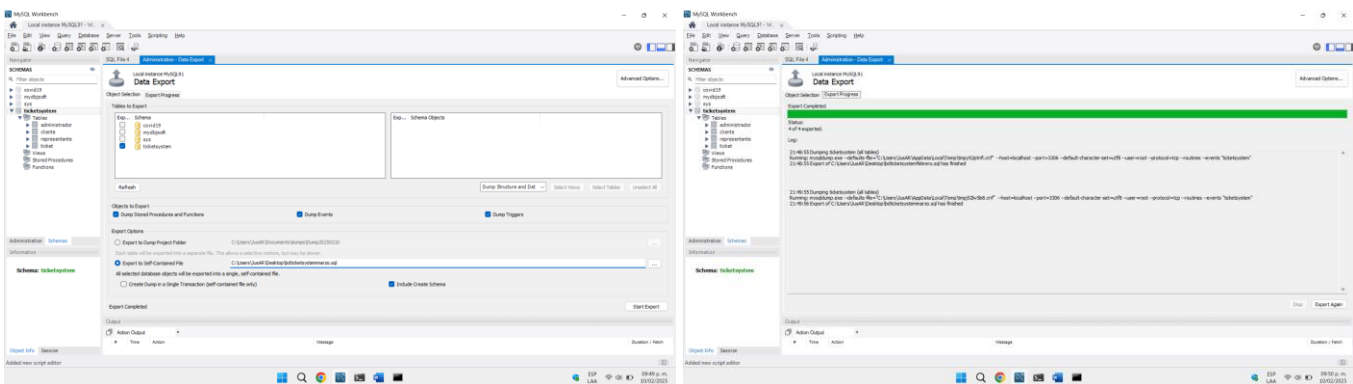
### 5.MysqIDump Cmd

## 2. MYSQL WORKBENCH

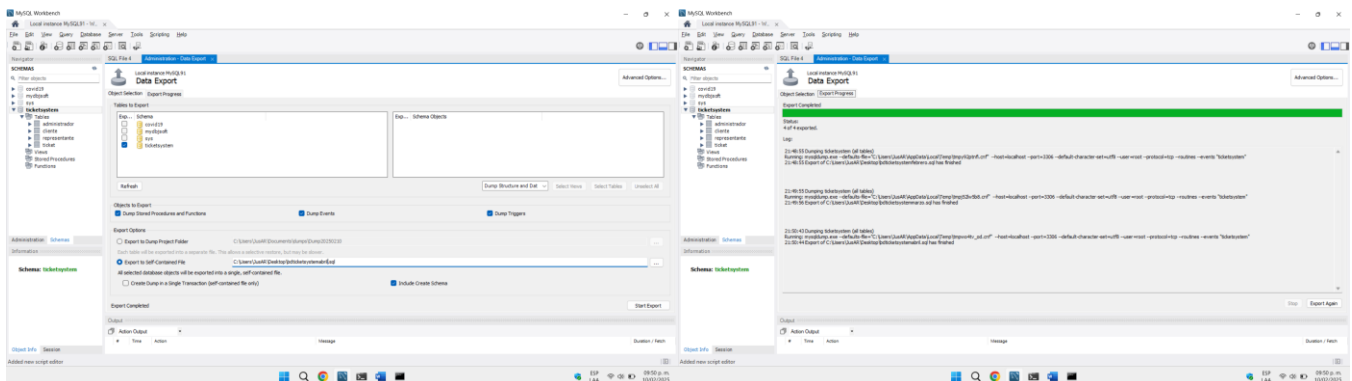
FEBRERO



MARZO



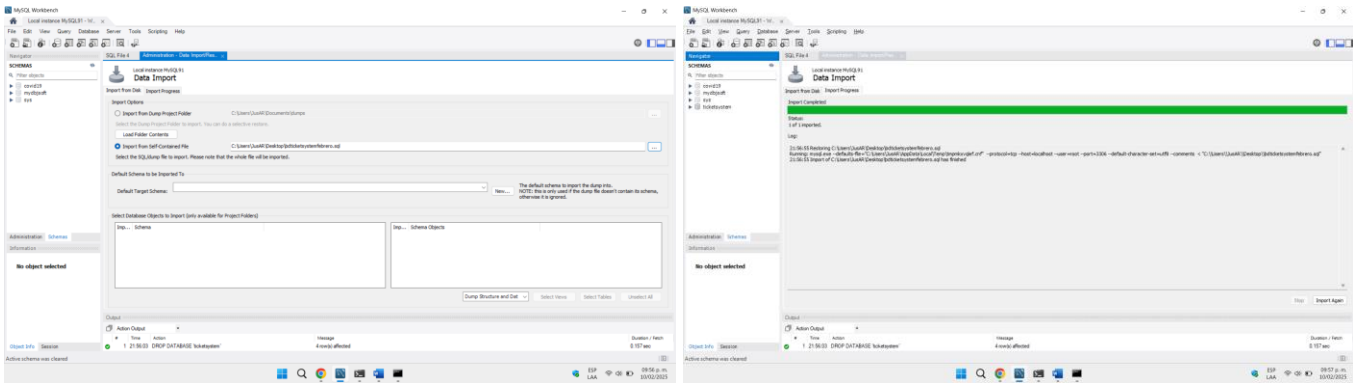
ABRIL



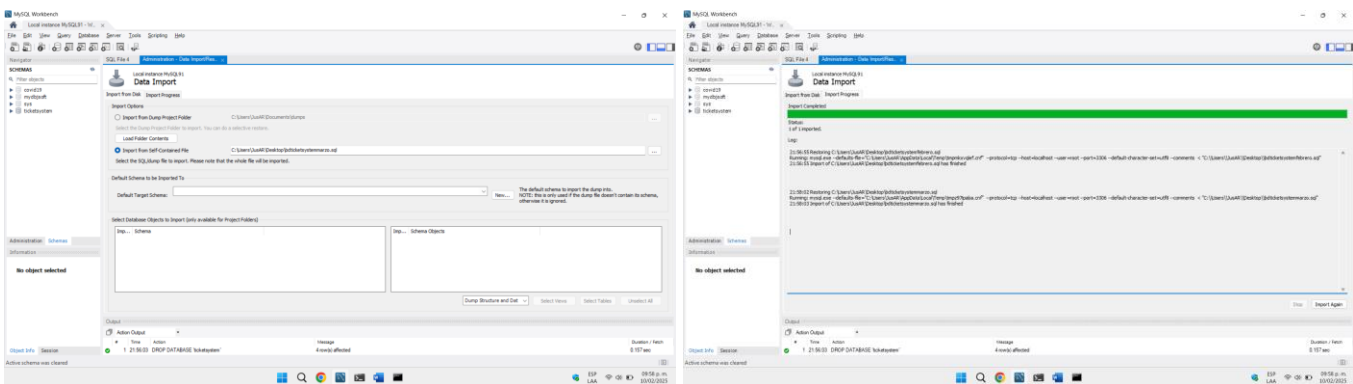
## 6.Respaldo Workbench

## 6. Proceso de restauración de los respaldos realizados

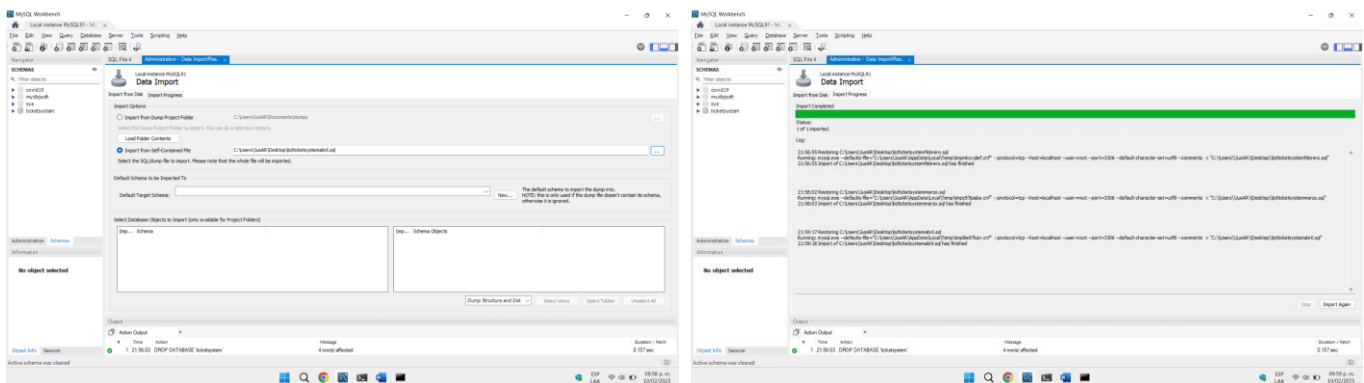
FEBRERO



MARZO



ABRIL



## 7.Restauración

## 7. Automatización de tareas

DELIMITER \$\$

CREATE EVENT evento\_horario\_fijo

ON SCHEDULE EVERY 1 DAY

STARTS '2025-02-10 10:12:00' -- hora en la que debe ejecutarse

DO

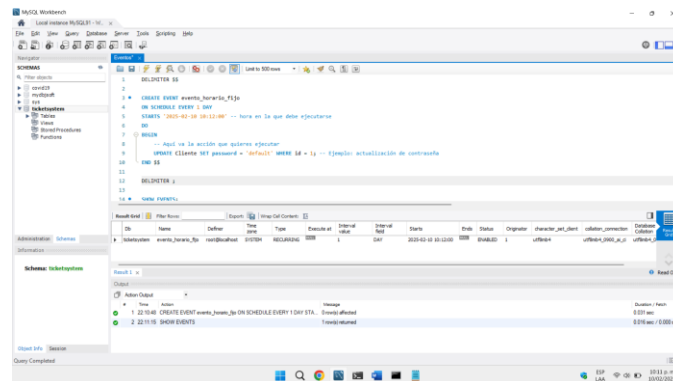
BEGIN

-- Aquí va la acción que quieres ejecutar

UPDATE Cliente SET password = 'default' WHERE id = 1; -- Ejemplo: actualización de contraseña

END \$\$

DELIMITER ;



## 8.Evento 1

DELIMITER \$\$

