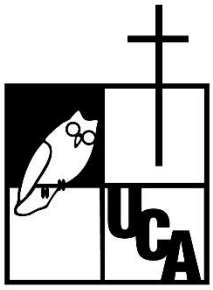


Universidad Centroamericana José Simeón Cañas

Administración de Base de Datos

Proyecto de Catedra

Equipo J

GABRIEL ALEJANDRO BATRES FLORES-00103923

GUILLERMO ALEJANDRO HERNANDEZ DUBON-00106423

ORLANDO EMMANUEL HERRERA ESTRADA-00084420

MARDEN ELIUTH LARIOS ORELLANA-00029223

DARWIN ORTIZ SANDOVAL-00042223

Fecha de entrega: 11/26/2025

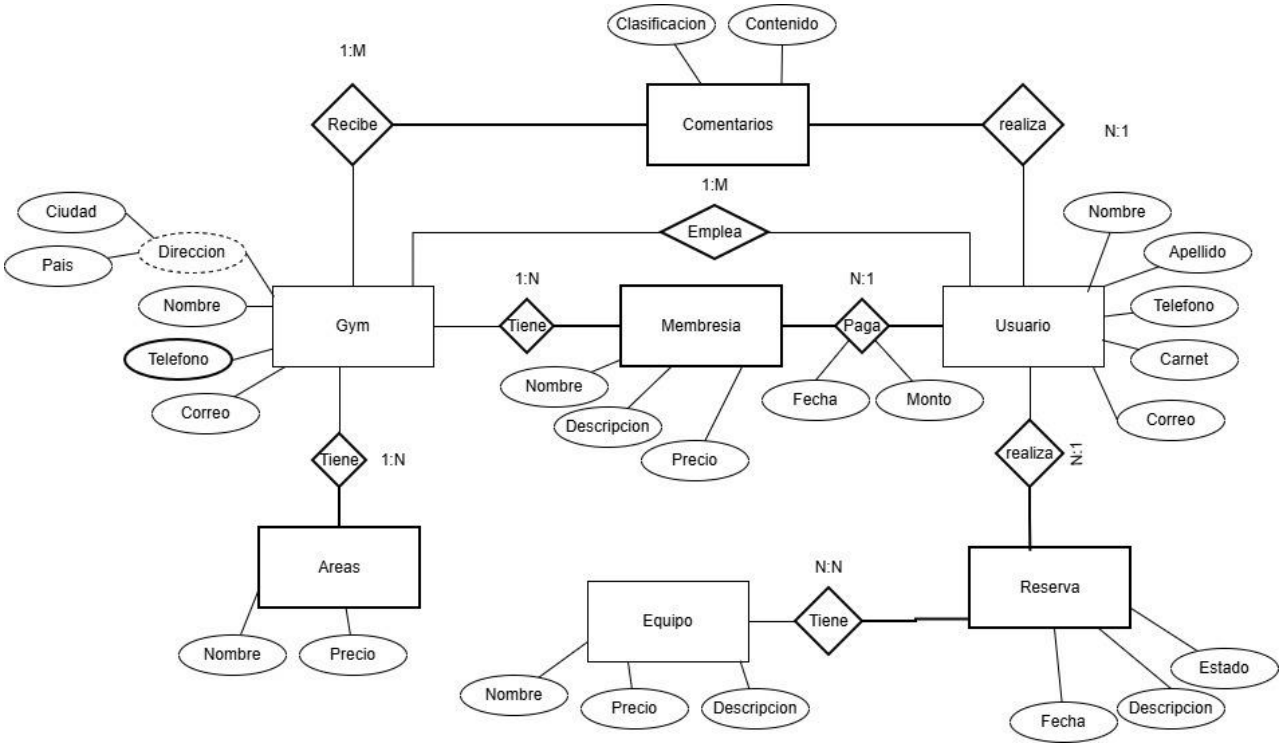
- ## Descripción del proyecto

El sistema de gestión de gimnasios centraliza la información y los procesos operativos de una cadena de gimnasios: administración de sucursales, gestión de membresías, registro y contacto de usuarios, reservas de espacios y equipos, pagos, comentarios de usuarios y un historial de acciones.

Está pensado para soportar operaciones comunes de front-office y back-office además de registro de clientes, compra de planes, control de vigencias, reservas y seguimiento de incidencias.

