PROYECTO FINAL

BASES DE DATOS

Danna Lopez, Andrew Paillacho, Handel Manobanda, Cristian Tambaco

PROBLEMATICA

Gestión de Salas de Conferencias en Empresas y Centros de Eventos

Contexto: Las empresas y centros de eventos suelen enfrentar dificultades en la reserva y administración de sus salas de conferencias. Los problemas comunes incluyen reservas duplicadas, conflictos de horarios, falta de control sobre el uso de las salas, y poca optimización de los recursos disponibles (equipos audiovisuales, capacidad de personas, disponibilidad de catering, etc.).

DEFINICION DE ENTIDADES, ATRIBUTOS Y RELACIONES

Entidad: Usuarios (Personas que reservan salas o administran el sistema)

Atributos:

- ID_Usuario (PK)
- nombres_completos
- correo
- Teléfono
- Rol (Administrador, Asistente(Registrado), Organizador, Invitado (Usuario no Registrado))
- Contraseña - Se va a encriptar después

Entidad: Salas (Espacios disponibles para conferencias)

Atributos:

- id_sala (PK)
- nombre_sala
- capacidad
- ubicacion
- equipamiento (Proyector, sonido, micrófono, etc.)
- disponibilidad (Sí/No)

Entidad: Eventos (Reservas de salas para conferencias o reuniones)

Atributos:

- ID_Evento (PK)
- Nombre
- Descripción
- Fecha
- Hora_Inicio
- Hora_Fin
- Estado (Pendiente, Confirmado, Cancelado)
- SalaID (Fk)
- OrganizadorID (FK)

Entidad: Reserva (Registro de reservas de salas por usuarios)

Atributos:

- id_reserva (PK)
- pago_id (FK)
- evento_id (FK)
- estado (pendiente, confirmada, cancelada)
- fecha_Reserva

DEFINICION DE ENTIDADES, ATRIBUTOS Y RELACIONES

Entidad: Pagos (Si el sistema incluye reservas pagadas)

Atributos:

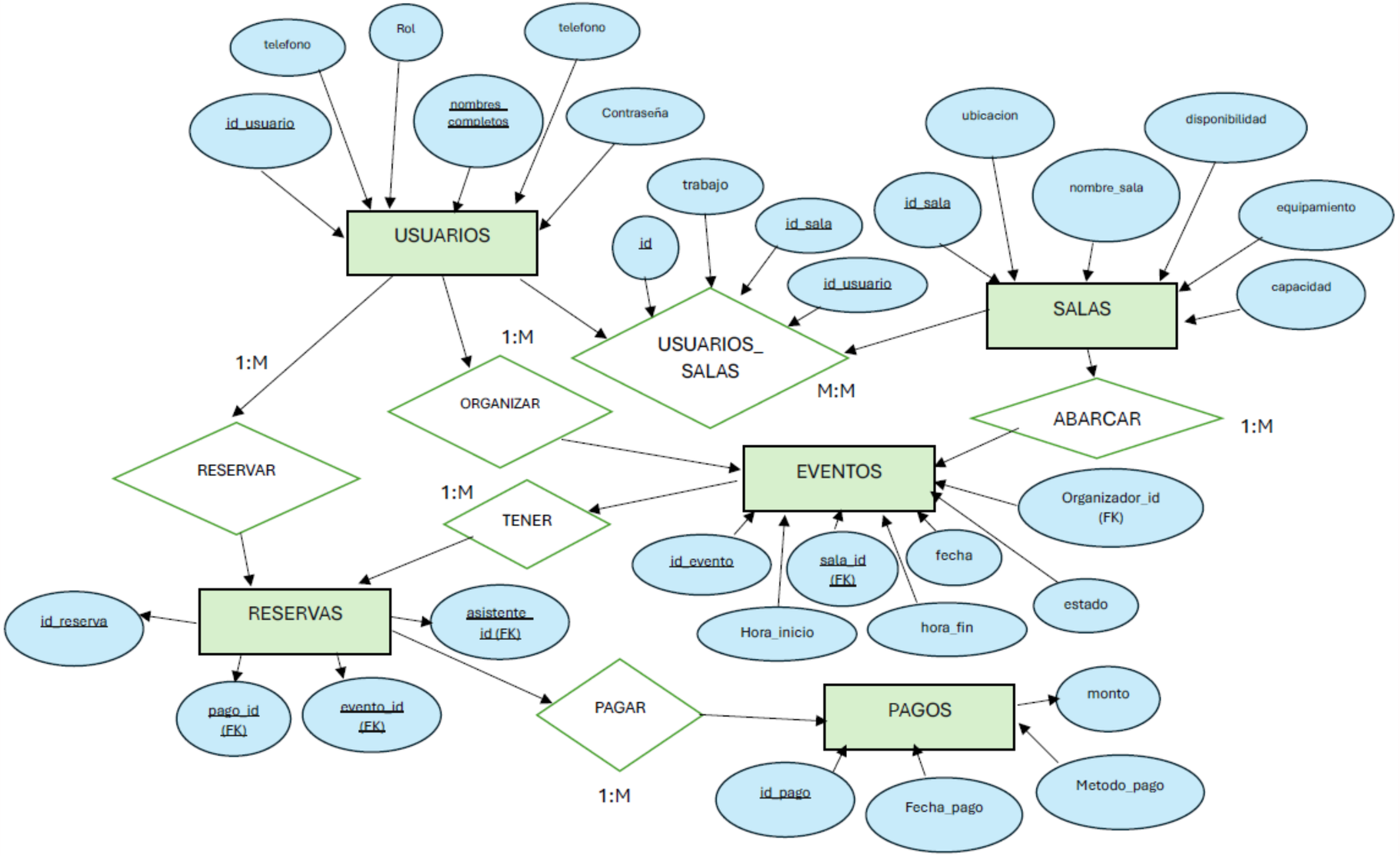
- id_Pago (PK)
- Monto
- Método_Pago (Tarjeta, PayPal, Transferencia)
- Fecha_Pago