CREATE EVENT evento\_fecha\_especificas

ON SCHEDULE

EVERY 1 DAY

STARTS '2025-02-10 08:00:00' -- Fecha de inicio

ENDS '2025-02-15 20:00:00' -- Fecha de fin

DO

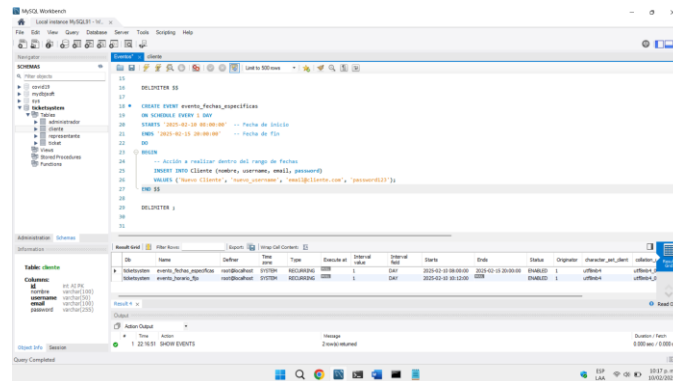
BEGIN

-- Acción a realizar dentro del rango de fechas

```
INSERT INTO Cliente (nombre, username, email, password) VALUES ('Nuevo Cliente', 'nuevo_username',
'email@cliente.com', 'password123');

END $$
```

DELIMITER ;



## 9.Evento 2

DELIMITER \$\$

```
CREATE EVENT evento_recurrente
```

```
ON SCHEDULE EVERY 1 WEEK
```

```
STARTS '2025-02-10 10:00:00' -- Fecha y hora de inicio
```

```
DO
```

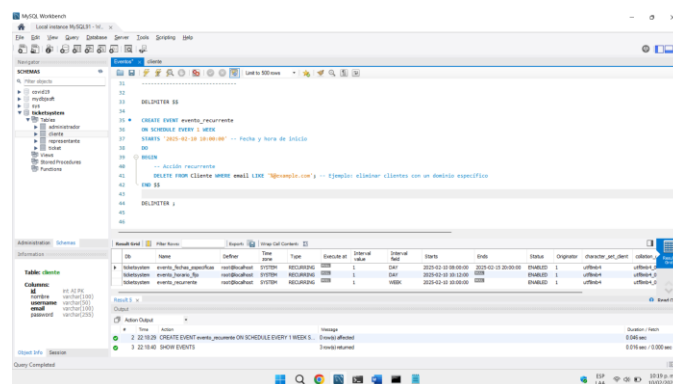
```
BEGIN
```

```
-- Acción recurrente
```

```
DELETE FROM Cliente WHERE email LIKE '%@example.com'; -- Ejemplo: eliminar clientes con un dominio específico
```

```
END $$
```

DELIMITER ;



