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)

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

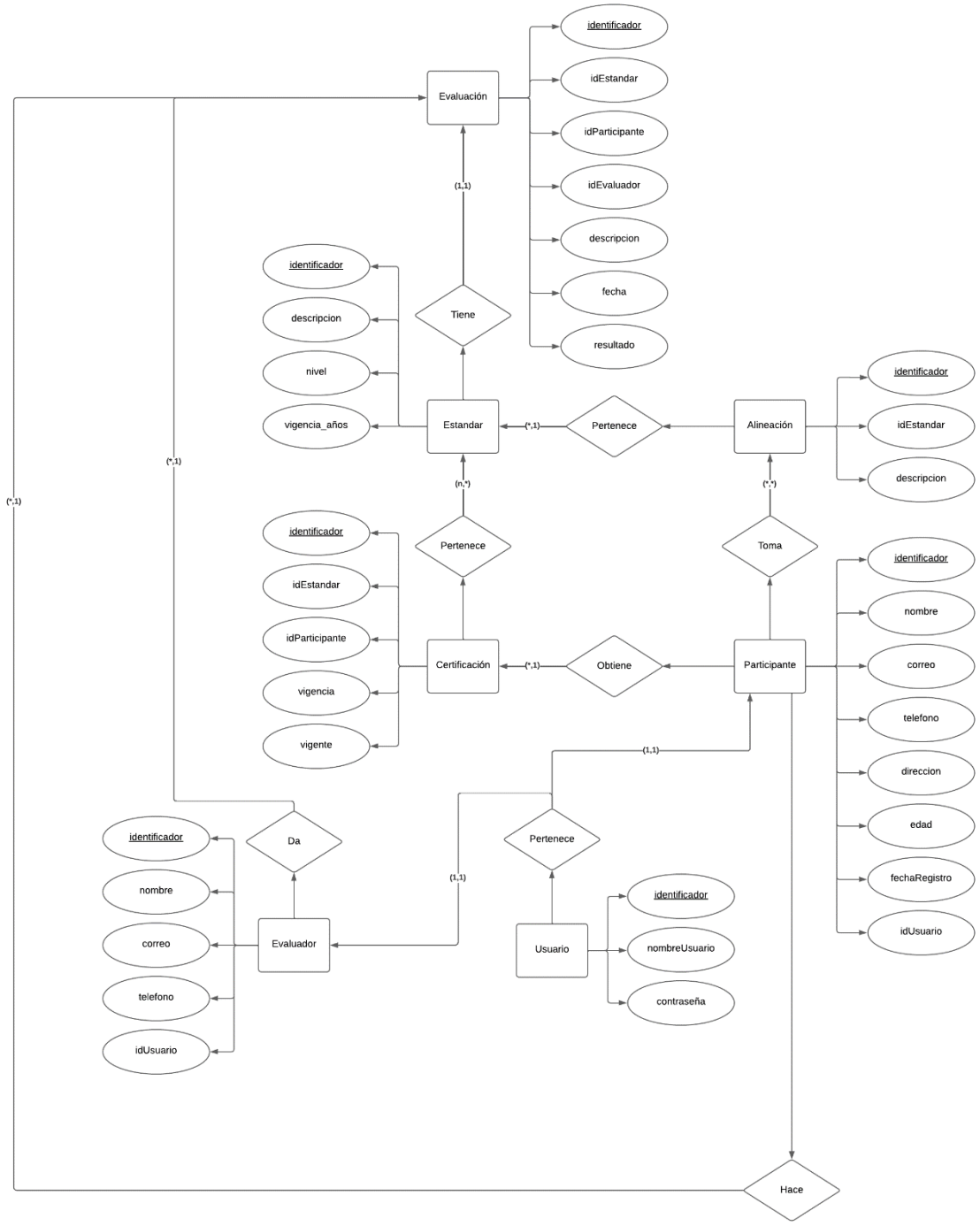
1. Automatizar los procesos de inscripción, evaluación y emisión de certificados.
2. Desarrollar mecanismos para el seguimiento continuo del progreso de los participantes y la validación de competencias.
3. Garantizar la accesibilidad y seguridad de la base de datos para usuarios autorizados.
4. Proveer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

