Universidad Autónoma Gabriel René Moreno

UAGRM SCHOOL OF ENGINEERING
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

MAESTRÍA EN DIRECCIÓN ESTRATÉGICA EN INGENIERÍA DE SOFTWARE

Módulo 3: Prueba de Software - MDEIS V2E1

"Tarea Final Grupo 5: BDD con Cucumber y TDD con Jest/XUnit"

Docente: MSc. Veronica Sasha Taboada Ovando

Integrantes:

1. Lisbeth Yasminka Quisbert Patzi
2. José Luis Quispe Salinas
3. Hugo Luis Silva Sierra
4. Javier Silva Sierra
5. Emma Tola Laca
6. Esnor Noel Enrique Vaca Moreno

Febrero, 2022

Bolivia

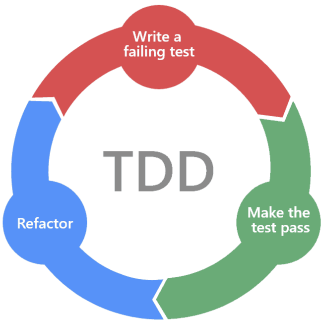
BDD con Cucumber y TDD con Jest

El código fuente correspondiente de esta tarea final del grupo 5, se encuentra publicado en GitHub en el enlace https://github.com/enevaca/demo-bdd-tdd que contiene:

- bdd-cucumber: Es un ejemplo de una Calculadora básica empleando BDD con Cucumber.
- tdd-jest: Es un ejemplo de una Calculadora básica empleando TDD con Jest.
- tdd-xunit-c#: Es un ejemplo de una Calculadora básica empleando TDD con XUnit.

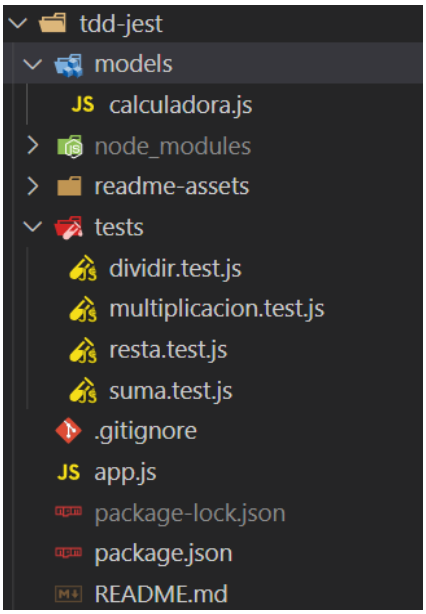
TDD con Jest

Tags: TDD, Jest, Node.JS



Código fuente: https://github.com/enevaca/demo-bdd-tdd/tree/main/tdd-jest

Video de explicación y Ejecución de la prueba TDD con Jest:
https://github.com/enevaca/demo-bdd-tdd/blob/main/tdd-jest/readme-assets/tdd-jest.mp4



Archivo: *suma.test.js*

```
const Calculadora = require('../models/calculadora');

describe('Sumar 2 números en la Calculadora', () => {
  var numero1, numero2;

  beforeEach(() => {
    numero1 = 1;
    numero2 = 2;
  });

  test(`Sumar ${numero1} y ${numero2} es igual a 3`, () => {
    // Arrange
    var calculadora = new Calculadora(numero1, numero2);

    // Act
    var resultado = calculadora.sumar();

    // Assert
    expect(resultado === 3).toBeTruthy();
  });

  test(`Sumar ${numero1} y ${numero2} No es igual a 50`, () => {
    // Arrange
    var calculadora = new Calculadora(numero1, numero2);

    // Act
    var resultado = calculadora.sumar();

    // Assert
    expect(resultado === 50).toBeFalsy();
  });

  test(`Sumar ${numero2} y ${numero1} (número al revés) es igual a 3`, () =>
{
    // Arrange
    var calculadora = new Calculadora(numero2, numero1);

    // Act
    var resultado = calculadora.sumar();

    // Assert
    expect(resultado).toEqual(3);
  });
})
```

Los otros archivos ***test (resta.test.js, multiplicacion.test.js*** y ***division.test.js)*** se encuentran en el repositorio antes mencionado.

