```
--
-- PostgreSQL database dump
--

-- Dumped from database version 17.5
-- Dumped by pg_dump version 17.5

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET transaction_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

--
-- Name: estado_enum; Type: TYPE; Schema: public; Owner: postgres
--

CREATE TYPE public.estado_enum AS ENUM (
    'verde',
    'amarillo',
    'rojo'
);


ALTER TYPE public.estado_enum OWNER TO postgres;

--
-- Name: tipo_sintoma_enum; Type: TYPE; Schema: public; Owner: postgres
--

CREATE TYPE public.tipo_sintoma_enum AS ENUM (
    'compuesto',
    'simple'
);


ALTER TYPE public.tipo_sintoma_enum OWNER TO postgres;

--
-- Name: tipo_unidad_enum; Type: TYPE; Schema: public; Owner: postgres
--

CREATE TYPE public.tipo_unidad_enum AS ENUM (
    'clínica',
    'hospital',
    'cesfam'
);


ALTER TYPE public.tipo_unidad_enum OWNER TO postgres;

--
```

```
-- Name: tipo_usuario_enum; Type: TYPE; Schema: public; Owner: postgres
--

CREATE TYPE public.tipo_usuario_enum AS ENUM (
    'administrador',
    'médico/funcionario',
    'paciente'
);


ALTER TYPE public.tipo_usuario_enum OWNER TO postgres;

--
-- Name: actualizar_registro_sintomas_diarios(); Type: FUNCTION;
Schema: public; Owner: postgres
--

CREATE FUNCTION public.actualizar_registro_sintomas_diarios() RETURNS
trigger
    LANGUAGE plpgsql
    AS $$
DECLARE
    paciente INT;
    fecha_reg DATE;
    puntaje_total INT;
    estado_semaforo estado_enum;
BEGIN
    -- Determinar paciente y fecha según tipo de operación
    paciente := COALESCE(NEW.paciente_id, OLD.paciente_id);
    fecha_reg := COALESCE(NEW.fecha, OLD.fecha);

    -- Calcular puntaje total para paciente y fecha
    SELECT COALESCE(SUM(puntuacion), 0)
    INTO puntaje_total
    FROM paciente_sintoma
    WHERE paciente_id = paciente
      AND fecha = fecha_reg;

    -- Asignar estado según el puntaje total (semáforo)
    IF puntaje_total >= 50 THEN
        estado_semaforo := 'rojo';
    ELSIF puntaje_total >= 20 THEN
        estado_semaforo := 'amarillo';
    ELSE
        estado_semaforo := 'verde';
    END IF;

    -- Validar existencia del paciente antes de insertar
    IF EXISTS (
        SELECT 1 FROM pacientes WHERE id = paciente
    ) THEN
        INSERT INTO registros_sintomas_diarios (paciente_id, fecha,
puntaje_total, estado)
        VALUES (paciente, fecha_reg, puntaje_total, estado_semaforo)
        ON CONFLICT (paciente_id, fecha) DO UPDATE
        SET puntaje_total = EXCLUDED.puntaje_total,
            estado = EXCLUDED.estado;
    END IF;
```

```
    RETURN NULL;
END;
$$;


ALTER FUNCTION public.actualizar_registro_sintomas_diarios() OWNER TO
postgres;

--
-- Name: disparar_boton_panico(integer); Type: PROCEDURE; Schema:
public; Owner: postgres
--

CREATE PROCEDURE public.disparar_boton_panico(IN p_paciente_id integer)
    LANGUAGE plpgsql
    AS $$
DECLARE
    v_medicos INT[];
    v_m_id INT;
BEGIN
    -- Validar existencia del paciente
    IF NOT EXISTS (
        SELECT 1 FROM pacientes WHERE id = p_paciente_id
    ) THEN
        RAISE EXCEPTION 'No existe el paciente con ID %',
p_paciente_id;
    END IF;

    -- Obtener hasta 3 médicos asignados al paciente
    SELECT ARRAY(
        SELECT medico_id
        FROM paciente_medico
        WHERE paciente_id = p_paciente_id
        LIMIT 3
    ) INTO v_medicos;

    -- Verificar que haya al menos un médico asignado
    IF array_length(v_medicos, 1) IS NULL THEN
        RAISE EXCEPTION 'El paciente % no tiene médicos asignados.',
p_paciente_id;
    END IF;

    -- Insertar un registro por cada médico notificado
    FOREACH v_m_id IN ARRAY v_medicos LOOP
        INSERT INTO boton_panico (paciente_id, profesional_notificado)
        VALUES (p_paciente_id, v_m_id);
    END LOOP;
END;
$$;


ALTER PROCEDURE public.disparar_boton_panico(IN p_paciente_id integer)
OWNER TO postgres;

--
-- Name: insertar_bitacora(integer, integer, text); Type: PROCEDURE;
Schema: public; Owner: postgres
```

```
--

CREATE PROCEDURE public.insertar_bitacora(IN p_paciente_id integer, IN
p_profesional_id integer, IN p_comentario text)
    LANGUAGE plpgsql
    AS $$
BEGIN
    -- Verificación de existencia del paciente
    IF NOT EXISTS (
        SELECT 1 FROM public.pacientes WHERE id = p_paciente_id
    ) THEN
        RAISE EXCEPTION 'No existe paciente con ID: %', p_paciente_id;
    END IF;

    -- Verificación de existencia del profesional (si se proporciona)
    IF p_profesional_id IS NOT NULL THEN
        IF NOT EXISTS (
            SELECT 1 FROM public.medico_funcionario WHERE id =
p_profesional_id
        ) THEN
            RAISE EXCEPTION 'No existe profesional con ID: %',
p_profesional_id;
        END IF;
    END IF;

    -- Validar comentario
    IF p_comentario IS NULL OR TRIM(p_comentario) = '' THEN
        RAISE EXCEPTION 'El comentario no puede estar vacío.';
    END IF;

    -- Inserción segura
    INSERT INTO public.bitacora (
        paciente_id, profesional_id, comentario
    )
    VALUES (
        p_paciente_id, p_profesional_id, p_comentario
    );
END;
$$;


ALTER PROCEDURE public.insertar_bitacora(IN p_paciente_id integer, IN
p_profesional_id integer, IN p_comentario text) OWNER TO postgres;

--
-- Name: insertar_enfermedad(character varying); Type: PROCEDURE;
Schema: public; Owner: postgres
--

CREATE PROCEDURE public.insertar_enfermedad(IN p_nombre character
varying)
    LANGUAGE plpgsql
    AS $$
BEGIN
    -- Validar que el nombre no esté vacío