El modelo está organizado alrededor de la entidad **Gym** que representa a cada sucursal que conecta con membresías, áreas y contactos. Los **Usuarios** consumen servicios: compran **Membresías** (gestionadas con registros de **Pago** que determinan la vigencia), realizan **Reservas** (que pueden incluir varios **Equipos**) y dejan **Comentarios** sobre las sucursales. Un **Historial** registra actividades relevantes del sistema. Las relaciones están diseñadas para mantener integridad referencial entre estas piezas.

- ## Diagrama Entidad Relación



- **Diccionario de datos**

Convenciones a utilizar:

- **PK** = clave primaria.
- **FK** = clave foránea.
- **NULL** indica si la columna admite valores nulos.
- **DEFAULT** indica valor por defecto en el DDL proporcionado.
- **CHK** indica restricciones CHECK implementadas en la BD.

1. **Tabla: Gym**

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_gym	INT IDENTITY (1,1)	NO	SI	—	—	PK	Identificador único de la sucursal.
nombre	NVARCHAR (100)	NO	—	—	—	—	Nombre de la sucursal.
direccion	NVARCHAR (255)	SÍ	—	—	—	—	Dirección física (calle, número, etc.).
ciudad	NVARCHAR (100)	SÍ	—	—	—	—	Ciudad donde está el gym.
pais	NVARCHAR (100)	SÍ	—	—	—	—	País de la sucursal.

2. **Tabla: Contacto_Gym**

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_contacto	INT IDENTITY	NO	Sí	—	—	PK	Identificador del contacto.
id_gym	INT	NO	—	Sí **Gym(id_gym)**	—	FK	Referencia a la sucursal.
telefono	NVARCHAR (20)	SÍ	—	—	—	—	Teléfono de contacto.
correo	NVARCHAR (100)	SÍ	—	—	—	—	Correo electrónico del gym.

3. **Tabla: Menbresia**

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_membresia	INT IDENTITY	NO	Sí	—	—	PK	ID de la membresía.
id_gym	INT	NO	—	Sí — Gym(id_gym)	—	FK	Sucursal que ofrece la membresía.
nombre	NVARCHAR (100)	NO	—	—	—	—	Nombre del plan (ej. “Mensual Plus”).
descripcion	NVARCHAR(MAX)	SÍ	—	—	—	—	Detalle del plan.
precio	DECIMAL (10,2)	NO	—	—	—	> = 0	Costo del plan.
duracion	INT	NO	—	—	—	—	Duración del plan.

4. Tabla: Usuario

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_usuario	INT IDENTITY	NO	Sí	—	—	PK	Identificador del usuario.
id_membresia	INT	SÍ	—	Sí — **Membresia(id_membresia)**	—	FK	Membresía asociada al usuario (si aplica).
nombre	NVARCHAR (100)	NO	—	—	—	—	Nombre propio.
apellido	NVARCHAR (100)	NO	—	—	—	—	Apellidos.
carnet	NVARCHAR (20)	SÍ	—	—	—	UNIQUE	Identificación externa / carnet (único).
rol	NVARCHAR (50)	SÍ	—	—	—	—	Rol en el sistema (cliente,empleado,admin).
contrasena	NVARCHAR (255)	NO	—	—	—	—	Hash de contraseña (almacena solo hash).

5. Tabla: Contacto_Usuario

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_contacto_usuario	INT IDENTITY	NO	Sí	—	—	PK	ID del contacto del usuario.
id_usuario	INT	NO	—	Sí — **Usuario(id_usuario)**	—	FK	Usuario asociado.
telefono	NVARCHAR (20)	SÍ	—	—	—	—	Teléfono del usuario.
email	NVARCHAR (100)	SÍ	—	—	—	—	Email alternativo.

6. Tabla: Area

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_area	INT IDENTITY	NO	Sí	—	—	PK	ID de área.
nombre	NVARCHAR (100)	NO	—	—	—	—	Nombre del área (ej. “Sala de clases”).
precio	DECIMAL (10,2)	NO	—	—	—	>= 0	Tarifa por uso/reserva del área.
id_gym	INT	NO	—	Sí **Gym(id_gym)**	—	FK	Sucursal a la que pertenece.