Diagrama entidad relación del proyecto

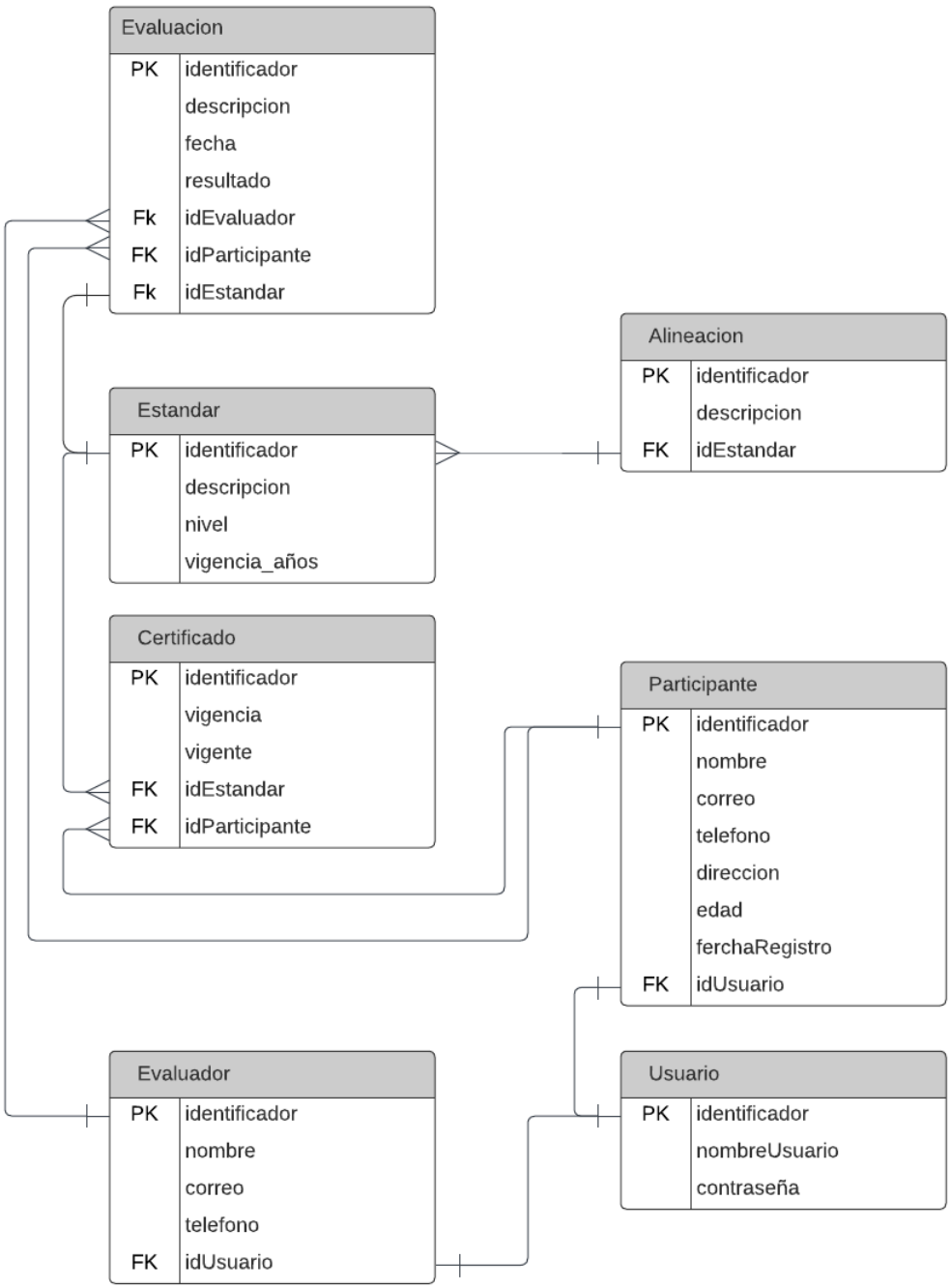
Entidades y atributos:

- 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

```sql
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,
```

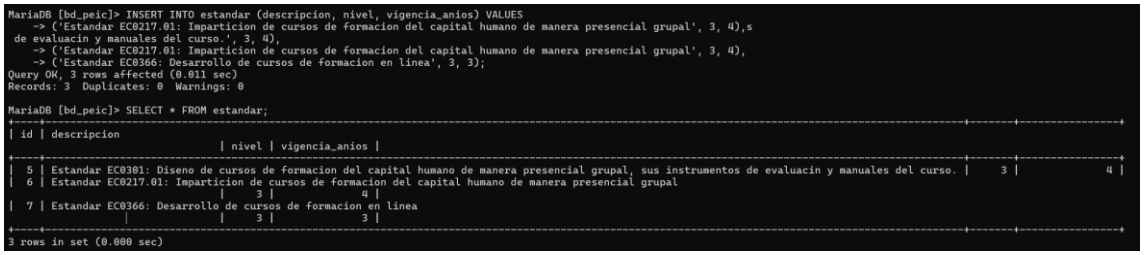
```
                DATE_ADD(CURDATE(), INTERVAL (SELECT vigencia_años FROM
estandar WHERE id = NEW.idEstandar) YEAR),
                'vigente'
            );
        END IF;
    END IF;
END$$

-- 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



```
MariaDB [bd_peic]> INSERT INTO estandar (descripcion, nivel, vigencia_anios) VALUES
    -> ('Estandar EC0217.01: Imparticion de cursos de formacion del capital humano de manera presencial grupal', 3, 4),s
 de evaluacin y manuales del curso.', 3, 4),
    -> ('Estandar EC0217.01: Imparticion de cursos de formacion del capital humano de manera presencial grupal', 3, 4),
    -> ('Estandar EC0366: Desarrollo de cursos de formacion en linea', 3, 3);
Query OK, 3 rows affected (0.011 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [bd_peic]> SELECT * FROM estandar;
+----+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------+---------------+
| id | descripcion                                                                                                                                                                         | nivel | vigencia_anios |
+----+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------+---------------+
|  5 | Estandar EC0301: Diseno de cursos de formacion del capital humano de manera presencial grupal, sus instrumentos de evaluacin y manuales del curso. |     3 |             4 |
|  6 | Estandar EC0217.01: Imparticion de cursos de formacion del capital humano de manera presencial grupal                                                                               |     3 |             4 |
|  7 | Estandar EC0366: Desarrollo de cursos de formacion en linea                                                                                                                         |     3 |             3 |
+----+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------+---------------+
3 rows in set (0.000 sec)
```

```
MariaDB [bd_peic]> INSERT INTO alineacion (descripcion, idEstandar) VALUES
    -> ('Alineacion al EC0301', 5),
    -> ('Alineacion al EC0217.01', 6),
    -> ('Alineacion al EC0366', 7);
Query OK, 3 rows affected (0.011 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [bd_peic]> SELECT * FROM alineacion;
+----+--------------------------+-------------+
| id | descripcion              | idEstandar |
+----+--------------------------+-------------+
|  1 | Alineacion al EC0301     |           5 |
|  2 | Alineacion al EC0217.01  |           6 |
|  3 | Alineacion al EC0366     |           7 |
+----+--------------------------+-------------+
3 rows in set (0.000 sec)

MariaDB [bd_peic]>
```

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

```
MariaDB [bd_peic]> INSERT INTO usuario (nombreusuario, contrasenia) VALUES
    -> ('admin', 'admin'),
    -> ('usuario1', 'clave1');
Query OK, 2 rows affected (0.001 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [bd_peic]> SELECT * FROM usuario;
+----+----------------+--------------+
| id | nombreusuario | contrasenia |
+----+----------------+--------------+
|  1 | admin          | admin        |
|  2 | usuario1       | clave1       |
+----+----------------+--------------+
2 rows in set (0.000 sec)
```

```
MariaDB [bd_peic]> INSERT INTO participante (nombre, correo, telefono, direccion, edad, fechaRegistro, idUsuario) VALUES
    -> ('Juan', 'juan@ejemplo.com', '1234567890', 'Calle Ejemplo', 26, '2024-06-01', 2);
Query OK, 1 row affected (0.011 sec)

MariaDB [bd_peic]> SELECT * FROM participante;
+----+--------+-----------------+------------+---------------+------+---------------+-----------+
| id | nombre | correo          | telefono   | direccion     | edad | fechaRegistro | idUsuario |
+----+--------+-----------------+------------+---------------+------+---------------+-----------+
|  1 | Juan   | juan@ejemplo.com | 1234567890 | Calle Ejemplo |   26 | 2024-06-01    |         2 |
+----+--------+-----------------+------------+---------------+------+---------------+-----------+
1 row in set (0.000 sec)
```

```
MariaDB [bd_peic]> INSERT INTO evaluador (nombre, correo, telefono, idUsuario) VALUES
    -> ('admin', 'admin@institutopuntoalejandria.com', '0000000000', 1);
Query OK, 1 row affected (0.004 sec)

MariaDB [bd_peic]> SELECT * FROM evaluadores;
ERROR 1146 (42S02): Table 'bd_peic.evaluadores' doesn't exist
MariaDB [bd_peic]> SELECT * FROM evaluador;
+----+--------+------------------------------------+------------+-----------+
| id | nombre | correo                             | telefono   | idUsuario |
+----+--------+------------------------------------+------------+-----------+
|  1 | admin  | admin@institutopuntoalejandria.com | 0000000000 |         1 |
+----+--------+------------------------------------+------------+-----------+
1 row in set (0.000 sec)
```

```
MariaDB [bd_peic]> INSERT INTO evaluacion (descripcion, idEvaluador, idParticipante, idEstandar, fecha, resultado) VALUES
    -> ('Evaluacion en EC0301', 1, 1, 5, '2024-06-12', 'competente');
Query OK, 1 row affected (0.012 sec)

MariaDB [bd_peic]> SELECT * FROM evaluacion;
+----+----------------------+-------------+----------------+------------+------------+------------+
| id | descripcion          | idEvaluador | idParticipante | idEstandar | fecha      | resultado  |
+----+----------------------+-------------+----------------+------------+------------+------------+
|  2 | Evaluacion en EC0301 |           1 |              1 |          5 | 2024-06-12 | competente |
+----+----------------------+-------------+----------------+------------+------------+------------+
1 row in set (0.000 sec)

MariaDB [bd_peic]> SELECT * FROM certificado;
+----+------------+----------------+------------+---------+
| id | idEstandar | idParticipante | vigencia   | vigente |
+----+------------+----------------+------------+---------+
|  1 |          5 |              1 | 2028-06-12 | vigente |
+----+------------+----------------+------------+---------+
1 row in set (0.000 sec)
```

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.

```
MariaDB [bd_peic]> SELECT
    -> participante.nombre AS NombreParticipante, estandar.descripcion AS Estandar, evaluacion.fecha AS FechaEvaluacion, evaluacion.resultado AS ResultadoEvaluacion
    -> FROM
    -> evaluacion
    -> JOIN
    -> participante ON evaluacion.idParticipante = participante.id
    -> JOIN
    -> estandar ON evaluacion.idEstandar = estandar.id;
+--------------------+-----------------------------------------------------------------------------------------------------------------------------------------+-----------------+---------------------
+
| NombreParticipante | Estandar                                                                                                                                | FechaEvaluacion | ResultadoEvaluacion
|
+--------------------+-----------------------------------------------------------------------------------------------------------------------------------------+-----------------+---------------------
+
| Juan               | Estandar EC0301: Diseno de cursos de formacion del capital humano de manera presencial grupal, sus instrumentos de evaluacin y manuales del curso. | 2024-06-12      | competente
|
+--------------------+-----------------------------------------------------------------------------------------------------------------------------------------+-----------------+---------------------
+
1 row in set (0.001 sec)
```

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

```
MariaDB [bd_peic]> SELECT
    -> participante.nombre AS NombreParticipante, usuario.nombreusuario AS NombreUsuario, usuario.contrasenia AS contrasenia
    -> FROM
    -> participante
    -> JOIN
    -> usuario ON participante.idUsuario = usuario.id;
+--------------------+---------------+-------------+
| NombreParticipante | NombreUsuario | contrasenia |
+--------------------+---------------+-------------+
| Juan               | usuario1      | clave1      |
+--------------------+---------------+-------------+
1 row in set (0.000 sec)
```

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.

```
MariaDB [bd_peic]> INSERT INTO usuario (nombreusuario, contrasenia) VALUES ('usuario2', 'clave2');
Query OK, 1 row affected (0.011 sec)

MariaDB [bd_peic]> SELECT * FROM usuario;
+----+---------------+-------------+
| id | nombreusuario | contrasenia |
+----+---------------+-------------+
|  1 | admin         | admin       |
|  2 | usuario1      | clave1      |
|  3 | usuario2      | clave2      |
+----+---------------+-------------+
3 rows in set (0.000 sec)

MariaDB [bd_peic]> INSERT INTO participante (nombre, correo, telefono, direccion, edad, fechaRegistro, idUsuario) VALUES
    -> ('Pepe', 'pepe@ejemplo.com', '321654987', 'Calle Ejemplo', 35, '2024-06-08', 3);
Query OK, 1 row affected (0.011 sec)

MariaDB [bd_peic]> SELECT * FROM participante;
+----+--------+-----------------+------------+---------------+------+---------------+-----------+
| id | nombre | correo          | telefono   | direccion     | edad | fechaRegistro | idUsuario |
+----+--------+-----------------+------------+---------------+------+---------------+-----------+
|  1 | Juan   | juan@ejemplo.com | 1234567890 | Calle Ejemplo |   26 | 2024-06-01    |         2 |
|  2 | Pepe   | pepe@ejemplo.com | 321654987  | Calle Ejemplo |   35 | 2024-06-08    |         3 |
+----+--------+-----------------+------------+---------------+------+---------------+-----------+
2 rows in set (0.000 sec)
```

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

```
MariaDB [bd_peic]> SELECT COUNT(*) AS CantidadUsuarios FROM usuario;
+------------------+
| CantidadUsuarios |
+------------------+
|                3 |
+------------------+
1 row in set (0.000 sec)
```

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

```
MariaDB [bd_peic]> SELECT SUM(edad) AS edades FROM participante;
+--------+
| edades |
+--------+
|     61 |
+--------+
1 row in set (0.000 sec)
```

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

```
MariaDB [bd_peic]> SELECT AVG(edad) FROM participante;
+-----------+
| AVG(edad) |
+-----------+
|   30.5000 |
+-----------+
1 row in set (0.000 sec)
```

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

```
MariaDB [bd_peic]> Select MIN(edad) AS EdadMinima FROM participante;
+------------+
| EdadMinima |
+------------+
|         26 |
+------------+
1 row in set (0.000 sec)
```

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

```
MariaDB [bd_peic]> Select MAX(edad) AS EdadMaxima FROM participante;
+------------+
| EdadMaxima |
+------------+
|         35 |
+------------+
1 row in set (0.000 sec)
```

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:

```
MariaDB [bd_peic]> CREATE TABLE bitacora (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     tabla VARCHAR(50),
    ->     operacion VARCHAR(10),
    ->     registro_id INT,
    ->     fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ->     usuario VARCHAR(50),
    ->     detalles TEXT
    -> );
Query OK, 0 rows affected (0.009 sec)

MariaDB [bd_peic]> SELECT * FROM bitacora;
Empty set (0.001 sec)
```

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$$
```

```
MariaDB [bd_peic]> INSERT INTO usuario (nombreusuario, contrasenia) VALUES
    -> ('usuario3', 'clave3'),
    -> ('usuario4', 'clave4');
Query OK, 2 rows affected (0.011 sec)
```

```
MariaDB [bd_peic]> INSERT INTO participante (nombre, correo, telefono, direccion, edad, fechaRegistro, idUsuario) VALUES
    -> ('Dario', 'dario@ejemplo.com', '951357462', 'Calle Ejemplo', 25, '2024-06-10', 4),
    -> ('Liliana', 'liliana@ejemplo.com', '6543219870', 'Calle Ejemplo', 48, '2024-06-11', 5);
Query OK, 2 rows affected (0.011 sec)
```

```
MariaDB [bd_peic]> INSERT INTO evaluacion (descripcion, idEvaluador, idParticipante, idEstandar, fecha, resultado) VALUES
    -> ('Evaluacion EC0217.01', 1, 4, 6, '2024-06-11', 'no competente'),
    -> ('Evaluacion EC366', 1, 3, 7, '2024-06-11', 'competente');
Query OK, 2 rows affected (0.003 sec)
```

Ahora corroborando los datos creados en la bitácora:

```
MariaDB [bd_peic]> SELECT * FROM bitacora;
+----+-------------+-----------+-------------+---------------------+-------------------------+------------------------------------------------------------------------------------------------------------------------------+
| id | tabla       | operacion | registro_id | fecha               | usuario                 | detalles                                                                                                                     |
+----+-------------+-----------+-------------+---------------------+-------------------------+------------------------------------------------------------------------------------------------------------------------------+
|  1 | evaluacion  | INSERT    |           3 | 2024-06-12 23:22:34 | NULL                    | descripcion: Evaluacion EC0217.01, idEvaluador: 1, idParticipante: 4, idEstandar: 6, fecha: 2024-06-11, resultado: no competente |
|  2 | certificado | INSERT    |           2 | 2024-06-12 23:22:34 | NULL                    | idEstandar: 7, idParticipante: 3, vigencia: 2027-06-12, vigente: vigente
|  3 | evaluacion  | INSERT    |           4 | 2024-06-12 23:22:34 | NULL                    | descripcion: Evaluacion EC366, idEvaluador: 1, idParticipante: 3, idEstandar: 7, fecha: 2024-06-11, resultado: competente       |
|  4 | evaluador   | INSERT    |           2 | 2024-06-12 23:30:50 | evaluador1@ejemplo.com | nombre: evaluador1, correo: evaluador1@ejemplo.com, telefono: 123456
|  5 | evaluador   | UPDATE    |           2 | 2024-06-12 23:32:25 | evaluador1@ejemplo.com | nombre: evaluador, correo: evaluador1@ejemplo.com, telefono: 654654321
|  6 | evaluador   | DELETE    |           2 | 2024-06-12 23:32:53 | evaluador1@ejemplo.com | nombre: evaluador, correo: evaluador1@ejemplo.com, telefono: 654654321
+----+-------------+-----------+-------------+---------------------+-------------------------+------------------------------------------------------------------------------------------------------------------------------+
6 rows in set (0.000 sec)
```

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”:

```
+----+------------+----------------+------------+---------+
| id | idEstandar | idParticipante | vigencia   | vigente |
+----+------------+----------------+------------+---------+
|  1 |          5 |              1 | 2028-06-12 | vigente |
|  2 |          7 |              3 | 2027-06-12 | vigente |
+----+------------+----------------+------------+---------+
2 rows in set (0.000 sec)
```

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.