Guía Taller Integral:

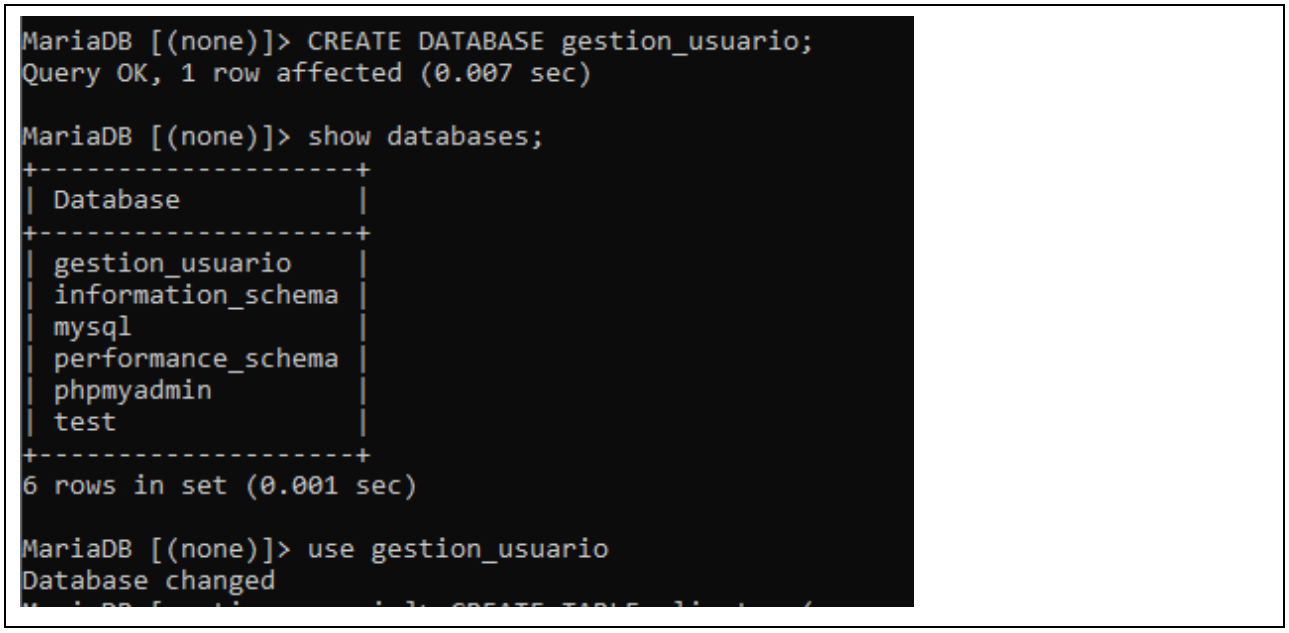
Guía Taller Integral: Gestión de Usuarios y Modelado de Datos en un DBMS

