ESPE UNIVERSIDAD DE LAS FUERZAS ARMADAS INNOVACIÓN PARA LA EXCELENCIA	INFORME DE PRÁCTICA DE LABORATORIO

DEPARTAMENTO:	**CIENCIAS DE LA COMPUTACIÓN**	**CARRERA:**	**INGENIERÍA EN SOFTWARE**		
ASIGNATURA:	Pruebas de Software	**NIVEL:**	6to	**FECHA:**	22/06/2025
DOCENTE:	Ing. Luis Castillo, Mgtr.	**PRÁCTICA N°:**	3	**CALIFICACIÓN:**	

Manejo de Pruebas Unitarias

Diego Alejandro Casignia Ruiz

RESUMEN

Se desarrollaron actividades orientadas al manejo de pruebas unitarias dentro del framework Angular, con el propósito de afianzar conocimientos sobre el aseguramiento de la calidad en el desarrollo de software. Inicialmente, se creó un entorno de desarrollo y se implementaron pruebas unitarias de forma manual utilizando JavaScript, guiándose por el ciclo de vida del Desarrollo Guiado por Pruebas (TDD). Posteriormente, se emplearon las herramientas Jasmine y Karma para la automatización de pruebas, permitiendo generar reportes de cobertura y validar el comportamiento de los componentes desarrollados. El propósito fundamental del laboratorio fue comprender cómo las pruebas unitarias permiten detectar errores de forma temprana y garantizar que cada módulo del sistema funcione correctamente antes de su integración. A través de la práctica, se concluyó que el uso de TDD y herramientas especializadas no solo mejora la calidad del código, sino que también facilita su mantenimiento y escalabilidad.

Palabras Claves: Pruebas unitarias, Angular, TDD (Desarrollo Guiado por Pruebas)

1. INTRODUCCIÓN:

En esta actividad, se crearán pruebas unitarias manejadas en el framework Angular, las mismas serán establecidas de forma manual y posteriormente se desarrollarán pruebas unitarias mediante el uso de las herramientas Karma y Jasmine. Estas herramientas son compatibles con Angular y nos permiten ejecutar pruebas y desplegar un reporte de cobertura de estas.

2. OBJETIVO:

2.1 Manejar el ciclo de vida de TDD.
2.2 Manejar la sintaxis para crear pruebas unitarias en Angular.
2.3 Usar el mantra de las 3 As para crear una buena prueba unitaria.

3. MARCO TEÓRICO:

Las pruebas unitarias son una técnica fundamental dentro del desarrollo de software moderno, especialmente en metodologías ágiles y enfoques como el Desarrollo Guiado por Pruebas (TDD, por sus siglas en inglés). Estas pruebas se centran en verificar el comportamiento correcto de las unidades más pequeñas del código, como funciones o métodos, de forma aislada del resto del sistema (Beizer, 1990).

En el contexto del desarrollo con Angular, se utilizan herramientas como Jasmine y Karma para la automatización de estas pruebas. Jasmine es un framework de pruebas para JavaScript que permite describir el comportamiento esperado del código mediante una sintaxis legible y expresiva, basada en funciones como describe(), it() y expect() (Jasmine Documentation, s. f.). Por otro lado, Karma actúa como un ejecutor de pruebas, integrando Jasmine con navegadores reales para la ejecución automatizada de los casos de prueba (Karma, s. f.).

El ciclo de vida del TDD consta de tres fases: escribir una prueba que falle (Red), escribir el código mínimo necesario para que la prueba pase (Green), y refactorizar el código manteniendo la funcionalidad (Refactor) (Beck, 2002). Este enfoque promueve la creación de código más limpio, mantenible y con menor número de errores.

Angular, como framework de desarrollo frontend, integra por defecto configuraciones para realizar pruebas unitarias. Herramientas como ng test permiten ejecutar los tests escritos en Jasmine y visualizar un reporte de cobertura con el comando ng test --code-coverage. Este reporte indica qué porcentaje del código ha sido probado, lo cual es clave para garantizar la calidad del software (Google Angular Docs, s. f.). Además, el uso de matchers en Jasmine, como toBe(), toEqual() o toBeNull(), permite realizar comparaciones precisas entre los resultados esperados y los obtenidos durante la ejecución del código. Estas herramientas proporcionan un entorno robusto para el aseguramiento de calidad desde las primeras etapas del desarrollo.

