Documentación Base de Datos

GuardiApp

Índice:

Documentación de la Base de Datos

La base de datos utilizada en la aplicación gestiona la asignación de horarios y guardias de los profesores. A continuación, se detalla la estructura de las tablas y las relaciones entre ellas, incluyendo las claves foráneas (Foreign Keys) que aseguran la integridad de los datos.

1. Tabla horario

Descripción:

La tabla horario se utiliza para almacenar la información referente a los horarios de los profesores. Cada fila representa una clase que un profesor tiene asignada en un día específico de la semana, junto con la asignatura que imparte.

Relaciones y Claves Foráneas:

profesor → users(id):

La columna profesor de la tabla horario es una clave foránea que se refiere a la columna id de la tabla users. Esto asegura que solo los usuarios registrados como profesores puedan ser asignados a los horarios.

clase → aula(id):

Este campo, de tipo bigint, hace referencia al campo id en la tabla aulas. Es decir, cada clase en la tabla horario está asociada con un aula en la tabla aulas.

Estructura:

Campo	Tipo de dato	Descripción
id	serial	Primary Key, Identificador único del horario. Es generado automáticamente por la base de datos.
dia	text	Día de la semana en el que el profesor tiene clase (por ejemplo: "Lunes", "Martes", etc.).
hora	text	Franja horaria de la clase (por ejemplo: "Primera Hora (08:30 - 09:25)").
profesor	int	Identificador del profesor asignado a este horario. Clave foránea que hace referencia a la columna id de la tabla users. Asegura que solo un profesor válido pueda ser asignado a un horario.
clase	int	Número de la clase asignada (por ejemplo, A-03-D).
asignatura	text	Nombre de la asignatura que el profesor imparte en ese horario (por ejemplo: "PSP", "SGE", etc.).
darclase	boolean	Indica si el profesor está asignado para dar clase en ese horario. Si es TRUE, el profesor está disponible para dar clases, si es FALSE, saldrá como inhabilitado

2. Tabla users

Descripción:

La tabla users almacena la información de todos los usuarios en el sistema, incluidos los profesores y administradores. Cada usuario tiene un rol asignado que define sus permisos en el sistema.

Estructura:

Campo	Tipo de dato	Descripción
id	serial	Identificador único del usuario. Es generado automáticamente por la base de datos.
nombre	text	Nombre de usuario
email	text	Correo del usuario
num_guardias	int	el número de guardias asignadas
rol	text	Dos posibilidades, Admin o Profesor
id_tabla	int	int con foreign key a la tabla auth.users y coge un número que se usa para la identificación y se usa para los triggers
activo	boolean	Si está TRUE al profesor se le podrán poner guardias, si está FALSE,
imagenperfil	text	Dirección en la que está una imagen para que sea la imagen de perfil del usuario

3. Tabla guardias

Descripción:

La tabla guardias almacena las guardias asignadas a los profesores. Cada fila de esta tabla representa una guardia que se asigna a un profesor en una hora y día específicos.

Relaciones y Claves Foráneas:

id_profesor → users(id):

id_profesor: La columna id_profesor de la tabla guardias está vinculada a la columna id de la tabla users. Esto garantiza que solo los usuarios registrados como profesores puedan ser asignados a guardias.

Estructura:

Campo	Tipo de dato	Descripción
id	int8	Id que representa a la guardia
id_profesor	int8	Id que representa al profesor de la guardia sobre la tabla users
dia	text	Día de la semana de la guardia
clase	int8	Id del aula en el que se hace la guardia sobre la tabla aula
hora	text	Hora a la que se realiza la guardia
tarea	text	Tarea dejada por un profesor para una guardia, puede dejarse a NULL
activa	bool	Dice si la guardia está asignada (TRUE) o no (FALSE).

4. Funciones y Triggers

Función generar_horario_vesper(profesor_id INT) y generar_horario_diurno(profesor_id INT) :

Esta función genera un horario vespertino aleatorio para un profesor específico. Se seleccionan 5 horas aleatorias por día para asignar una clase de una asignatura determinada, sin sobrescribir datos previos.

```
CREATE OR REPLACE FUNCTION generar_horario_diurno(profesor_id INT)
RETURNS VOID AS $$
DECLARE
    dias TEXT[] := ARRAY['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes'];
    horas TEXT[] := ARRAY['Primera Hora (08:30 - 9:25)', 'Segunda Hora (09:25 -
                          10:20)','Tercera Hora (10:20 - 11:15)', 'Cuarta Hora
(11:35 - 12:30)','Quinta Hora (12:30 - 13:25)', 'Sexta Hora (13:25 - 14:20)'];
    asignaturas TEXT[] := ARRAY['Matemáticas', 'PSP', 'Historia', 'Inglés',
'Ciencias','Filosofía', 'SGE', 'Móviles', 'Acceso a datos','Desarrollo de
interfaces', 'Empresa'];
    i INT;
    clase_aleatoria INT;
    horas_seleccionadas TEXT[];
    hora_actual TEXT;
BEGIN
    FOR i IN 1..array_length(dias, 1) LOOP -- Iterar sobre los días
        -- Seleccionar 5 horas aleatorias de las 6 disponibles
        horas_seleccionadas := ARRAY(
            SELECT horas[h]
            FROM generate_series(1, array_length(horas, 1)) AS h
            ORDER BY random()
            LIMIT 5
        );
-- Ordenar las horas seleccionadas según su posición en el array original
        horas_seleccionadas := ARRAY(
            SELECT h
            FROM unnest(horas_seleccionadas) AS h
            ORDER BY array_position(horas, h)
        );
        -- Insertar las horas seleccionadas en la tabla horario
        FOREACH hora_actual IN ARRAY horas_seleccionadas LOOP
            clase_aleatoria := FLOOR(RANDOM() * 49 + 1);
            INSERT INTO horario (dia, hora, profesor, clase, asignatura,
darClase)
            VALUES (dias[i], hora_actual, profesor_id, clase_aleatoria,
                    asignaturas[(random() * array_length(asignaturas, 1) +
1)::INT], TRUE);
        END LOOP;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Función sustituir_asignatura_nula():

Objetivo: Evitar valores NULL en la columna asignatura.

Si el valor de asignatura es NULL, se sustituye por 'Tutoría', se ejecuta mediante un trigger antes de insertar o actualizar.

