



# Disparador para registro de eventos (Log)

Elaboró: David González  
19 de octubre de 2020

## Contenido

|  |   |
|--|---|
| Estructura de tabla para registrar eventos ..... | 2 |
| Función disparadora.....                         | 2 |
| Crear función con PgAdmin .....                  | 3 |
| Crear función con SQL Shell (psql) .....         | 5 |
| Disparador (Trigger) .....                       | 6 |
| Crear disparador con PgAdmin .....               | 6 |
| Crear disparador por SQL Shell (psql) .....      | 7 |
| Sugerencias .....                                | 8 |

## Estructura de tabla para registrar eventos

```
CREATE TABLE public."registroLog"
(
    id bigint NOT NULL DEFAULT nextval('"registroLog_id_seq"'::regclass),
    fecha date NOT NULL,
    hora time(0) without time zone NOT NULL,
    operacion character varying(255) COLLATE pg_catalog."default" NOT NULL,
    disparador character varying(255) COLLATE pg_catalog."default" DEFAULT '-':character varying,
    tabla character varying(255) COLLATE pg_catalog."default" DEFAULT '-':character varying,
    esquema character varying(255) COLLATE pg_catalog."default" DEFAULT '-':character varying,
    descripcion character varying(255) COLLATE pg_catalog."default" DEFAULT '-':character varying,
    anterior json DEFAULT '{}':json,
    nuevo json DEFAULT '{}':json,
    CONSTRAINT "registroLog_pkey" PRIMARY KEY (id)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public."registroLog"
    OWNER to postgres;
```

Nota: La migración y el modelo para Laravel ya están creados, revisar el repositorio **roboshot-local** en la rama **devel**

## Función disparadora

Existe una única función que registrará las filas eliminadas, actualizadas o insertadas en todas las tablas que contengan un disparador programado.

La función a ejecutar es la siguiente:

```
CREATE FUNCTION public.registro_log()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
begin
    if(TG_OP = 'DELETE')then
        insert into "registroLog"(
            "fecha",
            "hora",
            "operacion",
            "disparador",
            "tabla",
            "esquema",
            "anterior")
            values(now(), now(), TG_OP, TG_NAME, TG_TABLE_NAME, TG_TABLE_SCHEMA,
to_json(OLD.*));
        return OLD;
    elseif(TG_OP = 'UPDATE')then
        insert into "registroLog"(
            "fecha",
            "hora",
            "operacion",
```

```

        "disparador",
        "tabla",
        "esquema",
        "anterior",
        "nuevo")
    values(now(), now(), TG_OP, TG_NAME, TG_TABLE_NAME, TG_TABLE_SCHEMA,
to_json(OLD.*), to_json(NEW.*));
    return OLD;
elseif(TG_OP = 'INSERT')then
insert into "registroLog"(
    "fecha",
    "hora",
    "operacion",
    "disparador",
    "tabla",
    "esquema",
    "nuevo")
    values(now(), now(), TG_OP, TG_NAME, TG_TABLE_NAME, TG_TABLE_SCHEMA,
to_json(NEW.*));
    return OLD;
end if;
end;
$BODY$;

```

```

ALTER FUNCTION public.registro_log()
    OWNER TO postgres;