4. DESCRIPCIÓN DEL PROCEDIMIENTO:

4.1 PARTE 1: Establecer el ambiente de desarrollo

4.1.1 Paso 1: Crear un proyecto nuevo de Angular.

a. Instalamos Angular CLI

```
PS C:\Users\PC> npm install -g @angular/cli
(node:12264) ExperimentalWarning: CommonJS module C:\Program Files\nodejs\node_modules\npm\node_modules\debug\src\node.js is loading ES Module C:\Program Files\nodejs\node_modules\npm\node_modules\supports-color\index.js using require().
Support for loading ES Module in require() is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)

added 28 packages, removed 19 packages, and changed 246 packages in 36s

52 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.9.0 -> 11.4.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.4.2
npm notice To update run: npm install -g npm@11.4.2
npm notice
```

```
PS C:\Users\PC> ng version
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see https://nodejs.org/en/about/previous-releases/.

     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 19.2.15
Node: 23.3.0 (Unsupported)
Package Manager: npm 11.4.2
OS: win32 x64

Angular:
...

Package                      Version
------------------------------------------------------
@angular-devkit/architect    0.1902.15 (cli-only)
@angular-devkit/core         19.2.15 (cli-only)
@angular-devkit/schematics   19.2.15 (cli-only)
@schematics/angular          19.2.15 (cli-only)

Warning: The current version of Node (23.3.0) is not supported by Angular.
```

b. Ejecutar el comando ng new *nombreProyecto*, para crear el nuevo proyecto
c. No manejamos ruteo y manejaremos únicamente CSS

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1> ng new lab1
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see https://nodejs.org/en/about/previous-releases/.
? Which stylesheet format would you like to use? CSS             [ https://developer.mozilla.org/docs/Web/CSS                    ]
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? Yes
? Would you like to use the Server Routing and App Engine APIs (Developer Preview) for this server application? Yes
CREATE lab1/angular.json (2838 bytes)
CREATE lab1/package.json (1222 bytes)
CREATE lab1/README.md (1527 bytes)
CREATE lab1/tsconfig.json (942 bytes)
CREATE lab1/.editorconfig (331 bytes)
CREATE lab1/.gitignore (629 bytes)
CREATE lab1/tsconfig.app.json (508 bytes)
CREATE lab1/tsconfig.spec.json (449 bytes)
CREATE lab1/.vscode/extensions.json (134 bytes)
CREATE lab1/.vscode/launch.json (490 bytes)
CREATE lab1/.vscode/tasks.json (980 bytes)
CREATE lab1/src/main.ts (256 bytes)
CREATE lab1/src/index.html (303 bytes)
CREATE lab1/src/styles.css (81 bytes)
CREATE lab1/src/main.server.ts (271 bytes)
CREATE lab1/src/server.ts (1702 bytes)
CREATE lab1/src/app/app.component.html (20239 bytes)
CREATE lab1/src/app/app.component.spec.ts (939 bytes)
CREATE lab1/src/app/app.component.ts (292 bytes)
CREATE lab1/src/app/app.component.css (0 bytes)
CREATE lab1/src/app/app.config.ts (447 bytes)
CREATE lab1/src/app/app.routes.ts (80 bytes)
CREATE lab1/src/app/app.config.server.ts (509 bytes)
CREATE lab1/src/app/app.routes.server.ts (174 bytes)
CREATE lab1/public/favicon.ico (15086 bytes)
✓ Packages installed successfully.
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
    Successfully initialized git.
```

4.1.2 Paso 2: Revisión de la creación del proyecto.

a. Abrir el proyecto en el editor de texto de preferencia
b. Ejecutar el proyecto con el comando ng s -o

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> ng serve -o
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For
more information, please see https://nodejs.org/en/about/previous-releases/.
Component HMR has been enabled, see https://angular.dev/hmr for more info.
Browser bundles
Initial chunk files   | Names           |  Raw size
polyfills.js          | polyfills       |  89.77 kB |
main.js               | main            |  47.99 kB |
styles.css            | styles          |  95 bytes |

                      | Initial total   | 137.86 kB


Server bundles
Initial chunk files   | Names           |  Raw size
polyfills.server.mjs  | polyfills.server| 567.22 kB |
main.server.mjs       | main.server     |  49.31 kB |
server.mjs            | server          |   1.86 kB |

Application bundle generation complete. [5.559 seconds]

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
  ➜  Local:   http://localhost:4200/
  ➜  press h + enter to show help
```

4.2 PARTE 2: Establecimiento de pruebas unitarias de forma manual con JavaScript guiadas por TDD

4.2.1 Paso 1: Crear la prueba.

a. Crear una prueba que defina un objeto de una clase Calculator()

```
import { Component, OnInit } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { Calculator } from './calculator';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent implements OnInit{
  title = 'lab1';

  ngOnInit(): void {
    let calculator = new Calculator();
    console.log("TEST CALCULATOR")

    let result = calculator.multiply(3, 7);
    console.log("1. Test Multiply");
    console.log("3 * 7 = 21 ?", result === 21);
    console.log("3 * 7 != 34 ?", result !== 34);
```

b. En una variable almacenar el resultado del método multiply() de la clase anteriormente creada
c. Mediante condicionales imprimir en consola si cumple con dos las condiciones propuestas

```
TEST CALCULATOR
1. Test Multiply
3 * 7 = 21 ? true
3 * 7 != 34 ? true
```

