Tecnicatura Universitaria en Programación
Base de Datos II

Equipo # 13

Integrantes:

Greco, Emmanuel Antonio - 31714
Mazzitelli, Marcos - 31770
Sánchez, Julieta Macarena - 31691
Taquino, Pedro - 31768

Explicación del Sistema

Nuestro sistema está diseñado para llevar el control de los turnos en una clínica, según los horarios disponibles de cada médico y la especialidad que éstos ejercen. Su propósito es facilitar el trabajo del personal administrativo de la clínica, ahorrándoles ciertas tareas tediosas como, por ejemplo, tener que chequear la disponibilidad de los doctores al dar un turno, o tener que llevar un control del historial clínico de los pacientes, entre otras.

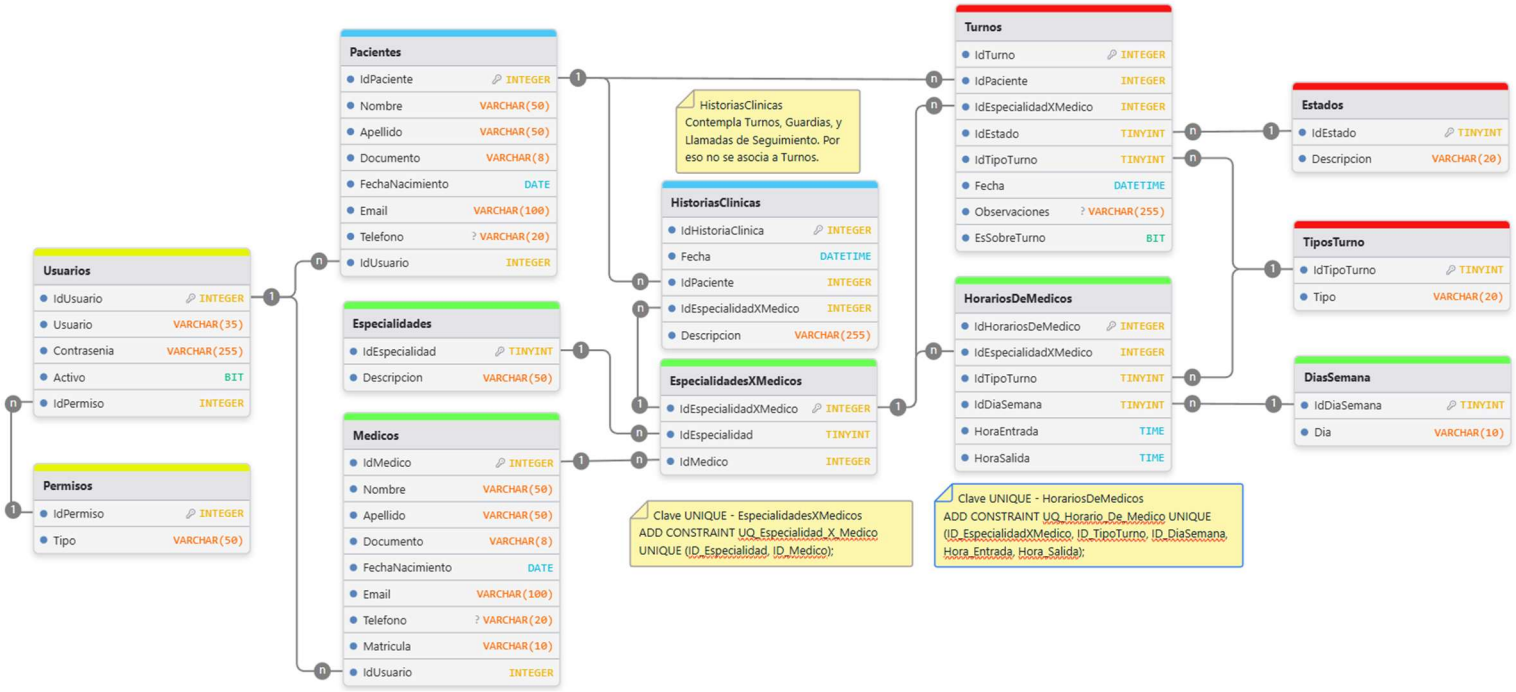
Funcionalidades principales

- **Registro de horarios disponibles:** Los médicos podrán establecer en qué horarios trabajan según el día de la semana. Permite además la distinción según el tipo de turno (presencial, virtual, urgencia) y según la especialidad. Esto significa que, para un mismo horario dado, un médico podría ofrecer atención en mas de un tipo de turno, y en más de una especialidad, pero no significa que pueda tener dos turnos en el mismo horario.
- **Registro de turno:** Se puede registrar un turno, asignado a un paciente y un médico, dentro de una especialidad y un tipo de turno dados. Como es de esperarse, el médico previamente tiene que haber registrado ese horario para ese tipo de turno y para esa especialidad.
- **Reporte por paciente:** Los empleados podrán ver los datos de contacto, la edad y un resumen de su historial de turnos.

- **Control automático de turnos:** En casos especiales, como, por ejemplo, si un médico se da de baja del sistema, todos los turnos asignados a él son cancelados. A su vez, se controla que no puedan ser modificados aquellos turnos ya concluidos o cancelados, así como también se registran con una observación especial aquellos turnos que sean de tipo “Sobreturno”, para conveniencia del médico y del paciente.

Se utiliza una base de datos relacional para administrar los datos y procesos mencionados. Para este fin, se utilizan 5 vistas, 3 procedimientos almacenados y 3 triggers:

Diagrama de Entidad Relación



Objetos de Base de Datos clave en el sistema

Vista 1 - V_Medicos_Especialidades

Reporte de administración de personal médico. Muestra un listado de médicos con sus datos personales (Email, Teléfono, Matrícula) y la Especialidad que atienden.
Además, también se puede visualizar su nombre de usuario y si el mismo se encuentra activo o no. Utiliza la función de usuario dbo.CalcularEdad() para facilitar la información al personal.

```
1 | CREATE VIEW V_Medicos_Especialidades
2 | AS
3 | SELECT
4 |     M.Apellido + ', ' + M.Nombre AS 'Nombre Completo Medico',
5 |
6 |     M.FechaNacimiento,
7 |     dbo.CalcularEdad(M.FechaNacimiento, GETDATE()) AS Edad,
8 |     M.Email,
9 |     M.Telefono,
10|     M.Matricula,
11|     E.Descripcion AS Especialidad,
12|     U.Usuario,
13|     U.Activo
14| FROM Medicos M
15| INNER JOIN EspecialidadesXMedicos EXM ON M.IdMedico = EXM.IdMedico
16| INNER JOIN Especialidades E ON EXM.IdEspecialidad = E.IdEspecialidad
17| INNER JOIN Usuarios U ON M.IdUsuario = U.IdUsuario;
18|
```

Vista 2 - V_Horarios_Detallados

Es un reporte operativo que permite ver una lista de horarios completa mostrando qué médico atiende qué especialidad, en qué día, en qué rango horario, y para qué tipo de turno (Presencial, Virtual, etc.)

Es una vista fundamental a la hora de crear un turno ya que estos datos informan sobre la disponibilidad horaria de cada médico.

```
1 | CREATE VIEW V_Horarios_Detallados
2 | AS
3 | SELECT
4 |     M.Apellido + ', ' + M.Nombre AS 'Nombre Completo Medico',
5 |     M.Matricula,
6 |     E.Descripcion AS Especialidad,
7 |     TT.Tipo AS TipoTurno,
8 |     DS.Dia AS DiaSemana,
9 |     HDM.HoraEntrada,
10|     HDM.HoraSalida,
11|     DS.IdDiaSemana
12| FROM HorariosDeMedicos HDM
13| INNER JOIN EspecialidadesXMedicos EXM ON HDM.IdEspecialidadXMedico =
EXM.IdEspecialidadXMedico
14| INNER JOIN Especialidades E ON EXM.IdEspecialidad = E.IdEspecialidad
15| INNER JOIN Medicos M ON EXM.IdMedico = M.IdMedico
16| INNER JOIN TiposTurno TT ON HDM.IdTipoTurno = TT.IdTipoTurno
17| INNER JOIN DiasSemana DS ON HDM.IdDiaSemana = DS.IdDiaSemana;
18|
```