Rebeca Pedrozo Cueto

Sistemas de bases de datos

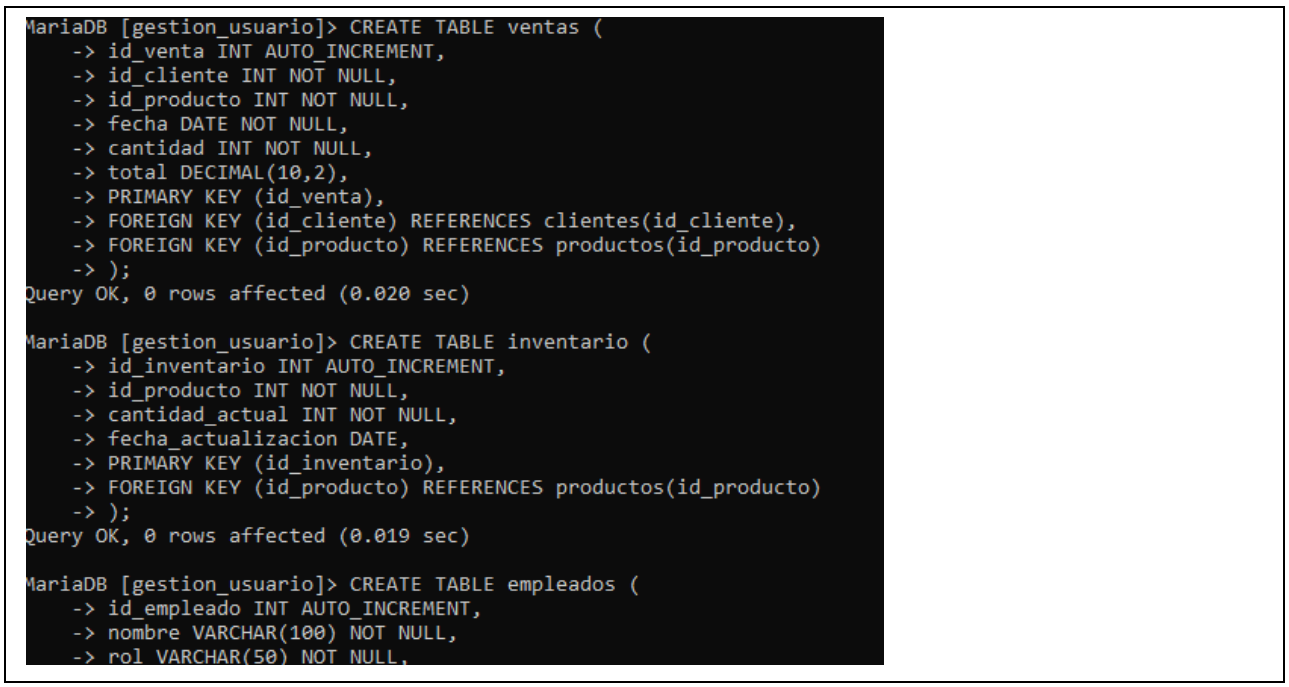
Creación de la base de datos

Se crean las tablas usando Xmapp y MariaDB



Diseño y creación de tablas

Se crean las tablas por medio de la consola de Xmapp



```
MariaDB [gestion_usuario]> CREATE TABLE ventas (
    -> id_venta INT AUTO_INCREMENT,
    -> id_cliente INT NOT NULL,
    -> id_producto INT NOT NULL,
    -> fecha DATE NOT NULL,
    -> cantidad INT NOT NULL,
    -> total DECIMAL(10,2),
    -> PRIMARY KEY (id_venta),
    -> FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente),
    -> FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
    -> );
Query OK, 0 rows affected (0.020 sec)

MariaDB [gestion_usuario]> CREATE TABLE inventario (
    -> id_inventario INT AUTO_INCREMENT,
    -> id_producto INT NOT NULL,
    -> cantidad_actual INT NOT NULL,
    -> fecha_actualizacion DATE,
    -> PRIMARY KEY (id_inventario),
    -> FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
    -> );
Query OK, 0 rows affected (0.019 sec)

MariaDB [gestion_usuario]> CREATE TABLE empleados (
    -> id_empleado INT AUTO_INCREMENT,
    -> nombre VARCHAR(100) NOT NULL,
    -> rol VARCHAR(50) NOT NULL,
```

Código: https://github.com/rebeca07-pedrozo/Talleres_Primer_Corte-BD-/blob/main/create%20table.sql.txt

Definición de roles

Se definen los roles

```
MariaDB [gestion_usuario]> CREATE ROLE 'administrador_sistema';
Query OK, 0 rows affected (0.004 sec)

MariaDB [gestion_usuario]> CREATE ROLE 'gerente';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE ROLE 'operador_inventario';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE ROLE 'analista_financiero';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE ROLE 'atencion_cliente';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE ROLE 'soporte_tecnico';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE ROLE 'auditor';
```

Código: https://github.com/rebeca07-pedrozo/Talleres_Primer_Corte-BD-/blob/main/Definicion%20roles.sql.txt

Asignación de permisos a los roles

Se asignan los permisos a los roles

```
MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.ventas TO 'gerente';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.clientes TO 'gerente';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.ventas TO 'analista_financiero';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.clientes TO 'atencion_cliente';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.empleados TO 'soporte_tecnico';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.ventas TO 'soporte_tecnico';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.inventario TO 'auditor';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.ventas TO 'auditor';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT SELECT ON gestion_usuario.clientes TO 'auditor';
```

Codigo: https://github.com/rebeca07-pedrozo/Talleres_Primer_Corte-BD-/blob/main/Permisos%20para%20roles.sql.txt

Creación de usuarios y asignación de roles

Se crean los usuarios y asignan roles

```
MariaDB [gestion_usuario]> CREATE USER 'usuario_inventario'@'localhost' IDENTIFIED BY 'ContraseñaSegura123!';
Query OK, 0 rows affected (0.003 sec)

MariaDB [gestion_usuario]> GRANT 'operador_inventario' TO 'usuario_inventario'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE USER 'usuario_admin'@'localhost' IDENTIFIED BY 'ContraseñaSuperSegura!';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT 'administrador_sistema' TO 'usuario_admin'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE USER 'usuario_gerente'@'localhost' IDENTIFIED BY 'ContraseñaSegura456!';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT 'gerente' TO 'usuario_gerente'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE USER 'usuario_financiero'@'localhost' IDENTIFIED BY 'ContraseñaSegura789!';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT 'analista_financiero' TO 'usuario_financiero'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> CREATE USER 'usuario_soporte'@'localhost' IDENTIFIED BY 'ContraseñaSegura321!';
Query OK, 0 rows affected (0.001 sec)

MariaDB [gestion_usuario]> GRANT 'atencion_cliente' TO 'usuario_soporte'@'localhost';
Query OK, 0 rows affected (0.001 sec)
```

Código: https://github.com/rebeca07-pedrozo/Talleres_Primer_Corte-BD-/blob/main/create%20user.sql.txt

Políticas de Seguridad y Auditoría en un DBMS

Sistemas de Bases de Datos

Rebeca Pedrozo Cueto

Plan para la Integración de la Autenticación Multifactor (MFA) en el Sistema

1. Paso 1: Evaluación del Sistema Actual

En este punto se debe establecer los requisitos mínimos para las contraseñas en el DBMS. La guía propone usar :

```
SET GLOBAL validate_password.length = 12;

SET GLOBAL validate_password.mixed_case_count = 1;

SET GLOBAL validate_password.number_count = 1;
SET GLOBAL validate_password.special_char_count = 1;
```

Sin embargo, por temas de compatibilidad se utilizó un plugin distinto. **En este caso usamos ´simple_password´**

```
SET GLOBAL simple_password_check_minimal_length = 12;

SET GLOBAL simple_password_check_digits = 1;

SET GLOBAL simple_password_check_letters_same_case = 1;

SET GLOBAL simple_password_check_other_characters = 1;
```

Link: https://github.com/rebeca07-pedrozo/BD-UL/blob/main/SQL/SET%20GLOBAL.sql

Como solución se propone también usar un trigger que evalúe la contraseña antes de posicionarse en la base de datos u otra tabla que evalúe esto antes:

```
CREATE TABLE control_usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario VARCHAR(50) UNIQUE NOT NULL,
    contrasena VARCHAR(255) NOT NULL,
    CONSTRAINT chk_contrasena CHECK (
        CHAR_LENGTH(contrasena) >= 12
```

```
    AND contrasena REGEXP '[A-Z]'
    AND contrasena REGEXP '[a-z]'
    AND contrasena REGEXP '[0-9]'
    AND contrasena REGEXP '[!@#$%^&*(),.?":{}|<>]'
  )
);
```

Todas las contraseñas deben cumplir con los mínimos establecidos.

2. Plan para la Integración de la Autenticación Multifactor (MFA) en el Sistema

En la actualidad, los sistemas de autenticación de un solo factor (como las contraseñas) ya no son suficientes para garantizar la seguridad de los usuarios. Los ciberataques se están volviendo cada vez más sofisticados, y es esencial fortalecer los mecanismos de protección de las cuentas de usuario. Una de las formas más efectivas de mejorar la seguridad es mediante la implementación de autenticación multifactor (MFA). MFA agrega una capa adicional de seguridad al requerir que los usuarios proporcionen dos o más factores de verificación antes de permitirles acceder a sus cuentas.

¿Qué es la Autenticación Multifactor (MFA)?

La autenticación multifactor (MFA) es un proceso de seguridad que requiere múltiples formas de identificación para acceder a una cuenta. Comúnmente, el proceso de autenticación de un usuario se basa en tres factores principales:

1. Algo que sabes: Normalmente, una contraseña o PIN.
2. Algo que tienes: Un dispositivo como un teléfono móvil, que genera un código de acceso o contiene un token físico.
3. Algo que eres: Características biométricas, como huellas dactilares o reconocimiento facial.

Paso 1: Evaluación del Sistema Actual

Antes de implementar MFA, es fundamental analizar cómo está configurado el sistema de autenticación actual. En muchos sistemas, el único factor de autenticación utilizado es la contraseña. Este es un punto vulnerable, ya que las contraseñas pueden ser adivinadas o robadas en ataques de phishing o fuerza bruta. El análisis de vulnerabilidades te permitirá determinar cómo MFA puede mejorar la seguridad de tu sistema. Un buen comienzo sería revisar los errores recurrentes, intentos fallidos de inicio de sesión y otras posibles debilidades en el sistema.

Paso 2: Selección del Método de MFA

Para integrar MFA de forma eficaz, se deben elegir los métodos de autenticación adecuados. Existen diversas opciones disponibles que incluyen:

1. Aplicaciones de Autenticación (como Google Authenticator):
    a. Google Authenticator es una aplicación gratuita que genera un código temporal cada 30 segundos. Este código se utiliza como el segundo factor de autenticación. Es uno de los métodos más utilizados debido a su facilidad de uso y configuración.
2. Envío de SMS o Correos Electrónicos:
    a. Aunque es común, el envío de códigos de autenticación por SMS o correo electrónico tiene ciertas vulnerabilidades, como ataques de SIM swapping o acceso no autorizado a la bandeja de entrada del correo electrónico. No obstante, sigue siendo un método adicional útil para mejorar la seguridad.
3. Métodos Biométricos:
    a. Si el sistema tiene compatibilidad con dispositivos de lectura de huellas dactilares o reconocimiento facial, estos son métodos de autenticación altamente seguros y convenientes para el usuario.

Para este ejemplo, vamos a centrarnos en la integración de Google Authenticator, ya que es fácil de implementar, gratuito y ampliamente utilizado.

Paso 3: Integración de Google Authenticator

3.1 Modificación de la Base de Datos

Para poder gestionar la autenticación multifactor, debemos almacenar el secreto de Google Authenticator de cada usuario. Esto es crucial porque este secreto es lo que se utiliza para generar los códigos de verificación que el usuario introducirá al iniciar sesión.

Imaginemos que ya tienes una tabla `usuarios` en tu base de datos MySQL. Lo primero será agregar una columna para almacenar el secreto MFA de cada usuario. Este secreto debe ser único por usuario y se genera en el momento de configurar MFA:

```
sql
CopiarEditar
ALTER TABLE usuarios ADD COLUMN mfa_secret VARCHAR(255) DEFAULT NULL;
```

3.2 Instalación de la Biblioteca de Google Authenticator

En el backend, puedes usar una librería de PHP como PHPGangsta/GoogleAuthenticator, que facilita la integración con Google Authenticator. Para instalarla, puedes usar Composer, el gestor de dependencias de PHP:

```
bash
CopiarEditar
composer require "phpgangsta/googleauthenticator"
```

3.3 Generación y Almacenamiento del Secreto

Una vez instalada la librería, al configurar MFA para un usuario, se generará un secreto único para ese usuario. Este secreto se almacenará en la base de datos y se utilizará para generar códigos de verificación. El código en PHP para generar el secreto sería el siguiente:

```
php
CopiarEditar
```

```
require_once 'vendor/autoload.php';
use PHPGangsta_GoogleAuthenticator;

$ga = new PHPGangsta_GoogleAuthenticator();
$secret = $ga->createSecret();

// Guardamos el secreto en la base de datos
$query = "UPDATE usuarios SET mfa_secret = ? WHERE id_usuario = ?";
$stmt = $conn->prepare($query);
$stmt->bind_param("si", $secret, $userId);
$stmt->execute();
```

Este secreto será el que el usuario escaneará en su aplicación de Google Authenticator.

3.4 Generación del Código QR para Google Authenticator

Para que el usuario configure su Google Authenticator, deberás generar un código QR que ellos puedan escanear con su aplicación. Aquí tienes un ejemplo de cómo generar este QR en PHP:

```
php
CopiarEditar
$qrCodeUrl = $ga->getQRCodeGoogleUrl('MiAplicación', $secret);
```

Este código QR contiene la información necesaria para que el usuario asocie su cuenta con la aplicación Google Authenticator. Al escanearlo, la aplicación generará códigos temporales cada 30 segundos, lo cual se utilizará en los inicios de sesión posteriores.

3.5 Verificación del Código

Cuando un usuario intente iniciar sesión, además de ingresar su contraseña, deberá proporcionar un código de verificación de Google Authenticator. El código PHP para verificar este código sería el siguiente:

```
php
CopiarEditar
```

```
$code = $_POST['mfa_code']; // Código ingresado por el usuario
$isValid = $ga->verifyCode($secret, $code, 2); // 2 es el margen de
tiempo permitido

if ($isValid) {
    // El código es correcto, permitir el acceso
} else {
    // El código es incorrecto, denegar el acceso
}
```

Este paso asegura que el usuario no solo tiene la contraseña correcta, sino que también tiene acceso al dispositivo en el que Google Authenticator está instalado.

Paso 4: Pruebas y Verificación

Antes de implementar MFA en producción, es crucial realizar pruebas exhaustivas para asegurarte de que todo funciona correctamente:

1. Verifica que el código QR se genere de manera correcta y que los usuarios puedan escanearlo sin problemas.
2. Asegúrate de que los códigos generados por Google Authenticator caduquen a los 30 segundos y sean válidos solo dentro de ese margen de tiempo.
3. Realiza pruebas de seguridad para asegurarte de que el secreto de MFA esté correctamente cifrado y protegido en la base de datos.

Paso 5: Beneficios de Implementar MFA

Implementar MFA, y en particular Google Authenticator, tiene varios beneficios clave:

1. Mayor Seguridad: La principal ventaja es que incluso si un atacante obtiene la contraseña de un usuario, aún necesitaría el código generado en el dispositivo del usuario. Esto hace que sea mucho más difícil acceder a una cuenta sin tener ambos factores.
2. Cumplimiento de Normativas: Muchas regulaciones de seguridad, como el GDPR o la Ley de Protección de Datos Personales, exigen medidas adicionales de seguridad para proteger la información sensible. MFA ayuda a cumplir con estas normativas.
3. Protección Contra Phishing y Fuerza Bruta: Dado que MFA requiere un segundo factor, los ataques de phishing y de fuerza bruta son mucho menos efectivos.

Incluso si un atacante obtiene una contraseña, no podrá acceder a la cuenta sin el código de Google Authenticator.

4. Reducción de Fraude: Si estás manejando transacciones financieras o datos sensibles, MFA puede reducir el riesgo de fraude al agregar una capa adicional de autenticación.

Paso 6: Revisar seguridad

Además de la implementación básica de MFA, hay algunas recomendaciones adicionales para fortalecer la seguridad:

1. Revisión de los Logs de Actividad: Es importante revisar los logs para identificar patrones sospechosos, como múltiples intentos fallidos de autenticación o accesos desde ubicaciones inusuales.
2. Implementación de Seguridad Adicional: Aunque Google Authenticator es un buen primer paso, también puedes considerar el uso de tokens físicos, como claves de seguridad USB (YubiKey), que son aún más seguras.
3. Educación al Usuario: Aunque MFA es una capa adicional de seguridad, los usuarios deben ser educados sobre cómo proteger su dispositivo móvil y sus aplicaciones de autenticación. Es vital que no compartan sus códigos o secretas con nadie.

Paso 4: Revisión de los backlogs

c:\xampp\mysql\bin\mysqld.exe, Version: 10.4.32-MariaDB-log (mariadb.org binary distribution). started with: TCP Port: 3306, Named Pipe: C:/xampp/mysql/mysql.sock Time Id Command Argument 250309 23:59:32 8 Connect root@localhost as anonymous on 8 Query set autocommit=1 8 Query SELECT current_user() 8 Query SET CHARACTER SET utf8 8 Query SET NAMES utf8 8 Query SHOW SESSION VARIABLES LIKE 'sql_mode' 8 Query SELECT CONNECTION_ID() 8 Query show character set where charset = 'utf8mb4' 8 Query SET NAMES 'utf8mb4' 8 Query SHOW SESSION STATUS LIKE 'Ssl_cipher' 8 Query USE `gestion_usuario` 8 Query set autocommit=1 9 Connect root@localhost as anonymous on 9 Query set autocommit=1 9 Query SELECT current_user() 9 Query SET CHARACTER SET utf8 9 Query SET NAMES utf8 9 Query SET SQL_SAFE_UPDATES=1 9 Query SELECT CONNECTION_ID() 9 Query show character set where charset = 'utf8mb4' 9 Query SET NAMES 'utf8mb4' 9 Query SHOW SESSION STATUS LIKE 'Ssl_cipher' 9 Query USE `gestion_usuario` 9 Query set autocommit=1 250310 0:01:29 9 Quit

8 Quit

c:\xampp\mysql\bin\mysqld.exe, Version: 10.4.32-MariaDB-log (mariadb.org binary distribution). started with: TCP Port: 3306, Named Pipe: C:/xampp/mysql/mysql.sock Time Id Command Argument 250310 5:03:39 8 Connect root@localhost as anonymous on 8 Query set autocommit=1 8 Query SELECT current_user() 8 Query SET CHARACTER SET utf8 8 Query SET NAMES utf8 8 Query SHOW SESSION VARIABLES LIKE 'sql_mode' 8 Query SELECT CONNECTION_ID() 8 Query show character set where charset = 'utf8mb4' 8 Query SET NAMES 'utf8mb4' 8 Query SHOW SESSION STATUS LIKE 'Ssl_cipher' 8 Query USE `gestion_usuario` 8 Query set autocommit=1 9 Connect root@localhost as anonymous on 9 Query set autocommit=1 9 Query SELECT current_user() 9 Query SET CHARACTER SET utf8 9 Query SET NAMES utf8 9 Query SET SQL_SAFE_UPDATES=1 9 Query SELECT CONNECTION_ID() 9 Query show character set where charset = 'utf8mb4' 9 Query SET NAMES 'utf8mb4' 9 Query SHOW SESSION STATUS LIKE 'Ssl_cipher' 9 Query USE `gestion_usuario` 9 Query set autocommit=1 9 Query SHOW SESSION VARIABLES LIKE 'sql_mode' 9 Query SHOW SESSION VARIABLES LIKE 'version_comment' 9 Query SHOW SESSION VARIABLES LIKE 'version' 9 Query SELECT current_user() 9 Query SHOW SESSION VARIABLES LIKE 'lower_case_table_names' 250310 5:03:40 8 Query SHOW DATABASES 9 Query show engines 9 Query show charset 9 Query show collation 9 Query show variables 9 Query SHOW SESSION VARIABLES LIKE 'version_compile_os' 9 Query SHOW SESSION VARIABLES LIKE 'offline_mode' 9 Query SHOW SESSION VARIABLES LIKE 'wait_timeout' 9 Query SHOW SESSION VARIABLES LIKE 'interactive_timeout' 8 Query SHOW FULL TABLES FROM `gestion_usuario` 8 Query SHOW PROCEDURE STATUS WHERE Db='gestion_usuario' 8 Query SHOW FUNCTION STATUS WHERE Db='gestion_usuario' 9 Query SHOW FULL COLUMNS FROM `gestion_usuario.clientes` 9 Query SHOW FULL COLUMNS FROM `gestion_usuario.empleados` 9 Query SHOW FULL COLUMNS FROM `gestion_usuario.inventario` 9 Query SHOW FULL COLUMNS FROM `gestion_usuario.productos` 9 Query SHOW FULL COLUMNS FROM `gestion_usuario.ventas` 250310 5:03:53 9 Query SELECT * FORM clientes 250310 5:03:57 9 Query SELECT * FORM clientes 250310 5:04:03 9 Query SELECT * FROM clientes LIMIT 0, 50 8 Query SHOW INDEX FROM `gestion_usuario.clientes` 250310 5:04:12 9 Query INSERT INTO clientes (nombre, correo, telefono) VALUES ('Juan Pérez', 'juan@correo.com', '1234567890') 250310 5:04:21 9 Query UPDATE productos SET precio = 'ABC' WHERE id_producto = 1 9 Query SHOW WARNINGS 250310 5:04:31 9 Query SET PASSWORD FOR 'root'@'localhost' = PASSWORD('nuevacontraseña') 250310 5:07:19 9 Query SELECT * FROM productos WHERE precio > 1000 LIMIT 1000 8 Query SHOW INDEX FROM `gestion_usuario.productos` 250310 5:07:28 9 Query INSERT INTO productos (descripcion, precio) VALUES ('Producto erróneo', 'ABC') 9 Query SHOW WARNINGS 250310 5:07:31 9 Query SELECT * FROM productos WHERE precio > 1000 LIMIT 1000 8 Query SHOW INDEX FROM `gestion_usuario.productos` 250310 5:07:35 9 Query INSERT INTO productos (descripcion, precio) VALUES ('Producto erróneo', 'ABC') 9 Query SHOW WARNINGS 250310 5:08:05 9 Query SELECT * FORM clientes 250310 5:08:24 9 Query SELECT * FROM clientes_inexistente LIMIT 0, 50 250310 5:08:46 9 Query INSERT INTO ventas (id_cliente, id_producto, fecha, cantidad, total) VALUES (999, 1, '2025-03-09', 2, 200) 250310 5:08:57 9 Query DELETE FROM productos WHERE

id_producto = 1 250310 5:09:17 9 Query SELECT * FROM clientes LIMIT 0, 50 8 Query SHOW INDEX FROM `gestion_usuario.clientes` 250310 5:09:25 9 Query SET PASSWORD FOR 'root'@'localhost' = PASSWORD('nuevacontraseña') 250310 5:09:31 9 Query DROP TABLE IF EXISTS clientes

Se detallan las operaciones realizadas, identificando consultas, configuraciones, posibles errores y advertencias que se generaron durante el proceso.

Conexiones y Configuración de Sesión

En el archivo de registro se observa que las conexiones al servidor se realizan bajo el usuario `root` desde `localhost`. En múltiples ocasiones, se observa que el usuario se conecta como anónimo, lo cual puede ser un aspecto a revisar en cuanto a la seguridad, ya que este tipo de conexión no requiere autenticación.

Una vez establecida la conexión, varias consultas de configuración son ejecutadas. Entre ellas, se destacan las siguientes:

- `SET autocommit=1`: Esta consulta habilita el autocompletado de transacciones, asegurando que cada operación en la base de datos se ejecute de manera independiente.
- `SET CHARACTER SET utf8` y `SET NAMES 'utf8mb4'`: Estas consultas aseguran que la codificación de caracteres se configure para manejar adecuadamente caracteres especiales y multilingües.

Por ejemplo, la ejecución de `SET CHARACTER SET utf8` es clave para asegurar que los datos en la base de datos se manejen correctamente en términos de codificación de texto, lo cual es especialmente relevante para bases de datos con caracteres internacionales.

Uso de la Base de Datos gestion_usuario

La base de datos seleccionada durante la mayoría de las operaciones es `gestion_usuario`, que parece ser el núcleo de las actividades realizadas. Consultas como `USE gestion_usuario` indican que las operaciones se realizaron bajo este esquema específico.

Una vez dentro de esta base de datos, se ejecutan varias consultas para obtener información sobre las tablas, como `SHOW FULL TABLES` y `SHOW FULL COLUMNS`, lo que indica una inspección detallada de la estructura de la base de datos. En particular, se consulta la estructura de las tablas `clientes`, `productos`, `empleados`, `inventario`, entre otras, probablemente para asegurarse de que los datos estén correctamente organizados y actualizados.

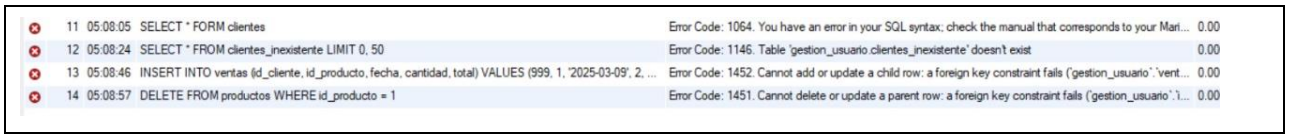
Consultas y Operaciones en la Base de Datos

Una de las consultas más frecuentes es `SELECT *`, utilizada para recuperar datos de diferentes tablas. Sin embargo, se han encontrado errores en algunas consultas. Por ejemplo:

- `SELECT * FORM clientes`: En lugar de la palabra clave `FROM`, se utilizó erróneamente `FORM`. Este error tipográfico podría haber ocasionado un fallo en la consulta, y es importante tener cuidado con este tipo de errores al escribir SQL.

Además, se observa que el proceso de inserción de datos no siempre fue exitoso. En particular, en la tabla `productos`, se intentó insertar un producto con un valor no válido en la columna `precio`:

- `INSERT INTO productos (descripcion, precio) VALUES ('Producto erróneo', 'ABC')`: Aquí, el valor `'ABC'` no es un número válido para el campo `precio`, lo que seguramente generó advertencias. En estos casos, el sistema probablemente emitió una advertencia, como lo muestra el registro con `SHOW WARNINGS`. Es importante asegurarse de que los datos sean del tipo correcto antes de ejecutar una operación de inserción.



11	05:08:05	SELECT * FORM clientes	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your Mari...	0.00
12	05:08:24	SELECT * FROM clientes_inexistente LIMIT 0, 50	Error Code: 1146. Table 'gestion_usuario.clientes_inexistente' doesn't exist	0.00
13	05:08:46	INSERT INTO ventas (id_cliente, id_producto, fecha, cantidad, total) VALUES (999, 1, '2025-03-09', 2, ...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`gestion_usuario`.`vent...	0.00
14	05:08:57	DELETE FROM productos WHERE id_producto = 1	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`gestion_usuario`.`i...	0.00

Cambios en Contraseña y Seguridad

Durante el análisis del registro, se observa que el usuario `root` cambia su contraseña varias veces utilizando la consulta:

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('nuevacontraseña').
```

Este tipo de operación es común en entornos de desarrollo para garantizar que el acceso a la base de datos esté controlado. Sin embargo, dado que se realizó varias veces, es recomendable evaluar si es necesario reforzar la política de contraseñas o considerar un enfoque más seguro para gestionar los accesos.

Eliminación de Datos

Un aspecto preocupante en el registro es la eliminación de tablas y registros. En particular, se ejecutaron las siguientes consultas:

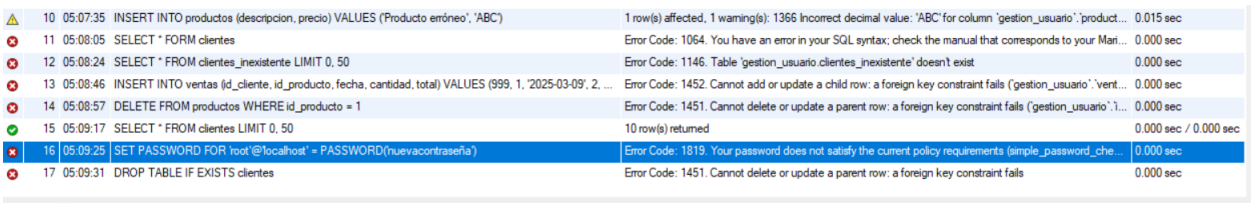
- `DROP TABLE IF EXISTS clientes`: Esta consulta elimina la tabla `clientes`, lo que puede generar la pérdida de datos valiosos si no se tiene cuidado al ejecutar este tipo de comandos.
- `DELETE FROM productos WHERE id_producto = 1`: También se observa la eliminación de un producto de la base de datos.

Es importante que se ejecute una validación adecuada antes de eliminar datos, ya que esta acción es irreversible y podría afectar la integridad de la base de datos.

Conclusiones y Recomendaciones

El registro muestra que las consultas realizadas en el servidor MariaDB fueron bastante variadas, incluyendo configuraciones de sesión, consultas de lectura, inserciones y eliminación de datos. Sin embargo, algunos puntos críticos deben ser corregidos o revisados:

1. Errores Tipográficos: Consultas como `SELECT * FORM clientes` deben ser corregidas para evitar fallos en la ejecución.
2. Validación de Datos: Es crucial asegurar que los datos sean del tipo correcto antes de realizar operaciones como inserciones o actualizaciones. En el caso del producto con un precio no válido ('ABC'), se deben manejar adecuadamente los valores incorrectos.
3. Seguridad: Aunque se cambió la contraseña de `root`, es importante revisar las configuraciones de seguridad del servidor para garantizar que solo usuarios autorizados tengan acceso.
4. Eliminación de Datos: Se debe tener cuidado al ejecutar comandos como `DROP TABLE` y `DELETE FROM`, ya que pueden resultar en la pérdida permanente de datos.



⚠	10	05:07:35	INSERT INTO productos (descripcion, precio) VALUES ('Producto erróneo', 'ABC')	1 row(s) affected, 1 warning(s): 1366 Incorrect decimal value: 'ABC' for column `gestion_usuario`.`product...	0.015 sec
✖	11	05:08:05	SELECT * FORM clientes	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your Mari...	0.000 sec
✖	12	05:08:24	SELECT * FROM clientes_inexistente LIMIT 0, 50	Error Code: 1146. Table 'gestion_usuario.clientes_inexistente' doesn't exist	0.000 sec
✖	13	05:08:46	INSERT INTO ventas (id_cliente, id_producto, fecha, cantidad, total) VALUES (999, 1, '2025-03-09', 2, ...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`gestion_usuario`.`vent...	0.000 sec
✖	14	05:08:57	DELETE FROM productos WHERE id_producto = 1	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`gestion_usuario`.`i...	0.000 sec
✔	15	05:09:17	SELECT * FROM clientes LIMIT 0, 50	10 row(s) returned	0.000 sec / 0.000 sec
✖	16	05:09:25	SET PASSWORD FOR 'root'@'localhost' = PASSWORD('nuevacontraseña')	Error Code: 1819. Your password does not satisfy the current policy requirements (simple_password_che...	0.000 sec
✖	17	05:09:31	DROP TABLE IF EXISTS clientes	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails	0.000 sec

El Error Code: 1819 es un error común que se presenta en sistemas que utilizan MariaDB como sistema de gestión de bases de datos. Este error está relacionado con las políticas de seguridad que regulan la creación y modificación de contraseñas. El mensaje específico "Your password does not satisfy the current policy requirements (simple_password_check)" indica que la contraseña que se intenta establecer no cumple con los requisitos de complejidad configurados en el servidor.

2. Descripción del Error

El error mencionado ocurre cuando un usuario intenta modificar o establecer una nueva contraseña en MariaDB, pero esta no cumple con los estándares de seguridad exigidos por la configuración de la base de datos. En este caso, la política activa es `simple_password_check`, que exige que las contraseñas sigan ciertos parámetros para ser consideradas válidas.

3. Requisitos de la Política simple_password_check

La política `simple_password_check` en MariaDB establece las siguientes restricciones para la creación de contraseñas:

- Longitud mínima: La contraseña debe tener al menos 8 caracteres.
- Complejidad: La contraseña debe incluir al menos:
  - Una letra mayúscula.
  - Una letra minúscula.
  - Un número.
  - Un carácter especial, como !, @, #, entre otros.

Estas reglas están diseñadas para garantizar que las contraseñas sean lo suficientemente seguras para proteger la base de datos de accesos no autorizados.

4. Causas del Error

El Error Code: 1819 se genera cuando la contraseña ingresada no satisface alguna de las condiciones establecidas por la política de seguridad. Esto puede ocurrir debido a que:

1. La contraseña no tiene la longitud mínima requerida.

2. La contraseña no incluye caracteres especiales o números.
3. La contraseña carece de una combinación de letras mayúsculas y minúsculas.

5. Soluciones Propuestas

Existen varias maneras de abordar este problema, las cuales se detallan a continuación:

5.1 Revisar y Modificar la Contraseña

La solución más directa es crear una contraseña que cumpla con los requisitos establecidos por la política de seguridad. Ejemplos de contraseñas válidas incluyen:

- `MiContraseña!123`
- `Contraseña@Segura1`

Estas contraseñas cumplen con los criterios de longitud y complejidad establecidos.

5.2 Verificar las Políticas de Contraseñas en MariaDB

Si el administrador desea confirmar las configuraciones actuales de las políticas de contraseñas, puede ejecutar el siguiente comando para obtener información detallada:

```
sql
CopiarEditar
SHOW VARIABLES LIKE 'validate_password%';
```

Este comando proporcionará detalles sobre los parámetros de validación de contraseñas, tales como la longitud mínima y los requisitos de complejidad.

5.3 Modificar la Política de Contraseñas (Solo en Entornos de Desarrollo)

Si el servidor está configurado para entornos de desarrollo y se desea flexibilizar las políticas de contraseñas, es posible cambiar la configuración de las mismas. Para hacerlo, se puede ejecutar el siguiente comando para reducir la estricta política:

```
sql
CopiarEditar
SET GLOBAL validate_password.policy = LOW;
```

Adicionalmente, si se quiere disminuir la longitud mínima de la contraseña, se puede usar el siguiente comando:

```
sql
CopiarEditar
SET GLOBAL validate_password.length = 6;
```

5.4 Reiniciar el Servicio de MariaDB

En el caso de que se modifiquen las políticas de contraseñas, es necesario reiniciar el servicio de MariaDB para que los cambios tengan efecto. Esto puede realizarse desde el panel de control de **XAMPP** o desde la línea de comandos con el siguiente comando:

```
bash
CopiarEditar
sudo systemctl restart mariadb
```