4.2.2 Paso 2: Crear el mínimo código para que la prueba pase o falle.

a. Crear una clase Calculator(), lo podemos hacer con el comando ng g cl calculator

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> ng generate class calculator
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
 see https://nodejs.org/en/about/previous-releases/.
CREATE src/app/calculator.spec.ts (177 bytes)
CREATE src/app/calculator.ts (30 bytes)
```

b. Importamos el componente a la clase donde estamos creando la prueba
c. Creamos el método multiply que multiplica dos números pasados por parámetro

```
export class Calculator {
    multiply(numberA: number, numberB: number){
        return numberA * numberB;
    }
```

4.2.3 Paso 3: Refactorizar el código.

a. Crear una prueba para un método que permita dividir dos números

```
import { Component, OnInit } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { Calculator } from './calculator';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent implements OnInit{
  title = 'lab1';

  ngOnInit(): void {
    let calculator = new Calculator();
    console.log("TEST CALCULATOR")

    let result = calculator.multiply(3, 7);
    console.log("1. Test Multiply");
    console.log("3 * 7 = 21 ?", result === 21);
    console.log("3 * 7 != 34 ?", result !== 34);

    let result2 = calculator.divide(10, 2);
    console.log("2. Test Divide");
    console.log("10 / 2 = 5 ?", result2 === 5);
    console.log("10 / 2 != 3 ?", result2 !== 3);

    let result3 = calculator.divide(10, 0);
    console.log("10 / 0 = null ?", result3 === null);
  }
}
```

b. Creamos el método en la clase Calculator(), con el mínimo código para que la prueba se ejecute

```
export class Calculator {
    multiply(numberA: number, numberB: number){
        return numberA * numberB;
    }

    divide(numberA: number, numberB: number){
        return numberA / numberB;
    }
}
```

c. Ahora usaremos la tercera etapa del ciclo de vida TDD que es la refactorización realizando una división para 0

```
export class Calculator {
    multiply(numberA: number, numberB: number){
        return numberA * numberB;
    }

    divide(numberA: number, numberB: number){
        if (numberB === 0){
            return null;
        }
        return numberA / numberB;
    }
}
```

d. Entonces lo que se desearía hacer es que al dividir para 0 pues se pueda responder de una manera diferente a este error, en este caso haremos que retorne un null. Y aquí es donde entra el paso de refactorización

```
TEST CALCULATOR
1. Test Multiply
3 * 7 = 21 ? true
3 * 7 != 34 ? true
2. Test Divide
10 / 2 = 5 ? true
10 / 2 != 3 ? true
10 / 0 = null ? true
```

4.3 PARTE 3: Establecimiento de pruebas unitarias de forma automática con las herramientas Karma y Jasmine

4.3.1 Paso 1: Comandos para ejecutar pruebas en Angular.

a. npm test

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> npm test

> lab1@0.0.0 test
> ng test

Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
 see https://nodejs.org/en/about/previous-releases/.
✓ Browser application bundle generation complete.
12 06 2025 12:42:15.676:WARN [karma]: No captured browser, open http://localhost:9876/
12 06 2025 12:42:15.779:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
12 06 2025 12:42:15.780:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
12 06 2025 12:42:15.797:INFO [launcher]: Starting browser Chrome
12 06 2025 12:42:18.464:INFO [Chrome 137.0.0.0 (Windows 10)]: Connected on socket wvnY-6T-aHbDzIKlAAAB with id 2199566
LOG: 'TEST CALCULATOR'
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.012 secs)
LOG: '1. Test Multiply'
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.012 secs)
LOG: '3 * 7 = 21 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.012 secs)
LOG: '3 * 7 != 34 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.012 secs)
LOG: '2. Test Divide'
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.012 secs)
LOG: '10 / 2 = 5 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.012 secs)
LOG: '10 / 2 != 3 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.012 secs)
LOG: '10 / 0 = null ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.012 secs)
Chrome 137.0.0.0 (Windows 10): Executed 5 of 5 SUCCESS (0.437 secs / 0.383 secs)
TOTAL: 5 SUCCESS
```