Vista 3 - V_Pacientes_Turnos

Permite ver un listado detallado de todos los turnos, ideal para un recepcionista, que incluye el estado del turno, los datos del paciente, y también los datos del médico, para facilitar con qué médico tiene el turno y en cuál de las especialidades que este médico dispone.

```
1 | CREATE VIEW V_Pacientes_Turnos
2 | AS
3 | SELECT
4 |     P.Nombre AS NombrePaciente,
5 |     P.Apellido AS ApellidoPaciente,
6 |     P.Documento AS Documento,
7 |     EP.Descripcion AS Especialidad,
8 |     M.Nombre AS NombreMedico,
9 |     M.Apellido AS ApellidoMedico,
10|     T.Fecha AS FechaYHora,
11|     TT.Tipo AS TipoDeTurno,
12|     T.IdTurno AS IdTurno,
13|     T.EsSobreTurno AS EsSobreTurno,
14|     ES.Descripcion AS Estado,
15|     T.Observaciones
16| FROM Pacientes P
17| INNER JOIN Turnos T ON P.IdPaciente = T.IdPaciente
18| INNER JOIN Estados ES ON ES.IdEstado = T.IdEstado
19| INNER JOIN TiposTurno TT ON TT.IdTipoTurno = T.IdTipoTurno
20| INNER JOIN EspecialidadesXMedicos EXM ON EXM.IdEspecialidadXMedico =
T.IdEspecialidadXMedico
21| INNER JOIN Medicos M ON M.IdMedico = EXM.IdMedico
22| INNER JOIN Especialidades EP ON EP.IdEspecialidad = EXM.IdEspecialidad;
23|
```

Vista 4 - V_Pacientes_Reporte

Muestra el listado maestro de pacientes con sus datos de contacto, su edad (usando la función de usuario ya mencionada, dbo.CalcularEdad()), mostrando también un contador de turnos completados y cancelados para conocer el comportamiento del paciente en el pasado.

```
CREATE VIEW V_Pacientes_Reporte
AS
SELECT
    P.IdPaciente,
    P.Nombre,
    P.Apellido,
    P.Documento,
    P.Email,
    P.Telefono,
    P.FechaNacimiento,
    dbo.CalcularEdad(P.FechaNacimiento, GETDATE()) AS Edad,
    COUNT (T.IdTurno) AS TurnosTotales,
    COUNT (CASE WHEN E.Descripcion = 'Cancelado' THEN 1
            END) AS TurnosCancelados
FROM Pacientes P
LEFT JOIN Turnos T ON P.IdPaciente = T.IdPaciente
LEFT JOIN Estados E ON T.IdEstado = E.IdEstado
GROUP BY
    P.IdPaciente, P.Nombre, P.Apellido, P.Documento,
    P.Email, P.Telefono, P.FechaNacimiento;
```

Vista 5 - V_Medicos_ConEstado

Muestra el listado de médicos y su estado (Activo / Inactivo), ya que es utilizada en la pantalla donde se realizan las altas y bajas correspondientes.

```
CREATE VIEW V_Medicos_ConEstado
AS
SELECT
    M.IdMedico,
    M.Nombre,
    M.Apellido,
    M.Documento,
    M.FechaNacimiento,
    M.Email,
    M.Telefono,
    M.Matricula,
    U.Activo AS EstadoUsuario
FROM Medicos M
INNER JOIN Usuarios U ON M.IdUsuario = U.IdUsuario;
```