    IF p_nombre IS NULL OR TRIM(p_nombre) = '' THEN
        RAISE EXCEPTION 'El nombre de la enfermedad no puede estar
vacío.';
```

```
    END IF;

    -- Verificar que no exista otra enfermedad con el mismo nombre
    IF EXISTS (
        SELECT 1 FROM public.enfermedad WHERE nombre = p_nombre
    ) THEN
        RAISE EXCEPTION 'Ya existe una enfermedad con el nombre: %',
p_nombre;
    END IF;

    -- Inserción segura
    INSERT INTO public.enfermedad (nombre)
    VALUES (p_nombre);
END;
$$;


ALTER PROCEDURE public.insertar_enfermedad(IN p_nombre character
varying) OWNER TO postgres;

--
-- Name: insertar_enfermedad_sintoma(integer, integer); Type:
PROCEDURE; Schema: public; Owner: postgres
--

CREATE PROCEDURE public.insertar_enfermedad_sintoma(IN p_id_enfermedad
integer, IN p_id_sintoma integer)
    LANGUAGE plpgsql
    AS $$
BEGIN
    -- Validar existencia de la enfermedad
    IF NOT EXISTS (
        SELECT 1 FROM public.enfermedad WHERE id = p_id_enfermedad
    ) THEN
        RAISE EXCEPTION 'No existe enfermedad con ID: %',
p_id_enfermedad;
    END IF;

    -- Validar existencia del síntoma
    IF NOT EXISTS (
        SELECT 1 FROM public.sintomas WHERE id = p_id_sintoma
    ) THEN
        RAISE EXCEPTION 'No existe síntoma con ID: %', p_id_sintoma;
    END IF;

    -- Verificar si ya existe la relación
    IF EXISTS (
        SELECT 1 FROM public.enfermedad_sintomas
        WHERE id_enfermedad = p_id_enfermedad AND id_sintoma =
p_id_sintoma
    ) THEN
        RAISE EXCEPTION 'La relación enfermedad-síntoma ya existe.';
    END IF;

    -- Inserción segura
    INSERT INTO public.enfermedad_sintomas (id_enfermedad, id_sintoma)
    VALUES (p_id_enfermedad, p_id_sintoma);
END;
```

```
$$;


ALTER PROCEDURE public.insertar_enfermedad_sintoma(IN p_id_enfermedad
integer, IN p_id_sintoma integer) OWNER TO postgres;

--
-- Name: insertar_indicacion(integer, text, character varying); Type:
PROCEDURE; Schema: public; Owner: postgres
--

CREATE PROCEDURE public.insertar_indicacion(IN p_paciente_id integer,
IN p_descripcion text, IN p_horarios character varying)
    LANGUAGE plpgsql
    AS $$
BEGIN
    -- Verificar que el paciente exista
    IF NOT EXISTS (
        SELECT 1 FROM public.pacientes WHERE id = p_paciente_id
    ) THEN
        RAISE EXCEPTION 'No existe un paciente con ID: %',
p_paciente_id;
    END IF;

    -- Verificar que la descripción no esté vacía
    IF p_descripcion IS NULL OR TRIM(p_descripcion) = '' THEN
        RAISE EXCEPTION 'La descripción de la indicación no puede estar
vacía.';
    END IF;

    -- Validación opcional de horarios
    IF p_horarios IS NOT NULL AND LENGTH(TRIM(p_horarios)) = 0 THEN
        RAISE EXCEPTION 'El campo de horarios no puede estar vacío.';
    END IF;

    -- Inserción segura
    INSERT INTO public.indicaciones (
        paciente_id, descripcion, horarios
    )
    VALUES (
        p_paciente_id, p_descripcion, p_horarios
    );
END;
$$;


ALTER PROCEDURE public.insertar_indicacion(IN p_paciente_id integer, IN
p_descripcion text, IN p_horarios character varying) OWNER TO postgres;

--
-- Name: insertar_medico_funcionario(character varying, character
varying, character varying, character varying, character varying,
character varying, integer); Type: PROCEDURE; Schema: public; Owner:
postgres
--

CREATE PROCEDURE public.insertar_medico_funcionario(IN p_rut_usuario
character varying, IN p_nombre_usuario character varying, IN
```

```
p_clave_usuario character varying, IN p_profesion character varying, IN
p_telefono character varying, IN p_email character varying, IN
p_unidad_referencia_id integer)
    LANGUAGE plpgsql
    AS $$
DECLARE
    v_usuario_id INT;
BEGIN
    -- Validación de teléfono
    IF p_telefono IS NULL OR TRIM(p_telefono) = '' THEN
        RAISE EXCEPTION 'El teléfono no puede estar vacío.';
    END IF;

    -- Validación de email
    IF p_email IS NULL OR TRIM(p_email) = '' THEN
        RAISE EXCEPTION 'El email no puede estar vacío.';
    END IF;

    -- Validación de unicidad de email
    IF EXISTS (
        SELECT 1 FROM public.medico_funcionario WHERE email = p_email
    ) THEN
        RAISE EXCEPTION 'El email ya está registrado: %', p_email;
    END IF;

    -- Validación de unidad de referencia
    IF p_unidad_referencia_id IS NOT NULL THEN
        IF NOT EXISTS (
            SELECT 1 FROM public.unidades_referencia WHERE id =
p_unidad_referencia_id
        ) THEN
            RAISE EXCEPTION 'No existe la unidad de referencia con ID:
%', p_unidad_referencia_id;
        END IF;
    END IF;

    -- Crear usuario automáticamente
    INSERT INTO public.usuarios (rut, nombre, clave, tipo_usuario)
    VALUES (p_rut_usuario, p_nombre_usuario, p_clave_usuario,
'médico/funcionario')
    RETURNING id INTO v_usuario_id;

    -- Inserción del médico con el ID del usuario creado
    INSERT INTO public.medico_funcionario (
        usuario_id, profesion, telefono, email, unidad_referencia_id
    )
    VALUES (
        v_usuario_id, p_profesion, p_telefono, p_email,
p_unidad_referencia_id
    );
END;
$$;


ALTER PROCEDURE public.insertar_medico_funcionario(IN p_rut_usuario
character varying, IN p_nombre_usuario character varying, IN
p_clave_usuario character varying, IN p_profesion character varying, IN
```

```
p_telefono character varying, IN p_email character varying, IN
p_unidad_referencia_id integer) OWNER TO postgres;

--