Entidad: Usuarios_Salas

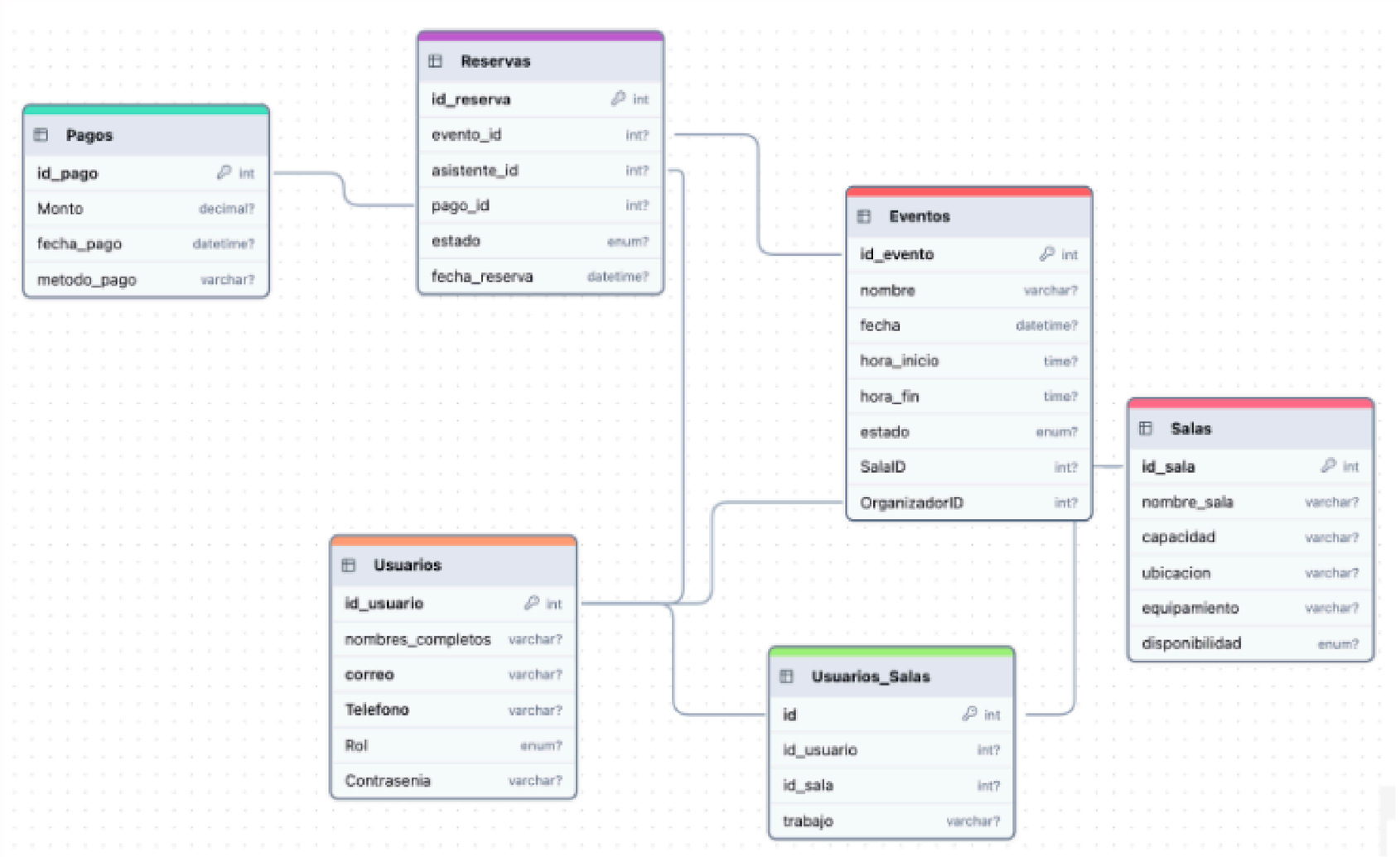
Atributos:

- ID PK
- id_usuario
- id_sala
- trabajo

MODELADO DE BASE DE DATOS - MODELO CONCEPTUAL



MODELADO DE BASE DE DATOS - MODELO LOGICO



MODELADO DE BASE DE DATOS - MODELO FISICO

```sql
create database gestion_conferencias;
use gestion_conferencias;
-- Crear la tabla para usuarios con diferentes roles
create table Usuarios(id_usuario int auto_increment primary key,
                      nombres_completos varchar(100) NOT NULL,
                      correo varchar(100) UNIQUE NOT NULL,
                      Telefono varchar(50) UNIQUE NOT NULL,        -- El correo y el telefono deberán ser unicos para cada usuario
                      Rol ENUM('Invitado', 'Organizador', 'Asistente', 'Administrador') NOT NULL,
                      Contrasenia varchar(255) NOT NULL);
-- Crear la tabla para salas
create table Salas(id_sala int auto_increment primary key,
                   nombre_sala varchar(60) NOT NULL,
                   capacidad varchar(100) NOT NULL,
                   ubicacion varchar(50) NOT NULL,
                   equipamiento varchar(50) NOT NULL,       -- Describir los equipos con los que cuenta la sala, EJ: PROYECTORES
                   disponibilidad ENUM('Si', 'No') NOT NULL);                      -- Verificar si la sala estará libre u ocupada
```

