TESTING

Angular

ANGULAR

TEST UNITARIOS

ANGULAR

- Los tests unitarios (o pruebas unitarias) son un tipo de prueba de software cuyo objetivo es verificar que la unidad más pequeña de código funciona correctamente.
- En la práctica, una unidad suele ser una función, método, clase o componente.
- En Angular (hasta la versión 20): Jasmine + Karma
  - Jasmine → Framework de pruebas.
  - Karma → Ejecutador de pruebas.

ANGULAR

- Vitest. A partir de la versión 21 de Angular:
  - Vitest → Framework de pruebas
  - Vitest → Ejecutador de pruebas
  - Utiliza jsdom y Happy DOM como simuladores de DOM (entornos que imitan un navegador) para ejecutar código JavaScript como si fuese un navegador real.
- La configuración se realiza en angular.json.

ANGULAR

- Los ficheros de pruebas unitarias **se crean automáticamente** al crear:
  - Componentes
  - Servicios
  - Directivas (las directivas son elementos que permiten modificar el comportamiento o la apariencia de los elementos del DOM)
  - Pipes (son funciones declarativas que permiten transformar valores directamente en las plantillas –templates-)
  - Guards (mecanismos de protección de rutas)
  - Interceptors (intercepta las peticiones y respuestas HTTP para modificarlas o reaccionar ante ellas.
  - Resolvers (mecanismo para obtener datos antes de activar una ruta)
- **No se crean** al crear:
  - Módulos
  - Interfaces
  - Clases simples
  - Enums

ANGULAR

- Test de componentes

```
import { ComponentFixture, TestBed } from '@angular/c
import { Componente1 } from './componente1';

describe('Componente1', () => {
  let component: Componente1;
  let fixture: ComponentFixture<Componente1>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [Componente1]
    })
    .compileComponents();

    fixture = TestBed.createComponent(Componente1);
    component = fixture.componentInstance;
    await fixture.whenStable();
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

describe→Agrupador de casos de prueba.
'Component1'→ Nombre asignado al caso de prueba

ANGULAR

- Test de componentes

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { Componente1 } from './componente1';

describe('Componente1', () => {
  let component: Componente1;
  let fixture: ComponentFixture<Componente1>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [Componente1]
    })
    .compileComponents();

    fixture = TestBed.createComponent(Componente1);
    component = fixture.componentInstance;
    await fixture.whenStable();
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Configuración

Creación del componente (fixture)

ANGULAR

- Test de componentes

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { Componente1 } from './componente1';

describe('Componente1', () => {
  let component: Componente1;
  let fixture: ComponentFixture<Componente1>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [Componente1]
    })
    .compileComponents();

    fixture = TestBed.createComponent(Componente1)
    component = fixture.componentInstance;
    await fixture.whenStable();
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

it→Función de Jasmine. Define un caso de prueba.
'shoud create'→Descripción de aquello que debería ocurrir al ejecutar el caso de prueba.
expect→Es la aserción. Recibe un valor real y aplica un 'matcher'.
toBeTruthy()→'Matcher' que evalua si un elemento es **true** (un objeto no vacío es true).

ANGULAR

- La ejecución de pruebas se realiza con el comando `ng test`.
- Se ejecutan todos los tests.
- Karma queda en modo '*watch*' a la espera de cambios para ejecutar los cambios que se produzcan en el código.

```
Watch mode enabled. Watching for file changes...

 DEV  v4.0.14 C:/Projects/angular/unit-tests-demo

 ✓ unit-tests-demo src/app/services/service1.spec.ts (1 test) 68ms
 ✓ unit-tests-demo src/app/components/componente1/componente1.spec.ts (1 test) 120ms
 ✓ unit-tests-demo src/app/app.spec.ts (2 tests) 229ms

 Test Files  3 passed (3)
      Tests  4 passed (4)
   Start at  12:19:04
   Duration  6.02s (transform 704ms, setup 4.68s, import 735ms, tests 417ms, environment 10.00s)

 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

ANGULAR

- Se pueden ejecutar test individuales:
  - `ng test --include='**/componente1.spec.ts'`

** es un comodín para indicar que busque el fichero de test en cualquier ubicación

ANGULAR

- Algunos 'matchers' de Jasmine:
  - toBe → Igualdad estricta (===)
  - toEqual → Igualdad en profundidad (datos)
  - toBeTruthy → El valor es 'truthy'
  - toBeFalsy → El valor es 'falsy'
  - toBeDefined → El valor no es undefined
  - toBeUndefined → El valor es undefined
  - toBeNull → El valor es null
  - toBeLessThan, toBeGreaterThan, toBeLessThanOrEqual…
  - toContain → Un array o cadena contiene determinado elemento.
  - toBeInstanceOf → Comprobación de instancia
  - toThrow → Lanza un error

ANGULAR

E2E (End to End Testing)

ANGULAR

- Las pruebas de extremo a extremo (E2E) son un tipo de prueba que se utiliza para asegurar que toda la aplicación funciona según lo previsto de principio a fin.
- Las pruebas E2E se diferencian de las pruebas unitarias en que están completamente desvinculadas de los detalles de implementación subyacentes del código.
- Se suelen utilizar para validar una aplicación de forma que simule la interacción del usuario con ella.