-- Name: insertar_paciente(character varying, character varying,
character varying, character varying, character varying, date, date,
integer); Type: PROCEDURE; Schema: public; Owner: postgres
--

CREATE PROCEDURE public.insertar_paciente(IN p_rut_usuario character
varying, IN p_nombre_usuario character varying, IN p_clave_usuario
character varying, IN p_direccion character varying, IN p_telefono
character varying, IN p_fecha_nacimiento date, IN p_fecha_diagnostico
date, IN p_id_enfermedad integer)
    LANGUAGE plpgsql
    AS $$
DECLARE
    v_usuario_id INT;
BEGIN
    -- Validar teléfono
    IF p_telefono IS NULL OR TRIM(p_telefono) = '' THEN
        RAISE EXCEPTION 'El teléfono no puede estar vacío.';
    END IF;

    -- Validar enfermedad
    IF NOT EXISTS (
        SELECT 1 FROM enfermedad WHERE id = p_id_enfermedad
    ) THEN
        RAISE EXCEPTION 'No existe una enfermedad con ID: %',
p_id_enfermedad;
    END IF;

    -- Crear usuario tipo paciente
    INSERT INTO public.usuarios (rut, nombre, clave, tipo_usuario)
    VALUES (p_rut_usuario, p_nombre_usuario, p_clave_usuario,
'paciente')
    RETURNING id INTO v_usuario_id;

    -- Insertar paciente con ID de usuario recién creado
    INSERT INTO public.pacientes (
        usuario_id, direccion, telefono, fecha_nacimiento,
fecha_diagnostico, id_enfermedad
    )
    VALUES (
        v_usuario_id, p_direccion, p_telefono, p_fecha_nacimiento,
p_fecha_diagnostico, p_id_enfermedad
    );
END;
$$;


ALTER PROCEDURE public.insertar_paciente(IN p_rut_usuario character
varying, IN p_nombre_usuario character varying, IN p_clave_usuario
character varying, IN p_direccion character varying, IN p_telefono
character varying, IN p_fecha_nacimiento date, IN p_fecha_diagnostico
date, IN p_id_enfermedad integer) OWNER TO postgres;

--
```

```
-- Name: insertar_paciente_medico(integer, integer); Type: PROCEDURE;
Schema: public; Owner: postgres
--

CREATE PROCEDURE public.insertar_paciente_medico(IN p_paciente_id
integer, IN p_medico_id integer)
    LANGUAGE plpgsql
    AS $$
BEGIN
    -- Validación de existencia del paciente
    IF NOT EXISTS (
        SELECT 1 FROM public.pacientes WHERE id = p_paciente_id
    ) THEN
        RAISE EXCEPTION 'No existe el paciente con ID: %',
p_paciente_id;
    END IF;

    -- Validación de existencia del médico
    IF NOT EXISTS (
        SELECT 1 FROM public.medico_funcionario WHERE id = p_medico_id
    ) THEN
        RAISE EXCEPTION 'No existe el médico con ID: %', p_medico_id;
    END IF;

    -- Inserción segura en la tabla puente
    INSERT INTO public.paciente_medico (paciente_id, medico_id)
    VALUES (p_paciente_id, p_medico_id);
END;
$$;


ALTER PROCEDURE public.insertar_paciente_medico(IN p_paciente_id
integer, IN p_medico_id integer) OWNER TO postgres;

--
-- Name: insertar_paciente_sintoma(integer, integer, numeric); Type:
PROCEDURE; Schema: public; Owner: postgres
--

CREATE PROCEDURE public.insertar_paciente_sintoma(IN p_paciente_id
integer, IN p_sintoma_id integer, IN p_valor_registrado numeric)
    LANGUAGE plpgsql
    AS $$
DECLARE
    s_tipo tipo_sintoma_enum;
    s_valor_referencia NUMERIC;
    s_puntuacion_min NUMERIC;
    s_puntuacion_max NUMERIC;
    puntuacion_final NUMERIC;
    valor_final NUMERIC;
BEGIN
    -- Obtener datos del síntoma
    SELECT tipo_sintoma, valor_referencia, puntuacion_min,
puntuacion_max
    INTO s_tipo, s_valor_referencia, s_puntuacion_min, s_puntuacion_max
    FROM Sintomas
    WHERE id = p_sintoma_id;
```

```
    IF s_tipo = 'compuesto' THEN
        IF p_valor_registrado > s_valor_referencia THEN
            puntuacion_final := s_puntuacion_max;
        ELSE
            puntuacion_final := s_puntuacion_min;
        END IF;
        valor_final := p_valor_registrado;
    ELSE -- caso sintoma simple
        puntuacion_final := s_puntuacion_max;
        valor_final := NULL; -- Sintomas de tipo simple no requieren de
un registro numérico por parte del paciente
    END IF;

    INSERT INTO paciente_sintoma (paciente_id, sintoma_id,
valor_registrado, puntuacion)
    VALUES (p_paciente_id, p_sintoma_id, valor_final, puntuacion_final)
      ON CONFLICT (paciente_id, sintoma_id, fecha)
    DO UPDATE SET
        valor_registrado = EXCLUDED.valor_registrado,
        puntuacion = EXCLUDED.puntuacion;
END;
$$;


ALTER PROCEDURE public.insertar_paciente_sintoma(IN p_paciente_id
integer, IN p_sintoma_id integer, IN p_valor_registrado numeric) OWNER
TO postgres;

--
-- Name: insertar_sintoma(character varying, public.tipo_sintoma_enum,
numeric, integer, integer); Type: PROCEDURE; Schema: public; Owner:
postgres
--

CREATE PROCEDURE public.insertar_sintoma(IN p_nombre character varying,
IN p_tipo public.tipo_sintoma_enum, IN p_valor_referencia numeric, IN
p_puntuacion_min integer, IN p_puntuacion_max integer)
    LANGUAGE plpgsql
    AS $$
BEGIN
    -- Validación de nombre
    IF p_nombre IS NULL OR TRIM(p_nombre) = '' THEN
        RAISE EXCEPTION 'El nombre del síntoma no puede estar vacío.';
    END IF;

    -- Validación del tipo
    IF p_tipo NOT IN ('simple', 'compuesto') THEN
        RAISE EXCEPTION 'Tipo de síntoma inválido: %', p_tipo;
    END IF;

    -- Validación de puntuaciones
    IF p_puntuacion_min IS NULL OR p_puntuacion_max IS NULL THEN
        RAISE EXCEPTION 'Las puntuaciones mínima y máxima son
obligatorias.';
    END IF;

    IF p_puntuacion_min >= p_puntuacion_max THEN
```

```
        RAISE EXCEPTION 'La puntuación mínima debe ser menor que la