Archivo: *calculadora.js*

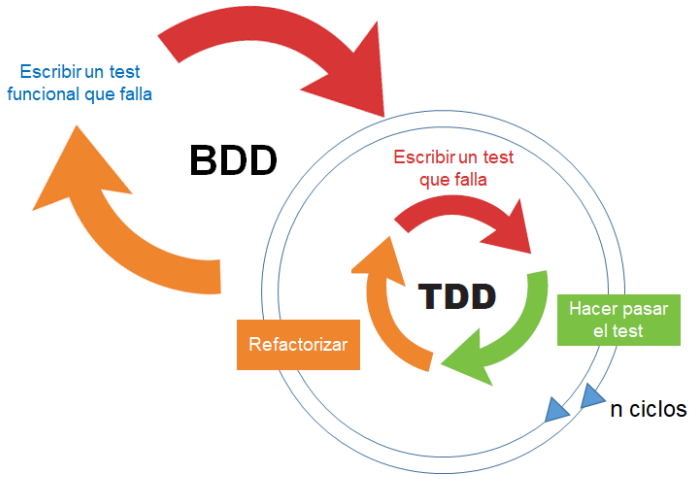
```
class Calculadora {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }

  sumar() {
    return this.x + this.y;
  }

  restar() {
    return this.x - this.y;
  }

  multiplicar() {
    return this.x * this.y;
  }

  dividir() {
    return this.x / this.y;
  }
}

module.exports = Calculadora;
```

Resultado de la ejecución TDD con Jest:

```
$ npm test

> tdd-jest@1.0.0 test
> jest

 PASS  tests/resta.test.js
 PASS  tests/multiplicacion.test.js
 PASS  tests/suma.test.js
 PASS  tests/dividir.test.js

Test Suites: 4 passed, 4 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        1.113 s
Ran all test suites.
```

BDD con Cucumber

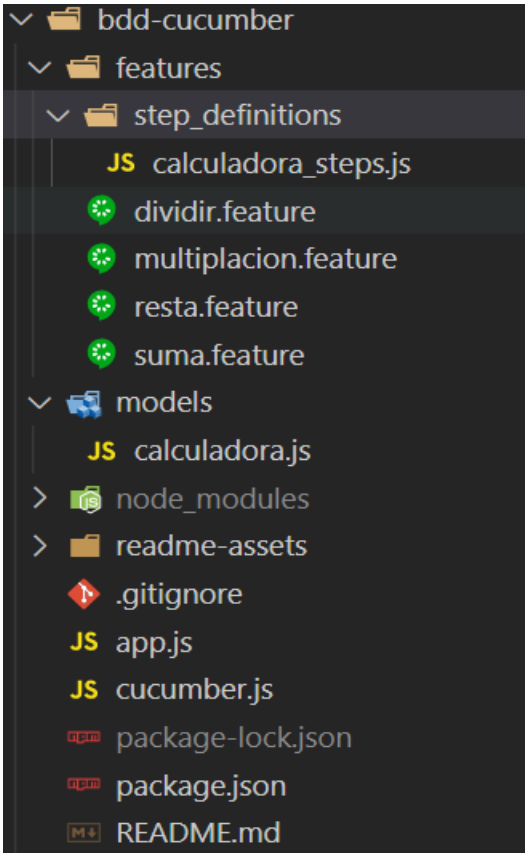
Tags: BDD, Cucumber, Node.JS



Código fuente: https://github.com/enevaca/demo-bdd-tdd/tree/main/bdd-cucumber

Video de explicación y Ejecución de la prueba BDD con Cucumber:
https://github.com/enevaca/demo-bdd-tdd/blob/main/bdd-cucumber/readme-assets/bdd-cucumber.mp4

Estructura del proyecto:



Archivo: *suma.feature*

```
@Sumar
Feature: Sumar 2 Números
  Como un usuario de la calculadora
  Quiero sumar 2 números

  Scenario: Sumar 2 números pequeños
    Given la calculadora con los números 5 y 8
    When quiero sumar ambos números
    Then el resultado debe ser 13

  Scenario: Sumar 2 números grandes
    Given la calculadora con los números 123 y 777
    When quiero sumar ambos números
    Then el resultado debe ser 900

  Scenario: Sumar 2 números con resultado incorrecto
    Given la calculadora con los números 5 y 8
    When quiero sumar ambos números
    Then el resultado debe ser distinto de 14
```

Los otros archivos de ***test funcionales*** *(**resta.feature**, **multiplicacion.feature** y **division.feature**)* se encuentran en el repositorio antes mencionado.