MODELADO DE BASE DE DATOS - MODELO FISICO

```sql
-- Crear la tabla para Eventos
create table Eventos(id_evento int auto_increment primary key,
                        nombre varchar(50) NOT NULL,       -- Nombre del evento
                        descripcion varchar(50) NOT NULL,                          -- Descripcion del evento
                        fecha datetime NOT NULL,                                    -- Fecha en la que sera llevada a cabo el evento
                        hora_inicio time NOT NULL,
                        hora_fin time NOT NULL,
                        Estado enum('Pendiente', 'Cancelado', 'Confirmado') NOT NULL,
                        SalaID int,
                        OrganizadorID int,
                        foreign key (OrganizadorID) references Usuarios(id_usuario));           -- Relacionar la columna OrganizadorI

-- Crear la tabla Pagos
CREATE TABLE Pagos (
    id_pago INT PRIMARY KEY,  -- ID único para el pago
    Monto DECIMAL(10, 2),  -- Monto del pago
    Fecha_Pago DATE,  -- Fecha en que se realizó el pago
    Metodo_Pago VARCHAR(50)  -- Método de pago (tarjeta, transferencia, etc.)
);
```

MODELADO DE BASE DE DATOS - MODELO FISICO

```sql
create table Reservas(id_reserva int auto_increment primary key,
                      evento_id int NOT NULL,    -- Clave foranea con la tabla Eventos
                      asistente_id int NOT NULL,   -- clave foranea con la tabla Usuarios
                      id_pago int NOT NULL,
                      estado enum('Pendiente', 'Cancelado', 'Confirmado') NOT NULL,
                      fecha_reserva datetime,
                  FOREIGN KEY (evento_ID) REFERENCES Eventos(id_evento),
                  FOREIGN KEY (asistente_id) REFERENCES Usuarios(id_usuario),
                  FOREIGN KEY (id_Pago) REFERENCES Pagos(id_pago)  -- Relación con el pago
);

-- Crear una tabla intermedia Usuarios_Salas para que un usuario este a cargo de varias salas
CREATE TABLE Usuarios_Salas (
    id int auto_increment primary key,
    id_usuario INT,
    id_sala INT,
    trabajo VARCHAR(50),  -- Ejemplo: "Administrador", "Soporte técnico"
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario),
    FOREIGN KEY (id_sala) REFERENCES Salas(id_sala)
);
```

BUENAS PRACTICAS PARA HACER ESCALABLES LOS MODELOS DE BASES DE DATOS

- Es importante estructurar las tablas de manera que se minimice la duplicidad de información.
- En este tipo de sistemas, las relaciones entre salas, eventos, usuarios y equipos son fundamentales. Utiliza claves foráneas correctamente para mantener la integridad referencial sin sacrificar la flexibilidad. Esto ayudará a gestionar los datos de forma coherente y reducirá la duplicidad de registros.
- Crear índices en las columnas que se usan con frecuencia en filtros y ordenamientos (como fechas de eventos, ID de salas, etc.). Un mal uso de los índices puede afectar el rendimiento, así que se debe tener cuidado con su cantidad y tipo.
- Diseñar el modelo de datos de manera modular, de forma que si se agregan nuevas funcionalidades, puedas hacerlo sin reestructurar toda la base de datos.

RESTRICCIONES DE INTEGRIDAD

Usuarios - Eventos: ON DELETE SET NULL, ON UPDATE CASCADE

- Si un usuario (organizador) es eliminado, su evento asociado tendrá el campo OrganizadorID establecido a NULL (no se elimina el evento).
- Si un usuario es actualizado, el OrganizadorID en los eventos relacionados se actualizará automáticamente.

```
-- Definición de las restricciones de integridad referencial (eliminación - update).

-- 1. Usuarios - Eventos: ON DELETE SET NULL, ON UPDATE CASCADE
-- Si un usuario (organizador) es eliminado, su evento asociado tendrá el campo OrganizadorID
-- establecido a NULL (no se elimina el evento).
-- Si un usuario es actualizado, el OrganizadorID en los eventos relacionados se actualizará automáticamente.

-- Modificar la tabla Eventos
ALTER TABLE Eventos
DROP FOREIGN KEY eventos_ibfk_1,
ADD CONSTRAINT fk_organizador_evento FOREIGN KEY (OrganizadorID) REFERENCES Usuarios(id_usuario)
ON DELETE SET NULL
ON UPDATE CASCADE;
```