```

```

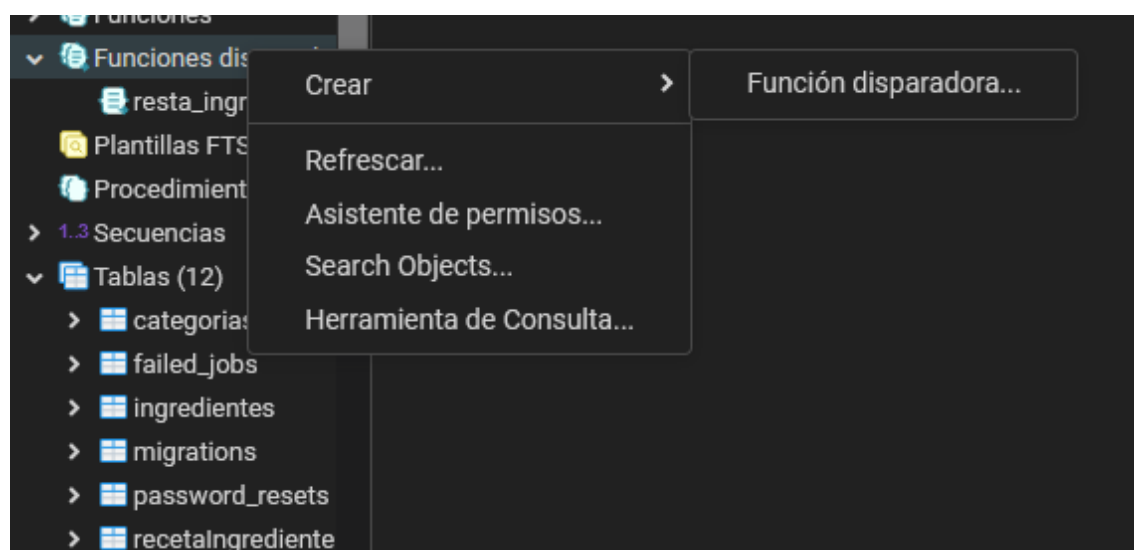
COMMENT ON FUNCTION public.registro_log()
    IS 'registra todos los eventos realizados en las tablas programadas con un disparador';

```

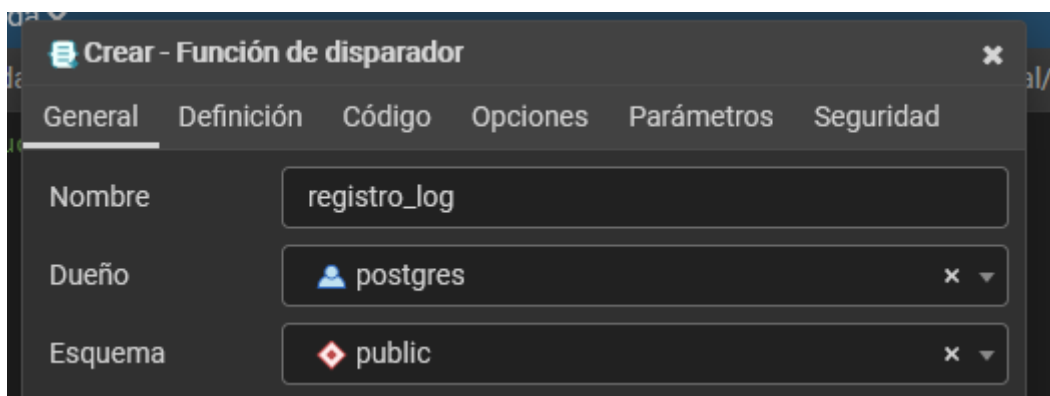
Se puede crear esta función en PostgreSQL de dos formas distintas:

### Crear función con PgAdmin

Se da clic derecho en **funciones disparadoras (functions Triggers)** -> **new** -> **function Trigger**



Se asigna el nombre del disparador (**registro\_log**)