7. Tabla: Equipo

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_equipo	INT IDENTITY	NO	Sí	—	—	PK	ID del equipo.
nombre	NVARCHAR (100)	NO	—	—	—	—	Nombre del equipo.
precio	DECIMAL (10,2)	NO	—	—	—	>= 0	Precio asociado al uso del equipo (si aplica).
descripcion	NVARCHAR(MAX)	SÍ	—	—	—	—	Detalle del equipo.

8. Tabla: Reserva

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_reserva	INT IDENTITY	NO	Sí	—	—	PK	ID de reserva.
id_usuario	INT	NO	—	Sí — **Usuario(id_usuario)**	—	FK	Usuario que realiza la reserva.
estado	NVARCHAR (50)	SÍ	—	—	—	—	Estado de la reserva (pendiente, confirmada, cancelada).
descripcion	NVARCHAR(MAX)	SÍ	—	—	—	—	Observaciones o propósito de la reserva.
fecha_reserva	DATETIME2	NO	—	—	—	—	Fecha y hora de la reserva.

9. Tabla: Extra (tabla puente Reserva-Equipo N: N)

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_reserva	INT	NO	Sí (parte)	Sí — **Reserva(id_reserva)**	—	PK compuesta	FK a la reserva.
id_equipo	INT	NO	Sí (parte)	Sí — **Equipo(id_equipo)**	—	PK compuesta	FK al equipo.

10. Tabla: Comentarios

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_comentario	INT IDENTITY	NO	Sí	—	—	PK	ID del comentario.
id_usuario	INT	NO	—	Sí — **Usuario(id_usuario)**	—	FK	Autor del comentario.
id_gym	INT	NO	—	Sí — **Gym(id_gym)**	—	FK	Gym al que se refiere.
comentario	NVARCHAR(MAX)	SÍ	—	—	—	—	Texto libre del comentario.
clasificacion	INT	SÍ	—	—	—	CHECK **(clasificacion BETWEEN 1 AND 5)**	Valor numérico de 1 a 5.

11. Tabla: Pago

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_pago	INT IDENTITY	NO	Sí	—	—	PK	ID del pago.
id_usuario	INT	NO	—	Sí — **Usuario(id_usuario)**	—	FK	Usuario que realiza el pago.
id_gym	INT	NO	—	Sí — **Gym(id_gym)**	—	FK	Sucursal donde se realiza/gestiona el pago.
id_membresia	INT	NO	—	Sí — **Membresia(id_membresia)**	—	FK	Membresía adquirida.
fecha_pago	DATETIME2	NO	—	—	GETDATE ()	—	Fecha/hora del pago
monto_pagado	DECIMAL (10,2)	NO	—	—	—	>= 0	Importe abonado.
fecha_inicio_vigencia	DATE	NO	—	—	—	—	Inicio de vigencia de la membresía.
fecha_fin_vigencia	DATE	NO	—	—	—	—	Fin de vigencia.
estado_pago	NVARCHAR (50)	NO	—	—	—	—	Estado (completado,pendiente,reembolsado, etc.).

12. Tabla: Historial

Columna	Tipo	NULL	PK	FK	DEFAULT	Restricciones	Descripción
id_historial	INT IDENTITY	NO	Sí	—	—	PK	ID del registro.
id_reserva	INT	SÍ	—	Sí — **Reserva(id_reserva)**	—	FK opcional	Reserva relacionada (si aplica).
id_usuario	INT	SÍ	—	Sí — **Usuario(id_usuario)**	—	FK opcional	Usuario actor o relacionado (si aplica).
tipo_accion	NVARCHAR(100)	SÍ	—	—	—	—	Tipo descriptivo de la acción.
fecha_registro	DATETIME2	NO	—	—	GETDATE()	—	Timestamp del evento.
descripcion	NVARCHAR(MAX)	SÍ	—	—	—	—	Detalle adicional.

- **Roles y privilegios**

Las tablas presentes muestran los usuarios (login → user → rol → privilegios), y la descripción de los Roles y permisos respectivamente.