b. ng test --watch=false

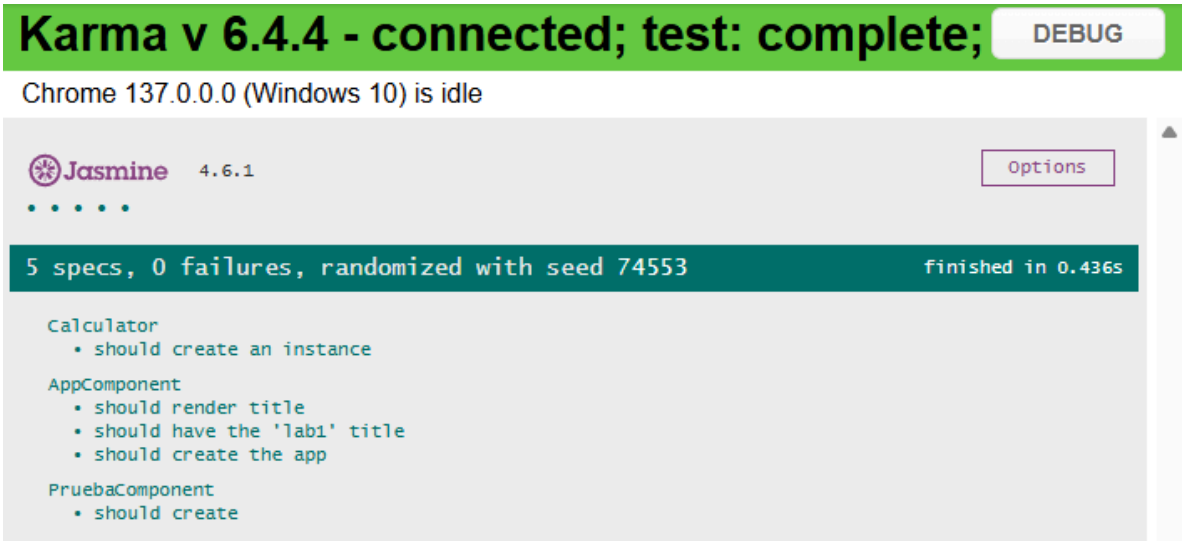
```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> ng test --watch=false
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
 see https://nodejs.org/en/about/previous-releases/.
✓ Browser application bundle generation complete.
12 06 2025 12:48:21.842:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
12 06 2025 12:48:21.846:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
12 06 2025 12:48:21.858:INFO [launcher]: Starting browser Chrome
12 06 2025 12:48:23.869:INFO [Chrome 137.0.0.0 (Windows 10)]: Connected on socket deQ3oOD1gnJfZuF7AAAB with id 13752351
LOG: 'TEST CALCULATOR'
Chrome 137.0.0.0 (Windows 10): Executed 0 of 5 SUCCESS (0 secs / 0 secs)
LOG: '1. Test Multiply'
Chrome 137.0.0.0 (Windows 10): Executed 0 of 5 SUCCESS (0 secs / 0 secs)
LOG: '3 * 7 = 21 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 0 of 5 SUCCESS (0 secs / 0 secs)
LOG: '3 * 7 != 34 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 0 of 5 SUCCESS (0 secs / 0 secs)
LOG: '2. Test Divide'
Chrome 137.0.0.0 (Windows 10): Executed 0 of 5 SUCCESS (0 secs / 0 secs)
LOG: '10 / 2 = 5 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 0 of 5 SUCCESS (0 secs / 0 secs)
LOG: '10 / 2 != 3 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 0 of 5 SUCCESS (0 secs / 0 secs)
LOG: '10 / 0 = null ?', true
Chrome 137.0.0.0 (Windows 10): Executed 0 of 5 SUCCESS (0 secs / 0 secs)
Chrome 137.0.0.0 (Windows 10): Executed 5 of 5 SUCCESS (0.013 secs / 0.238 secs)
TOTAL: 5 SUCCESS
```

c. ng test --code-coverage

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> ng test --code-coverage
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
 see https://nodejs.org/en/about/previous-releases/.
✓ Browser application bundle generation complete.
12 06 2025 12:49:41.261:WARN [karma]: No captured browser, open http://localhost:9876/
12 06 2025 12:49:41.360:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
12 06 2025 12:49:41.362:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
12 06 2025 12:49:41.384:INFO [launcher]: Starting browser Chrome
12 06 2025 12:49:43.348:INFO [Chrome 137.0.0.0 (Windows 10)]: Connected on socket nj_pDsas1KHUbYQjAAAB with id 1165817
LOG: 'TEST CALCULATOR'
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.292 secs)
LOG: '1. Test Multiply'
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.292 secs)
LOG: '3 * 7 = 21 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.292 secs)
LOG: '3 * 7 != 34 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.292 secs)
LOG: '2. Test Divide'
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.292 secs)
LOG: '10 / 2 = 5 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.292 secs)
LOG: '10 / 2 != 3 ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.292 secs)
LOG: '10 / 0 = null ?', true
Chrome 137.0.0.0 (Windows 10): Executed 1 of 5 SUCCESS (0 secs / 0.292 secs)
Chrome 137.0.0.0 (Windows 10): Executed 5 of 5 SUCCESS (0.435 secs / 0.382 secs)
TOTAL: 5 SUCCESS