máxima.';
    END IF;

    -- Validación específica según tipo
    IF p_tipo = 'compuesto' THEN
        IF p_valor_referencia IS NULL THEN
            RAISE EXCEPTION 'Los síntomas compuestos requieren un valor
de referencia.';
        END IF;
    ELSE -- tipo simple
        IF p_valor_referencia IS NULL OR p_valor_referencia <> 0 THEN
            RAISE EXCEPTION 'Los síntomas simples deben tener valor de
referencia igual a 0.';
        END IF;

        IF p_puntuacion_min <> 0 THEN
            RAISE EXCEPTION 'Los síntomas simples deben tener
puntuación mínima igual a 0.';
        END IF;
    END IF;

    -- Inserción segura
    INSERT INTO public.sintomas (
        nombre, tipo_sintoma, valor_referencia,
        puntuacion_min, puntuacion_max
    )
    VALUES (
        p_nombre, p_tipo, p_valor_referencia,
        p_puntuacion_min, p_puntuacion_max
    );
END;
$$;


ALTER PROCEDURE public.insertar_sintoma(IN p_nombre character varying,
IN p_tipo public.tipo_sintoma_enum, IN p_valor_referencia numeric, IN
p_puntuacion_min integer, IN p_puntuacion_max integer) OWNER TO
postgres;

--
-- Name: insertar_unidad_referencia(character varying, text,
public.tipo_unidad_enum, character varying); Type: PROCEDURE; Schema:
public; Owner: postgres
--

CREATE PROCEDURE public.insertar_unidad_referencia(IN p_nombre
character varying, IN p_direccion text, IN p_tipo
public.tipo_unidad_enum, IN p_telefono character varying)
    LANGUAGE plpgsql
    AS $$
BEGIN
    -- Validación de nombre
    IF p_nombre IS NULL OR TRIM(p_nombre) = '' THEN
        RAISE EXCEPTION 'El nombre de la unidad no puede estar vacío.';
    END IF;

    -- Validación de dirección
```

```
    IF p_direccion IS NULL OR TRIM(p_direccion) = '' THEN
        RAISE EXCEPTION 'La dirección no puede estar vacía.';
    END IF;

    -- Validación de teléfono
    IF p_telefono IS NULL OR TRIM(p_telefono) = '' THEN
        RAISE EXCEPTION 'El teléfono no puede estar vacío.';
    END IF;

    -- Validación opcional del tipo
    IF p_tipo NOT IN ('clínica','hospital','cesfam') THEN
        RAISE EXCEPTION 'Tipo de unidad no válido: %', p_tipo;
    END IF;

    -- Inserción
    INSERT INTO public.unidades_referencia (
        nombre, direccion, tipo, telefono
    )
    VALUES (
        p_nombre, p_direccion, p_tipo, p_telefono
    );
END;
$$;


ALTER PROCEDURE public.insertar_unidad_referencia(IN p_nombre character
varying, IN p_direccion text, IN p_tipo public.tipo_unidad_enum, IN
p_telefono character varying) OWNER TO postgres;

--
-- Name: insertar_usuario(character varying, character varying,
character varying, public.tipo_usuario_enum); Type: PROCEDURE; Schema:
public; Owner: postgres
--

CREATE PROCEDURE public.insertar_usuario(IN p_rut character varying, IN
p_nombre character varying, IN p_clave character varying, IN
p_tipo_usuario public.tipo_usuario_enum)
    LANGUAGE plpgsql
    AS $$
BEGIN
    -- Validaciones básicas
    IF p_rut IS NULL OR TRIM(p_rut) = '' THEN
        RAISE EXCEPTION 'El RUT no puede estar vacío.';
    END IF;

    IF p_nombre IS NULL OR TRIM(p_nombre) = '' THEN
        RAISE EXCEPTION 'El nombre no puede estar vacío.';
    END IF;

    IF p_clave IS NULL OR TRIM(p_clave) = '' THEN
        RAISE EXCEPTION 'La clave no puede estar vacía.';
    END IF;

    -- Validación explícita del tipo ENUM (opcional pero útil)
    IF p_tipo_usuario NOT IN ('administrador', 'médico/funcionario',
'paciente') THEN
        RAISE EXCEPTION 'Tipo de usuario inválido: %', p_tipo_usuario;
```

```
    END IF;

    -- Inserción segura
    INSERT INTO public.usuarios (rut, nombre, clave, tipo_usuario)
    VALUES (p_rut, p_nombre, p_clave, p_tipo_usuario);
END;
$$;


ALTER PROCEDURE public.insertar_usuario(IN p_rut character varying, IN
p_nombre character varying, IN p_clave character varying, IN
p_tipo_usuario public.tipo_usuario_enum) OWNER TO postgres;

--
-- Name: limitar_medicos_por_paciente(); Type: FUNCTION; Schema:
public; Owner: postgres
--

CREATE FUNCTION public.limitar_medicos_por_paciente() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
    IF (
        SELECT COUNT(*) FROM paciente_medico
        WHERE paciente_id = NEW.paciente_id
    ) >= 3 THEN
        RAISE EXCEPTION 'Un paciente no puede tener más de 3 médicos
asignados';
    END IF;
    RETURN NEW;
END;
$$;


ALTER FUNCTION public.limitar_medicos_por_paciente() OWNER TO postgres;

--
-- Name: limitar_pacientes_por_medico(); Type: FUNCTION; Schema:
public; Owner: postgres
--

CREATE FUNCTION public.limitar_pacientes_por_medico() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
    IF (
        SELECT COUNT(*) FROM paciente_medico
        WHERE medico_id = NEW.medico_id
    ) >= 10 THEN
        RAISE EXCEPTION 'Un médico no puede tener más de 10 pacientes
asignados';
    END IF;
    RETURN NEW;
END;
$$;


ALTER FUNCTION public.limitar_pacientes_por_medico() OWNER TO postgres;
```

```
--
-- Name: notificar_alerta(); Type: FUNCTION; Schema: public; Owner:
postgres
--

CREATE FUNCTION public.notificar_alerta() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
DECLARE
    v_medico_id INT;
    v_fecha_actual DATE := NEW.fecha::date;
BEGIN
    -- Verificar si el estado cambió a 'rojo'
    IF NEW.estado = 'rojo'::estado_enum AND (OLD.estado IS DISTINCT
FROM 'rojo'::estado_enum OR OLD.estado IS NULL) THEN

        -- Recorrer hasta 3 médicos asignados al paciente
        FOR v_medico_id IN