Login (SQL Server)	Usuario en BD	Rol asignado	Privilegios principales (resumen)
admin_gym	admin_gym	Role_AdminGym	**DML total** en el esquema dbo: SELECT, INSERT, UPDATE, DELETE (permiso amplio para administración).
recepcion1	recepcion1	Role_Recepcion	SELECT, INSERT, UPDATE en **Reserva**; SELECT, INSERT en **Pago**; SELECT en **Usuario** y **Comentarios** (operaciones propias de recepción).
trainer1	trainer1	Role_Entrenador	SELECT en **Usuario**, **Reserva** y **Comentarios** (acceso lectura a clientes, reservas y feedback).
cliente1	cliente1	Role_Cliente	SELECT, INSERT en **Reserva**; SELECT, INSERT en **Comentarios** (cliente puede consultar y crear reservas y dejar comentarios).
gerencia1	gerencia1	Role_Gerencia	SELECT en todo el esquema dbo (acceso lectura amplia para reportes/análisis).
auditor1	auditor1	Role_ReadOnly	SELECT en todo el esquema dbo (usuario auditor con solo permisos de lectura).

Rol	Permisos GRANT (tal como aparece en script)	Ámbito / Observaciones
Role_AdminGym	GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA:dbo	Permisos DML completos sobre todas las tablas del esquema dbo.
Role_Recepcion	GRANT SELECT, INSERT, UPDATE ON Reserva; GRANT SELECT, INSERT ON Pago; GRANT SELECT ON Usuario; GRANT SELECT ON Comentarios	Operaciones de recepción: gestionar reservas y registrar pagos, ver usuarios y comentarios.
Role_Entrenador	GRANT SELECT ON Usuario; GRANT SELECT ON Reserva; GRANT SELECT ON Comentarios	Acceso exclusivamente de lectura a clientes, reservas y feedback.
Role_Cliente	GRANT SELECT, INSERT ON Reserva; GRANT SELECT, INSERT ON Comentarios	Clientes pueden leer/crear reservas y comentarios.
Role_Gerencia	GRANT SELECT ON SCHEMA:dbo	Lectura sobre todo el esquema (informes, métricas).
Role_ReadOnly	GRANT SELECT ON SCHEMA:dbo	Acceso de solo lectura (auditoría / consultas).

- **Políticas de seguridad**

1. Control de acceso y gestión de identidades:

- **Roles definidos:** Role_AdminGym, Role_Recepcion, Role_Entrenador, Role_Cliente, Role_Gerencia, Role_ReadOnly. Esto implementa el principio de menor privilegio ya que cada actor recibe únicamente los permisos necesarios (administrador: DML en dbo; recepción: gestionar reservas y pagos; entrenador: lectura; cliente: crear reservas/comentarios; gerencia/auditor: solo lectura).
- **Logins y usuarios de servidor/BD**: por ejemplo**.** *admin_gym***,** *recepcion1***,** *auditor1* y users asociados en la base de datos del Gym.
- **Asignación de permisos:** mediante ALTER ROLE ... ADD MEMBER y GRANT por objeto o esquema ej. *GRANT SELECT ON SCHEMA:dbo TO Role_Gerencia.*

2. Protección de operaciones críticas:

- **Bloqueo de eliminaciones de usuarios:** existe un *INSTEAD OF DELETE* (*Auditoria_TR_EliminarUsuario*) que registra intentos y evita borrar usuarios. Esto protege integridad y cumplimiento.

- **Procedimientos almacenados (SPs) para operaciones sensibles:** CRUD de usuario y módulo de pagos están centralizados en SPs (*sp_CrearUsuario, sp_RegistrarPago, sp_ActualizarEstadoPago*, etc.).

Esto permite controlar lógicas y registrar auditoría centralizada en SPs antes de cambios directos en tablas.

3. Comunicaciones y notificaciones

- **Database Mail** está habilitado y configurado para notificaciones cuentas/perfiles NotificacionGimnasio / PerfilNotificaciones o CuentaOutlook / PerfilOutlook según scripts que se muestran en el repositorio.
- **Triggers que envían correos:** *NotificarCambioUsuario*, *NotificarCambioReserva*, *NotificarCambioPago*, *NotificarCambioMembresia*. Estos envían emails registrando inserciones, updates y deletes, lo cual es util para alertas operativas y auditoría proactiva.

- ## Diseño de esquemas

1. Esquemas y separación lógica

- **Esquema dbo** para objetos operativos (tablas principales: Usuario, Pago, Reserva, etc.).