============================== Coverage summary ==============================
Statements   : 100% ( 21/21 )
Branches     : 100% ( 1/1 )
Functions    : 100% ( 3/3 )
Lines        : 100% ( 19/19 )
==============================================================================
```

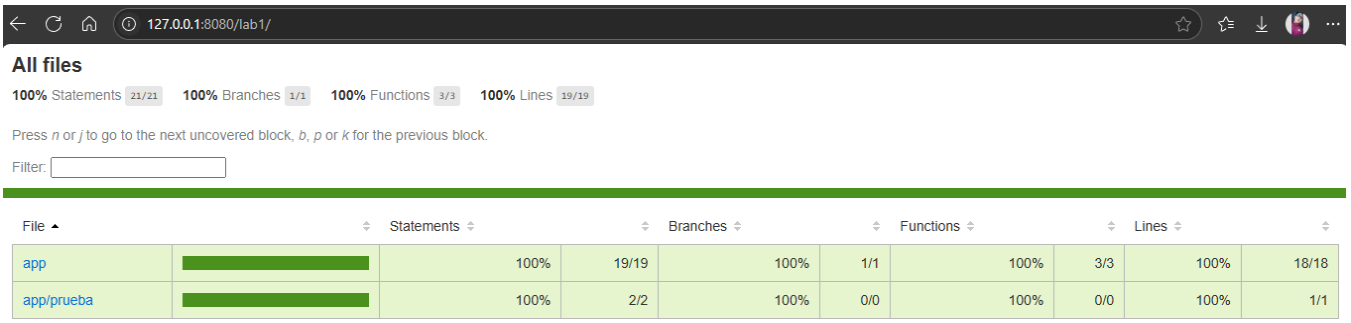
4.3.2 Paso 2: Archivos de configuración para la ejecución de pruebas.

a. Revisión del archivo karma.conf.js
b. Revisión del archivo test.ts
c. Revisar el reporte de coverage que se crea, en la carpeta correspondiente, al ejecutar el comando ng test --code-coverage



d. Para ejecutar sin estar buscando el archivo en el explorador, es con el comando http-server coverage/

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> http-server coverage/
Starting up http-server, serving coverage/

http-server version: 14.1.1

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  http://192.168.56.1:8080
  http://10.40.14.187:8080
  http://127.0.0.1:8080
Hit CTRL-C to stop the server
```



e. Pero para que este funcione debe instalarse npm install -g http-server

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> npm install -g http-server
(node:11996) ExperimentalWarning: CommonJS module C:\Users\PC\AppData\Roaming\npm\node_modules\npm\node_modules\debug\src\n
ode.js is loading ES Module C:\Users\PC\AppData\Roaming\npm\node_modules\npm\node_modules\supports-color\index.js using req
uire().
Support for loading ES Module in require() is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)

added 48 packages in 6s

15 packages are looking for funding
  run `npm fund` for details
```

4.3.3 Paso 3: Realización de pruebas con Karma y Jasmine.

a. Siempre se debe tener en cuenta la sintaxis, en la mayoría de las herramientas se tiene un describe y un it, y se realiza la comparación del valor esperado con el valor real

```
import { Calculator } from './calculator';

describe('Test for Calculator', () => {
  describe('Test for multiply', () => {
    it('should return twelve', () => {
      // Arrange
      let calculator = new Calculator();
      // Act
      let result = calculator.multiply(3, 4);
      // Assert
      expect(result).toEqual(12);
    });
  });
});
```

```
describe('Test for divide', () => {
  it('divide for a number', () => {
    let calculator = new Calculator();
    let result = calculator.divide(35, 7);
    expect(result).toEqual(5);
  });
  it('should return ten', () => {
    let calculator = new Calculator();
    let result = calculator.divide(100, 10);
    expect(result).toEqual(10);
  });
});
```

b. Creamos nuevamente el reporte de cobertura y vemos los resultados arrojados

All files

100% Statements 9/9 **100%** Branches 1/1 **100%** Functions 3/3 **100%** Lines 7/7

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
app	100%	7/7	100%	1/1	100%	3/3	100%	6/6
app/prueba	100%	2/2	100%	0/0	100%	0/0	100%	1/1

All files / app calculator.ts

100% Statements 4/4 **100%** Branches 1/1 **100%** Functions 2/2 **100%** Lines 4/4

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

```
1    export class Calculator {
2        multiply(numberA: number, numberB: number){
3  2x        return numberA * numberB;
4        }
5
6        divide(numberA: number, numberB: number){
7  3x        if (numberB === 0){
8  2x            return null;
9            }
10 1x        return numberA / numberB;
11       }
12   }
13
```

c. Seguimos ejecutando más casos de prueba, haciendo que las pruebas fallen

All files

88.88% Statements 8/9 **0%** Branches 0/1 **100%** Functions 3/3 **85.71%** Lines 6/7

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
app	85.71%	6/7	0%	0/1	100%	3/3	83.33%	5/6
app/prueba	100%	2/2	100%	0/0	100%	0/0	100%	1/1

All files

66.66% Statements 6/9 **0%** Branches 0/1 **66.66%** Functions 2/3 **57.14%** Lines 4/7

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
app	57.14%	4/7	0%	0/1	66.66%	2/3	50%	3/6
app/prueba	100%	2/2	100%	0/0	100%	0/0	100%	1/1

d. Probar los matcher de Jasmine

