

“TDD”

TEST DRIVEN DEVELOPMENT DESARROLLO BASADO EN PRUEBAS

Claudia Pereira Cuba





TABLE DE CONTENIDO



01.

¿QUÉ ES TDD ?

02.

¿CÓMO SE PRACTICA TDD?

03.

BENEFICIOS DEL
DESARROLLO BASADO EN
PRUEBAS






TABLE DE CONTENIDO



04.

**EJEMPLO DE TDD EN
JSCRIPT**

05.

BIBLIOGRAFÍA



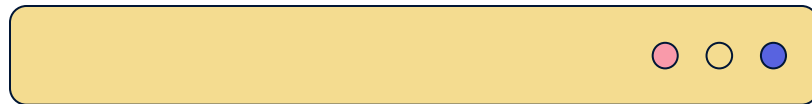


TDD

¿QUÉ ES EL TEST DRIVEN DEVELOPMENT?

Desarrollo basado en
pruebas





TDD

El desarrollo guiado por pruebas (TDD), conocido como Test Development Dirven, es una metodología de desarrollo de software que se enfoca en escribir primero las pruebas automatizada antes de implementar el código. Los desarrolladores deben crear pruebas para cada función del software antes de implementar las mismas, lo que asegura que el código cumpla con los requisitos especificados y detectar errores tempranamente en el proceso de desarrollo.

¿CÓMO SE PRACTICA TDD?



El desarrollo basado en pruebas implica pasar por tres fases, una y otra vez:

Fase roja

Fase verde

Fase azul



CICLO RED-GREEN-REFACTOR

RED

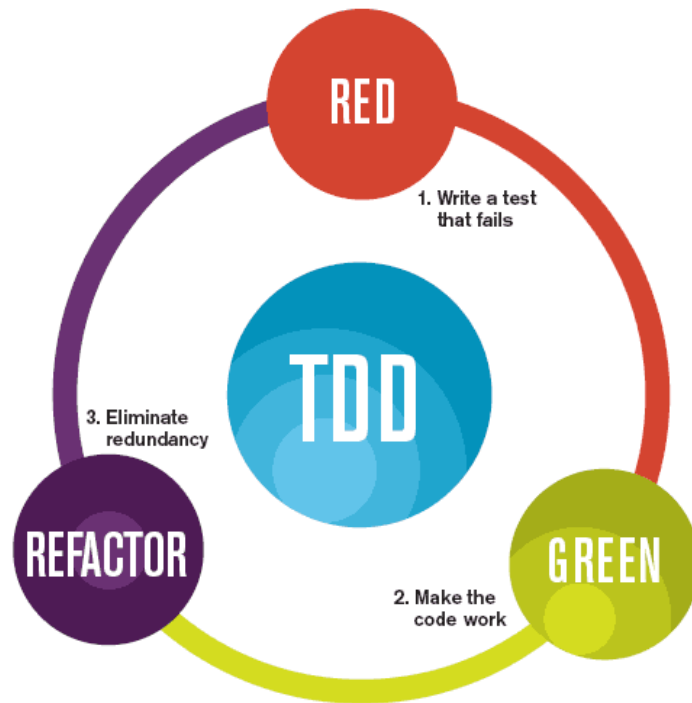
Se escribe una prueba. La prueba debe fallar inicialmente porque la funcionalidad aún no está implementada.

GREEN

Se escribe el código, se implementa el código necesario para hacer que la prueba pase.

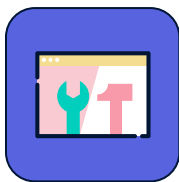
REFACTOR

Se refactoriza, para mejorar el código sin cambiar su comportamiento o funcionalidad. La refactorización asegura que el código se mantenga limpio y mantenible. Es importante asegurarse de que los cambios realizados no rompan las pruebas existentes.



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

BENEFICIOS DEL DESARROLLO BASADO EN PRUEBAS



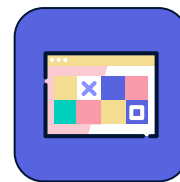
REDUCIR ERRORES

Permite a los desarrolladores detectar errores y otros problemas.



IDENTIFICAR PROBLEMAS DE FUNCIONALIDAD

Lo que permite solucionarlos más rápidamente



EVITA LA DUPLICACIÓN DE CÓDIGO

El resultado es una base de código ordenada y libre de duplicaciones innecesarias.

BENEFICIOS DEL DESARROLLO BASADO EN PRUEBAS



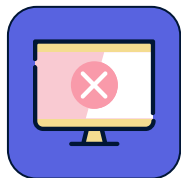
SIMPLIFICAR EL CÓDIGO

TDD requiere que los desarrolladores escriban código en respuesta a los requisitos de las pruebas. Este enfoque promueve la simplificación del código.



CREAR SOFTWARE EXTENSIBLE

Este enfoque modular del desarrollo de software contribuye a crear software que sea más flexible y extensible.



COMPRUEBA LA CALIDAD DEL CÓDIGO

El equipo de desarrolladores puede usar TDD para determinar de manera efectiva qué tan bueno es el código.



EJEMPLO TDD CON JEST EN JAVASCRIPT



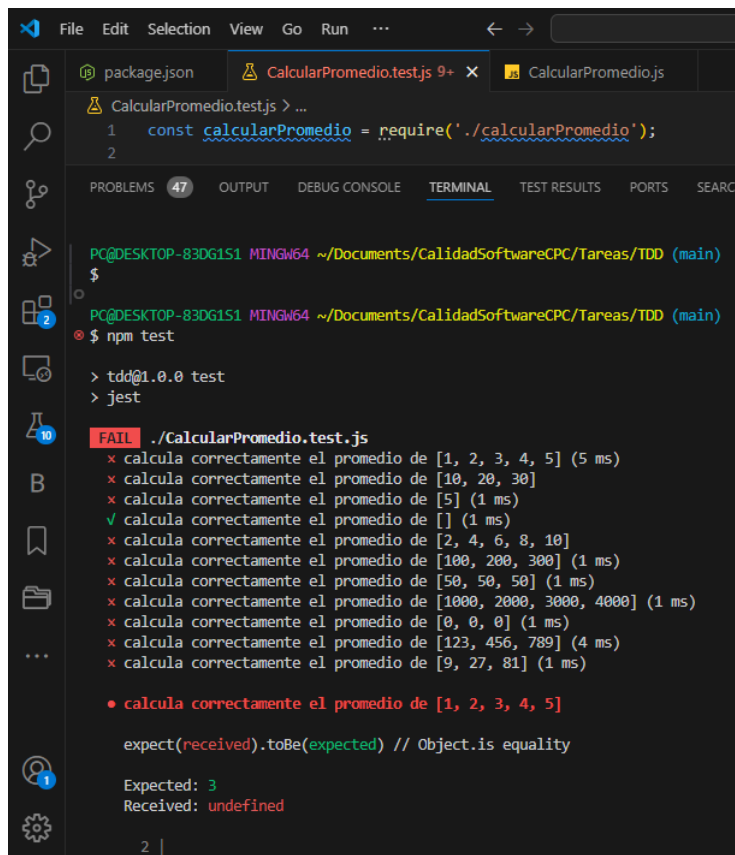
1. Red: Se escribe una prueba y falla ya que no hay implementación o la implementación no cumple con los requisitos.

```
PS C:\Users\PC\Documents\CalidadSoftware\PC\Faraoas\TDD>
PS C:\Users\PC\Documents\CalidadSoftware\PC\Faraoas\TDD> npm install --save-dev jest
you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated glob@7.2.3: glob versions prior to v9 are no longer supported

added 276 packages, and audited 277 packages in 44s
32 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

```
@ package.json  CalcularPromedio.test.js 9+  CalcularPromedio.js
CalcularPromedio.test.js > ...
1  const calcularPromedio = require('./calcularPromedio');
2
3  test('calcula correctamente el promedio de [1, 2, 3, 4, 5]', () => {
4    expect(calcularPromedio([1, 2, 3, 4, 5])).toBe(3);
5  });
6
7  test('calcula correctamente el promedio de [10, 20, 30]', () => {
8    expect(calcularPromedio([10, 20, 30])).toBe(20);
9  });
10
11 test('calcula correctamente el promedio de [5]', () => {
12   expect(calcularPromedio([5])).toBe(5);
13 });
14
15 test('calcula correctamente el promedio de []', () => {
16   expect(calcularPromedio([])).toBe(0);
17 });
18
19 test('calcula correctamente el promedio de [2, 4, 6, 8, 10]', () => {
20   expect(calcularPromedio([2, 4, 6, 8, 10])).toBe(6);
21 });
22
23 test('calcula correctamente el promedio de [100, 200, 300]', () => {
24   expect(calcularPromedio([100, 200, 300])).toBe(200);
25 });
26
27 test('calcula correctamente el promedio de [50, 50, 50]', () => {
28   expect(calcularPromedio([50, 50, 50])).toBe(50);
29 });
30
31 test('calcula correctamente el promedio de [1000, 2000, 3000, 4000]', () => {
32   expect(calcularPromedio([1000, 2000, 3000, 4000])).toBe(2500);
33 }
```

EJEMPLO TDD CON JEST EN JAVASCRIPT



```
File Edit Selection View Go Run ...
package.json CalcularPromedio.test.js 9+ x CalcularPromedio.js
CalcularPromedio.test.js > ...
1 const calcularPromedio = require('./calcularPromedio');
2

PROBLEMS 47 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS SEARCH
PC@DESKTOP-83DG1S1 MINGW64 ~/Documents/CalidadSoftwareCPC/Tareas/TDD (main)
$
PC@DESKTOP-83DG1S1 MINGW64 ~/Documents/CalidadSoftwareCPC/Tareas/TDD (main)
$ npm test

> tdd@1.0.0 test
> jest

FAIL ./CalcularPromedio.test.js
  x calcula correctamente el promedio de [1, 2, 3, 4, 5] (5 ms)
  x calcula correctamente el promedio de [10, 20, 30]
  x calcula correctamente el promedio de [5] (1 ms)
  ✓ calcula correctamente el promedio de [] (1 ms)
  x calcula correctamente el promedio de [2, 4, 6, 8, 10]
  x calcula correctamente el promedio de [100, 200, 300] (1 ms)
  x calcula correctamente el promedio de [50, 50, 50] (1 ms)
  x calcula correctamente el promedio de [1000, 2000, 3000, 4000] (1 ms)
  x calcula correctamente el promedio de [0, 0, 0] (1 ms)
  x calcula correctamente el promedio de [123, 456, 789] (4 ms)
  x calcula correctamente el promedio de [9, 27, 81] (1 ms)

  • calcula correctamente el promedio de [1, 2, 3, 4, 5]

    expect(received).toBe(expected) // Object.is equality

    Expected: 3
    Received: undefined
```

EJEMPLO TDD CON JEST EN JAVASCRIPT

```
package.json  CalcularPromedio.test.js 9+ x  CalcularPromedio.js
CalcularPromedio.test.js > ...
1  const calcularPromedio = require('./calcularPromedio');
2

PROBLEMS 47 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS SEARCH ERROR GITLENS DEVD8

2 |
3 | test('calcula correctamente el promedio de [1, 2, 3, 4, 5]', () => {
> 4 |     expect(calcularPromedio([1, 2, 3, 4, 5])).toBe(3);
    |                                             ^
5 | });
6 |
7 | test('calcula correctamente el promedio de [10, 20, 30]', () => {

at Object.toBe (CalcularPromedio.test.js:4:47)

• calcula correctamente el promedio de [10, 20, 30]

expect(received).toBe(expected) // Object.is equality

Expected: 20
Received: undefined

6 |
7 | test('calcula correctamente el promedio de [10, 20, 30]', () => {
> 8 |     expect(calcularPromedio([10, 20, 30])).toBe(20);
    |                                             ^
9 | });
10 |
11 | test('calcula correctamente el promedio de [5]', () => {

at Object.toBe (CalcularPromedio.test.js:8:44)

• calcula correctamente el promedio de [5]

expect(received).toBe(expected) // Object.is equality
```

EJEMPLO TDD CON JEST EN JAVASCRIPT

1. Green: se escribe el código, se implementa el código necesario para hacer que la prueba pase.

The screenshot shows a VS Code editor with a file named `CalcularPromedio.js` containing the following JavaScript code:

```
1 function calcularPromedio(numeros) {  
2  
3   if (numeros.length === 0) return 0;  
4  
5   return numeros.reduce((acumulador, valor) => acumulador + valor, 0) / numeros.length;  
6 }  
7  
8 module.exports = calcularPromedio;  
9
```

The left sidebar shows the file explorer with the following structure:

- OPEN EDITORS
 - package.json
 - CalcularPromedio.test.js
 - CalcularPromedio.js
- TDD
 - node_modules
 - CalcularPromedio.js
 - CalcularPromedio.test.js
 - package-lock.json
 - package.json
- OUTLINE
- TIMELINE
- MYSQL

The bottom panel shows the TEST RESULTS tab with the following output:

```
TestRun "TDD:watch-tests-0:process-start:0 (20)" started  
PASS ./CalcularPromedio.test.js  
✓ calcula correctamente el promedio de [1, 2, 3, 4, 5] (12 ms)  
✓ calcula correctamente el promedio de [10, 20, 30]  
✓ calcula correctamente el promedio de [5] (2 ms)  
✓ calcula correctamente el promedio de []  
✓ calcula correctamente el promedio de [2, 4, 6, 8, 10] (4 ms)  
✓ calcula correctamente el promedio de [100, 200, 300] (1 ms)  
✓ calcula correctamente el promedio de [50, 50, 50]  
✓ calcula correctamente el promedio de [1000, 2000, 3000, 4000] (1 ms)  
✓ calcula correctamente el promedio de [0, 0, 0] (1 ms)  
✓ calcula correctamente el promedio de [123, 456, 789] (2 ms)  
✓ calcula correctamente el promedio de [9, 27, 81] (1 ms)  
  
Test Suites: 1 passed, 1 total  
Tests: 11 passed, 11 total  
Snapshots: 0 total  
Time: 2 s  
Ran all test suites related to changed files.
```

The right sidebar shows a list of test runs with the following details:

- Test run at 8/16/2024, 6:40:48 AM
 - ✓ calcula correctamente el promedio de [1, 2, 3, 4, 5]
 - ✓ calcula correctamente el promedio de [10, 20, 30]
 - ✓ calcula correctamente el promedio de [5]
 - ✓ calcula correctamente el promedio de []
 - ✓ calcula correctamente el promedio de [2, 4, 6, 8, 10]
 - ✓ calcula correctamente el promedio de [100, 200, 300]
 - ✓ calcula correctamente el promedio de [50, 50, 50]
 - ✓ calcula correctamente el promedio de [1000, 2000, 3000, 4000]
 - ✓ calcula correctamente el promedio de [0, 0, 0]
 - ✓ calcula correctamente el promedio de [123, 456, 789]
 - ✓ calcula correctamente el promedio de [9, 27, 81]
- Test run at 8/16/2024, 6:40:42 AM
- Test run at 8/16/2024, 6:40:33 AM
- Test run at 8/16/2024, 6:37:29 AM
- Test run at 8/16/2024, 6:34:21 AM
- Test run at 8/16/2024, 6:34:17 AM
- Test run at 8/16/2024, 6:26:53 AM

EJEMPLO TDD CON JEST EN JAVASCRIPT



3. Refactor: Se refactoriza, para mejorar el código sin cambiar su comportamiento o funcionalidad.

```
9  */
10
11 function calcularPromedio(numeros) {
12
13     if (!Array.isArray(numeros)) throw new Error('El input debe ser un array');
14     if (numeros.length === 0) return 0;
15
16     if (!numeros.every(num => typeof num === 'number')) {
17         throw new Error('Todos los elementos del array deben ser números');
18     }
19
20     const suma = numeros.reduce((acumulador, valor) => acumulador + valor, 0);
21     return suma / numeros.length;
22 }
23
24 module.exports = calcularPromedio;
```

51

OUTPUT

DEBUG CONSOLE

TERMINAL

TEST RESULTS

PORTS

SEARCH ERROR

GITLENS

DEVDB

```
✓ calcula correctamente el promedio de [100, 200, 300]
✓ calcula correctamente el promedio de [50, 50, 50]
✓ calcula correctamente el promedio de [1000, 2000, 3000, 4000] (1 ms)
✓ calcula correctamente el promedio de [0, 0, 0]
✓ calcula correctamente el promedio de [123, 456, 789] (1 ms)
✓ calcula correctamente el promedio de [9, 27, 81]

Test Suites: 1 passed, 1 total
Tests: 11 passed, 11 total
Snapshots: 0 total
Time: 0.869 s, estimated 2 s
Ran all test suites.
PS C:\Users\PC\Documents\CalidadSoftwareCPC\Tareas\TDD>
```

EJEMPLO TDD CON JEST EN JAVASCRIPT



4. Se puede agregar más pruebas con otros valores, y seguir probando.

```
48 test('calcula correctamente el promedio de [-5, -10, -15]', () => {
49   expect(calcularPromedio([-5, -10, -15])).toBe(-10);
50 });
51
52 test('calcula correctamente el promedio de [1.5, 2.5, 3.5]', () => {
53   expect(calcularPromedio([1.5, 2.5, 3.5])).toBe(2.5);
54 });
55
56 test('calcula correctamente el promedio de [10, -10, 5, -5]', () => {
57   expect(calcularPromedio([10, -10, 5, -5])).toBe(0);
58 });
59
60 test('calcula correctamente el promedio de [-1.5, 0, 1.5]', () => {
61   expect(calcularPromedio([-1.5, 0, 1.5])).toBe(0);
62 });
```

PROBLEMS 51 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS SEARCH ERROR

```
✓ calcula correctamente el promedio de [2, 4, 6, 8, 10] (1 ms)
✓ calcula correctamente el promedio de [100, 200, 300]
✓ calcula correctamente el promedio de [50, 50, 50] (1 ms)
✓ calcula correctamente el promedio de [1000, 2000, 3000, 4000] (1 ms)
✓ calcula correctamente el promedio de [0, 0, 0] (1 ms)
✓ calcula correctamente el promedio de [123, 456, 789] (1 ms)
✓ calcula correctamente el promedio de [9, 27, 81] (1 ms)
✓ calcula correctamente el promedio de [-5, -10, -15]
✓ calcula correctamente el promedio de [1.5, 2.5, 3.5] (1 ms)
✓ calcula correctamente el promedio de [10, -10, 5, -5]
✓ calcula correctamente el promedio de [-1.5, 0, 1.5] (3 ms)
```

Test Suites: 1 passed, 1 total
Tests: 15 passed, 15 total



BIBLIOGRAFÍA



Hanes, M. (2019, diciembre 10). Why test-driven development (TDD). Marsner Technologies.
<https://marsner.com/blog/why-test-driven-development-tdd/>

López, M. (2023, abril 10). Qué es TDD: Metodología de Diseño de Software - IMMUNE.
Immune Technology Institute. <https://immune.institute/blog/que-es-tdd-desarrollo-software/>

Venema, M. (2022, diciembre 22). What is test driven development (TDD)? Nimblework;
NimbleWork, Inc. <https://www.nimblework.com/agile/test-driven-development-tdd/>