Procedimiento Almacenado 1 - SP_Agregar_Horarios_De_Medicos

SP de Acción, que permite registrar el horario laboral de un médico. El mismo recibe como parámetros IdMedico, IdEspecialidad, IdTipoTurno, IdDiaSemana, HoraEntrada y HoraSalida. Verifica que todos los IDs existan en sus tablas, luego verifica que la HoraEntrada sea anterior a la HoraSalida, pero lo más importante es que verifica que el médico (IdMedico) esté realmente asignado a esa especialidad (IdEspecialidad) consultando la tabla EspecialidadesXMedicos. Además, otra validación clave que realiza es verificar que este nuevo horario no se pise o interfiera con ningún otro horario ya existente para ese médico en ese día y también que el mismo esté dado de alta.

Utiliza TRY - CATCH - THROW para devolver el error a la aplicación web.

```
1 | CREATE PROCEDURE SP_Agregar_Horarios_De_Medicos(
2 |     @IdMedico INT,
3 |     @IdEspecialidad TINYINT,
4 |     @IdTipoTurno TINYINT,
5 |     @IdDiaSemana TINYINT,
6 |     @HoraEntrada TIME,
7 |     @HoraSalida TIME
8 | )
9 | AS
10| BEGIN
11|     BEGIN TRY
12|         IF @IdMedico NOT IN (SELECT IdMedico FROM Medicos) BEGIN
13|             RAISERROR('Medico inexistente.', 16, 1);
14|         END
15|             IF @IdEspecialidad NOT IN (SELECT IdEspecialidad FROM
Especialidades) BEGIN
16|             RAISERROR('Especialidad inexistente.', 16, 1);
17|         END
18|         IF @IdTipoTurno NOT IN (SELECT IdTipoTurno FROM TiposTurno) BEGIN
19|             RAISERROR('Tipo de turno inexistente.', 16, 1);
20|         END
21|         IF @IdDiaSemana NOT IN (SELECT IdDiaSemana FROM DiasSemana) BEGIN
22|             RAISERROR('Dia de semana inexistente.', 16, 1);
23|         END
24|
25|         IF @HoraEntrada >= @HoraSalida BEGIN
26|             RAISERROR('No puede haber un horario con duracion menor o
igual a 0.', 16, 1);
27|         END
28|
29|
30|          IF (SELECT COUNT(*) FROM EspecialidadesXMedicos WHERE IdMedico
= @IdMedico  AND IdEspecialidad = @IdEspecialidad) = 0 BEGIN
31|                 RAISERROR('El medico no esta dado de alta en esta
especialidad.', 16, 1);
32|         END
33|         ELSE BEGIN
34|             DECLARE @IdEspecialidadXMedico INT
35|             SELECT @IdEspecialidadXMedico = IdEspecialidadXMedico FROM
EspecialidadesXMedicos WHERE IdMedico = @IdMedico AND IdEspecialidad =
@IdEspecialidad
36|         END
37|
```

```
38|         IF (SELECT COUNT(*) FROM HorariosDeMedicos
39|                 WHERE IdTipoTurno = @IdTipoTurno
40|                 AND IdEspecialidadXMedico = @IdEspecialidadXMedico
41|                 AND IdDiaSemana = @IdDiaSemana
42|                 AND HoraEntrada < @HoraSalida
43|                 AND HoraSalida > @HoraEntrada) > 0
44|         BEGIN
45|            RAISERROR('El horario seleccionado interfiere con otro horario
establecido.', 16, 1);
46|         END
47|
48|         IF (SELECT COUNT (*) FROM EspecialidadesXMedicos EXM
49|             INNER JOIN Medicos M ON EXM.IdMedico = M.IdMedico
50|             INNER JOIN Usuarios U ON M.IdUsuario = U.IdUsuario
51|             WHERE IdEspecialidadXMedico = @IdEspecialidadXMedico
52|             AND U.Activo = 0 ) > 0
53|         BEGIN
54|             RAISERROR('El medico se encuentra dado de baja.' , 16, 1)
55|         END
56|
57|        INSERT INTO HorariosDeMedicos (IdEspecialidadXMedico, IdTipoTurno,
IdDiaSemana, HoraEntrada, HoraSalida)
58|                             VALUES (@IdEspecialidadXMedico, @IdTipoTurno,
@IdDiaSemana, @HoraEntrada, @HoraSalida)
59|
60|   END TRY
61|      BEGIN CATCH
62|          DECLARE @msg NVARCHAR(4000) = ERROR_MESSAGE();
63|          THROW 50000, @msg, 1;
64|      END CATCH
65| END
```