- **Esquema Auditoria** creado específicamente para registros de auditoría (*Auditoria.LogAccesos*), esta separación física y semántica entre datos operativos y registros de auditoría y facilita aplicar políticas distintas de permisos y retención (por ejemplo: permitir SELECT a *Role_Gerencia* sobre dbo pero restringir acceso a Auditoria a perfiles concretos de auditoría).

2. Diseño orientado a seguridad

- **Restricción de DML directo**: uso de SPs para crear/actualizar/consultar (*sp_CrearUsuario, sp_RegistrarPago, sp_CrearReserva*, etc.) permite:
  - Encapsular validaciones,
  - Garantizar que cada operación escriba en *Auditoria.LogAccesos*,
  - Simplificar la revocación de permisos (se puede DENY UPDATE/DELETE directo en la tabla y permitir sólo ejecución de SPs).

- **Triggers de protección y auditoría:**

  - *INSTEAD OF DELETE* sobre Usuario bloquea borrados accidentales y registra intento en la auditoría.
  - Triggers AFTER sobre tablas claves registran y notifican cambios.

- **Vistas y roles de solo lectura:** se crearon vistas (*vw_PagosPorGym, vw_ResumenReservasPorUsuario, vw_RankingIngresosGym, vw_HistorialEventos*) que sirven para exponer datos agregados sin otorgar permisos directos a las tablas subyacentes. Permitir SELECT sobre vistas a *Role_Gerencia/Role_ReadOnly* es más seguro que dar acceso a tablas base.

- **Índices para mantenimiento y rendimiento:** índices en columnas FK y campos de filtrado (*IX_Pago_id_usuario, IX_Reserva_estado, IX_Historial_reserva*, etc.). Mantiene la política de mantenimiento de índices rebuild y reorganize) en el plan operativo.

- ## Auditoría y Logs

  En esta sección se muestra como y en donde se audita dentro del sistema del sistema de Gimnasio.

- **Tabla central de auditoría:** *Auditoria.LogAccesos* con columnas: *id_auditoria* (PK), *usuario* (SYSTEM_USER o SUSER_SNAME), *luego accion* (INSERT/UPDATE/DELETE/SELECT/otros), *tabla_afectada*, *fecha* (GETDATE default), y por último el *detalle* donde colocamos un texto libre.

- **Fuentes de registros en la tabla de auditoría:**Triggers *Auditoria_TR_EliminarUsuario* y *Auditoria_TR_UpdateMembresia* insertan registros automáticos al producirse eliminaciones o actualizaciones.SPs (CRUD usuario, Pago, Reservas) insertan entradas de auditoría tras sus operaciones (cada SP hace INSERT INTO *Auditoria.LogAccesos ...*).*INSTEAD OF DELETE* en *Usuario* registra intentos de delete y evita el borrado real.

- **Notificaciones:** los triggers *NotificarCambio** envían correo con resumen del cambio (además de que SPs y triggers escriben en *Auditoria.LogAccesos*).

- **Consultas optimizadas e Índices**

En esta sección mostraremos algunas de las consultas de auditoria esenciales del sistema además de recalcar que han sido optimizadas haciendo uso de índices el resto de consultas son mostrados con más detalle en el repositorio de GitHub.

- **Índices:**

```
CREATE INDEX IX_Pago_id_usuario ON Pago(id_usuario);
CREATE INDEX IX_Pago_fecha_pago ON Pago(fecha_pago);
CREATE INDEX IX_Pago_estado ON Pago(estado_pago);
```

- **Consulta 1:**

```
CREATE OR ALTER PROCEDURE sp_RegistrarPago
   @id_usuario INT,
   @id_gym INT,
   @id_membresia INT,
   @monto_pagado DECIMAL(10,2),
   @fecha_inicio DATE,
   @fecha_fin DATE,
   @estado NVARCHAR(20)
AS
BEGIN
   SET NOCOUNT ON;

   INSERT INTO Pago (id_usuario, id_gym, id_membresia, monto_pagado,
                     fecha_pago, fecha_inicio_vigencia, fecha_fin_vigencia,
estado_pago)
   VALUES (@id_usuario, @id_gym, @id_membresia, @monto_pagado,
GETDATE(),
          @fecha_inicio, @fecha_fin, @estado);

   INSERT INTO Auditoria.LogAccesos (usuario, accion, tabla_afectada,
fecha, detalle)
   VALUES (SUSER_SNAME(), 'INSERT', 'Pago', GETDATE(),
          'Nuevo pago registrado para usuario=' + CAST(@id_usuario AS
NVARCHAR(10)));
END;
GO
```