**Crear - Función de disparador**

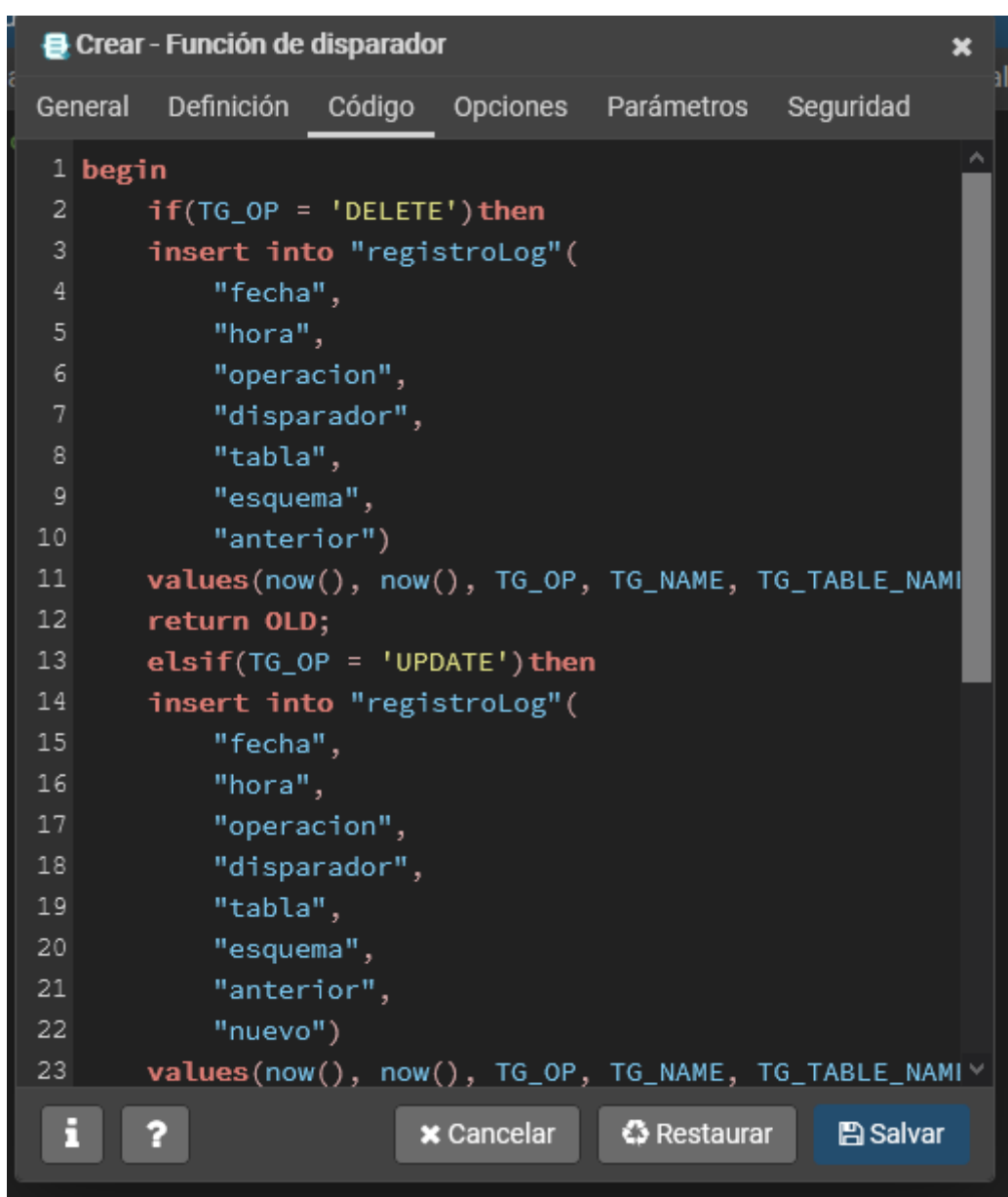
General Definición Código Opciones Parámetros Seguridad

Nombre: registro\_log

Dueño: postgres

Esquema: public

En el apartado **Código / SQL** se pega y copia el código resaltado en azul arriba escrito





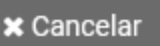


**Crear - Función de disparador**

General Definición Código Opciones Parámetros Seguridad

```

1 begin
2     if(TG_OP = 'DELETE')then
3         insert into "registroLog"(
4             "fecha",
5             "hora",
6             "operacion",
7             "disparador",
8             "tabla",
9             "esquema",
10            "anterior")
11        values(now(), now(), TG_OP, TG_NAME, TG_TABLE_NAME);
12        return OLD;
13    elseif(TG_OP = 'UPDATE')then
14        insert into "registroLog"(
15            "fecha",
16            "hora",
17            "operacion",
18            "disparador",
19            "tabla",
20            "esquema",
21            "anterior",
22            "nuevo")
23        values(now(), now(), TG_OP, TG_NAME, TG_TABLE_NAME);

```

Clic en **Salvar / Save**.