```
describe('Test', () => {
  describe('Jasmine Matchers', () => {
    it('test of Matchers', () => {
      let name1 = 'luis';
      let name2;
      expect(name1).toBeDefined();
      expect(name2).toBeUndefined();

      expect(1+5==6).toBeTruthy();
      expect(1+1==3).toBeFalsy();

      expect(8).toBeLessThan(40);
      expect(50).toBeGreaterThan(10);

      expect('abc').toMatch(/abc/);

      expect(["tables", "chairs", "sofa"]).toContain('chairs');
    });
  });
});
```

e. Ver el uso de beforeEach()

```
describe('Test for Calculator', () => {

  let calculator: Calculator;

  beforeEach(() => {
    calculator = new Calculator();
  });

  describe('Test for multiply', () => {
    it('should return twelve', () => {
      let result = calculator.multiply(3, 4);
      expect(result).toEqual(12);
    });
    it('should return ten', () => {
      let result = calculator.multiply(2, 5);
      expect(result).toEqual(10);
    });
  });

  describe('Test for divide', () => {
    it('divide for a number', () => {
      let result = calculator.divide(35, 7);
      expect(result).toEqual(5);
    });
    it('divide for a zero', () => {
      expect(calculator.divide(35, 0)).toBeNull;
      expect(calculator.divide(7, 0)).toBeNull;
    });
  });
```

5. PREGUNTAS/ACTIVIDADES:

Realizar 5 casos de prueba para elementos o componentes Angular (orientada a elementos HTML)

5.1 Creamos un componente de prueba

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> ng generate component prueba
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see
 https://nodejs.org/en/about/previous-releases/.
CREATE src/app/prueba/prueba.component.html (22 bytes)
CREATE src/app/prueba/prueba.component.spec.ts (615 bytes)
CREATE src/app/prueba/prueba.component.ts (225 bytes)
CREATE src/app/prueba/prueba.component.css (0 bytes)
```

5.2 Creamos prueba.component.html y prueba.component.ts

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div>
        <h1 id="titulo">Bienvenido al componente de prueba</h1>
        <input type="text" placeholder="Escribe tu nombre" [(ngModel)]="nombre" id="inputNombre" />
        <button (click)="saludar()" id="btnSaludar">Saludar</button>
        <p id="mensaje">{{ mensaje }}</p>
    </div>
</body>
</html>
```

```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-prueba',
  templateUrl: './prueba.component.html',
  styleUrls: ['./prueba.component.css']
})
export class PruebaComponent {
  nombre: string = '';
  mensaje: string = '';

  saludar(): void {
    this.mensaje = `Hola, ${this.nombre}`;
  }
}
```

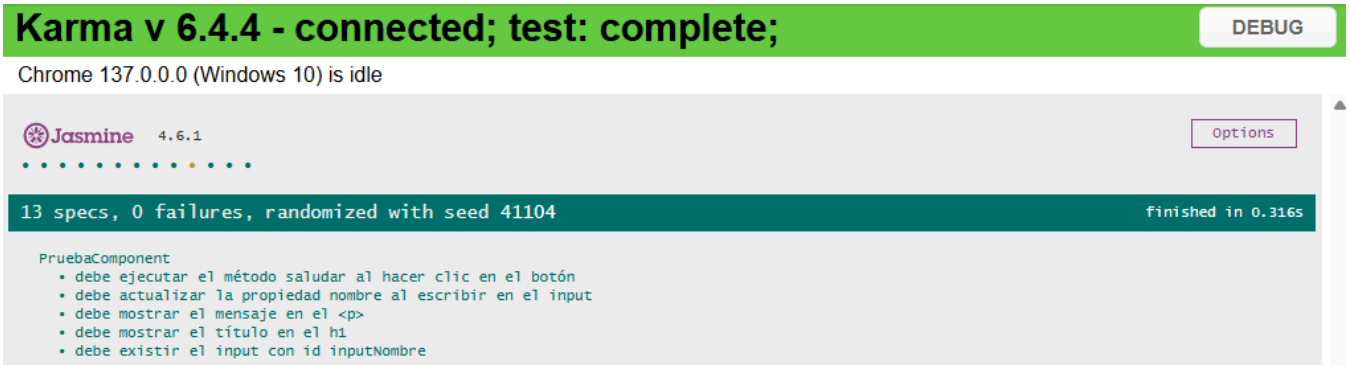
5.3 Creamos prueba.componetn.spec.ts con los 5 casos de prueba

```typescript
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { FormsModule } from '@angular/forms';
import { By } from '@angular/platform-browser';
import { PruebaComponent } from './prueba.component';

describe('PruebaComponent', () => {
  let component: PruebaComponent;
  let fixture: ComponentFixture<PruebaComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [PruebaComponent],
      imports: [FormsModule] // Necesario para usar [(ngModel)]
    }).compileComponents();

    fixture = TestBed.createComponent(PruebaComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });
```

