Pruebas unitarias en Postgres

Jorge Gómez Reus

24 de febrero de 2019

Índice

1. Introducción

1.1. Pruebas Unitarias

Las pruebas son una parte fundamental del desarrollo de software ya que nos permiten asegurar en parte la calidad de este. Las pruebas unitarias son un nivel de pruebas de software que involucra probar los componentes/unidades del software. **Una unidad es la parte más pequeña del software capaz de ser probada**.

Existen frameworks pruebas unitarias para el nivel de aplicaciones, en el caso de java: Junit, TestNG, entre otros, pero estos frameworks solo son del nivel de aplicación, por ende necesitamos una herramienta que nos permita hacer pruebas a un nivel más bajo en el stack de tecnologías i.e. la base de datos.

1.2. pgTAP

pgTAP es un conjunto de funciones de base de datos que nos facilitan escribir pruebas estilo TAP en scripts de psql o funciones estilo XUnit. El formato de salida estilo **TAP** nos permite obtener, analizar y reportar la información de las pruebas usando herramientas estandarizadas, como por ejemplo **pg_prove**. La gran ventaja de pgTAP es que las funciones de prueba fueron escritas en pgplsql, por no tienen el *overhead* de un driver de postgres o un ORM.
Ejemplo:

```
1 -- Comenzamos una transacción
2 BEGIN;
3 -- Seleccionamos el número de pruebas a hacer, en este caso 2
4 SELECT plan(2);
5 --Variables
6 \set var_id_alumno 1
7
8 -- Prueba de inserción
9 SELECT ok(
```

```
    insertar_alumno(:src_id, 'nombre', 'apellido_p', 'apellido_m'),
    'insertar_alumno()_debería_regresar_true'
);

-- Prueba de consulta
SELECT is(
    ARRAY(
    SELECT nb_alumno FROM tal01_alumno WHERE id_alumno = :id_alumno;
    ),
    ARRAY['character_varying'],
    'Consultamos_el_nombre_de_un_alumno'
);
--Le decimos a pgTAP que las pruebas fueron completadas,
--esto para que indique los resultados
SELECT * FROM finish();
--Hacemos un rollback de la transacción
ROLLBACK;
```

1.3. ¿Qué nos permite hacer pgTAP?

- Probar existencia, tipo de lenguaje, tipo de retorno y "salud" de una función.

```
   BEGIN;
       SELECT plan(4);
       SELECT has_function(
       'public',
       'spsce_1',
       ARRAY[ 'integer', 'integer', 'integer' ],
       'Probar_la_existencia_de_la_función_spsce_1(idCiclo,_idNivel,_idGrupo)'
       );
       SELECT function_lang_is(
       'public',
       'spsce_1',
       ARRAY[ 'integer', 'integer', 'integer' ],
       'plpgsql',
       'Probar_que_la_función_spsce_1_está_escrita_el_plpgsql'
       );
       SELECT function_returns(
       'public',
       'spsce_1',
       ARRAY[ 'integer', 'integer', 'integer' ],
       'setof_tce03_grupo',
       'Probar_que_la_función_spsce_1_retorna_setof_tce03_grupo'
       );
       \set id_ciclo 1
       \set id_nivel 1
       \set id_grupo 1
       SELECT lives_ok(
           FORMAT(
           'SELECT_*_FROM_public.spsce_1(_%s,_%s,_%s)',
```

```
            :id_ciclo, :id_nivel, :id_grupo),
            'Probar que la función spsce 1 no genere error en tiempo de ejecución'
        );
        SELECT * FROM finish();
    ROLLBACK;
```

- Probar configuraciones del gestor de base de datos.
- Probar consultas.