

**Centro de Enseñanza Técnica Industrial**

**Desarrollo de Software**

**Proyecto Final**

**Jesús Alberto Aréchiga Carrillo**

**22310439 4N**

**Profesor**

**José Luis García Cerpas**

**Junio 2024**

**Guadalajara, Jalisco**

índice

[Descripción de la aplicación (base de datos) 2](#_Toc169117515)

[Planteamiento del problema 2](#_Toc169117516)

[Objetivo general 2](#_Toc169117517)

[Objetivos específicos 2](#_Toc169117518)

[Diagrama entidad relación del proyecto 3](#_Toc169117519)

[Diagrama Entidad-Relación 5](#_Toc169117520)

[Modelo Relacional 6](#_Toc169117521)

# Descripción de la aplicación (base de datos)

La base de datos está diseñada para gestionar la información relacionada con la certificación de estándares de competencia, facilitando la inscripción a certificaciones, el seguimiento del progreso de los participantes, la validación de competencias adquiridas y la emisión de certificados. Su objetivo es centralizar y automatizar los procesos relacionados con las certificaciones, mejorando la eficiencia operativa y asegurando la precisión de los registros.

# Planteamiento del problema

La empresa certificadora enfrenta desafíos en la gestión eficiente y en la operacionalización de sus procesos de certificación debido a sistemas desactualizados y la falta de un método centralizado para el manejo de la información. Esto resulta en ineficiencias operativas, errores en los registros, demoras en la emisión de certificados y dificultades para adaptarse rápidamente a cambios en los estándares de competencia.

# Objetivo general

Desarrollar e implementar una base de datos robusta y escalable que centralice y automatice la gestión de certificaciones en estándares de competencia, mejorando la eficiencia operativa, asegurando la precisión de los registros, y facilitando el proceso de certificación para todos los interesados.

# Objetivos específicos

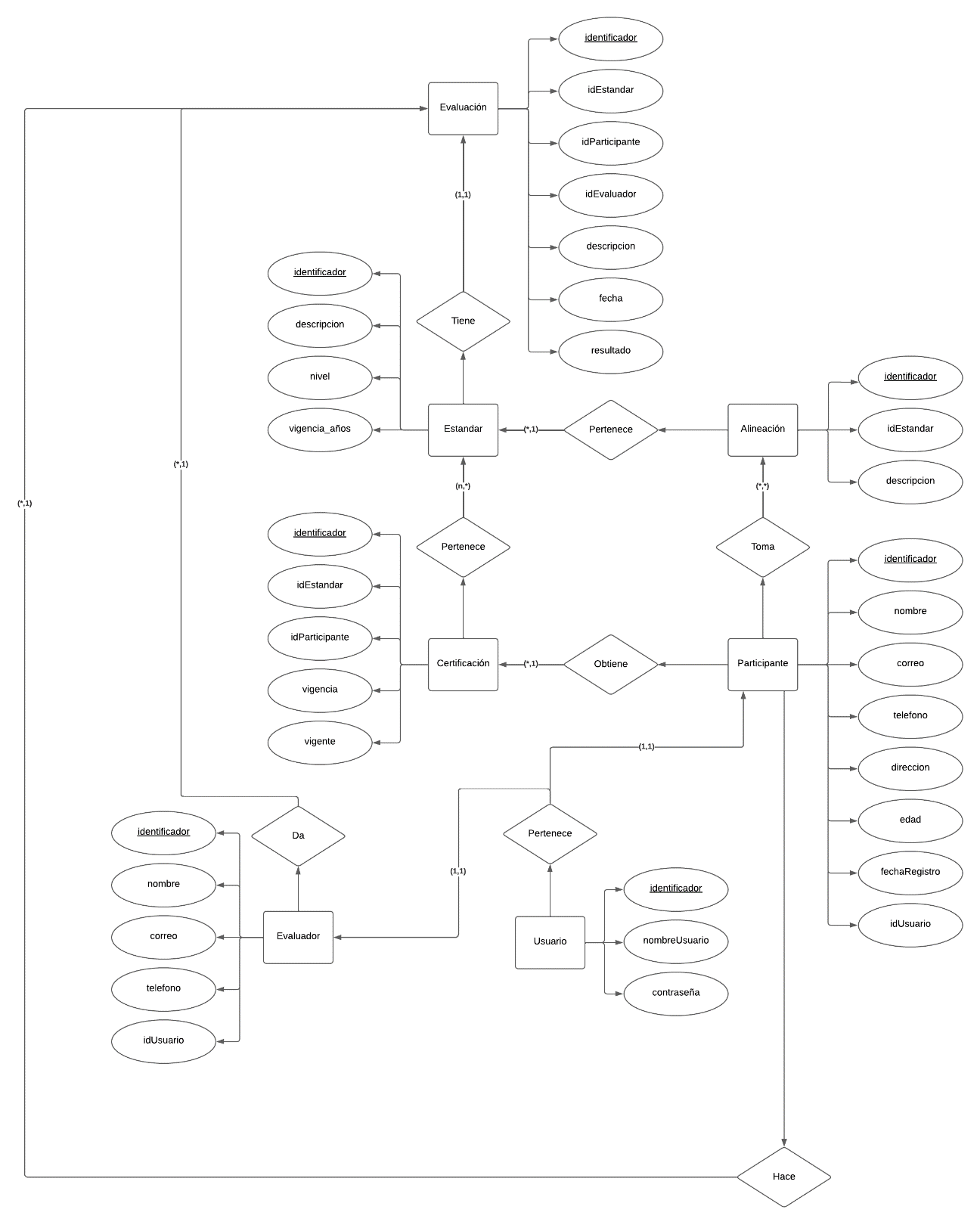
1. Automatizar los procesos de inscripción, evaluación y emisión de certificados.
2. Desarrollar mecanismos para el seguimiento continuo del progreso de los participantes y la validación de competencias.
3. Garantizar la accesibilidad y seguridad de la base de datos para usuarios autorizados.
4. Proveer capacidades de generación de reportes y análisis de datos para apoyar la toma de decisiones y la mejora continua.