```
  // 1. Verificar que el título se renderiza correctamente
  it('debe mostrar el título en el h1', () => {
    const titulo = fixture.debugElement.query(By.css('#titulo')).nativeElement;
    expect(titulo.textContent).toContain('Bienvenido al componente de prueba');
  });

  // 2. Verificar que el input existe en el DOM
  it('debe existir el input con id inputNombre', () => {
    const input = fixture.debugElement.query(By.css('#inputNombre'));
    expect(input).toBeTruthy();
  });

  // 3. Verificar que ngModel actualiza la propiedad "nombre"
  it('debe actualizar la propiedad nombre al escribir en el input', () => {
    const input = fixture.debugElement.query(By.css('#inputNombre')).nativeElement;
    input.value = 'Diego';
    input.dispatchEvent(new Event('input'));
    fixture.detectChanges();
    expect(component.nombre).toBe('Diego');
  });

  // 4. Verificar que al hacer clic en el botón se ejecuta el método saludar
  it('debe ejecutar el método saludar al hacer clic en el botón', () => {
    component.nombre = 'Diego';
    const boton = fixture.debugElement.query(By.css('#btnSaludar')).nativeElement;
    boton.click();
    fixture.detectChanges();
    expect(component.mensaje).toBe('Hola, Diego');
  });

  // 5. Verificar que el mensaje se muestre en el elemento <p>
  it('debe mostrar el mensaje en el <p>', () => {
    component.nombre = 'Diego';
    component.saludar();
    fixture.detectChanges();
    const mensaje = fixture.debugElement.query(By.css('#mensaje')).nativeElement;
    expect(mensaje.textContent).toBe('Hola, Diego');
  });
});
```

5.4 ng test

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> ng test
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
 see https://nodejs.org/en/about/previous-releases/.
✓ Browser application bundle generation complete.
22 06 2025 20:30:06.701:WARN [karma]: No captured browser, open http://localhost:9876/
22 06 2025 20:30:06.780:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
22 06 2025 20:30:06.782:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
22 06 2025 20:30:06.803:INFO [launcher]: Starting browser Chrome
22 06 2025 20:30:08.955:INFO [Chrome 137.0.0.0 (Windows 10)]: Connected on socket YLA4F7a91EeVp8mlAAAB with id 18853575
WARN: 'Spec 'Test for Calculator Test for divide divide for a zero' has no expectations.'
Chrome 137.0.0.0 (Windows 10): Executed 9 of 13 SUCCESS (0 secs / 0.258 secs)
Chrome 137.0.0.0 (Windows 10): Executed 13 of 13 SUCCESS (0.317 secs / 0.262 secs)
TOTAL: 13 SUCCESS
```