Archivo: *calculadora_steps.js*

```
const assert = require('assert');
const { Given, When, Then } = require('@cucumber/cucumber');
const Calculadora = require('../../models/calculadora');

var calculadora;

Given('la calculadora con los números {float} y {float}', (numero1, numero2)
=> {
  calculadora = new Calculadora(numero1, numero2);
});

When('quiero sumar ambos números', () => {
  calculadora.sumar();
});

When('quiero restar ambos números', () => {
  calculadora.restar();
});

When('quiero multiplicar ambos números', () => {
  calculadora.multiplicar();
});

When('quiero dividir ambos números', () => {
  calculadora.dividir();
});

Then('el resultado debe ser {float}', (expected) => {
```

```
  assert.equal(calculadora.getResultado(), expected);
});

Then('el resultado debe ser distinto de {float}', (expected) => {
  assert.notEqual(calculadora.getResultado(), expected);
});

Then('la división por Cero no está permitida', () => {
  assert.equal(calculadora.getResultado(), Infinity);
});
```

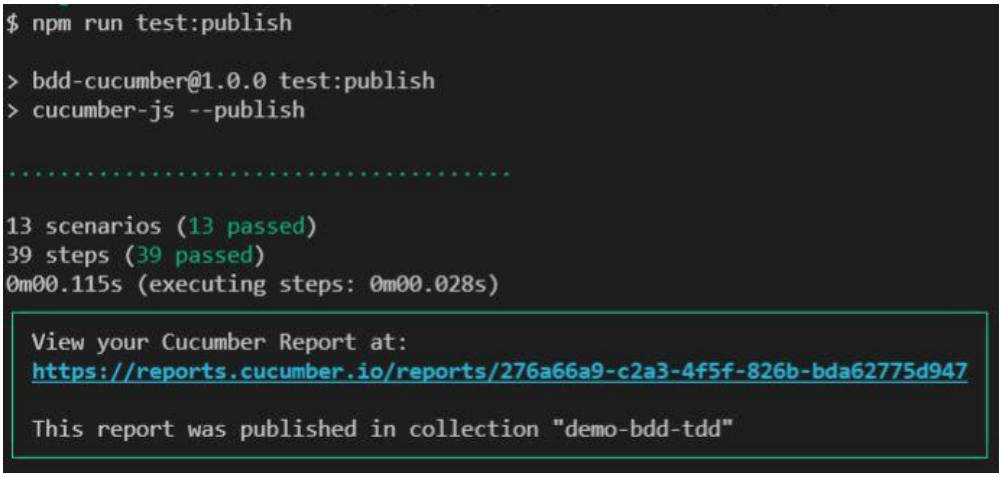
Archivo: calculadora.js

```
class Calculadora {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }

  sumar() {
    this.result = this.x + this.y;
  }

  restar() {
    this.result = this.x - this.y;
  }

  multiplicar() {
    this.result = this.x * this.y;
  }

  dividir() {
    this.result = this.x / this.y;
  }

  getResultado() {
    return this.result;
  }
}

module.exports = Calculadora;
```