RESTRICCIONES DE INTEGRIDAD

Eventos - Reservas: ON DELETE CASCADE, ON UPDATE CASCADE

- Si un evento es eliminado, todas las reservas asociadas a él se eliminarán automáticamente.
- Si un evento es actualizado, las reservas asociadas se actualizarán automáticamente.

```
-- 2. Eventos - Reservas: ON DELETE CASCADE, ON UPDATE CASCADE
-- Si un evento es eliminado, todas las reservas asociadas a él se eliminarán automáticamente.
-- Si un evento es actualizado, las reservas asociadas se actualizarán automáticamente.

-- Modificar la tabla Reservas
ALTER TABLE Reservas
DROP FOREIGN KEY reservas_ibfk_1,
ADD CONSTRAINT fk_evento_reserva FOREIGN KEY (evento_id) REFERENCES Eventos(id_evento)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

RESTRICCIONES DE INTEGRIDAD

Eventos - Reservas: ON DELETE CASCADE, ON UPDATE CASCADE

- Si un evento es eliminado, todas las reservas asociadas a él se eliminarán automáticamente.
- Si un evento es actualizado, las reservas asociadas se actualizarán automáticamente.

```
-- 2. Eventos - Reservas: ON DELETE CASCADE, ON UPDATE CASCADE
-- Si un evento es eliminado, todas las reservas asociadas a él se eliminarán automáticamente.
-- Si un evento es actualizado, las reservas asociadas se actualizarán automáticamente.

-- Modificar la tabla Reservas
ALTER TABLE Reservas
DROP FOREIGN KEY reservas_ibfk_1,
ADD CONSTRAINT fk_evento_reserva FOREIGN KEY (evento_id) REFERENCES Eventos(id_evento)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

RESTRICCIONES DE INTEGRIDAD

Usuarios - Reservas: ON DELETE CASCADE, ON UPDATE CASCADE

- Si un usuario (asistente) es eliminado, sus reservas asociadas también serán eliminadas.
- Si un usuario es actualizado, las reservas asociadas se actualizarán automáticamente.

```
-- 3. Usuarios - Reservas: ON DELETE CASCADE, ON UPDATE CASCADE
-- Si un usuario (asistente) es eliminado, sus reservas asociadas también serán eliminadas.
-- Si un usuario es actualizado, las reservas asociadas se actualizarán automáticamente.

-- Modificar la tabla Reservas
ALTER TABLE Reservas
DROP FOREIGN KEY reservas_ibfk_2,
ADD CONSTRAINT fk_asistente_reserva FOREIGN KEY (asistente_id) REFERENCES Usuarios(id_usuario)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

RESTRICCIONES DE INTEGRIDAD

Pagos - Reservas: ON DELETE RESTRICT, ON UPDATE CASCADE

- Si un usuario (asistente) es eliminado, sus reservas asociadas también serán eliminadas.
- Si un usuario es actualizado, las reservas asociadas se actualizarán automáticamente.

```
-- 4. Pagos - Reservas: ON DELETE RESTRICT, ON UPDATE CASCADE
-- No se puede eliminar un pago si está asociado a alguna reserva.
-- Si un pago es actualizado, la reserva relacionada se actualizará automáticamente.