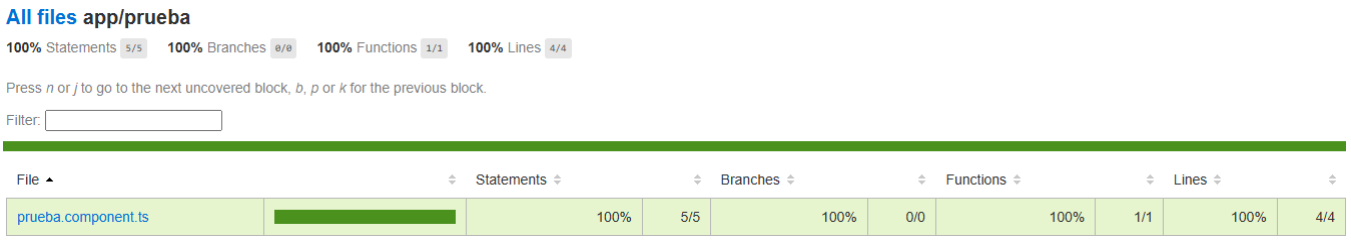


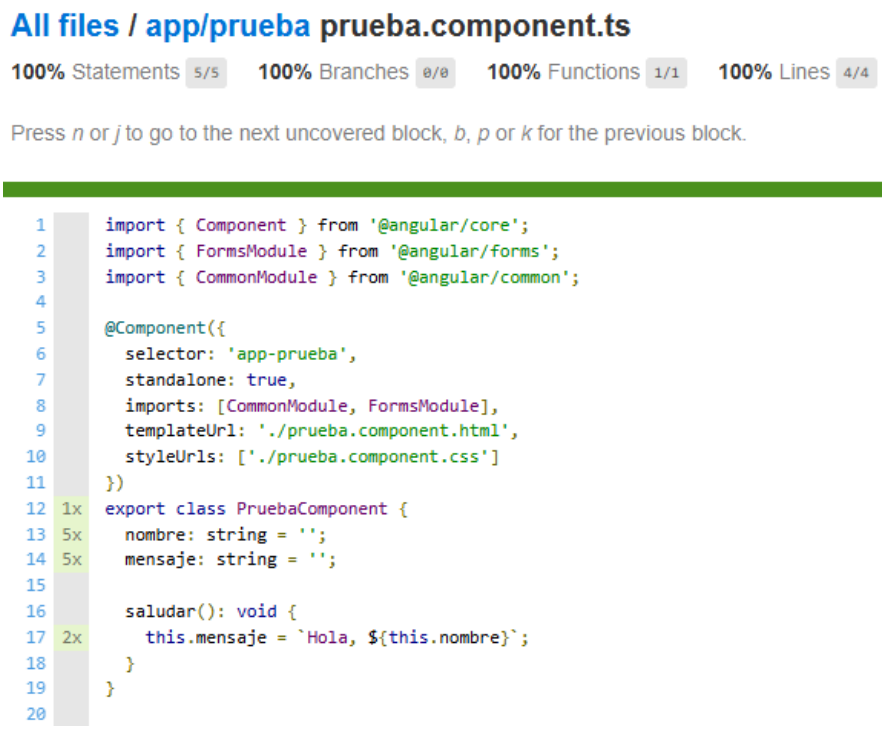
5.5 ng test –code-coverage

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> ng test --code-coverage
Node.js version v23.3.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
 see https://nodejs.org/en/about/previous-releases/.
✓ Browser application bundle generation complete.
22 06 2025 20:37:18.105:WARN [karma]: No captured browser, open http://localhost:9876/
22 06 2025 20:37:18.195:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
22 06 2025 20:37:18.196:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
22 06 2025 20:37:18.211:INFO [launcher]: Starting browser Chrome
22 06 2025 20:37:20.604:INFO [Chrome 137.0.0.0 (Windows 10)]: Connected on socket BDmaKxRSSDDAlKr_AAAB with id 82970455
WARN: 'Spec 'Test for Calculator Test for divide divide for a zero' has no expectations.'
Chrome 137.0.0.0 (Windows 10): Executed 0 of 13 SUCCESS (0 secs / 0 secs)
Chrome 137.0.0.0 (Windows 10): Executed 13 of 13 SUCCESS (0.327 secs / 0.278 secs)
TOTAL: 13 SUCCESS

============================== Coverage summary ==============================
Statements   : 100% ( 12/12 )
Branches     : 100% ( 1/1 )
Functions    : 100% ( 4/4 )
Lines        : 100% ( 10/10 )
==============================================================================
```

5.6 http-server coverage/

```
PS D:\DIEGO\Documents\SOFTWARE\PRUEBAS\u2\Lab1\lab1> http-server coverage/
Starting up http-server, serving coverage/

http-server version: 14.1.1

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  http://192.168.56.1:8080
  http://192.168.1.46:8080
  http://127.0.0.1:8080
Hit CTRL-C to stop the server
```





6. CONCLUSIONES:

6.1 La implementación de pruebas unitarias en Angular utilizando herramientas como Jasmine y Karma permite garantizar el correcto funcionamiento de los componentes desde las etapas iniciales del desarrollo, contribuyendo a la detección temprana de errores y mejorando la calidad del software.

6.2 La aplicación del ciclo de vida del Desarrollo Guiado por Pruebas (TDD) facilita la construcción de código más estructurado, mantenible y confiable, al fomentar una programación orientada a resultados esperados antes de escribir la lógica funcional.

6.3 Las pruebas realizadas de forma manual con JavaScript y luego automatizadas con Jasmine y Karma demostraron la importancia de validar no solo la lógica de negocio, sino también el comportamiento de los elementos visuales en aplicaciones frontend modernas.

7. RECOMENDACIONES:

7.1 Es recomendable adoptar el enfoque TDD en proyectos de desarrollo de software desde las primeras etapas, ya que promueve buenas prácticas de codificación y facilita el mantenimiento del código a largo plazo.

7.2 Se sugiere profundizar en el uso avanzado de Jasmine, especialmente en el manejo de *matchers personalizados* y funciones como beforeEach() y afterEach(), lo cual permite estructurar pruebas más robustas y reutilizables.

7.3 Es importante mantener actualizados los entornos de desarrollo y las dependencias del proyecto Angular, para asegurar la compatibilidad con las versiones más recientes de Jasmine y Karma, evitando errores durante la ejecución de pruebas.

7.4 Se recomienda documentar cada prueba unitaria con comentarios claros sobre su propósito, lo cual mejora la comprensión del código por parte de otros desarrolladores y facilita futuras modificaciones.

8. BIBLIOGRAFÍA:

Beck, K. (2002). Test-Driven Development: By Example. Addison-Wesley.

Beizer, B. (1990). Software Testing Techniques (2nd ed.). Van Nostrand Reinhold.

Google Angular Docs. (s. f.). Testing. Recuperado el 22 de junio de 2025, de https://angular.io/guide/testing

Jasmine Documentation. (s. f.). Introduction. Recuperado el 22 de junio de 2025, de https://jasmine.github.io/pages/docs_home.html

Karma. (s. f.). Introduction. Recuperado el 22 de junio de 2025, de https://karma-runner.github.io/latest/index.html