## Crear función con SQL Shell (psql)

Se abre la consola Shell de PostgreSQL y se ingresan las credenciales correspondientes, al igual que el nombre de la base de datos que se ha de trabajar

```
SQL Shell (psql)
Server [localhost]:
Database [postgres]: roboshotLocal
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:
psql (11.9)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.
roboshotLocal=#
```

Con el comando `\e` se abrirá la aplicación de bloc de notas. En ese archivo se copia y pega todo el código arriba escrito (verde y azul).

```
roboshotLocal=# \e

*psql.edit.11576.sql: Bloc de notas
Archivo Edición Formato Ver Ayuda
CREATE FUNCTION public.registro_log()
  RETURNS trigger
  LANGUAGE 'plpgsql'
  COST 100
  VOLATILE NOT LEAKPROOF
AS $BODY$
begin
    if(TG_OP = 'DELETE')then
    insert into "registroLog"(
        "fecha",
        "hora",
        "operacion",
        "disparador",
        "tabla",
        "esquema",
        "anterior")
    values(now(), now(), TG_OP, TG_NAME, TG_TABLE_NAME, TG_TABLE_SCHEMA, to_json(OLD.*));
    return OLD;
    elseif(TG_OP = 'UPDATE')then
    insert into "registroLog"(
        "fecha",
        "hora",
        "operacion",
        "disparador",
        "tabla",
        "esquema",
        "anterior",
```

Se guarda el archivo y se cierra. Una vez cerrado, en consola aparecerá la leyenda **CREATE FUNCTION.**

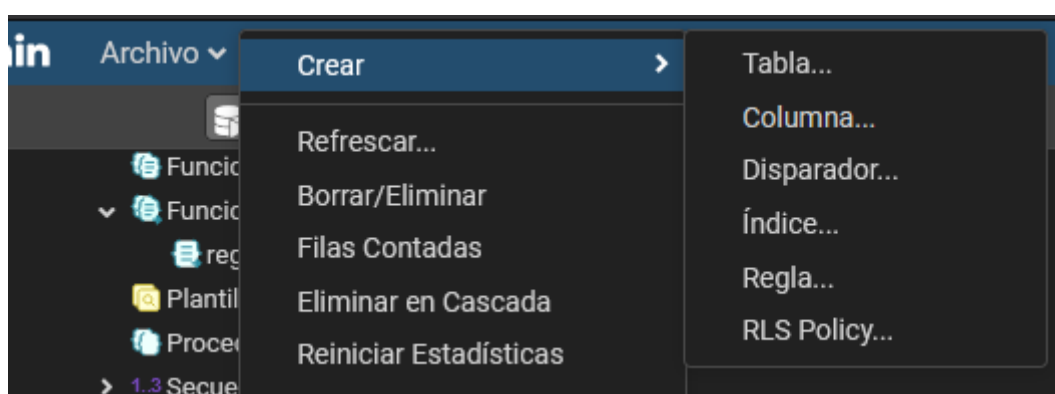
## Disparador (Trigger)

El disparador o *Trigger* será el encargado de ejecutar la función descrita arriba. Deberá ser creado directamente en la tabla que se requiera.