# Diagrama entidad relación del proyecto

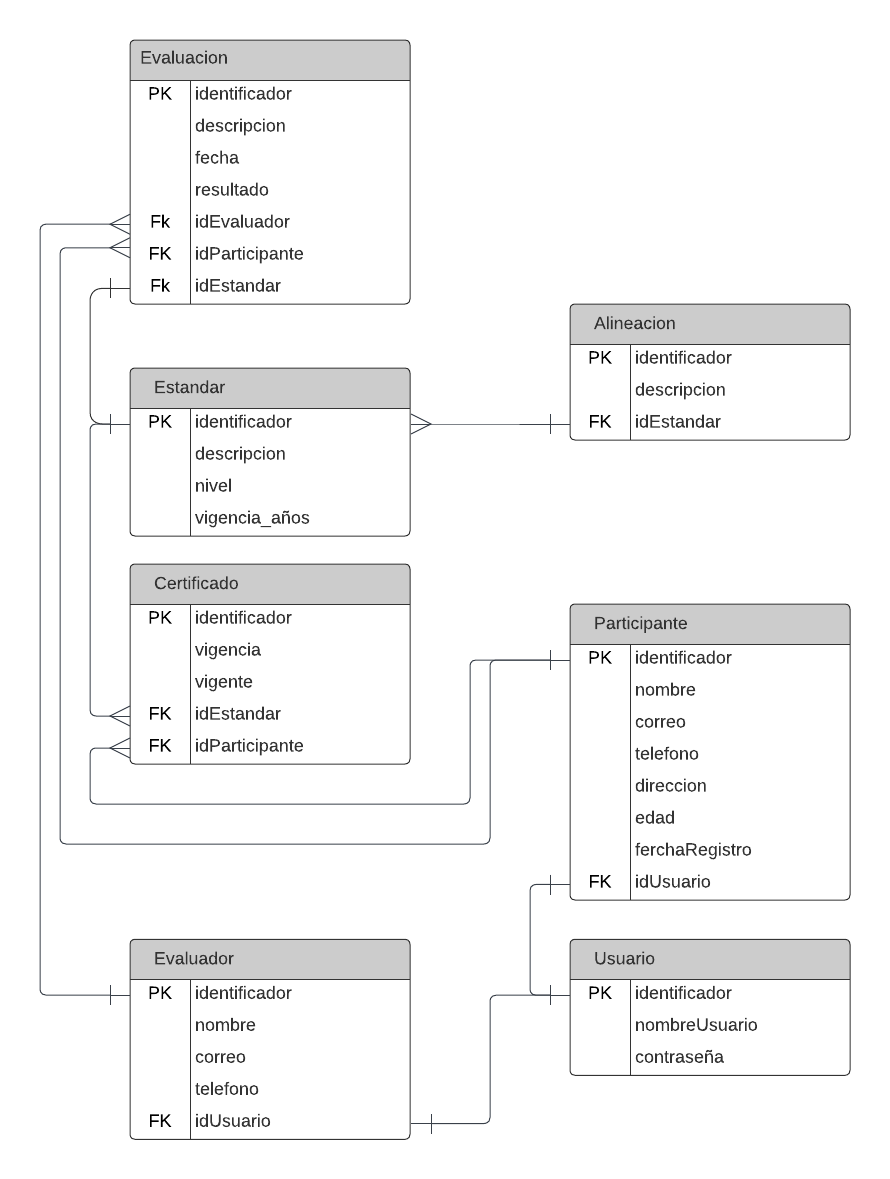
Entidades y atributos:

* Participantes: ID, nombre completo, correo electrónico, número de teléfono, dirección, fecha de registro, edad. Un participante puede inscribirse a más de un alineamiento y evaluación. Una evaluación sólo puede ser realizada por un cliente y un alineamiento puede ser cursada por más de un cliente.
* Certificaciones: ID, estándar certificado, ID del participante, nombre completo del participante. Una certificación puede tener muchos participantes y puede estar relacionada con un solo estándar de competencia. Un participante puede tener más de un certificado. Un estándar de competencia puede expedir más de un certificado.
* Estándares de Competencia: ID, descripción, nivel o categoría.
* Alineaciones: ID, descripción, estándar de competencia asociado. Una alineación puede estar relacionada con un solo estándar de competencia. Un estándar de competencia puede tener solo una alineación.
* Evaluaciones: ID, descripción, estándar de competencia asociado. Una evaluación puede estar relacionado a un solo estándar de competencia. Un estándar de competencia puede tener solo una evaluación.
* Resultados: ID de evaluación, ID del participante, fecha de realización, resultado obtenido. Un resultado puede pertenecer a una sola evaluación. Una evaluación sólo puede tener un resultado.

## Diagrama Entidad-Relación



## Modelo Relacional



# Base de datos física

DROP DATABASE IF EXISTS bd\_peic;

CREATE DATABASE bd\_peic;

USE bd\_peic;

-- Tabla usuario

CREATE TABLE usuario (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    nombreusuario VARCHAR(50),

    contraseña VARCHAR(50)

);

-- Tabla participante

CREATE TABLE participante (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    nombre VARCHAR(100),

    correo VARCHAR(100),

    telefono VARCHAR(20),

    direccion VARCHAR(255),

    edad INT,

    ferchaRegistro DATE,

    idUsuario INT,

    FOREIGN KEY (idUsuario) REFERENCES usuario(id)

);

-- Tabla evaluador

CREATE TABLE evaluador (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    nombre VARCHAR(100),

    correo VARCHAR(100),

    telefono VARCHAR(20),

    idUsuario INT,

    FOREIGN KEY (idUsuario) REFERENCES usuario(id)

);

-- Tabla estandar

CREATE TABLE estandar (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    descripcion VARCHAR(255),

    nivel INT,

    vigencia\_años INT  -- Añadir columna para la vigencia en años

);

-- Tabla alineacion

CREATE TABLE alineacion (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    descripcion VARCHAR(255),

    idEstandar INT,

    FOREIGN KEY (idEstandar) REFERENCES estandar(id)

);

-- Tabla evaluacion

CREATE TABLE evaluacion (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    descripcion VARCHAR(255),

    idEvaluador INT,

    idParticipante INT,

    idEstandar INT,

    fecha DATE,

    resultado VARCHAR(15),

    FOREIGN KEY (idEvaluador) REFERENCES evaluador(id),

    FOREIGN KEY (idParticipante) REFERENCES participante(id),

    FOREIGN KEY (idEstandar) REFERENCES estandar(id)

);

CREATE TABLE certificado (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    idEstandar INT,

    idParticipante INT,

    vigencia DATE,

    vigente VARCHAR(10) DEFAULT 'vigente',

    FOREIGN KEY (idEstandar) REFERENCES Estandar(id),

    FOREIGN KEY (idParticipante) REFERENCES Participante(id)

);

-- TRIGGERS

DELIMITER $$

-- Trigger para inserción en evaluacion

CREATE TRIGGER trg\_after\_evaluacion\_insert

AFTER INSERT ON evaluacion

FOR EACH ROW

BEGIN

    IF NEW.resultado = 'competente' THEN

        IF NOT EXISTS (

            SELECT 1

            FROM certificado

            WHERE idParticipante = NEW.idParticipante

              AND idEstandar = NEW.idEstandar

              AND vigencia >= CURDATE()

        ) THEN

            INSERT INTO certificado (idEstandar, idParticipante, vigencia, vigente)

            VALUES (

                NEW.idEstandar,

                NEW.idParticipante,

                DATE\_ADD(CURDATE(), INTERVAL (SELECT vigencia\_años FROM estandar WHERE id = NEW.idEstandar) YEAR),

                'vigente'

            );

        END IF;

    END IF;

END$$

-- Trigger para actualización en evaluacion

CREATE TRIGGER trg\_after\_evaluacion\_update

AFTER UPDATE ON evaluacion

FOR EACH ROW

BEGIN

    IF NEW.resultado = 'competente' THEN

        IF NOT EXISTS (

            SELECT 1

            FROM certificado

            WHERE idParticipante = NEW.idParticipante

              AND idEstandar = NEW.idEstandar

              AND vigencia >= CURDATE()

        ) THEN

            INSERT INTO certificado (idEstandar, idParticipante, vigencia, vigente)

            VALUES (

                NEW.idEstandar,

                NEW.idParticipante,

                DATE\_ADD(CURDATE(), INTERVAL (SELECT vigencia\_años FROM estandar WHERE id = NEW.idEstandar) YEAR),

                'vigente'

            );

        END IF;

    END IF;

END$$

-- Trigger para actualizar el estado de vigencia de los certificados

CREATE TRIGGER trg\_update\_certificado\_vigencia

BEFORE UPDATE ON certificado

FOR EACH ROW

BEGIN

    IF NEW.vigencia < CURDATE() THEN

        SET NEW.vigente = 'vencido';

    END IF;

END$$

DELIMITER ;

# Insertar datos de catálogo

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

Se insertan datos en las tablas de estándar y alineación. Estos datos se quedan y no son editables.

# Insertar datos en la base de datos

A computer screen shot of a black screen

Description automatically generated

A black screen with white text

Description automatically generated

A computer screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

# Consultas de varias tablas (join)

Consulta de el nombre del participante, la descripción del estándar, la fecha de la evaluación y el resultado de la evaluación.

A black screen with white text

Description automatically generated

Consulta de los nombres de los participantes, su usuario y contraseña.

A screen shot of a computer

Description automatically generated

# Utilizar funciones de agregación (COUNT, SUM, AVG, MIN y MAX)

Se va a agregar un participante más para poder hacer las consultas con las funciones.

A screenshot of a computer program

Description automatically generated

Función COUNT, se quiere saber cuántos usuarios hay en la base de datos:

A black screen with white text

Description automatically generated

Función SUM, se quiere saber la suma de las edades de todos participantes:

A black screen with white text

Description automatically generated

Función AVG, se quiere saber el promedio de las edades de los participantes:

A black screen with white text

Description automatically generated

Función MIN, se quiere saber la edad mínima entre los participantes:

A black screen with white text

Description automatically generated

Función MAX, se quiere saber la edad máxima entre los participantes:

A black screen with white text

Description automatically generated

# Triggers

Para la inclusión de los triggers se va a implementar una tabla llamada bitácora en la que se registren todos los cambios en la base de datos:

A screenshot of a computer program

Description automatically generated

Ahora los triggers van a hacer una inserción de datos cada vez que haya un cambio a alguna de las tablas.

-- Trigger para inserción en la tabla usuario

CREATE TRIGGER trg\_usuario\_insert

AFTER INSERT ON usuario

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('usuario', 'INSERT', NEW.id, NEW.nombreusuario, CONCAT('nombreusuario: ', NEW.nombreusuario, ', contraseña: ', NEW.contraseña));

END$$

-- Trigger para actualización en la tabla usuario

CREATE TRIGGER trg\_usuario\_update

AFTER UPDATE ON usuario

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('usuario', 'UPDATE', NEW.id, NEW.nombreusuario, CONCAT('nombreusuario: ', NEW.nombreusuario, ', contraseña: ', NEW.contraseña));

END$$

-- Trigger para eliminación en la tabla usuario

CREATE TRIGGER trg\_usuario\_delete

AFTER DELETE ON usuario

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('usuario', 'DELETE', OLD.id, OLD.nombreusuario, CONCAT('nombreusuario: ', OLD.nombreusuario, ', contraseña: ', OLD.contraseña));

END$$

-- Trigger para inserción en la tabla participante

CREATE TRIGGER trg\_participante\_insert

AFTER INSERT ON participante

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('participante', 'INSERT', NEW.id, NEW.correo, CONCAT('nombre: ', NEW.nombre, ', correo: ', NEW.correo, ', telefono: ', NEW.telefono, ', direccion: ', NEW.direccion, ', edad: ', NEW.edad, ', fechaRegistro: ', NEW.ferchaRegistro));

END$$

-- Trigger para actualización en la tabla participante

CREATE TRIGGER trg\_participante\_update

AFTER UPDATE ON participante

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('participante', 'UPDATE', NEW.id, NEW.correo, CONCAT('nombre: ', NEW.nombre, ', correo: ', NEW.correo, ', telefono: ', NEW.telefono, ', direccion: ', NEW.direccion, ', edad: ', NEW.edad, ', fechaRegistro: ', NEW.ferchaRegistro));

END$$

-- Trigger para eliminación en la tabla participante

CREATE TRIGGER trg\_participante\_delete

AFTER DELETE ON participante

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('participante', 'DELETE', OLD.id, OLD.correo, CONCAT('nombre: ', OLD.nombre, ', correo: ', OLD.correo, ', telefono: ', OLD.telefono, ', direccion: ', OLD.direccion, ', edad: ', OLD.edad, ', fechaRegistro: ', OLD.ferchaRegistro));

END$$

-- Trigger para inserción en la tabla evaluador

CREATE TRIGGER trg\_evaluador\_insert

AFTER INSERT ON evaluador

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('evaluador', 'INSERT', NEW.id, NEW.correo, CONCAT('nombre: ', NEW.nombre, ', correo: ', NEW.correo, ', telefono: ', NEW.telefono));

END$$

-- Trigger para actualización en la tabla evaluador

CREATE TRIGGER trg\_evaluador\_update

AFTER UPDATE ON evaluador

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('evaluador', 'UPDATE', NEW.id, NEW.correo, CONCAT('nombre: ', NEW.nombre, ', correo: ', NEW.correo, ', telefono: ', NEW.telefono));

END$$

-- Trigger para eliminación en la tabla evaluador

CREATE TRIGGER trg\_evaluador\_delete

AFTER DELETE ON evaluador

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('evaluador', 'DELETE', OLD.id, OLD.correo, CONCAT('nombre: ', OLD.nombre, ', correo: ', OLD.correo, ', telefono: ', OLD.telefono));

END$$

-- Trigger para inserción en la tabla estandar

CREATE TRIGGER trg\_estandar\_insert

AFTER INSERT ON estandar

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('estandar', 'INSERT', NEW.id, NULL, CONCAT('descripcion: ', NEW.descripcion, ', nivel: ', NEW.nivel, ', vigencia\_años: ', NEW.vigencia\_años));

END$$

-- Trigger para actualización en la tabla estandar

CREATE TRIGGER trg\_estandar\_update

AFTER UPDATE ON estandar

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('estandar', 'UPDATE', NEW.id, NULL, CONCAT('descripcion: ', NEW.descripcion, ', nivel: ', NEW.nivel, ', vigencia\_años: ', NEW.vigencia\_años));

END$$

-- Trigger para eliminación en la tabla estandar

CREATE TRIGGER trg\_estandar\_delete

AFTER DELETE ON estandar

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('estandar', 'DELETE', OLD.id, NULL, CONCAT('descripcion: ', OLD.descripcion, ', nivel: ', OLD.nivel, ', vigencia\_años: ', OLD.vigencia\_años));

END$$

-- Trigger para inserción en la tabla evaluacion

CREATE TRIGGER trg\_evaluacion\_insert

AFTER INSERT ON evaluacion

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('evaluacion', 'INSERT', NEW.id, NULL, CONCAT('descripcion: ', NEW.descripcion, ', idEvaluador: ', NEW.idEvaluador, ', idParticipante: ', NEW.idParticipante, ', idEstandar: ', NEW.idEstandar, ', fecha: ', NEW.fecha, ', resultado: ', NEW.resultado));

END$$

-- Trigger para actualización en la tabla evaluacion

CREATE TRIGGER trg\_evaluacion\_update

AFTER UPDATE ON evaluacion

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('evaluacion', 'UPDATE', NEW.id, NULL, CONCAT('descripcion: ', NEW.descripcion, ', idEvaluador: ', NEW.idEvaluador, ', idParticipante: ', NEW.idParticipante, ', idEstandar: ', NEW.idEstandar, ', fecha: ', NEW.fecha, ', resultado: ', NEW.resultado));

END$$

-- Trigger para eliminación en la tabla evaluacion

CREATE TRIGGER trg\_evaluacion\_delete

AFTER DELETE ON evaluacion

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('evaluacion', 'DELETE', OLD.id, NULL, CONCAT('descripcion: ', OLD.descripcion, ', idEvaluador: ', OLD.idEvaluador, ', idParticipante: ', OLD.idParticipante, ', idEstandar: ', OLD.idEstandar, ', fecha: ', OLD.fecha, ', resultado: ', OLD.resultado));

END$$

-- Trigger para inserción en la tabla certificado

CREATE TRIGGER trg\_certificado\_insert

AFTER INSERT ON certificado

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('certificado', 'INSERT', NEW.id, NULL, CONCAT('idEstandar: ', NEW.idEstandar, ', idParticipante: ', NEW.idParticipante, ', vigencia: ', NEW.vigencia, ', vigente: ', NEW.vigente));

END$$

-- Trigger para actualización en la tabla certificado

CREATE TRIGGER trg\_certificado\_update

AFTER UPDATE ON certificado

FOR EACH ROW

BEGIN

    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('certificado', 'UPDATE', NEW.id, NULL, CONCAT('idEstandar: ', NEW.idEstandar, ', idParticipante: ', NEW.idParticipante, ', vigencia: ', NEW.vigencia, ', vigente: ', NEW.vigente));

END$$

-- Trigger para eliminación en la tabla certificado

CREATE TRIGGER trg\_certificado\_delete

AFTER DELETE ON certificado

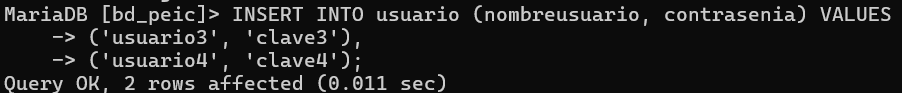
FOR EACH ROW

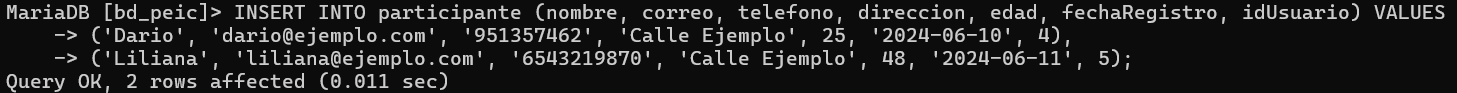
BEGIN

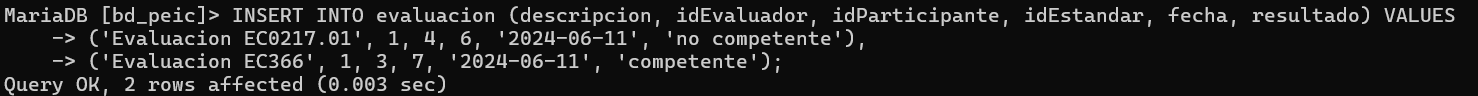
    INSERT INTO bitacora (tabla, operacion, registro\_id, usuario, detalles)

    VALUES ('certificado', 'DELETE', OLD.id, NULL, CONCAT('idEstandar: ', OLD.idEstandar, ', idParticipante: ', OLD.idParticipante, ', vigencia: ', OLD.vigencia, ', vigente: ', OLD.vigente));

END$$







Ahora corroborando los datos creados en la bitácora:

A black and white screen

Description automatically generated

Y en el caso de los triggers de los certificados, se habían hecho un par de inserciones en la tabla “evaluador” que disparan el trigger para insertar datos en la tabla “certificado”:

A black screen with white text

Description automatically generated

# Script para hacer el respaldo lógico y físico de la base de datos:

El script para Windows se haría en un archivo con extensión .bat:

@echo off

REM Configuración

set usuario=root

set base\_de\_datos=bd\_peic

set ruta\_destino=C:\

REM Respaldar la base de datos con mysqldump

mysqldump -u%usuario% %base\_de\_datos% > "%ruta\_destino%\respaldo.sql"

REM Copiar los archivos físicos de la base de datos

xcopy "C:\xampp\mysql\data\%base\_de\_datos%" "%ruta\_destino%\%base\_de\_datos%" /E /I

echo Respaldo completo realizado exitosamente en %date% %time%

El script para Linux sería:

#!/bin/bash

# Configuración

usuario="tu\_usuario"

contraseña="tu\_contraseña"

nombre\_base\_de\_datos="tu\_base\_de\_datos"

ruta\_destino="/ruta/donde/guardar/backups"

# Respaldar la base de datos con mysqldump

mysqldump -u $usuario -p$contraseña $nombre\_base\_de\_datos > $ruta\_destino/respaldo.sql

# Copiar los archivos físicos de la base de datos

sudo cp -R /var/lib/mysql/$nombre\_base\_de\_datos $ruta\_destino

echo "Respaldo completo realizado exitosamente en $(date)"

# Conclusiones

Las bases de datos son herramientas muy eficientes a la hora de gestionar los datos que se utilizan. En este caso se han utilizado 8 tablas que se relacionan entre ellas, permite registrar usuarios y participantes, evaluar estándares de competencia mediante evaluaciones, y emitir certificados basados en los resultados obtenidos. Los triggers implementados aseguran que los certificados se emitan automáticamente cuando un participante demuestra competencia en un estándar específico.

Para mantener la integridad y seguridad de los datos, se implementó un sistema de respaldo lógico y físico. El respaldo lógico mediante mysqldump nos permite generar scripts SQL completos para restaurar la base de datos en caso necesario, mientras que el respaldo físico copia los archivos de datos directamente para una restauración rápida.