-- Modificar la tabla Reservas
ALTER TABLE Reservas
DROP FOREIGN KEY reservas_ibfk_3,
ADD CONSTRAINT fk_pago_reserva FOREIGN KEY (id_pago) REFERENCES Pagos(id_pago)
ON DELETE RESTRICT
ON UPDATE CASCADE;
```

RESTRICCIONES DE INTEGRIDAD

4.6.5 Usuarios - Usuarios_Salas: ON DELETE CASCADE, ON UPDATE CASCADE

- Si un usuario es eliminado, todas sus asociaciones con salas (en la tabla intermedia Usuarios_Salas) serán eliminadas.
- Si un usuario es actualizado, las asociaciones en Usuarios_Salas se actualizarán automáticamente.

```
-- 5. Usuarios - Usuarios_Salas: ON DELETE CASCADE, ON UPDATE CASCADE
-- Si un usuario es eliminado, todas sus asociaciones con salas (en la tabla intermedia Usuarios_Salas) serán eliminadas.
-- Si un usuario es actualizado, las asociaciones en Usuarios_Salas se actualizarán automáticamente.

-- Modificar la tabla Usuarios_Salas
ALTER TABLE Usuarios_Salas
DROP FOREIGN KEY usuarios_salas_ibfk_1,
ADD CONSTRAINT fk_usuario_sala FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

RESTRICCIONES DE INTEGRIDAD

6. Salas - Usuarios_Salas: ON DELETE CASCADE, ON UPDATE CASCADE

- Si una sala es eliminada, todas las asociaciones de usuarios con esa sala (en la tabla Usuarios_Salas) también serán eliminadas.
- Si una sala es actualizada, las asociaciones en Usuarios_Salas se actualizarán automáticamente.

```sql
-- 6. Salas - Usuarios_Salas: ON DELETE CASCADE, ON UPDATE CASCADE
-- Si una sala es eliminada, todas las asociaciones de usuarios con esa sala (en la tabla Usuarios_Salas) también serán eliminadas.
-- Si una sala es actualizada, las asociaciones en Usuarios_Salas se actualizarán automáticamente.

-- Modificar la tabla Usuarios_Salas
ALTER TABLE Usuarios_Salas
DROP FOREIGN KEY usuarios_salas_ibfk_2,
ADD CONSTRAINT fk_sala_usuario FOREIGN KEY (id_sala) REFERENCES Salas(id_sala)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

LA IMPLEMENTACIÓN DE POLÍTICAS DE ACCESO Y SEGURIDAD

La implementación de políticas de acceso y seguridad es esencial para proteger los activos de información de una organización y garantizar la continuidad del negocio. Estas políticas ayudan a prevenir accesos no autorizados, proteger contra malware y otras amenazas, y garantizar que los empleados comprendan sus responsabilidades en materia de seguridad de la información.

Al seguir un proceso estructurado para la implementación de políticas de acceso y seguridad, las organizaciones pueden reducir significativamente su riesgo de sufrir incidentes de seguridad y proteger sus activos de información de manera efectiva.

IMPLEMENTACIÓN DE POLÍTICAS DE ACCESO Y SEGURIDAD

CREACION DE ROLES

```
-- Implementación de políticas de acceso y seguridad.

CREATE ROLE 'Invitado';
CREATE ROLE 'Organizador';
CREATE ROLE 'Asistente';
CREATE ROLE 'Administrador';
```

ASIGNACION DE PRIVILEGIOS PERTENECIENTES A CADA ROL

```
-- El rol de Invitado: Solo le permite ver eventos y salas
GRANT SELECT ON gestion_conferencias.Eventos TO 'Invitado';
GRANT SELECT ON gestion_conferencias.Salas TO 'Invitado';

-- el rol de Organizador: le permite crear, modificar y eliminar eventos, y gestionar salas
GRANT ALL PRIVILEGES ON gestion_conferencias.Eventos TO 'Organizador';
GRANT ALL PRIVILEGES ON gestion_conferencias.Salas TO 'Organizador';

-- el rol de sistente: le permite realizar reservar eventos y ver su información
GRANT SELECT, INSERT, UPDATE ON gestion_conferencias.Reservas TO 'Asistente';
GRANT SELECT ON gestion_conferencias.Eventos TO 'Asistente';

-- y por ultimo el rol de dministrador: no tiene restricciones, tiene todos los permisos..
GRANT ALL PRIVILEGES ON gestion_conferencias.* TO 'Administrador';
```

CIFRADO DE CONTRASEÑA

```sql
-- Implementación del cifrado de contraseñas
-- Se crea una columna en la tabla
ALTER TABLE Usuarios ADD COLUMN Contrasenia_encriptada VARBINARY(255);

-- Insertar un usuario con contraseña cifrada
INSERT INTO Usuarios (nombres_completos, correo, Telefono, Rol, Contrasenia, Contrasenia_encriptada)
VALUES ('Juan Pérez', 'juan.perez@mail.com', '123456789', 'Organizador', 'mi_password', AES_ENCRYPT('mi_password', 'clave_secreta'));

-- Recuperar y descifrar la contraseña para validación
SELECT nombres_completos, correo,
       AES_DECRYPT(Contrasenia_encriptada, 'clave_secreta') AS Contrasenia_Descifrada
FROM Usuarios;

-- Convertir el resultado a texto
-- Cambia la consulta para que MySQL convierta el resultado a CHAR:
SELECT nombres_completos, correo,
       CONVERT(AES_DECRYPT(Contrasenia_encriptada, 'clave_secreta') USING utf8) AS Contrasenia_Descifrada
FROM Usuarios;

-- Esta es la consulta general que muestra contraseña no encriptada y encriptada
SELECT id_usuario, nombres_completos, correo, Telefono, Rol, Contrasenia, Contrasenia_encriptada FROM Usuarios;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id_usuario	nombres_completos	correo	Telefono	Rol	Contrasenia	Contrasenia_encriptada
1	Juan Pérez	juan.perez@mail.com	123456789	Organizador	mi_password	BLOB

HABILITAR AUDITORÍA Y REGISTRAR EVENTOS.

HABILITAR EL LOG DE CONSULTAS GENERALES Y LENTAS

```
SET GLOBAL general_log = 'ON';

SET GLOBAL slow_query_log = 'ON';
SET GLOBAL long_query_time = 2; -- Registra consultas que tarden más de 2 segundos
```

CREAR UNA TABLA PARA ALMACENAR LOS LOGS DE AUDITORÍA

```
CREATE TABLE Auditoria (
    id_auditoria INT AUTO_INCREMENT PRIMARY KEY,
    id_usuario INT,
    accion VARCHAR(100),
    tabla_afectada VARCHAR(100),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario)
);
```

HABILITAR AUDITORÍA Y REGISTRAR EVENTOS.

CREAR UN TRIGGER PARA REGISTRAR LAS ACCIONES DE LOS USUARIOS

```
DELIMITER //
CREATE TRIGGER after_insert_eventos
AFTER INSERT ON Eventos
FOR EACH ROW
BEGIN
    INSERT INTO Auditoria (id_usuario, accion, tabla_afectada)
    VALUES (NEW.OrganizadorID, 'INSERT', 'Eventos');
END//
DELIMITER ;
```

REPETIR EL PROCESO PARA OTRAS ACCIONES Y TABLAS

```
DELIMITER //
CREATE TRIGGER after_update_eventos
AFTER UPDATE ON Eventos
FOR EACH ROW
BEGIN
    INSERT INTO Auditoria (id_usuario, accion, tabla_afectada)
    VALUES (NEW.OrganizadorID, 'UPDATE', 'Eventos');
END//
DELIMITER ;
```

```
DELIMITER //

CREATE TRIGGER after_delete_eventos
AFTER DELETE ON Eventos
FOR EACH ROW
BEGIN
    INSERT INTO Auditoria (id_usuario, accion, tabla_afectada)
    VALUES (OLD.OrganizadorID, 'DELETE', 'Eventos');
END//
DELIMITER ;
```