Procedimiento Almacenado 2 - SP_Registrar_Turno

SP de Acción, que resuelve el alta de un nuevo turno para un paciente agregando lógica de negocio para impedir que se ingresen registros invalidos a través de algunas validaciones. El mismo recibe como parámetros IdPaciente, IdEspecialidadXMedico, IdTipoTurno, Fecha, Observaciones y EsSobreTurno.

Verifica que el IdPaciente y el IdEspecialidadXMedico existan. También verifica que el médico atienda en ese día y hora. Usa DATEPART(WEEKDAY, @Fecha) para comparar el día de la semana y BETWEEN para considerar un rango horario dentro de la hora de entrada y la de salida del médico. Una de las validaciones clave es que verifica que no exista otro turno para ese médico en un rango de 29 minutos antes y después del nuevo turno que se intenta registrar, a excepción de que sea un sobreturno, en este último caso, sí se permite solapamiento con otros turnos. Otro dato muy importante es que también se verifica que tanto el médico como el paciente se encuentren dados de alta.

Usa ROLLBACK TRANSACTION y RETURN en cada validación fallida, y TRY - CATCH - THROW para devolver el error a la aplicación web.

```
1 | CREATE PROCEDURE SP_Registrar_Turno
2 |     @IdPaciente INT,
3 |     @IdEspecialidadXMedico INT,
4 |     @IdTipoTurno TINYINT,
5 |     @Fecha DATETIME,
6 |     @Observaciones VARCHAR(255) = NULL,
7 |     @EsSobreTurno BIT
8 | AS
9 | BEGIN
10|
11|
12|     BEGIN TRY
13|
14|         BEGIN TRANSACTION;
15|             DECLARE @HoraTurno TIME;
16|             DECLARE @IdEstadoPendiente TINYINT;
17|             DECLARE @IdEstadoCancelado TINYINT;
18|             DECLARE @IdEstadoAtendido TINYINT;
19|
20|             SET @HoraTurno = CONVERT(TIME, @Fecha);
21|
22|                IF (SELECT COUNT (*) FROM Pacientes WHERE IdPaciente =
@IdPaciente) = 0
23|             BEGIN
24|                 RAISERROR('El paciente especificado no existe.', 16, 1);
25|                  ROLLBACK TRANSACTION;
26|                  RETURN;
27|             END
28|
29|             IF (SELECT COUNT (*) FROM Pacientes P
30|             INNER JOIN Usuarios U ON P.IdUsuario = U.IdUsuario
31|             WHERE IdPaciente = @IdPaciente
32|             AND U.Activo = 0) > 0
33|             BEGIN
```

```
34|                 RAISERROR('El paciente se encuentra dado de baja.', 16,
1);
35|                 ROLLBACK TRANSACTION;
36|                 RETURN;
37|             END
38|
39|
40|             -- Validacion que el medico tenga esa especialidad asignada.
41|                IF (SELECT COUNT (*) FROM EspecialidadesXMedicos WHERE
IdEspecialidadXMedico = @IdEspecialidadXMedico) = 0
42|             BEGIN
43|                   RAISERROR('La combinacion de medico y especialidad no
existe.', 16, 1);
44|                 ROLLBACK TRANSACTION;
45|                 RETURN;
46|             END
47|
48|             IF @Fecha <= GETDATE()
49|             BEGIN
50|                RAISERROR('No se puede registrar un turno en el pasado.',
16, 1);
51|                 ROLLBACK TRANSACTION;
52|                 RETURN;
53|             END
54|
55|             -- Validacion medico dado de baja.
56|             IF (SELECT COUNT (*) FROM EspecialidadesXMedicos EXM