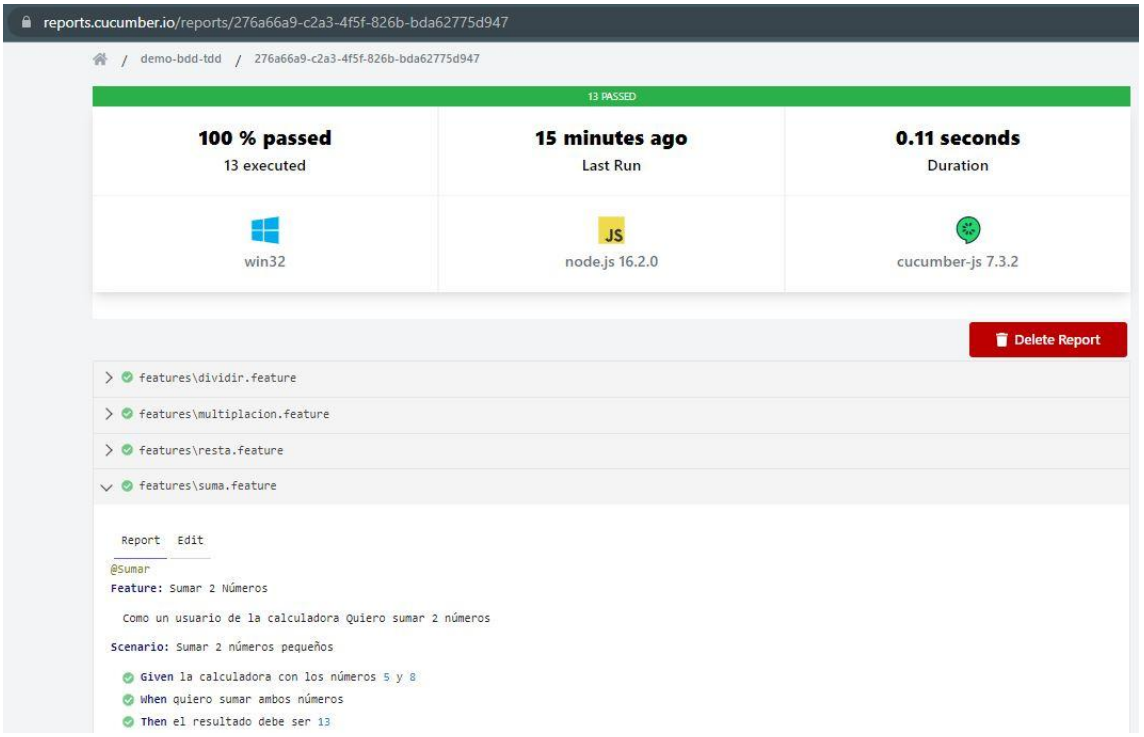
Resultado de la ejecución BDD con Cucumber:

```
$ npm test

> bdd-cucumber@1.0.0 test
> cucumber-js

.....................................

13 scenarios (13 passed)
39 steps (39 passed)
0m00.117s (executing steps: 0m00.021s)
```

Publicación de los resultado en reporte de Cucumber:



Captura de pantalla del reporte:
https://reports.cucumber.io/reports/276a66a9-c2a3-4f5f-826b-bda62775d947

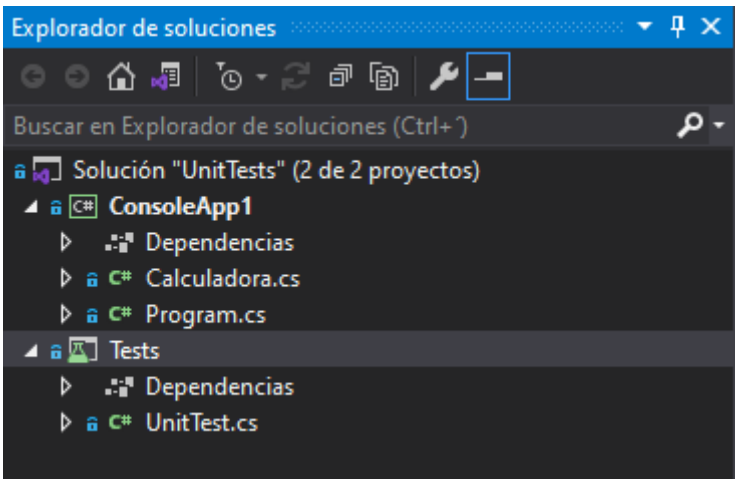


TDD con XUnit (Extra 2)

Tags: TDD, XUnit, C#, Visual Studio 2022

Código fuente: https://github.com/enevaca/demo-bdd-tdd/tree/main/tdd-xunit-c%23

Estructura del Proyecto:



Archivo: *UnitTest.cs*

```
using ConsoleApp1;
using Xunit;

namespace Tests
{
    public class UnitTest
    {
        [Fact]
        public void TestSuma()
        {
            // Arrange
            Calculadora calculadora = new Calculadora(4.5, 5.6);

            // Act
            double resultado = calculadora.Suma();

            // Assert
            Assert.Equal(10.1, resultado);
        }

        [Fact]
        public void TestResta()
        {
            // Arrange
            Calculadora calculadora = new Calculadora(100, 40);

            // Act
            double resultado = calculadora.Resta();

            // Assert
            Assert.Equal(60, resultado);
```

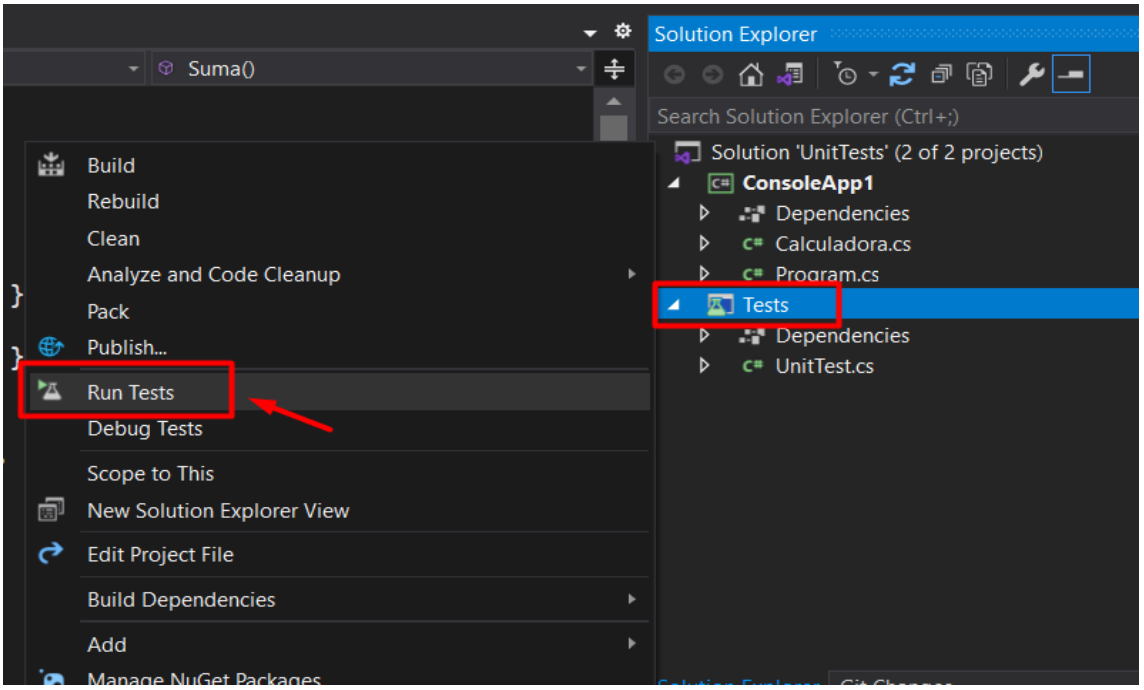
```
        }


        [Fact]
        public void TestMultiplicacion()
        {
            // Arrange
            Calculadora calculadora = new Calculadora(5, 30);

            // Act
            double resultado = calculadora.Multiplicacion();

            // Assert
            Assert.Equal(150, resultado);
        }

        [Fact]
        public void TestDivision()
        {
            // Arrange
            Calculadora calculadora = new Calculadora(300, 6);

            // Act
            double resultado = calculadora.Division();

            // Assert
            Assert.Equal(50, resultado);
        }
    }
}
```

Archivo: *Calculadora.cs*

```
namespace ConsoleApp1
{
    public class Calculadora
    {
        public double A { get; set; }
        public double B { get; set; }

        public Calculadora(double a, double b)
        {
            this.A = a;
            this.B = b;
        }

        public double Suma()
        {
            return A + B;
        }

        public double Resta()
        {
            return A - B;
        }

        public double Multiplicacion()
```

```
        {
            return A * B;
        }

        public double Division()
        {
            return A / B;
        }
    }
}
```

Forma de ejecución:



Resultado de la ejecución TDD con XUnit: