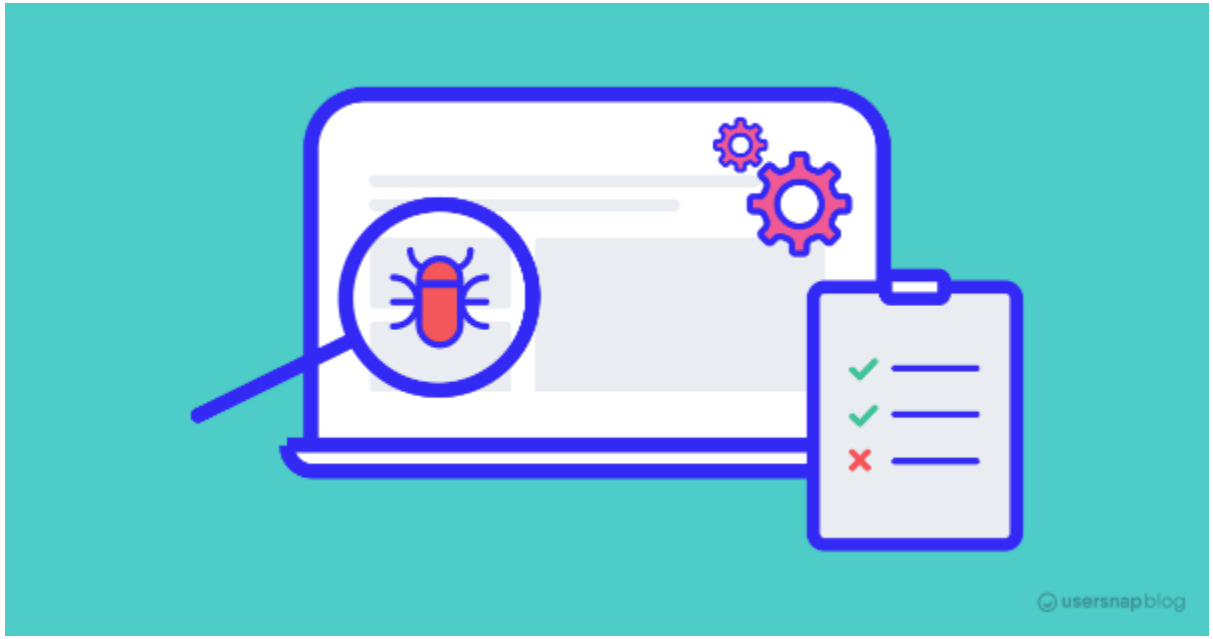
Testing de software



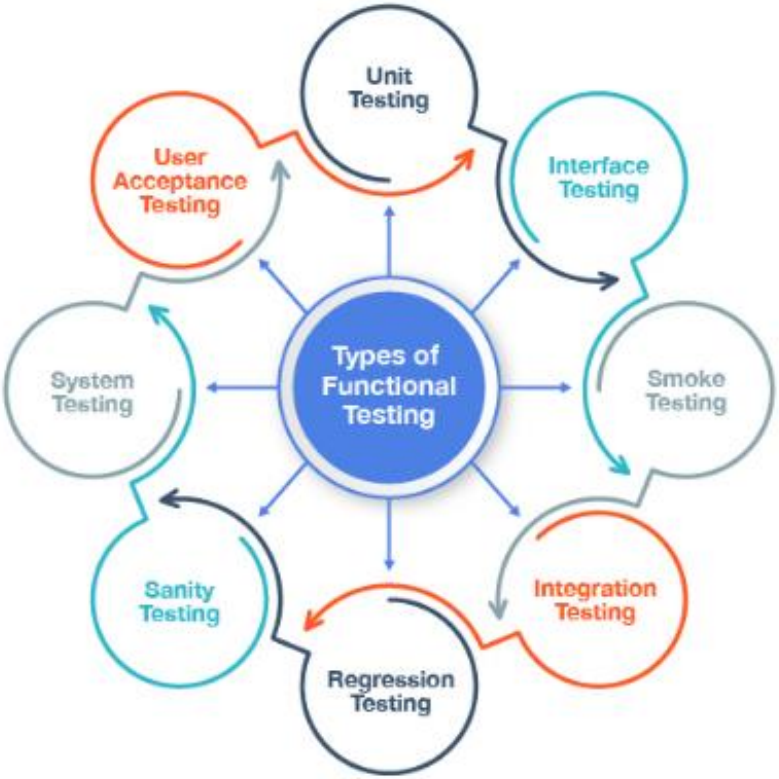
Proceso de evaluar y verificar que un producto o aplicación de software hace lo que se supone que debe hacer.

¿Para qué?

Prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento.

Tipos de Pruebas

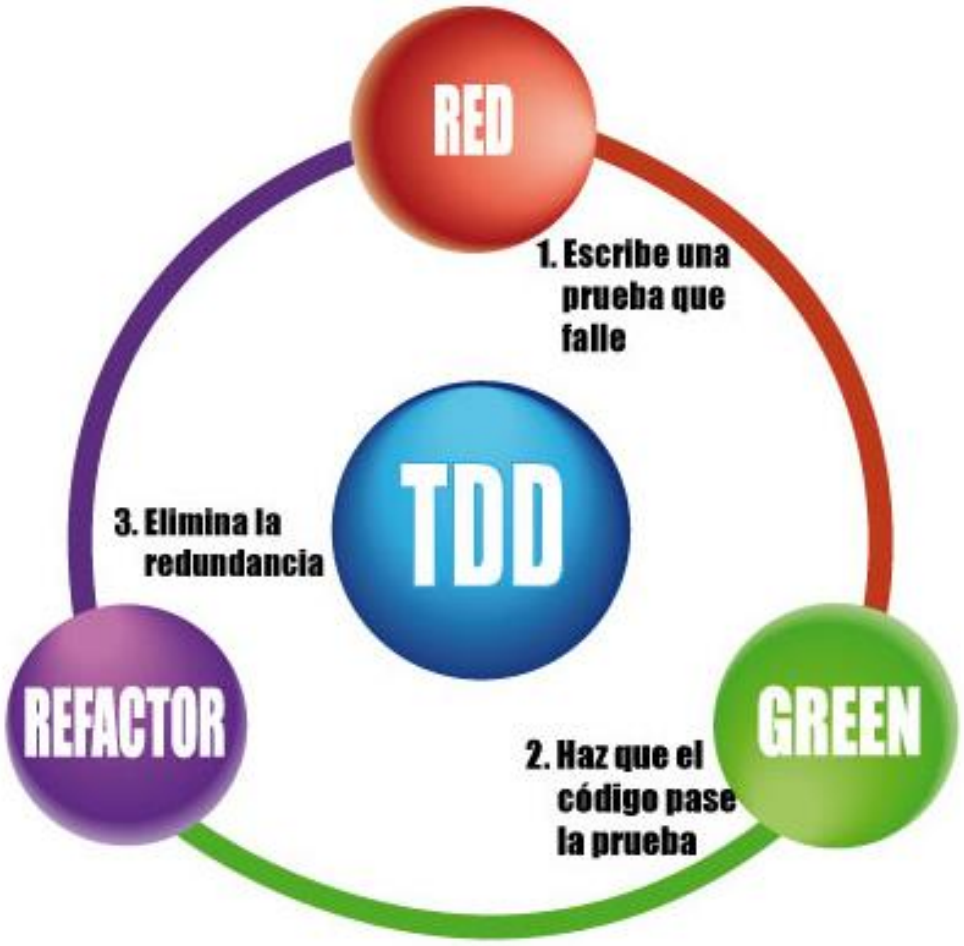
Funcionales



No funcionales

Y TDD…?

Escribir pruebas



Refactorizar código

Librerías JS para Testing

- npm install -g jest.
- **index.js**
- **index.test.js**:

```
test("Esto suma dos numeros", () =>
{
        expect(1 + 1).toBe(2);
});
```

- package-json:

```
{
"scripts": {
"test": "jest"
}
}
```

- **Testing de backend**
- npm install supertest --save-dev
- const req = require(“supertest”)

```
test('webapi remota', async () 
=> { const res = await 
req('https://dog.ceo')
.get('/api/breeds/image/random
');
  console.log(res.body);
);
```

Testing de backend

```
describe("GET /api/articulosfamilias", function () {
  it("Devolveria todos los artciulosfamilias", async function () {
    const res = await request(app)
    .get("/api/articulosfamilias")
    .set("Accept", "application/json");
    expect(res.headers["content-type"]).toEqual(
      "application/json; charset=utf-8"
      );
    expect(res.statusCode).toEqual(200);
    expect(res.body).toEqual(
      expect.arrayContaining([
        expect.objectContaining({
          IdArticuloFamilia: expect.any(Number),
          Nombre: expect.any(String),
          }),
      ])
    );
  });
});
```

Caso práctico: Testing

- Implementar en un aplicación backend Node/Express, la funcionalidad de:
  - Listar **todos productos** cargados en una base de datos sqlite.
  - Listar un **producto por id**
  - Crear **nuevo producto**
- Utilizar **Supertest** para validar los **escenarios de éxito y falla** para los endpoints (crear archivo *index.test.js*):
  - GET /productos
  - GET /productos/:id
  - POST /productos