57|             INNER JOIN Medicos M ON EXM.IdMedico = M.IdMedico
58|             INNER JOIN Usuarios U ON M.IdUsuario = U.IdUsuario
59|             WHERE IdEspecialidadXMedico = @IdEspecialidadXMedico
60|             AND U.Activo = 0) > 0
61|             BEGIN
62|                 RAISERROR('El medico se encuentra dado de baja', 16, 1);
63|                 ROLLBACK TRANSACTION;
64|                 RETURN;
65|             END
66|            -- Validacion que el medico tenga esa fecha y horario asignado
con esa especialidad y ese tipo de turno
67|             SET DATEFIRST 1;  -- para que el weekday 1 sea lunes.
68|             IF (SELECT COUNT (*) FROM HorariosDeMedicos H
69|               INNER JOIN DiasSemana DS ON H.IdDiaSemana = DS.IdDiaSemana
70|                 WHERE H.IdEspecialidadXMedico = @IdEspecialidadXMedico
71|                   AND H.IdTipoTurno = @IdTipoTurno
72|                   AND DATEPART(WEEKDAY, @Fecha) = DS.IdDiaSemana
73|                  AND @HoraTurno BETWEEN H.HoraEntrada AND H.HoraSalida)
= 0
74|             BEGIN
75|                     RAISERROR('El medico no atiende en el dia y hora
seleccionados, o el tipo de turno no es correcto.', 16, 1);
76|                 ROLLBACK TRANSACTION;
77|                 RETURN;
78|             END
79|
80|              -- Validacion que el medico no tenga un turno asignado en
esa fecha y 30 minutos antes y despues
81|               DECLARE @FechaInicioRango DATETIME = DATEADD(MINUTE, -29,
@Fecha); --  29 min antes que comience el nuevo turno
82|                DECLARE @FechaFinRango DATETIME = DATEADD(MINUTE, 29,
@Fecha);     --  29  min despues del nuevo turno
83|            SET @IdEstadoCancelado = (SELECT IdEstado FROM Estados WHERE
Descripcion = 'Cancelado');
```

```
84|             SET @IdEstadoAtendido = (SELECT IdEstado FROM Estados WHERE
Descripcion = 'Completado');
85|
86|             IF (SELECT COUNT (*) FROM Turnos T
87|                 WHERE T.IdEspecialidadXMedico = @IdEspecialidadXMedico
88|                 AND T.Fecha BETWEEN @FechaInicioRango AND @FechaFinRango
89|                           AND  T.IdEstado  NOT  IN  (@IdEstadoCancelado,
@IdEstadoAtendido)
90|                   AND @EsSobreTurno = 0) > 0
91|             BEGIN
92|                  RAISERROR('El medico ya tiene un turno asignado en esa
fecha y hora.', 16, 1);
93|                 ROLLBACK TRANSACTION;
94|                 RETURN;
95|             END
96|
97|            SET @IdEstadoPendiente = (SELECT IdEstado FROM Estados WHERE
Descripcion = 'Pendiente');
98|
99|                 INSERT  INTO  Turnos  (IdPaciente,  IdEspecialidadXMedico,
IdEstado, IdTipoTurno, Fecha, Observaciones, EsSobreTurno)
100|                          VALUES   (@IdPaciente,   @IdEspecialidadXMedico,
@IdEstadoPendiente, @IdTipoTurno, @Fecha, @Observaciones, @EsSobreTurno);
101|
102|         COMMIT TRANSACTION;
103|
104|     END TRY
105|     BEGIN CATCH
106|         ROLLBACK TRANSACTION;
107|         THROW;
108|     END CATCH
109| END
110|
```

Procedimiento Almacenado 3 - SP_Reporte_Historia_Paciente

SP de reporte parametrizado, que recibe un único parámetro, el IdPaciente.

