

# Testing en Giswater

Arnau Urgeles Arribas i Dani Marín Bocanegra

BGEO

02-09-2025

# Tabla de contenidos

---

**01** Qué es Giswater

**02** Importancia del  
CI/Testing

**03** Qué es pgTAP

**04** Clasificación de  
tests

**05** GitHub Actions

**06** Ejecución de tests

**07** Resultados

**08** Conclusiones



# Qué es Giswater

---

- Plataforma open-source para redes hidráulicas.
- QGIS + PostgreSQL/PostGIS.
- Modelos: UD (Urban Drainage) y WS (Water Supply)

# Importancia del testing en BBDD

---



## Sin tests

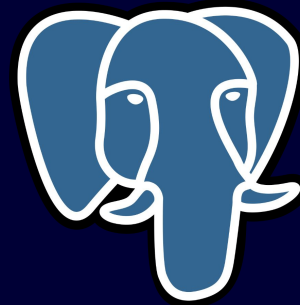
- Errores en triggers o funciones.
- Estructura de BBDD rota.
- Datos inconsistentes.
- Migraciones que rompen modelos.
- Error humano.



## Con tests

- Repetible y confiable.
- Detecta errores temprano.
- Integración con CI/CD.
- Automatizable.
- Documenta el comportamiento esperado de la BD.

# Qué es pgTAP



## Características

- Framework de testing para PostgreSQL.
- Tests escritos en SQL.
- Compatible con CI/CD.

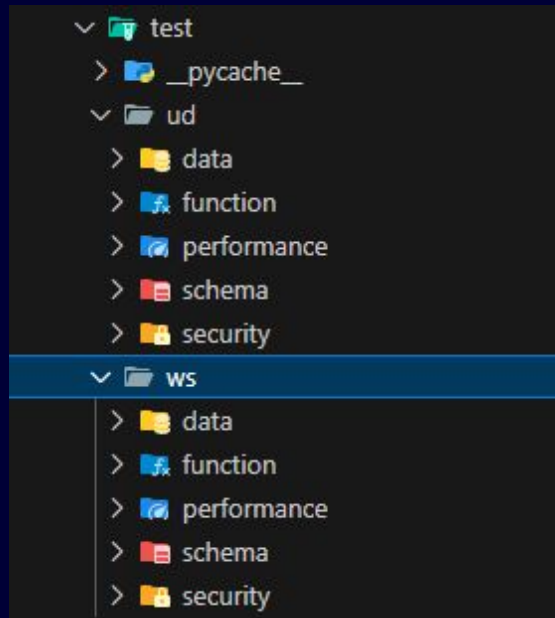
```
BEGIN;  
  
SELECT * FROM no_plan();  
  
SELECT has_table('anl_arc'::name, 'Table anl_arc should exist');  
-- check columns names  
  
SELECT columns_are(  
    'anl_arc',  
    ARRAY[  
        'id', 'arc_id', 'arccat_id', 'state', 'arc_id_aux', 'expl_id', 'fid', 'cur_user', 'the_geom'  
    ],  
    'Table anl_arc should have the correct columns'  
);  
  
SELECT * FROM finish();  
  
ROLLBACK;
```

# Clasificación de tests



## Tipos de tests

- Data
- Function
- Performance
- Schema
- Security



# Clasificación de tests

---

## Data



Se enfocan en la integridad y consistencia de los datos.

## Function



Se enfocan en validar la lógica de las funciones y procedimientos almacenados.

## Security



Evalúan la protección de la base de datos frente a accesos no autorizados o vulnerabilidades.

## Performance



Evalúan la eficiencia de consultas y operaciones sobre la base de datos.

## Schema



Validan que la estructura de la base de datos sea correcta.

# GitHub Actions

---

- Plataforma de automatización de flujos de trabajo (CI/CD) integrada en GitHub.
- Definición de flujos en archivos YAML dentro del repositorio.
- Ejecución automática cuando ocurre un evento (push, pull request, release, etc.).
- Basado en jobs y steps que corren en runners (máquinas virtuales).



GitHub Actions



# Ejecución de los tests

## Archivo workflow

Contiene los pasos de ejecución de tests, instalación de dependencias, y configuración del entorno PostgreSQL/PostGIS.

Ejemplo de pasos:

- Setup Ubuntu + PostgreSQL + PostGIS + pgTAP
- Ejecutar scripts SQL para crear esquema de Giswater
- Ejecutar tests con pgTAP
- Reportar resultados

```
name: CI Test WS database

on:
  push:
    branches: [dev, master]
  pull_request:
    branches: [dev, master]
  workflow_dispatch:
    inputs:
      tests:
        description: "Tests to run"
        required: false
        default: "all"

jobs:
  ci_test_ws_db:
    runs-on: ubuntu-latest
    strategy:
      fail-fast: false
      matrix:
        pg_version: [16, 17]

    env:
      PGPASSWORD: ${ secrets.POSTGRES_PASSWORD }

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Show which PG version we're testing
        run: echo "Testing on PostgreSQL version ${ matrix.pg_version }"

      - name: Set up Python
        uses: actions/setup-python@v2
        with:
```

# Ejecución de tests en Giswater (Github)

Push

## Entorno

Se inicia el workflow i se crea un runner con:

- A. Ubuntu
- B. Python
- C. PostgreSQL + PostGIS + pgTAP

## Creación BBDD

Se ejecutan los archivos para crear una base de datos de Giswater:

## Ejecución tests

Se ejecutan los tests sobre el schema creado:

- A. Schema
- B. Security
- C. Function
- D. Data
- E. Performance

## Resultados

Se muestran los resultados de cada test ejecutado, mostrando detalles en caso de error.

Final

# Ejecución de tests en Giswater (Act)

---

- Act permite ejecutar los workflows de GitHub Actions en tu máquina local.
- Usa el mismo archivo .yaml usado por github
- Ventajas:
  - Mismo resultado que en la nube.
  - Desarrollo más rápido (no hace falta hacer push).
  - Facilita la depuración de errores.

# Resultados

- Cada test genera un resultado PASS / FAIL automáticamente.
- Los logs muestran detalles de errores:
  - Qué test falla
  - Valores esperados vs obtenidos
  - Línea o función SQL implicada

## Resultado de tests

```
psql:test/ud/function/test_gw_fct_setarcddivide.sql:52: ERROR:  query string argument of EXECUTE is null
CONTEXT:  PL/pgSQL function gw_fct_setarcddivide(json) line 1102 at EXECUTE
test/ud/function/test_gw_fct_setarcddivide.sql .....
Dubious, test returned 3 (wstat 768, 0x300)
Failed 1/1 subtests
test/ud/function/test_gw_fct_setarcfusion.sql ..... ok
test/ud/function/test_gw_fct_setarcreverse.sql ..... ok
test/ud/function/test_gw_fct_setcatalog.sql ..... ok
```



# 12.205

Tests ejecutados en cada push

# 90%

Porcentaje del modelo cubierto

# +30

Tests ejecutados por cada tabla, función, trigger...

# Conclusiones

---



Calidad y confiabilidad



Detección temprana de  
problemas



Documentación viva del  
sistema



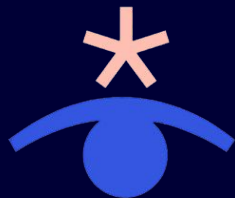
Cobertura amplia



Automatización y eficiencia



Flexibilidad en la  
ejecución



jornadas**sig**libre

Geotech/spatial data science

# Gracias!

---

Alguna pregunta?

Arnau Urgeles Arribas  
arnauurgeles.bgeo@gmail.com

Dani Marín Bocanegra  
dani.bgeo@gmail.com