```
CREATE OR REPLACE FUNCTION sustituir_asignatura_nula()
RETURNS TRIGGER AS $$
BEGIN
    -- Verificar si la columna 'asignatura' es NULL
    IF NEW.asignatura IS NULL THEN
        NEW.asignatura := 'Tutoría'; -- Sustituir NULL por 'Tutoría'
    END IF;

    RETURN NEW; -- Devolver la fila modificada
END;
$$ LANGUAGE plpgsql;
Trigger:
CREATE TRIGGER trigger_sustituir_asignatura_nula
BEFORE INSERT OR UPDATE ON horario
FOR EACH ROW
EXECUTE FUNCTION sustituir_asignatura_nula();
```

Función crearUsuario():

Cuando un usuario se inserta en la tabla auth.users, esta función inserta automáticamente un registro en la tabla public.users (o cualquier tabla correspondiente) utilizando el id del usuario.

```
CREATE OR REPLACE FUNCTION crearUsuario()
RETURNS trigger AS $$
BEGIN
  INSERT INTO public.users (id_tabla)
  VALUES (NEW.id);
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER creacion
AFTER INSERT ON public.users
FOR EACH ROW EXECUTE FUNCTION crearUsuario();
```

5. Cálculo de Estadísticas de Guardias

Se han desarrollado varias funciones para calcular estadísticas sobre las guardias realizadas por los profesores.

Guardias Activas e Inactivas

Existen dos funciones principales encargadas de contabilizar el número de guardias activas e inactivas:

f1(): Calcula la cantidad de guardias activas en cada día y franja horaria. Primero vacía la tabla de estadísticas y reinicia su contador de identificadores, luego agrupa las guardias activas por día y hora y almacena el total en la tabla de estadísticas. Finalmente, calcula la suma total de todas las guardias activas.

f2(): Realiza el mismo proceso que f1(), pero para las guardias inactivas.

```
CREATE OR REPLACE FUNCTION f1()
RETURNS VOID AS $$
DECLARE
    seq_name TEXT;
    total_sum BIGINT;
BEGIN
    TRUNCATE stats1;
    SELECT pg_get_serial_sequence('stats1', 'id') INTO seq_name;
    IF seq_name IS NOT NULL THEN
        EXECUTE 'ALTER SEQUENCE ' || seq_name || ' RESTART WITH 1;';
    END IF;

    -- Insertar los datos de guardias en stats1
    INSERT INTO stats1 (dia, hora, total_guardias)
    SELECT dia, hora, COUNT(*) AS total_guardias
    FROM guardias
    WHERE activa = TRUE
    GROUP BY dia, hora
    ORDER BY dia, hora;

    -- Calcular la suma de total_guardias
    SELECT SUM(total_guardias) INTO total_sum FROM stats1;

    -- Actualizar la columna suma_total con la suma calculada
    UPDATE stats1 SET suma_total = total_sum;

END;
$$ LANGUAGE plpgsql;
```

Tiempo de Guardias por Profesor

Para analizar cuánto tiempo pasa cada profesor realizando guardias, se han creado las siguientes funciones:

f3(): Agrupa las guardias activas por profesor y franja horaria y almacena el tiempo total de guardias realizadas por cada docente.

```
CREATE OR REPLACE FUNCTION f3()
RETURNS VOID AS $$
DECLARE
    seq_name TEXT;
BEGIN
    TRUNCATE stats3;
    SELECT pg_get_serial_sequence('stats3', 'id') INTO seq_name;
    IF seq_name IS NOT NULL THEN
        EXECUTE 'ALTER SEQUENCE ' || seq_name || ' RESTART WITH 1;';
    END IF;
    INSERT INTO stats3 (id_profe, hora, tiempo_guardias)
    SELECT id_profesor, hora, COUNT(*) AS tiempo_guardia
    FROM guardias
    WHERE activa = TRUE  -- Asumiendo que "activa" indica si es una guardia
    GROUP BY id_profesor, hora
    ORDER BY id_profesor, hora;
END;
$$ LANGUAGE plpgsql;
```

f4(): Amplía la información de f3(), contabilizando no solo el tiempo total de guardias por profesor, sino también la cantidad de guardias que ha realizado en cada clase.

```
CREATE OR REPLACE FUNCTION f4()
RETURNS VOID AS $$
DECLARE
    seq_name TEXT;
BEGIN
    TRUNCATE stats4;
    SELECT pg_get_serial_sequence('stats4', 'id') INTO seq_name;
    IF seq_name IS NOT NULL THEN
        EXECUTE 'ALTER SEQUENCE ' || seq_name || ' RESTART WITH 1;';
    END IF;
    INSERT INTO stats4 (id_profesor, clase ,hora, tiempo_guardia)
    SELECT id_profesor, clase, hora, COUNT(*) AS tiempo_guardia
    FROM guardias
    WHERE activa = TRUE  -- Asumiendo que "activa" indica si es una
guardia
    GROUP BY id_profesor, clase, hora
    ORDER BY id_profesor, clase, hora;
END;
$$ LANGUAGE plpgsql;
```