Este procedimiento realiza INNER JOINS con las tablas HistoriasClinicas, Pacientes, EspecialidadesXMedicos, Medicos y Especialidades con la finalidad de devolver un antecedente clínico completo de ese paciente para que el Médico pueda acceder a una información completa y ordenada por fecha de forma descendente sobre todos los registros que se le hicieron, con qué médicos y a qué especialidad acudió, que descripción se anotó, etc.

```
1 | CREATE PROCEDURE SP_Reporte_Historia_Paciente(
2 | @IdPaciente INT)
3 | AS
4 | BEGIN
5 |
6 |     SELECT
7 |         HC.IdHistoriaClinica AS Id,
8 |         HC.Fecha,
9 |         P.Nombre AS NombrePaciente,
10|         P.Apellido AS ApellidoPaciente,
11|         P.FechaNacimiento AS FechaNacimiento,
12|         dbo.CalcularEdad(P.FechaNacimiento, GETDATE()) AS Edad,
13|         P.Documento,
14|         M.Nombre AS NombreMedico,
15|         M.Apellido AS ApellidoMedico,
16|         E.Descripcion AS Especialidad,
17|         HC.Descripcion AS DescripcionHistoria
18|     FROM HistoriasClinicas HC
19|     INNER JOIN Pacientes P ON HC.IdPaciente = P.IdPaciente
20|     INNER JOIN EspecialidadesXMedicos EXM ON HC.IdEspecialidadXMedico =
EXM.IdEspecialidadXMedico
21|     INNER JOIN Medicos M ON EXM.IdMedico = M.IdMedico
22|     INNER JOIN Especialidades E ON EXM.IdEspecialidad = E.IdEspecialidad
23|     WHERE HC.IdPaciente = @IdPaciente
24|     ORDER BY
25|         HC.Fecha DESC;
26| END
27|
```

Trigger 1 - TR_MedicoInactivado_CancelarTurnos

Es un trigger de automatización tras una eliminación. Sin embargo, en lugar de un borrado físico, implementa una baja lógica para no perder integridad de las demás tablas.
Este trigger se dispara en lugar de (Instead OF) un comando DELETE sobre la tabla Medicos, evitando que el médico sea borrado físicamente de la base de datos para, en lugar de eso, realiza un UPDATE sobre la tabla Usuarios seteando el atributo Activo en 0 (baja lógica).
Además, realiza una automatización donde busca todos los turnos futuros (fecha mayor a la actual) con estado “Pendientes” y los actualiza a cancelados.

```
1 | CREATE TRIGGER TR_MedicoInactivado_CancelarTurnos ON Medicos
2 | INSTEAD OF DELETE
3 | AS
4 | BEGIN
5 |
6 |     DECLARE @IdUsuario INT = (SELECT IdUsuario FROM DELETED);
7 |     DECLARE @IdMedico INT = (SELECT IdMedico FROM Deleted);
8 |      DECLARE @IdEstadoCancelado TINYINT = (SELECT IdEstado FROM Estados
WHERE Descripcion = 'Cancelado');
9 |      DECLARE @IdEstadoPendiente TINYINT = (SELECT IdEstado FROM Estados
WHERE Descripcion = 'Pendiente');
10|
11|     -- Seteamos el usuario en 0 para inactivarlo.
12|     UPDATE Usuarios SET Activo = 0 WHERE IdUsuario = @IdUsuario;
13|
14|        -- Actualizamos los estados de los turnos pendientes futuros de
este medico a Cancelado
15|     UPDATE Turnos
16|     SET
17|         IdEstado = @IdEstadoCancelado
18|     WHERE
19|            IdEspecialidadXMedico IN (SELECT IdEspecialidadXMedico FROM
EspecialidadesXMedicos WHERE IdMedico = @IdMedico)
20|         AND Fecha > GETDATE()  -- (Solo para turnos futuros y con estado
pendiente)
21|          AND IdEstado = @IdEstadoPendiente;
22| END
23|
```

Trigger 2 - TR_Observacion_Sobreturno

Es un trigger de automatización tras una inserción en la tabla turnos.