## 10.Evento 3

DELIMITER \$\$

CREATE EVENT evento\_instruccion\_unica

ON SCHEDULE EVERY 1 DAY

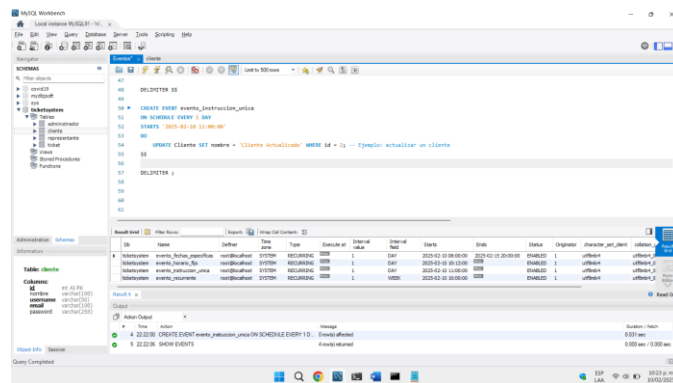
STARTS '2025-02-10 11:00:00'

DO

UPDATE Cliente SET nombre = 'Cliente Actualizado' WHERE id = 2; -- Ejemplo: actualizar un cliente

\$\$

DELIMITER ;



## 11.Evento 4

DELIMITER \$\$

CREATE PROCEDURE actualizar\_cliente()

BEGIN

UPDATE Cliente SET email = 'nuevoemail@cliente.com' WHERE id = 3;

INSERT INTO Cliente (nombre, username, email, password)

VALUES ('Cliente Extra', 'extra\_username', 'extra@cliente.com', 'password\_extra');

END \$\$

DELIMITER ;

DELIMITER \$\$

CREATE EVENT evento\_con\_procedimiento



ON SCHEDULE EVERY 1 DAY

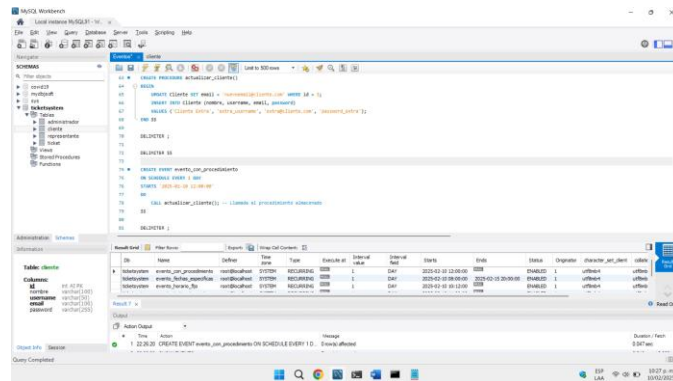
STARTS '2025-02-10 12:00:00'

DO

CALL actualizar\_cliente(); -- Llamada al procedimiento almacenado

\$\$

DELIMITER ;



## 12.Evento 5

### 8. Lineamientos de seguridad en la base de datos relacional

1. Usuario: admin\_user

Nombre de Usuario: admin\_user

Contraseña: admin\_password123

Base de Datos: ticketssystem

Objetos que puede manipular:

Todas las tablas: ticket, representante, cliente, administrador.

Vista general de la base de datos y tablas.

Permisos:

SELECT: Leer datos de todas las tablas.

INSERT: Insertar datos en todas las tablas.

UPDATE: Actualizar datos en todas las tablas.

DELETE: Eliminar datos de todas las tablas.

CREATE: Crear nuevas tablas y objetos.

DROP: Eliminar tablas y objetos.

ALTER: Modificar la estructura de las tablas.

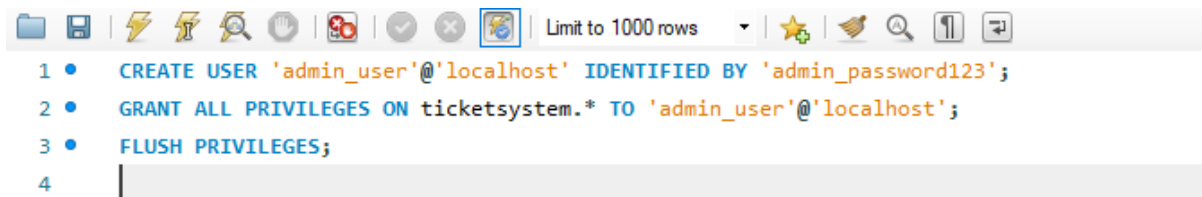
GRANT: Asignar permisos a otros usuarios.

Comando para creación y asignación de permisos en MySQL:

```
CREATE USER 'admin_user'@'localhost' IDENTIFIED BY 'admin_password123';
```

```
GRANT ALL PRIVILEGES ON ticketssystem.* TO 'admin_user'@'localhost';
```

```
FLUSH PRIVILEGES;
```



## 2. Usuario: support\_user

Nombre de Usuario: support\_user

Contraseña: support\_password123

Base de Datos: ticketssystem

Objetos que puede manipular:

Tablas: ticket, representante, cliente.

Permisos:

SELECT: Leer datos de las tablas ticket, representante, cliente.

INSERT: Insertar nuevos tickets en la tabla ticket.

UPDATE: Actualizar información de los tickets en la tabla ticket.

DELETE: Eliminar tickets (si es necesario).

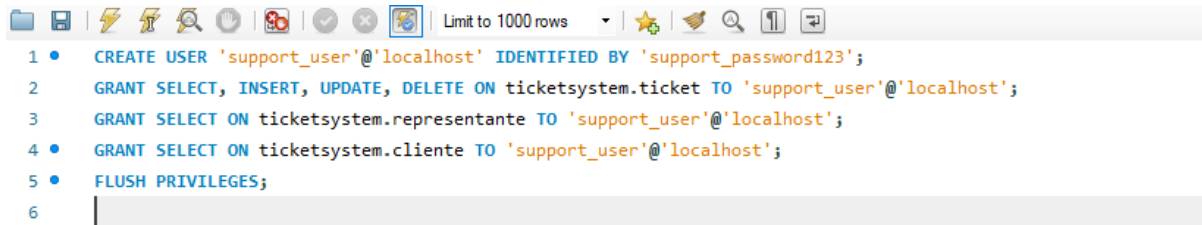
Nota: No tiene acceso a la tabla administrador ni a la creación o eliminación de objetos

.