ANGULAR

- Descarga e instalación:
  - ng e2e
  - Solicita el framework que se quiere utilizar para hacer las pruebas e2e:

```
Would you like to add a package with "e2e" capabilities now?
No
› Cypress
Nightwatch
WebdriverIO
Playwright
Puppeteer
```

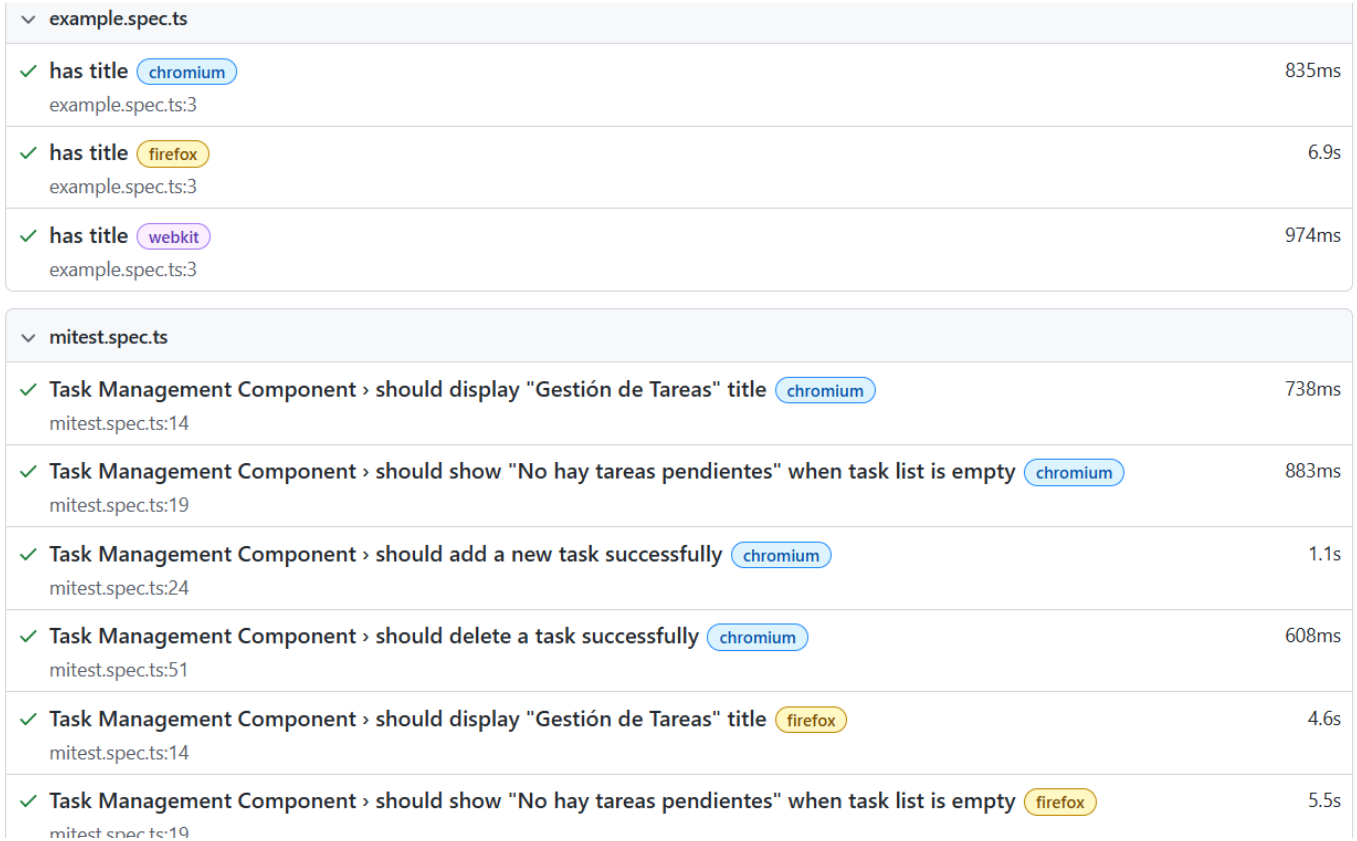
ANGULAR

- Ejecución (ejemplo Playwright):
  - Instalación de los navegadores:
    - npx playwright install
  - Ejecución de las pruebas:
    - ng e2e
  - Consulta del informe:
    - npx playwright show-report

Es el mismo comando que se utiliza para la instalación

ANGULAR

- Ejecución (ejemplo Playwright):



ANGULAR

- Enlaces de interés:
  - Testing (angular.dev):
    - https://angular.dev/guide/testing
  - Angular - Testing Unitario. Aprende Jasmine y Karma Paso a Paso
    - https://youtu.be/4y9KHXqv3p0?si=bh63a1aG8FUz3Z2m
  - Jasmine: https://jasmine.github.io/
    - Matchers → https://jasmine.github.io/api/edge/matchers.html
    - Matchers en Jasmine → https://anamartinezaguilar.medium.com/matchers-en-jasmine-984e952aeca9
  - Karma: https://karma-runner.github.io/latest/index.html
  - Vitest: https://vitest.dev/
  - Jsdom : https://github.com/jsdom/jsdom
  - Happy dom: https://github.com/capricorn86/happy-dom

ANGULAR

- Enlaces de interés:
  - End to End Testing:
    - https://angular.dev/tools/cli/end-to-end
  - Introducción a Playwright y VS Code (edición 2025)
    - https://youtu.be/WvsLGZnHmzw?si=QBXQClAb8jFBb050
  - Modern E2E Testing for Angular Apps with Playwright
    - https://angular.love/modern-e2e-testing-for-angular-apps-with-playwright
  - Playwright:
    - https://playwright.dev/
  - Cypress:
    - https://www.cypress.io/