- **Consulta 2:**

```
CREATE OR ALTER PROCEDURE sp_ConsultarPagos
```

```
    @id_usuario INT = NULL
AS
BEGIN
    SET NOCOUNT ON;

    SELECT p.*, u.nombre, u.apellido
    FROM Pago p
    INNER JOIN Usuario u ON u.id_usuario = p.id_usuario
    WHERE (@id_usuario IS NULL OR p.id_usuario = @id_usuario)
    ORDER BY p.fecha_pago DESC;

    INSERT INTO Auditoria.LogAccesos (usuario, accion, tabla_afectada,
fecha, detalle)
    VALUES (SUSER_SNAME(), 'SELECT', 'Pago', GETDATE(),
        'Consulta de pagos para usuario=' + ISNULL(CAST(@id_usuario AS
NVARCHAR(10)), 'TODOS'));
END;
GO
```

- **Estrategia de dimensionamiento, respaldo y recuperación.**

1. Adimensionamiento:

loos números que se utilizan son estimaciones de capacidad para un despliegue inicial de la base de datos. Ajustado a los datos obtenidos durante el desarrollo del sistema el factor por índices que se utilizó es 1.25 (25% de overhead por índices y estructuras internas).

Ejemplo de cálculo:

- **id_usuario = 4**
- **id_membresia = 4**
- **nombre = 60**
- **apellido = 60**
- **carnet = 20**
- **rol = 20**
- **contrasena = 120**
- **overhead fila (metadatos, nulos, etc.) = 34**

1. 4 + 4 = 8
2. 8 + 60 = 68
3. 68 + 60 = 128
4. 128 + 20 = 148
5. 148 + 20 = 168
6. 168 + 120 = 288
7. 288 + 34 = 322 bytes por fila

Con 10,000 filas:

- 322 × 10,000 = 3,220,000 bytes (dato total)

Conversión a megabytes (1 MB = 1,048,576 bytes):

- 3,220,000 ÷ 1,048,576 =
  - 1,048,576 × 3 = 3,145,728
  - Resto = 3,220,000 − 3,145,728 = 74,272
  - Fracción = 74,272 ÷ 1,048,576 ≈ 0.0708
  - Resultado ≈ 3 + 0.0708 = 3.0708 MB (≈ 3.07 MB)

Aplicando factor por índices 1.25:

- 3,220,000 × 1.25 = 4,025,000 bytes total
- 4,025,000 ÷ 1,048,576 ≈ 3.84 MB

Por tanto, Usuario = 3.07 MB de datos y 3.84 MB incluyendo índice/overhead estimado.

Tabla	Filas estimadas	Tamaño fila estimado (bytes)	Tamaño datos (MB)	Tamaño con índices (MB, factor 1.25)
Gym	10	264 B	0.003	0.004
Contacto_gym	30	~88 B	0.003	0.004
Membresia	50	477 B	0.023	0.029
Usuario	10,000	322 B	3.07 MB	3.84 MB
Contacto_usuario	5,000	88 B	0.42	~0.53
Area	50	73 B	0.003	0.004
Equipo	500	269 B	0.13	0.16
Reserva	50,000	~236 B	11.23	14.04
Extra	80,000	8 B (2 INT)	0.61	0.76
Comentarios	20,000	212 B	4.05	5.06
Pago	30,000	55 B	1.57	1.96
Historial	200,000	280 B	53.62	67.03
TOTAL	—	—	74.25 MB	92.95 B

2. Estrategia de respaldo de la base de datos:

El script y los detalles completos se encuentra en el repositorio de GitHub, pero en general:

Backups y Jobs (SQL Agent):

- Full diario (02:00) con CHECKSUM + RESTORE VERIFYONLY + notificación por mail.
- Differential diario (14:00).
- Log backups cada 15 minutos.
- Job de monitor de espacio (*VW_DiscoSQL*) cada 10 minutos.
- Scripts de job con notificaciones en caso de fallo (DB Mail).

Retención y RPO/RTO implícitos: FULL diario + DIFF diario + LOGs cada 15 min. Esto da un RPO cercano a 15 minutos y un RTO basado en el tiempo de restauración de: último FULL + último DIFF + aplicar LOGs.