Éste se dispara automáticamente luego de las inserciones en la tabla turnos, verificando que, si fueron creadas como sobreturno (atributo BIT), se añade la palabra “Sobreturno:” concatenado con la observación inicial sobre el campo Observaciones, para facilitar la información al médico o recepcionista.

```
1 | CREATE TRIGGER TR_Observacion_Sobreturno ON Turnos
2 | AFTER INSERT
3 | AS
4 | BEGIN
5 |     -- Se obtiene el id del turno que se acaba de crear
6 |     DECLARE @IdTurno INT = (SELECT IdTurno FROM inserted);
7 |
8 |     -- Si es sobre turno...
9 |     IF (SELECT EsSobreTurno FROM inserted) = 1 BEGIN
10|         -- Agregar la etiqueta sobreturno en la observacion del turno.
11|         UPDATE Turnos SET Observaciones = 'SOBRETURNO: ' + Observaciones
WHERE IdTurno = @IdTurno;
12|     END
13| END
14|
```

Trigger 3 - TR_No_Modificar_Turnos_Cerrados

Este trigger se dispara luego de una modificación (UPDATE) en la tabla Turnos y funciona de la siguiente manera:

Verifica que la actualización sea sobre la columna Estado y si el valor antiguo (tabla deleted) de ese turno fue “Cancelado” o “Completado” significa que esos Turnos no deberían modificarse, por lo que finaliza la operación inmediatamente.

Si pasa esa validación, lo siguiente que hace es verificar que el nuevo valor sea “Cancelado” y en caso de que sea así, en el atributo Observaciones, coloca “Turno cancelado” para facilitar tanto al personal médico como a los pacientes.

Este trigger es clave en ámbito de seguridad porque evita que un usuario realice modificaciones sobre Turnos que ya fueron cerrados, asegurando integridad en el historial.

```
1 | CREATE TRIGGER TR_No_Modificar_Turnos_Cerrados ON Turnos
2 | AFTER UPDATE
3 | AS
4 | BEGIN
5 |     -- Obtengo los Ids de los estados Cancelado o Completado
6 |     DECLARE @IdEstadoCancelado TINYINT = (SELECT IdEstado FROM Estados
WHERE Descripcion = 'Cancelado');
7 |     DECLARE @IdEstadoCompletado TINYINT = (SELECT IdEstado FROM Estados
WHERE Descripcion = 'Completado');
8 |
9 |     -- Si hay al menos un registro (> 0) donde el id de estado del
registro corresponda a cancelado o completado...
10|     IF (SELECT COUNT(*) FROM inserted I JOIN deleted D ON I.IdTurno =
D.IdTurno WHERE D.IdEstado IN (@IdEstadoCancelado, @IdEstadoCompletado)) >
0
11|     BEGIN
12|         -- Se muestra un error, y se hace rollback, cancelando la
actualizacion de los registros.
13|         RAISERROR('Los turnos cerrados no pueden ser modificados.', 16,
1);
14|         ROLLBACK TRANSACTION;
15|         RETURN;
16|     END
17|
18|     UPDATE T
19|     SET Observaciones = 'Turno cancelado' FROM Turnos T
20|     INNER JOIN inserted I ON T.IdTurno = I.IdTurno
21|     WHERE I.IdEstado = @IdEstadoCancelado
22|
23| END
24|
```

Links a los recursos

Repositorio con scripts de creación de BD, creación de objetos, carga de datos, y con el desarrollo web (en .ASPX) de presentación:

https://github.com/MarcosMazzitelli/TPI-DB-II-Grupo13

Video de demostración del sistema:

https://youtu.be/wIgvOO-d8x0

El código pegado en este documento fue generado con la herramienta https://www.syntax-highlighter.dev/