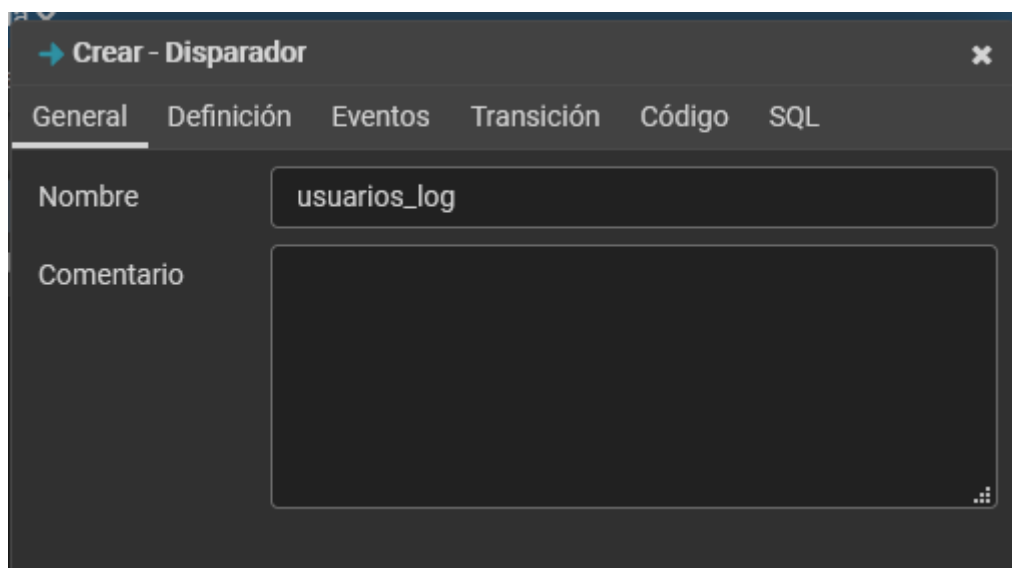
```
create trigger registro_ingredientes
before insert or update or delete
on ingredientes
for each row
execute procedure registro_log();
```

## Crear disparador con PgAdmin

Se da clic derecho sobre **nombre\_tabla** -> **new** -> **Trigger...**



Se añade un nombre al disparador (**usuarios\_log**)



Se le asigna la función disparadora creada con anterioridad, además de que se deja seleccionada la casilla **¿Disparador de fila?**

→ Crear - Disparador

General Definición Eventos Transición Código SQL

¿Disparador de fila? ☒ Si

¿Disparador de restricción? ☐ No

¿Aplazable? ☐ No

¿Aplazado? ☐ No

Función de disparador public.registro\_log

Se añaden los eventos que ha de ejecutar, además de cuándo se va a ejecutar. Se ha establecido que se realice antes de ejecutar el comando SQL (**before**).

→ Crear - Disparador

General Definición **Eventos** Transición Código SQL

Dispara BEFORE

**Eventos**

INSERT ☒ Si UPDATE ☒ Si

DELETE ☒ Si TRUNCATE ☐ No

Cuando 1

Columnas Select an item...

Clic en **Salvar / Save**.

### Crear disparador por SQL Shell (psql)

Se siguen los mismos pasos que en la creación de la función. Se ejecuta la consola y se escribe el comando \e

```
roboshotLocal=# \e
```



En el archivo de bloc de notas se copia el código arriba escrito (amarillo). Hay que realizar los cambios correspondientes en cuanto al nombre de la tabla y el nombre del disparador. En caso de que no se cambien habrá conflicto entre los disparadores o registros repetidos.

```
*psql.edit.11576.sql: Bloc de notas
Archivo Edición Formato Ver Ayuda
create trigger usuarios_log
before insert or update or delete
on usuarios
for each row
execute procedure registro_log();
```

Se guarda el archivo y se cierra. En consola aparecerá la leyenda **CREATE TRIGGER**.

```
roboshotLocal-# \e
CREATE TRIGGER
```

## Sugerencias

En el caso de que se haga un trigger erróneo se puede eliminar de la tabla con el siguiente comando:

```
roboshotLocal=# drop trigger usuarios_log on ingredientes;
DROP TRIGGER
```

Se puede consultar la lista de funciones realizadas con el comando **/df**

```
roboshotLocal=# \df
Listado de funciones
Esquema | Nombre | Tipo de dato de salida | Tipos de datos de argumentos | Tipo
-----+-----+-----+-----+-----
public | registro_log | trigger | | func
(1 fila)
```

**No confundir los disparadores con disparadores por evento, ya que estos últimos trabajan de forma global en la base de datos y no son requeridos por el momento.**