            SELECT pm.medico_id
            FROM paciente_medico pm
            WHERE pm.paciente_id = NEW.paciente_id
            LIMIT 3
        LOOP
            -- Verificar si ya existe una alerta para este paciente en
ese día
            IF EXISTS (
                SELECT 1 FROM alertas
                WHERE paciente_id = NEW.paciente_id
                  AND profesional_notificado = v_medico_id
                  AND fecha::date = v_fecha_actual
            ) THEN
                -- Actualizar alerta existente
                UPDATE alertas
                SET
                    motivo = 'Estado rojo actualizado',
                    registro_sintomas_id = NEW.id,
                    fecha = NEW.fecha  -- puedes mantener la original
si lo prefieres
                WHERE paciente_id = NEW.paciente_id
                  AND profesional_notificado = v_medico_id
                  AND fecha::date = v_fecha_actual;
                RAISE NOTICE 'Alerta actualizada para paciente % y
médico %', NEW.paciente_id, v_medico_id;
            ELSE
                -- Insertar nueva alerta
                INSERT INTO alertas (
                    paciente_id, fecha, motivo, registro_sintomas_id,
profesional_notificado
                ) VALUES (
                    NEW.paciente_id, NEW.fecha, 'Estado rojo
detectado', NEW.id, v_medico_id
                );
                RAISE NOTICE 'Alerta insertada para paciente % y médico
%', NEW.paciente_id, v_medico_id;
            END IF;
        END LOOP;
```

```
        -- Si no hay médicos asignados
        IF NOT FOUND THEN
            RAISE NOTICE 'No se encontró médico asignado para el
paciente %', NEW.paciente_id;
        END IF;
    END IF;

    RETURN NEW;
END;
$$;


ALTER FUNCTION public.notificar_alerta() OWNER TO postgres;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- Name: alertas; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.alertas (
    id integer NOT NULL,
    paciente_id integer,
    fecha date NOT NULL,
    motivo text NOT NULL,
    registro_sintomas_id integer,
    profesional_notificado integer
);


ALTER TABLE public.alertas OWNER TO postgres;

--
-- Name: alertas_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.alertas_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.alertas_id_seq OWNER TO postgres;

--
-- Name: alertas_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.alertas_id_seq OWNED BY public.alertas.id;
```

```
--
-- Name: bitacora; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.bitacora (
    id integer NOT NULL,
    paciente_id integer NOT NULL,
    fecha timestamp without time zone DEFAULT now() NOT NULL,
    profesional_id integer,
    comentario text NOT NULL
);


ALTER TABLE public.bitacora OWNER TO postgres;

--
-- Name: bitacora_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.bitacora_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.bitacora_id_seq OWNER TO postgres;

--
-- Name: bitacora_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.bitacora_id_seq OWNED BY public.bitacora.id;


--
-- Name: boton_panico; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.boton_panico (
    id integer NOT NULL,
    paciente_id integer NOT NULL,
    fecha timestamp without time zone DEFAULT now() NOT NULL,
    profesional_notificado integer NOT NULL
);


ALTER TABLE public.boton_panico OWNER TO postgres;

--
-- Name: boton_panico_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--
```

```
CREATE SEQUENCE public.boton_panico_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.boton_panico_id_seq OWNER TO postgres;

--
-- Name: boton_panico_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.boton_panico_id_seq OWNED BY
public.boton_panico.id;


--
-- Name: enfermedad; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.enfermedad (
    id integer NOT NULL,
    nombre character varying(100) NOT NULL
);


ALTER TABLE public.enfermedad OWNER TO postgres;

--
-- Name: enfermedad_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.enfermedad_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.enfermedad_id_seq OWNER TO postgres;

--
-- Name: enfermedad_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.enfermedad_id_seq OWNED BY public.enfermedad.id;


--
```

```
-- Name: enfermedad_sintomas; Type: TABLE; Schema: public; Owner:
postgres
--

CREATE TABLE public.enfermedad_sintomas (
    id_enfermedad integer NOT NULL,
    id_sintoma integer NOT NULL
);


ALTER TABLE public.enfermedad_sintomas OWNER TO postgres;

--
-- Name: indicaciones; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.indicaciones (
    id integer NOT NULL,
    paciente_id integer NOT NULL,
    descripcion text NOT NULL,
    horarios character varying(150)
);


ALTER TABLE public.indicaciones OWNER TO postgres;

--
-- Name: indicaciones_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.indicaciones_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.indicaciones_id_seq OWNER TO postgres;

--
-- Name: indicaciones_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.indicaciones_id_seq OWNED BY
public.indicaciones.id;


--
-- Name: medico_funcionario; Type: TABLE; Schema: public; Owner:
postgres
--

CREATE TABLE public.medico_funcionario (
    id integer NOT NULL,
```

```
    usuario_id integer NOT NULL,
    profesion character varying(50) NOT NULL,
    telefono character varying(20) NOT NULL,
    email character varying(100) NOT NULL,
    unidad_referencia_id integer NOT NULL
);


ALTER TABLE public.medico_funcionario OWNER TO postgres;

--
-- Name: medico_funcionario_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.medico_funcionario_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.medico_funcionario_id_seq OWNER TO postgres;

--
-- Name: medico_funcionario_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.medico_funcionario_id_seq OWNED BY
public.medico_funcionario.id;


--
-- Name: paciente_medico; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.paciente_medico (
    paciente_id integer NOT NULL,
    medico_id integer NOT NULL
);


ALTER TABLE public.paciente_medico OWNER TO postgres;

--
-- Name: paciente_sintoma; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.paciente_sintoma (
    paciente_id integer NOT NULL,
    sintoma_id integer NOT NULL,
    fecha date DEFAULT CURRENT_DATE NOT NULL,
    valor_registrado numeric,
    puntuacion numeric NOT NULL
);
```

```
ALTER TABLE public.paciente_sintoma OWNER TO postgres;

--
-- Name: pacientes; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.pacientes (
    id integer NOT NULL,
    usuario_id integer NOT NULL,
    direccion text NOT NULL,
    telefono character varying(20) NOT NULL,
    fecha_nacimiento date NOT NULL,
    fecha_diagnostico date NOT NULL,
    id_enfermedad integer NOT NULL
);


ALTER TABLE public.pacientes OWNER TO postgres;

--
-- Name: pacientes_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.pacientes_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.pacientes_id_seq OWNER TO postgres;

--
-- Name: pacientes_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.pacientes_id_seq OWNED BY public.pacientes.id;


--
-- Name: registros_sintomas_diarios; Type: TABLE; Schema: public;
Owner: postgres
--

CREATE TABLE public.registros_sintomas_diarios (
    id integer NOT NULL,
    paciente_id integer NOT NULL,
    fecha date NOT NULL,
    puntaje_total integer NOT NULL,
    estado public.estado_enum DEFAULT 'verde'::public.estado_enum NOT
NULL
);
```

```
ALTER TABLE public.registros_sintomas_diarios OWNER TO postgres;

--
-- Name: registros_sintomas_diarios_id_seq; Type: SEQUENCE; Schema:
public; Owner: postgres
--

CREATE SEQUENCE public.registros_sintomas_diarios_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.registros_sintomas_diarios_id_seq OWNER TO
postgres;

--
-- Name: registros_sintomas_diarios_id_seq; Type: SEQUENCE OWNED BY;
Schema: public; Owner: postgres
--

ALTER SEQUENCE public.registros_sintomas_diarios_id_seq OWNED BY
public.registros_sintomas_diarios.id;


--
-- Name: sintomas; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.sintomas (
    id integer NOT NULL,
    nombre character varying(100) NOT NULL,
    tipo_sintoma public.tipo_sintoma_enum NOT NULL,
    valor_referencia numeric,
    puntuacion_min numeric NOT NULL,
    puntuacion_max numeric NOT NULL
);


ALTER TABLE public.sintomas OWNER TO postgres;

--
-- Name: sintomas_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.sintomas_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER SEQUENCE public.sintomas_id_seq OWNER TO postgres;

--
-- Name: sintomas_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.sintomas_id_seq OWNED BY public.sintomas.id;


--
-- Name: unidades_referencia; Type: TABLE; Schema: public; Owner:
postgres
--

CREATE TABLE public.unidades_referencia (
    id integer NOT NULL,
    nombre character varying(100) NOT NULL,
    direccion text NOT NULL,
    tipo public.tipo_unidad_enum NOT NULL,
    telefono character varying(20) NOT NULL
);


ALTER TABLE public.unidades_referencia OWNER TO postgres;

--
-- Name: unidades_referencia_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.unidades_referencia_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.unidades_referencia_id_seq OWNER TO postgres;

--
-- Name: unidades_referencia_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.unidades_referencia_id_seq OWNED BY
public.unidades_referencia.id;


--
-- Name: usuarios; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.usuarios (
```

```
    id integer NOT NULL,
    rut character varying(12) NOT NULL,
    nombre character varying(100) NOT NULL,
    clave character varying(255) NOT NULL,
    tipo_usuario public.tipo_usuario_enum NOT NULL
);


ALTER TABLE public.usuarios OWNER TO postgres;

--
-- Name: usuarios_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.usuarios_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.usuarios_id_seq OWNER TO postgres;

--
-- Name: usuarios_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--

ALTER SEQUENCE public.usuarios_id_seq OWNED BY public.usuarios.id;


--
-- Name: alertas id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.alertas ALTER COLUMN id SET DEFAULT
nextval('public.alertas_id_seq'::regclass);


--
-- Name: bitacora id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.bitacora ALTER COLUMN id SET DEFAULT
nextval('public.bitacora_id_seq'::regclass);


--
-- Name: boton_panico id; Type: DEFAULT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.boton_panico ALTER COLUMN id SET DEFAULT
nextval('public.boton_panico_id_seq'::regclass);
```

```
--
-- Name: enfermedad id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.enfermedad ALTER COLUMN id SET DEFAULT
nextval('public.enfermedad_id_seq'::regclass);


--
-- Name: indicaciones id; Type: DEFAULT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.indicaciones ALTER COLUMN id SET DEFAULT
nextval('public.indicaciones_id_seq'::regclass);


--
-- Name: medico_funcionario id; Type: DEFAULT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.medico_funcionario ALTER COLUMN id SET DEFAULT
nextval('public.medico_funcionario_id_seq'::regclass);


--
-- Name: pacientes id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.pacientes ALTER COLUMN id SET DEFAULT
nextval('public.pacientes_id_seq'::regclass);


--
-- Name: registros_sintomas_diarios id; Type: DEFAULT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.registros_sintomas_diarios ALTER COLUMN id SET
DEFAULT nextval('public.registros_sintomas_diarios_id_seq'::regclass);


--
-- Name: sintomas id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.sintomas ALTER COLUMN id SET DEFAULT
nextval('public.sintomas_id_seq'::regclass);


--
-- Name: unidades_referencia id; Type: DEFAULT; Schema: public; Owner:
postgres
--
```

```
ALTER TABLE ONLY public.unidades_referencia ALTER COLUMN id SET DEFAULT
nextval('public.unidades_referencia_id_seq'::regclass);


--
-- Name: usuarios id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.usuarios ALTER COLUMN id SET DEFAULT
nextval('public.usuarios_id_seq'::regclass);


--
-- Name: alertas alertas_pkey; Type: CONSTRAINT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.alertas
    ADD CONSTRAINT alertas_pkey PRIMARY KEY (id);


--
-- Name: bitacora bitacora_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.bitacora
    ADD CONSTRAINT bitacora_pkey PRIMARY KEY (id);


--
-- Name: boton_panico boton_panico_pkey; Type: CONSTRAINT; Schema:
public; Owner: postgres
--

ALTER TABLE ONLY public.boton_panico
    ADD CONSTRAINT boton_panico_pkey PRIMARY KEY (id);


--
-- Name: enfermedad enfermedad_nombre_key; Type: CONSTRAINT; Schema:
public; Owner: postgres
--

ALTER TABLE ONLY public.enfermedad
    ADD CONSTRAINT enfermedad_nombre_key UNIQUE (nombre);


--
-- Name: enfermedad enfermedad_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.enfermedad
    ADD CONSTRAINT enfermedad_pkey PRIMARY KEY (id);


--
```

```
-- Name: enfermedad_sintomas enfermedad_sintomas_pkey; Type:
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.enfermedad_sintomas
    ADD CONSTRAINT enfermedad_sintomas_pkey PRIMARY KEY (id_enfermedad,
id_sintoma);


--
-- Name: indicaciones indicaciones_pkey; Type: CONSTRAINT; Schema:
public; Owner: postgres
--

ALTER TABLE ONLY public.indicaciones
    ADD CONSTRAINT indicaciones_pkey PRIMARY KEY (id);


--
-- Name: medico_funcionario medico_funcionario_email_key; Type:
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.medico_funcionario
    ADD CONSTRAINT medico_funcionario_email_key UNIQUE (email);


--
-- Name: medico_funcionario medico_funcionario_pkey; Type: CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.medico_funcionario
    ADD CONSTRAINT medico_funcionario_pkey PRIMARY KEY (id);


--
-- Name: paciente_medico paciente_medico_pkey; Type: CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.paciente_medico
    ADD CONSTRAINT paciente_medico_pkey PRIMARY KEY (paciente_id,
medico_id);


--
-- Name: paciente_sintoma paciente_sintoma_pkey; Type: CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.paciente_sintoma
    ADD CONSTRAINT paciente_sintoma_pkey PRIMARY KEY (paciente_id,
sintoma_id, fecha);


--
```

```
-- Name: pacientes pacientes_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.pacientes
    ADD CONSTRAINT pacientes_pkey PRIMARY KEY (id);


--
-- Name: registros_sintomas_diarios
registros_sintomas_diarios_paciente_id_fecha_key; Type: CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.registros_sintomas_diarios
    ADD CONSTRAINT registros_sintomas_diarios_paciente_id_fecha_key
UNIQUE (paciente_id, fecha);


--
-- Name: registros_sintomas_diarios registros_sintomas_diarios_pkey;
Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.registros_sintomas_diarios
    ADD CONSTRAINT registros_sintomas_diarios_pkey PRIMARY KEY (id);


--
-- Name: sintomas sintomas_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.sintomas
    ADD CONSTRAINT sintomas_pkey PRIMARY KEY (id);


--
-- Name: unidades_referencia unidades_referencia_pkey; Type:
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.unidades_referencia
    ADD CONSTRAINT unidades_referencia_pkey PRIMARY KEY (id);


--
-- Name: usuarios usuarios_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.usuarios
    ADD CONSTRAINT usuarios_pkey PRIMARY KEY (id);


--
-- Name: usuarios usuarios_rut_key; Type: CONSTRAINT; Schema: public;
Owner: postgres
```

```
--

ALTER TABLE ONLY public.usuarios
    ADD CONSTRAINT usuarios_rut_key UNIQUE (rut);


--
-- Name: paciente_sintoma actualizar_registro_sintomas_diarios; Type:
TRIGGER; Schema: public; Owner: postgres
--

CREATE TRIGGER actualizar_registro_sintomas_diarios AFTER INSERT OR
DELETE OR UPDATE ON public.paciente_sintoma FOR EACH ROW EXECUTE
FUNCTION public.actualizar_registro_sintomas_diarios();


--
-- Name: registros_sintomas_diarios notificar_alerta; Type: TRIGGER;
Schema: public; Owner: postgres
--

CREATE TRIGGER notificar_alerta AFTER UPDATE OF estado ON
public.registros_sintomas_diarios FOR EACH ROW EXECUTE FUNCTION
public.notificar_alerta();


--
-- Name: paciente_medico trigger_limitar_medicos_por_paciente; Type:
TRIGGER; Schema: public; Owner: postgres
--

CREATE TRIGGER trigger_limitar_medicos_por_paciente BEFORE INSERT ON
public.paciente_medico FOR EACH ROW EXECUTE FUNCTION
public.limitar_medicos_por_paciente();


--
-- Name: paciente_medico trigger_limitar_pacientes_por_medico; Type:
TRIGGER; Schema: public; Owner: postgres
--

CREATE TRIGGER trigger_limitar_pacientes_por_medico BEFORE INSERT ON
public.paciente_medico FOR EACH ROW EXECUTE FUNCTION
public.limitar_pacientes_por_medico();


--
-- Name: alertas alertas_paciente_id_fkey; Type: FK CONSTRAINT; Schema:
public; Owner: postgres
--

ALTER TABLE ONLY public.alertas
    ADD CONSTRAINT alertas_paciente_id_fkey FOREIGN KEY (paciente_id)
REFERENCES public.pacientes(id) ON DELETE CASCADE;


--
```

```
-- Name: alertas alertas_profesional_notificado_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.alertas
    ADD CONSTRAINT alertas_profesional_notificado_fkey FOREIGN KEY
(profesional_notificado) REFERENCES public.medico_funcionario(id) ON
DELETE CASCADE;


--
-- Name: alertas alertas_registro_sintomas_id_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.alertas
    ADD CONSTRAINT alertas_registro_sintomas_id_fkey FOREIGN KEY
(registro_sintomas_id) REFERENCES
public.registros_sintomas_diarios(id);


--
-- Name: bitacora bitacora_paciente_id_fkey; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.bitacora
    ADD CONSTRAINT bitacora_paciente_id_fkey FOREIGN KEY (paciente_id)
REFERENCES public.pacientes(id) ON DELETE CASCADE;


--
-- Name: bitacora bitacora_profesional_id_fkey; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.bitacora
    ADD CONSTRAINT bitacora_profesional_id_fkey FOREIGN KEY
(profesional_id) REFERENCES public.medico_funcionario(id) ON DELETE
CASCADE;


--
-- Name: boton_panico boton_panico_paciente_id_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.boton_panico
    ADD CONSTRAINT boton_panico_paciente_id_fkey FOREIGN KEY
(paciente_id) REFERENCES public.pacientes(id) ON DELETE CASCADE;


--
-- Name: boton_panico boton_panico_profesional_notificado_fkey; Type:
FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.boton_panico
```

```
    ADD CONSTRAINT boton_panico_profesional_notificado_fkey FOREIGN KEY
(profesional_notificado) REFERENCES public.medico_funcionario(id);


--
-- Name: enfermedad_sintomas enfermedad_sintomas_id_enfermedad_fkey;
Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.enfermedad_sintomas
    ADD CONSTRAINT enfermedad_sintomas_id_enfermedad_fkey FOREIGN KEY
(id_enfermedad) REFERENCES public.enfermedad(id);


--
-- Name: enfermedad_sintomas enfermedad_sintomas_id_sintoma_fkey; Type:
FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.enfermedad_sintomas
    ADD CONSTRAINT enfermedad_sintomas_id_sintoma_fkey FOREIGN KEY
(id_sintoma) REFERENCES public.sintomas(id);


--
-- Name: pacientes fk_enfermedad; Type: FK CONSTRAINT; Schema: public;
Owner: postgres
--

ALTER TABLE ONLY public.pacientes
    ADD CONSTRAINT fk_enfermedad FOREIGN KEY (id_enfermedad) REFERENCES
public.enfermedad(id);


--
-- Name: indicaciones indicaciones_paciente_id_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.indicaciones
    ADD CONSTRAINT indicaciones_paciente_id_fkey FOREIGN KEY
(paciente_id) REFERENCES public.pacientes(id) ON DELETE CASCADE;


--
-- Name: medico_funcionario
medico_funcionario_unidad_referencia_id_fkey; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.medico_funcionario
    ADD CONSTRAINT medico_funcionario_unidad_referencia_id_fkey FOREIGN
KEY (unidad_referencia_id) REFERENCES public.unidades_referencia(id);


--
-- Name: medico_funcionario medico_funcionario_usuario_id_fkey; Type:
FK CONSTRAINT; Schema: public; Owner: postgres
```

```
--

ALTER TABLE ONLY public.medico_funcionario
    ADD CONSTRAINT medico_funcionario_usuario_id_fkey FOREIGN KEY
(usuario_id) REFERENCES public.usuarios(id) ON DELETE CASCADE;


--
-- Name: paciente_medico paciente_medico_medico_id_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.paciente_medico
    ADD CONSTRAINT paciente_medico_medico_id_fkey FOREIGN KEY
(medico_id) REFERENCES public.medico_funcionario(id) ON DELETE CASCADE;


--
-- Name: paciente_medico paciente_medico_paciente_id_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.paciente_medico
    ADD CONSTRAINT paciente_medico_paciente_id_fkey FOREIGN KEY
(paciente_id) REFERENCES public.pacientes(id) ON DELETE CASCADE;


--
-- Name: paciente_sintoma paciente_sintoma_paciente_id_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.paciente_sintoma
    ADD CONSTRAINT paciente_sintoma_paciente_id_fkey FOREIGN KEY
(paciente_id) REFERENCES public.pacientes(id) ON DELETE CASCADE;


--
-- Name: paciente_sintoma paciente_sintoma_sintoma_id_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.paciente_sintoma
    ADD CONSTRAINT paciente_sintoma_sintoma_id_fkey FOREIGN KEY
(sintoma_id) REFERENCES public.sintomas(id);


--
-- Name: pacientes pacientes_usuario_id_fkey; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.pacientes
    ADD CONSTRAINT pacientes_usuario_id_fkey FOREIGN KEY (usuario_id)
REFERENCES public.usuarios(id) ON DELETE CASCADE;


--
```

```
-- Name: registros_sintomas_diarios
registros_sintomas_diarios_paciente_id_fkey; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.registros_sintomas_diarios
    ADD CONSTRAINT registros_sintomas_diarios_paciente_id_fkey FOREIGN
KEY (paciente_id) REFERENCES public.pacientes(id) ON DELETE CASCADE;


--
-- PostgreSQL database dump complete
--
```