Comando para creación y asignación de permisos en MySQL:

```
CREATE USER 'support_user'@'localhost' IDENTIFIED BY 'support_password123';
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ticketsystem.ticket TO 'support_user'@'localhost';
GRANT SELECT ON ticketsystem.representante TO 'support_user'@'localhost';
GRANT SELECT ON ticketsystem.cliente TO 'support_user'@'localhost';
FLUSH PRIVILEGES;
```



```
1 • CREATE USER 'support_user'@'localhost' IDENTIFIED BY 'support_password123';
2   GRANT SELECT, INSERT, UPDATE, DELETE ON ticketsystem.ticket TO 'support_user'@'localhost';
3   GRANT SELECT ON ticketsystem.representante TO 'support_user'@'localhost';
4 • GRANT SELECT ON ticketsystem.cliente TO 'support_user'@'localhost';
5   FLUSH PRIVILEGES;
6
```

### 3. Usuario: readonly\_user

**Nombre de Usuario:** readonly\_user

**Contraseña:** readonly\_password123

**Base de Datos:** ticketsystem

**Objetos que puede manipular:**

**Tablas:** ticket, representante, cliente, administrador.

**Permisos:**

**SELECT:** Solo tiene permisos de lectura en todas las tablas.

**Nota:** Este usuario no puede modificar ni eliminar ningún dato, solo visualizar.

**Comando para creación y asignación de permisos en MySQL:**

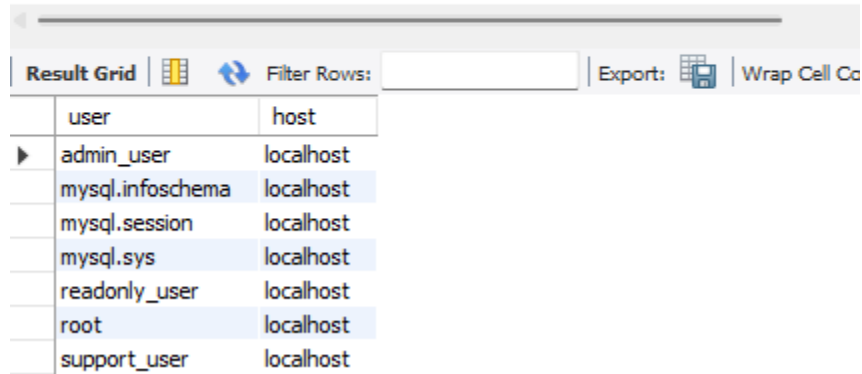
```
CREATE USER 'readonly_user'@'localhost' IDENTIFIED BY 'readonly_password123';
GRANT SELECT ON ticketsystem.* TO 'readonly_user'@'localhost';
FLUSH PRIVILEGES;
```



```
1 • CREATE USER 'readonly_user'@'localhost' IDENTIFIED BY 'readonly_password123';
2   GRANT SELECT ON ticketsystem.* TO 'readonly_user'@'localhost';
3   FLUSH PRIVILEGES;
4
```

## Muestra de los 3 Usuarios Creados:

```
1 • SELECT user, host FROM mysql.user;  
2
```



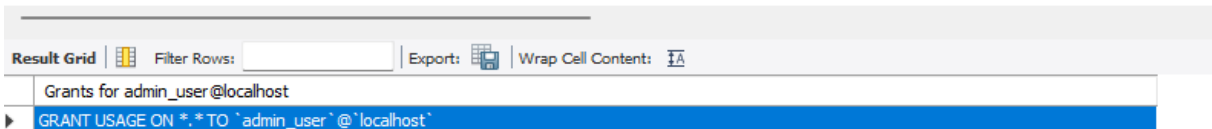
The screenshot shows a database client interface with a 'Result Grid' tab. The grid displays the output of the query 'SELECT user, host FROM mysql.user;'. The columns are 'user' and 'host'. The rows list several MySQL users, all with 'localhost' as the host. The 'admin\_user' row is highlighted with a mouse cursor.

user	host
admin_user	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
readonly_user	localhost
root	localhost
support_user	localhost

## PERMISOS INDIVIDUALES

### admin\_user

```
1 • SHOW GRANTS FOR 'admin_user'@'localhost';  
2
```

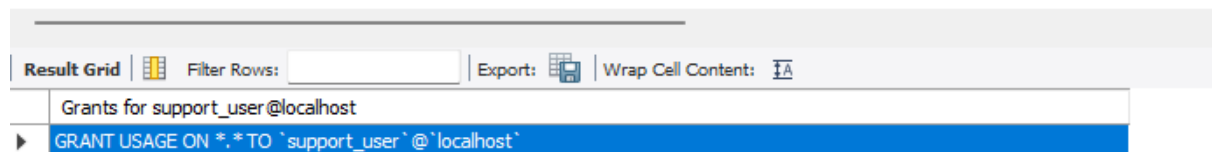


The screenshot shows a database client interface with a 'Result Grid' tab. The grid displays the output of the query 'SHOW GRANTS FOR 'admin\_user'@'localhost';'. The first row is a header: 'Grants for admin\_user@localhost'. The second row shows the grant: 'GRANT USAGE ON \*.\* TO `admin\_user` @ `localhost`'.

Grants for admin_user@localhost
GRANT USAGE ON *.* TO `admin_user` @ `localhost`

### PERMISOS DE support\_user

```
1 • SHOW GRANTS FOR 'support_user'@'localhost';  
2
```



The screenshot shows a database client interface with a 'Result Grid' tab. The grid displays the output of the query 'SHOW GRANTS FOR 'support\_user'@'localhost';'. The first row is a header: 'Grants for support\_user@localhost'. The second row shows the grant: 'GRANT USAGE ON \*.\* TO `support\_user` @ `localhost`'.

Grants for support_user@localhost
GRANT USAGE ON *.* TO `support_user` @ `localhost`

## PERMISOS DE readonly\_user

```
1 • SHOW GRANTS FOR 'readonly_user'@'localhost';  
2
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Grants for readonly_user@localhost				
GRANT USAGE ON *.* TO `readonly_user`@`localhost`				

## 9. Revocación de permisos de un usuario

REVOKE INSERT, UPDATE, DELETE ON \*.\* FROM 'trabajador'@'%';

Output				
Action Output				
#	Time	Action	Message	
✓ 1	22:38:03	REVOKE INSERT, UPDATE, DELETE ON *.* FROM 'trabajador'@'%'	0 row(s) affected	

## 13. Remover permisos

## 10. Borrado de 1 de los usuarios

DROP USER 'trabajador'@'%';

Output				
Action Output				
#	Time	Action	Message	
✓ 1	22:40:49	SELECT User, Host FROM mysql.user LIMIT 0, 500	7 row(s) returned	
✓ 2	22:40:53	DROP USER 'trabajador'@'%'	0 row(s) affected	

## 14. Borrar Usuario

## 11. Reporte del rendimiento de una base de datos relacional con herramienta de monitoreo

### 1. Terminal (showprocesslist, mysqladmin)

Comando Usando showprocesslist

-Muestra los procesos en ejecución en MySQL, útil para detectar bloqueos o consultas lentas.

```
mysql> SHOW PROCESSLIST;
```

Id	User	Host	db	Command	Time	State	Info
5	event_scheduler	localhost	NULL	Daemon	106979	Waiting on empty queue	NULL
9	root	localhost:52427	NULL	Sleep	281		NULL
10	root	localhost:52428	NULL	Sleep	281		NULL
11	root	localhost:52430	NULL	Sleep	2921		NULL
12	root	localhost:52431	NULL	Sleep	1		NULL
14	root	localhost:52704	ticketsystem	Query	0	init	SHOW PROCESSLIST

```
6 rows in set, 1 warning (0.00 sec)

mysql> |
```

## 15.Show Processlist

SHOW STATUS LIKE 'Threads%';

-Revela el estado de los hilos del servidor.

```
mysql> SHOW STATUS LIKE 'Threads%';
```

Variable_name	Value
Threads_cached	0
Threads_connected	5
Threads_created	5
Threads_running	2

```
4 rows in set (0.00 sec)

mysql> |
```

## 16.Show Status

Muestra estadísticas generales de MySQL (consultas por segundo, hilos activos, etc.).

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqladmin -u root -p status
Enter password: ****
Uptime: 107363 Threads: 6 Questions: 1373 Slow queries: 0 Opens: 233 Flush tables: 3 Open tables: 148 Queries per second avg: 0.012
```

**Propósito:** Permite ver las consultas que se están ejecutando actualmente en la base de datos.

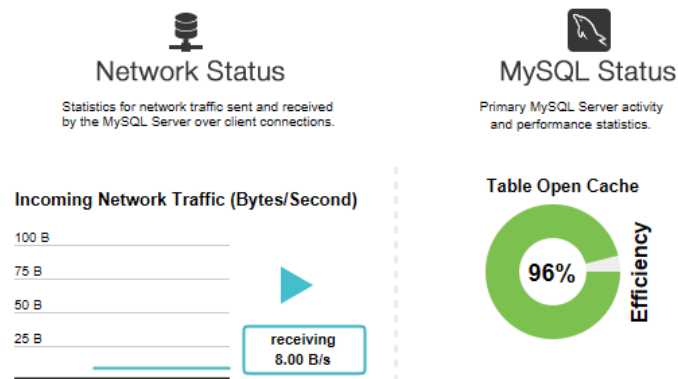
**Información clave:**

- **ID de proceso:** Identificador único de la consulta.
- **Usuario y host:** Muestra el usuario que está ejecutando la consulta y desde qué host.
- **Estado:** Estado actual de la consulta (por ejemplo, "sleeping", "executing").
- **Tiempo de ejecución:** Duración en segundos de la consulta actual.

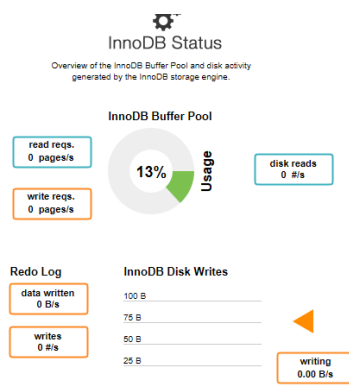
## Comando mysqladmin status:

- **Propósito:** Muestra estadísticas generales del servidor de MySQL.
- **Información clave:**
  - **Threads connected:** Número de hilos activos y conexiones.
  - **Queries per second:** Consultas por segundo.
  - **Slow queries:** Número de consultas que tardan más de lo recomendado.
  - **Open tables:** Número de tablas abiertas en memoria.

## 2. MYSQL WORKBENCH (dashboard, Client Connection)



### 17.Métricas clave



### 18.Métricas clave 2

## Client Connections



Local instance MySQL80  
Client Connections

Threads Connected: 4 Threads Running: 2 Threads Created: 5 Threads Cached: 1 Rejected (over limit): 0  
Total Connections: 17 Connection Limit: 151 Aborted Clients: 0 Aborted Connections: 1 Errors: 0 ⓘ

Id	User	Host	DB	Command	Time	State	Threa...	Type	Name	Paren...	Instrumented
5	event_sched...	localhost	None	Daemon	107612	Waiting on e...	44	FOREGROUND	thread/sql/e...	1	YES
7	None	None	None	Daemon	107612	Suspending	47	FOREGROUND	thread/sql/c...	1	YES
9	root	localhost	None	Sleep	178	None	50	FOREGROUND	thread/sql/o...	0	YES
10	root	localhost	None	Sleep	178	None	51	FOREGROUND	thread/sql/o...	48	YES
16	root	localhost	None	Query	0	executing	57	FOREGROUND	thread/sql/o...	0	YES
17	root	localhost	None	Sleep	3	None	58	FOREGROUND	thread/sql/o...	0	YES

## Consultas por segundo (QPS):

- **Propósito:** Mide la cantidad de consultas ejecutadas por segundo.
- **Interpretación:** Si es alto, puede indicar una carga pesada en el servidor y necesidad de optimización.

## Uso de conexiones:

- **Propósito:** Muestra cuántas conexiones están activas en el servidor.
- **Interpretación:** Un número elevado de conexiones podría sugerir que el servidor está manejando muchas peticiones simultáneas, lo que puede impactar el rendimiento si no está optimizado.

## Uso del buffer pool de InnoDB:

- **Propósito:** Mide cuánta memoria está utilizando InnoDB para almacenar datos en caché.
- **Interpretación:** Si el uso es muy bajo, puede ser útil aumentar el tamaño del buffer pool para mejorar el rendimiento de las consultas.

## 3. phpMyAdmin (Status).

Server: localhost

Databases SQL **Status** Export Import Settings Variables

Server Query statistics All status variables Monitor Advisor

Network traffic since startup: 15 MiB

This MySQL server has been running for 29 days, 12 hours, 4 minutes and 52 seconds. It started up on Mar 23, 2015 at (

Traffic ⓘ	per hour	
Received	4.9 MiB	7.1 KiB
Sent	10.1 MiB	
Total	15 MiB	21.7 KiB

Connections		per hour		%
max. concurrent connections	3	---	---	---
Failed attempts	0	0	0%	
Aborted	147	0.21	0.79%	
Total	19 k	26.19	100.00%	

Processes	ID	User	Host	Database	Command	Time	Status	SQL query ← T →
Kill	18540	whitespa	localhost	None	Sleep	0 ---	---	
Kill	18541	whitespa	localhost	None	Query	0 ---		SHOW PROCESSLIST



**Conexiones activas:**

- **Propósito:** Muestra cuántas conexiones están abiertas en el servidor MySQL.
- **Interpretación:** Un número alto de conexiones simultáneas podría significar que el servidor necesita más recursos para manejar la carga.

**Estadísticas de consultas:**

- **Propósito:** Muestra el número total de consultas ejecutadas en la base de datos.
- **Interpretación:** Un número elevado de consultas puede indicar que el sistema está bajo una carga alta o que se necesita optimización en las consultas.

**Consultas lentas:**

- **Propósito:** Muestra las consultas que superan el tiempo de ejecución definido como "lento".
- **Interpretación:** Las consultas lentas deben ser optimizadas, ya que afectan el rendimiento general del sistema. Las optimizaciones pueden incluir la creación de índices o la reescritura de consultas.

## Conclusión

En este proyecto, se ha diseñado y gestionado una base de datos relacional con el objetivo de centralizar y organizar información de manera eficiente y segura, asegurando un óptimo desempeño a lo largo del tiempo. El modelo de la base de datos fue normalizado hasta la tercera forma normal (3FN) para garantizar la eliminación de redundancias, mejorar la integridad de los datos y optimizar el espacio de almacenamiento. La estructura de las tablas y sus relaciones se definieron de acuerdo con principios de diseño relacional, donde se utilizaron claves primarias y foráneas para establecer vínculos claros entre las distintas entidades, como es el caso de la relación entre clientes y tickets mediante el uso de claves foráneas.

El esquema de la base de datos fue cuidadosamente diseñado para asegurar la correcta implementación de las tablas y sus relaciones. Este esquema fue implementado en un entorno MySQL, utilizando herramientas como MySQL Workbench, PhpMyAdmin y la terminal con el comando LOAD DATA para realizar la carga de datos. A través de estos procesos, se garantizó la integridad y la correcta migración de la información, asegurando que solo se incluyeran datos válidos y actualizados en la base de datos.

El plan de respaldo de la base de datos fue una de las tareas fundamentales para garantizar la disponibilidad de los datos en caso de incidentes imprevistos. Se definieron fechas, horas y responsables para realizar respaldos periódicos, y se implementaron procedimientos específicos con herramientas como MySQL Workbench y PhpMyAdmin. Además, se llevó a cabo el proceso de restauración de los respaldos utilizando diferentes métodos, asegurando que los datos pudieran ser recuperados sin comprometer la integridad del sistema.

Para la automatización de tareas, se implementaron cinco eventos con diferentes tipos de configuración, tales como horarios fijos, fechas de inicio y fin, eventos recurrentes y procedimientos almacenados. Esto permitió gestionar tareas automáticas de manera eficiente y reducir la intervención manual en procesos críticos.

En cuanto a la seguridad de la base de datos, se definieron lineamientos estrictos para la gestión de usuarios, contraseñas y permisos. Se administraron tres usuarios, asignándoles permisos específicos para garantizar el acceso adecuado a los objetos de la base de datos, y se realizó la revocación de permisos y el borrado de un usuario para asegurar la protección de los datos sensibles.

Finalmente, se implementaron estrategias de monitoreo del rendimiento de la base de datos utilizando herramientas como el comando show processlist, el dashboard de MySQL Workbench y los estados de PhpMyAdmin. Estos procedimientos permitieron obtener información valiosa para evaluar la disponibilidad, el rendimiento y la confiabilidad de la base de datos.

En resumen, este proyecto no solo abarcó el diseño y la normalización de una base de datos relacional, sino también la implementación de procesos cruciales para la carga, el respaldo, la automatización y la seguridad de los datos. Las herramientas y metodologías empleadas aseguran un manejo adecuado de la base de datos, garantizando su rendimiento y protección a lo largo del tiempo.

## Reflexiones

- ¿Cómo estructuraste las tablas en la base de datos utilizando el modelo relacional y cómo se manejan las relaciones entre ellas? Proporciona ejemplos de claves primarias y foráneas implementadas.

### 1. Tablas y sus claves primarias:

- **Administrador:** La clave primaria es id, que es un campo entero con incremento automático, lo que garantiza que cada registro de administrador tenga un identificador único.

id INT AUTO\_INCREMENT PRIMARY KEY

- **Cliente:** La clave primaria es también id, con un incremento automático para asegurar la unicidad.

id INT AUTO\_INCREMENT PRIMARY KEY

- **Representante:** Al igual que las otras tablas, la clave primaria es id, garantizando un identificador único.

id INT AUTO\_INCREMENT PRIMARY KEY

- **Ticket:** La clave primaria es id, asegurando la unicidad de cada ticket.

id INT AUTO\_INCREMENT PRIMARY KEY

### 2. Relaciones entre las tablas:

Las relaciones entre las tablas se manejan utilizando **claves foráneas (FK)**. Estas claves foráneas vinculan una tabla con otra, creando dependencias y ayudando a mantener la integridad referencial.

- **Relación Cliente - Ticket:** Un cliente puede tener varios tickets, pero cada ticket está asociado a un único cliente. La relación se establece mediante la clave foránea cliente\_id en la tabla Ticket, que referencia el id de la tabla Cliente. Se usa la opción ON DELETE CASCADE para que, si un cliente es eliminado, también se eliminen los tickets asociados.

cliente\_id INT, -- Relación con el Cliente

FOREIGN KEY (cliente\_id) REFERENCES Cliente(id) ON DELETE CASCADE

- **Relación Representante - Ticket:** Un representante puede estar asignado a varios tickets, pero cada ticket tiene solo un representante. La relación se establece mediante la clave foránea representante\_id en la tabla Ticket, que referencia el id de la tabla Representante. Se usa la opción ON DELETE SET NULL para que, si un representante es eliminado, los tickets asociados tengan el campo representante\_id establecido en NULL en lugar de eliminar los tickets.

representante\_id INT, -- Relación con el Representante

FOREIGN KEY (representante\_id) REFERENCES Representante(id) ON DELETE SET NULL

### 3. Ejemplo de Relaciones:

- Un **ticket** podría estar relacionado con un **cliente** y un **representante** mediante sus respectivas claves foráneas. Por ejemplo:
  - **Ticket 1:** Creado por el **Cliente 2** y asignado al **Representante 3**.
  - La tabla Ticket tendría cliente\_id = 2 y representante\_id = 3, lo que indica que este ticket está relacionado con esos dos registros.
- **Explica el proceso de normalización que seguiste para diseñar la base de datos. ¿Qué formas normales aplicaste y por qué decidiste llevar las tablas a esos niveles de normalización?**

#### 1. Primera Forma Normal (1FN):

Para que la base de datos esté en la **Primera Forma Normal (1FN)**, deben cumplirse los siguientes requisitos:

- Todos los valores de las columnas deben ser atómicos, es decir, no deben contener múltiples valores en una sola celda.
- Las columnas deben tener un valor único para cada registro en la tabla.

#### Aplicación de la 1FN:

- Las tablas Administrador, Cliente, Representante, y Ticket están estructuradas de tal manera que cada columna tiene un único valor para cada fila (por ejemplo, los valores en la columna nombre no tienen múltiples valores).
- En la tabla Ticket, el campo mensaje y solucion están diseñados para contener texto de longitud variable, pero son atómicos, ya que representan un solo mensaje o solución.

#### Ejemplo:

- La columna estado en la tabla Ticket tiene valores definidos como ENUM('Abierto', 'En Proceso', 'Cerrado'), lo que asegura que no se almacenen múltiples estados para un solo ticket.

#### 2. Segunda Forma Normal (2FN):

La **Segunda Forma Normal (2FN)** se logra cuando:

- La base de datos ya está en 1FN.
- Todos los atributos no clave dependen completamente de la clave primaria (es decir, no deben existir dependencias parciales).

#### Aplicación de la 2FN:

- En las tablas Administrador, Cliente, y Representante, la clave primaria es el id, y todas las columnas dependen directamente de esta clave. No hay dependencias parciales.

- En la tabla Ticket, las columnas serie, fecha, departamento, asunto, mensaje, y solucion dependen completamente de la clave primaria id. No existen atributos que dependan solo de una parte de la clave primaria (ya que solo tiene un campo id como clave primaria).

### 3. Tercera Forma Normal (3FN):

Para que una base de datos esté en **Tercera Forma Normal (3FN)**, debe cumplir con:

- Estar en 2FN.
- No tener dependencias transitivas, es decir, los atributos no clave no deben depender de otros atributos no clave.

#### Aplicación de la 3FN:

- En las tablas Administrador, Cliente, y Representante, no hay dependencias transitivas. Por ejemplo, el email y el username de los administradores dependen solo de la clave primaria id, no de otros atributos como nombre.
- En la tabla Ticket, no hay dependencias transitivas entre los campos. La relación entre ticket, cliente\_id, y representante\_id está gestionada mediante claves foráneas, lo que evita dependencias redundantes.
- **Describe el proceso que implementaste para la carga inicial de datos en la base de datos. ¿Cómo aseguraste la integridad de los datos durante este proceso y qué estrategias utilizaste para respaldar y restaurar la base de datos de manera eficiente?**

#### 1. Carga Inicial de Datos:

Para cargar los datos en la base de datos, se utilizó el comando LOAD DATA de MySQL, que es eficiente y rápido para cargar grandes cantidades de datos desde archivos de texto (por ejemplo, archivos CSV) a las tablas de la base de datos.

#### Proceso de carga de datos con LOAD DATA:

- **Preparación de archivos CSV:** Los datos iniciales de las tablas Administrador, Cliente, Representante y Ticket se almacenaron en archivos CSV, donde cada fila representaba un registro y cada columna correspondía a un campo en las tablas de la base de datos.
- **Formato del archivo CSV:** Los archivos fueron estructurados de manera que los valores estuvieran separados por comas, con cada línea representando un nuevo registro. Los campos que contienen texto se aseguraron de estar rodeados por comillas dobles para evitar conflictos con los delimitadores de campo.
- **Carga de datos:** Utilicé el comando LOAD DATA INFILE para cargar los datos desde el archivo CSV a la tabla correspondiente. Ejemplo:

```
LOAD DATA INFILE '/path/to/archivo.csv'  
INTO TABLE Administrador  
FIELDS TERMINATED BY ','  
ENCLOSED BY '"'  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;
```

Este comando carga los datos desde un archivo CSV, define el delimitador de campo (,), la comilla de envoltura para los textos ("), y asegura que se ignore la primera fila que contiene los nombres de las columnas.

## 2. Asegurar la Integridad de los Datos:

Durante el proceso de carga, se implementaron varias estrategias para asegurar que los datos se insertaran de forma correcta y consistente:

- **Validación de datos antes de la carga:** Antes de cargar los archivos CSV, se realizaron validaciones previas sobre los datos, como asegurarse de que:
  - No existieran valores duplicados en campos únicos, como username o email.
  - Los valores de los campos fecha y estado en la tabla Ticket estuvieran en el formato adecuado.
  - Las claves foráneas (cliente\_id y representante\_id) coincidieran con los registros existentes en las tablas Cliente y Representante.
- **Integridad referencial:** Durante la carga, MySQL asegura la integridad referencial gracias a las restricciones de las claves foráneas (FOREIGN KEY). Esto significa que, si intentamos insertar un cliente\_id o representante\_id que no existe en sus respectivas tablas, la base de datos rechazará la inserción y no permitirá la carga de datos incorrectos.
- **Comprobación de errores:** Se revisaron los errores generados durante el proceso de carga, ya sea en la consola de MySQL o en el archivo de log de MySQL, para detectar registros que no pudieron ser cargados debido a violaciones de restricciones o problemas con los datos.

## 3. Respaldo de la Base de Datos:

Para respaldar la base de datos de manera eficiente, se utilizaron las herramientas de **importación y exportación** de **MySQL Workbench**. Estas herramientas permiten realizar respaldos completos de la base de datos y restaurarlos de manera rápida en caso de ser necesario.

### Proceso de Respaldo:

- **Exportación de la base de datos:** Utilicé la opción **"Data Export"** en **MySQL Workbench** para realizar un respaldo completo de la base de datos. Esta opción permite exportar tanto la estructura de las tablas como los datos.

- Seleccioné las tablas Administrador, Cliente, Representante, y Ticket para exportar.
- Elegí el formato SQL para que la base de datos pueda ser restaurada posteriormente con los mismos datos y estructuras.
- Configuré la opción para incluir tanto la estructura como los datos de las tablas.
- **Respaldo en archivos comprimidos:** Para optimizar el espacio de almacenamiento y facilitar la transferencia, utilicé la opción de exportar en formato comprimido (por ejemplo, .sql.gz), que reduce el tamaño del archivo de respaldo.

**Comando SQL de respaldo:** En la línea de comandos de MySQL, el comando para exportar la base de datos completa puede verse de la siguiente manera:

```
mysqldump -u username -p TicketSystem > respaldo_ticket_system.sql
```

#### 4. Restauración de la Base de Datos:

Para restaurar la base de datos desde el respaldo:

- **Importación de la base de datos:** Se utilizó la opción **"Data Import"** de **MySQL Workbench** para importar el archivo de respaldo generado anteriormente.
  - Seleccioné el archivo de respaldo (.sql o .sql.gz).
  - Al importar, las tablas y los datos se restauran tal como estaban en el momento del respaldo.

**Comando SQL de restauración:** En caso de usar la línea de comandos, el comando para restaurar desde un archivo de respaldo sería:

```
mysql -u username -p TicketSystem < respaldo_ticket_system.sql
```

#### 5. Estrategias de Respaldo y Restauración Eficientes:

- **Automatización de respaldos:** Se podría programar un respaldo automático a intervalos regulares utilizando herramientas de automatización, como cronjobs en Linux, para asegurar que siempre haya un respaldo reciente disponible.
- **Verificación de respaldos:** Después de realizar un respaldo, es recomendable verificar que el archivo de respaldo se pueda restaurar correctamente. Esto se puede hacer restaurando el respaldo en un entorno de pruebas y comprobando que todos los datos y relaciones se restauren adecuadamente.
- **Monitoreo de la base de datos:** Configurar alertas para monitorear la base de datos durante la carga de datos o la restauración para detectar problemas como violaciones de integridad de datos, caídas de la base de datos o errores de recursos.
- **¿Cómo gestionaste los eventos en la base de datos? Explica si utilizaste triggers o procedimientos almacenados para realizar acciones automáticas**

**ante ciertos eventos (como inserciones, actualizaciones o eliminaciones de registros).**

#### **1. Evento con horario fijo (sin recurrencia):**

Este evento se ejecuta en un **momento específico** y no tiene recurrencia. En mi ejemplo, creé un evento que insertaba un nuevo cliente en la base de datos en una fecha y hora específicas. Este tipo de evento es útil cuando necesitas realizar acciones puntuales en momentos predefinidos, sin que se repitan.

#### **2. Evento con fechas de inicio y fin:**

Este evento tiene un **rango de fechas definido**, en el que se ejecuta repetidamente. Se configura con fechas de **inicio** y **fin** específicas. En mi ejemplo, el evento realiza una acción repetitiva de forma diaria durante ese rango de tiempo.

#### **3. Evento recurrente:**

En este caso, el evento se ejecuta **de manera continua** en intervalos específicos, como un evento que se ejecuta cada día. Este tipo de eventos es útil para tareas como mantenimiento periódico o actualizaciones programadas.

#### **4. Evento con una instrucción SQL:**

Este evento es simple y consiste en la ejecución de una **única instrucción SQL**, como una actualización o eliminación.

#### **5. Evento con bloque de instrucciones y procedimiento almacenado:**

Finalmente, en este caso, utilicé un **bloque de instrucciones SQL** que se ejecutan de manera más compleja, e incluso llamando a un **procedimiento almacenado** para realizar múltiples tareas o acciones.

- **¿Qué medidas de seguridad implementaste en la base de datos para proteger los datos sensibles de la empresa? Además, ¿qué herramientas o técnicas utilizaste para monitorear el rendimiento de la base de datos y garantizar su disponibilidad y confiabilidad?**

#### **Medidas de Seguridad Implementadas:**

Para proteger los datos sensibles de la empresa, implementamos las siguientes medidas de seguridad:

##### **1. Creación de usuarios con permisos específicos:**

Se crearon tres usuarios con diferentes niveles de acceso:

- **admin\_user:** Con todos los permisos, para gestionar la base de datos.



- **support\_user:** Solo puede ver y modificar datos relacionados con los tickets, pero no puede alterar la estructura de la base de datos.
- **readonly\_user:** Solo tiene permisos de lectura en todas las tablas, sin poder modificar nada.

## 2. **Contraseñas seguras:**

Se asignaron contraseñas fuertes a cada uno de los usuarios para evitar accesos no autorizados.

## 3. **Control de acceso:**

Los usuarios solo pueden acceder a lo que realmente necesitan, lo que ayuda a reducir el riesgo de que alguien vea o modifique información que no les corresponde.

---

## **Herramientas y Técnicas para Monitorear el Rendimiento:**

Para garantizar que la base de datos funcione correctamente y se mantenga confiable, utilizamos las siguientes herramientas:

### 1. **Comandos de terminal (SHOW PROCESSLIST y mysqladmin status):**

Estos comandos nos permiten ver qué procesos están en ejecución, cuántas conexiones están activas y si hay consultas lentas o bloqueos que puedan afectar el rendimiento.

### 2. **MySQL Workbench (Dashboard):**

Con esta herramienta, pudimos monitorear en tiempo real las consultas por segundo, el uso de las conexiones y la memoria utilizada por el sistema, lo que nos ayuda a detectar posibles problemas de rendimiento.

### **phpMyAdmin (Status):**

Usamos la sección de "Status" en phpMyAdmin para revisar el número de consultas realizadas, las conexiones activas y las consultas lentas. Esto nos permite ver si hay algo que necesita ser optimizado para mejorar la eficiencia.

## Referencias

- Administrar una BD Relacional - documentación de Apuntes de Informática (FP) - 2025. (s/f). Apuntesinformaticafp.com. Recuperado el 11 de febrero de 2025, de [https://www.apuntesinformaticafp.com/cursos/administrar\\_una\\_bdr.html](https://www.apuntesinformaticafp.com/cursos/administrar_una_bdr.html)
- Administración de bases de datos. (s/f). Academia-lab.com. Recuperado el 11 de febrero de 2025, de <https://academia-lab.com/enciclopedia/administracion-de-bases-de-datos/>
- Fernández-Avilés y José-María Montero, G. (s/f). Capítulo 5 Gestión de bases de datos relacionales. Github.io. Recuperado el 11 de febrero de 2025, de <https://cdr-book.github.io/datos-sql.html>
- Gestión de Bases de Datos Relacionales con SQL. (s/f). Entredata.org. Recuperado el 11 de febrero de 2025, de <https://www.entredata.org/fundamentos-de-sql/gestion-de-bases-de-datos-relacionales-con-sql>