



red.es

PROFESIONALES
DIGITALES

Formación
Continua

***El FSE** invierte en tu futuro*



UNIÓN EUROPEA

Testing



red.es

PROFESIONALES
DIGITALES

Formación
Continua



UNIÓN EUROPEA

El FSE invierte en tu futuro

Testing en Angular

Realizar test en proyectos cuando superar cierto tamaño es muy importante para probar la aplicación sin tener que hacerlo manualmente. Además si se combina con la integración continua se puede minimizar el riesgo de futuros bugs

Tipos de test que existen:

- **Tests Unitarios:** Consiste en probar unidades pequeñas (componentes por ejemplo).
- **Tests End to End (E2E):** Consiste en probar toda la aplicación simulando la acción de un usuario, es decir, por ejemplo para desarrollo web, mediante herramientas automáticas, se abrirá el navegador y navegará y usará la página como lo haría un usuario normal.
- **Tests de Integración:** Consiste en probar el conjunto de la aplicación asegurando la correcta comunicación entre los distintos elementos de la aplicación. Por ejemplo, en Angular observando cómo se comunican los servicios con la API y con los componentes.



red.es

PROFESIONALES
DIGITALES

Formación
Continua

El FSE invierte en tu futuro



UNIÓN EUROPEA

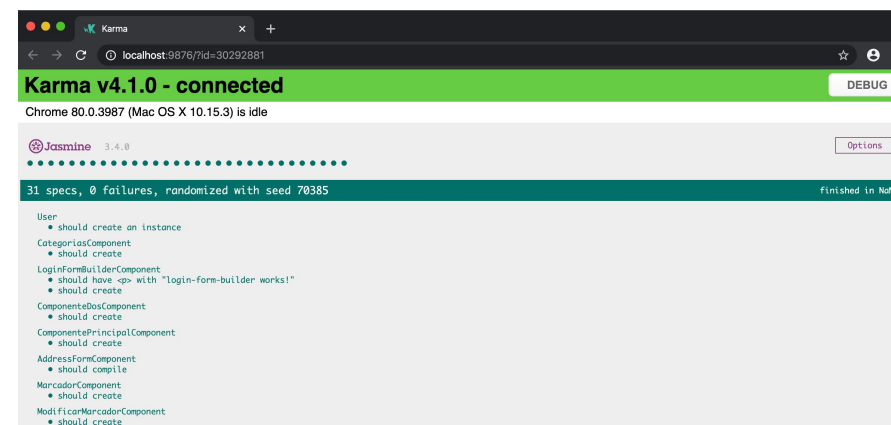
Angular CLI descarga e instala todo lo necesario para testear la aplicación con el framework de *Jasmine*. Para realizar un test de nuestra aplicación:

ng test

Al introducir este comando Angular realizará un testeo de toda la aplicación e irá recorriendo los diferentes *.spec.ts* de cada clase.

```
06 03 2020 10:34:24.502:WARN [karma]: No captured browser, open http://localhost:9876/
06 03 2020 10:34:24.536:INFO [Chrome 80.0.3987 (Mac OS X 10.15.3)]: Connected on socket Kso_QN7yZLDEc4PjAAAA with id 30292881
WARN: 'The "slide" event cannot be bound because Hammer.JS is not loaded and no custom loader has been specified.'
Chrome 80.0.3987 (Mac OS X 10.15.3): Executed 11 of 31 SUCCESS (0 secs / 0.43 secs)
WARN: 'The "slideend" event cannot be bound because Hammer.JS is not loaded and no custom loader has been specified.'
Chrome 80.0.3987 (Mac OS X 10.15.3): Executed 11 of 31 SUCCESS (0 secs / 0.43 secs)
WARN: 'The "slidestart" event cannot be bound because Hammer.JS is not loaded and no custom loader has been specified.'
Chrome 80.0.3987 (Mac OS X 10.15.3): Executed 11 of 31 SUCCESS (0 secs / 0.43 secs)
WARN: 'The "longpress" event cannot be bound because Hammer.JS is not loaded and no custom loader has been specified.'
Chrome 80.0.3987 (Mac OS X 10.15.3): Executed 16 of 31 SUCCESS (0 secs / 0.518 secs)
Chrome 80.0.3987 (Mac OS X 10.15.3): Executed 31 of 31 SUCCESS (1.009 secs / 0.884 secs)
TOTAL: 31 SUCCESS
TOTAL: 31 SUCCESS
TOTAL: 31 SUCCESS

===== Coverage summary =====
Statements : 75.29% ( 128/170 )
Branches   : 5.88% ( 1/17 )
Functions  : 68.54% ( 61/89 )
Lines      : 70% ( 98/140 )
```





red.es

PROFESIONALES
DIGITALES

Formación
Continua



UNIÓN EUROPEA

El FSE invierte en tu futuro

Integración continua

Los servicios de integración continua nos ayudan a mantener nuestro proyecto libre de bugs. Enlazando el repositorio del proyecto con alguna de las herramientas de integración continua se realizarán los test cada vez que se haga un *commit* y un *pull request*.

Las herramientas recomendadas por Angular son Circle CI, Travis CI o Jenkins

Informe de cobertura

Angular CLI puede ejecutar pruebas unitarias y crear informes de cobertura. Estos informes muestran cualquier parte de nuestro código que no pueda ser probada adecuadamente por las pruebas unitarias. Para generar el informe:



red.es

PROFESIONALES
DIGITALES

Formación
Continua



UNIÓN EUROPEA

El FSE invierte en tu futuro

```
ng test --code-coverage
```

Este comando creará una carpeta llamada *coverage* en la raíz del proyecto. Al abrir el *index.html* que está dentro de esta carpeta se podrán ver los resultados. Si queremos que se genere automáticamente un informe en cada test debemos de activar esta opción en el *angular.json*:

```
"test": {  
  "options": {  
    "codeCoverage": true  
  }  
}
```



red.es

PROFESIONALES
DIGITALES

Formación
Continua



UNIÓN EUROPEA

El FSE invierte en tu futuro

Jasmine

Para hacer los test en Angular se suele usar Jasmine. Jasmine es un framework Javascript, para la definición de test usando un lenguaje natural entendible por todo tipo de personas.

Un ejemplo de test sería:

```
describe("A suite name", function() {  
  it("contains spec with an expectation", function() {  
    expect(true).toBe(true);  
  });  
});
```

- **describe:** Define una colección de tests. Ésta función recibe dos parámetros, un string con el nombre de la colección y una función donde definiremos los tests.
- **it:** Define un test. Recibe como parámetro el nombre del test y una función a ejecutar por el test.
- **expect:** Lo que espera recibir el test. Es decir, con expect hacemos la comprobación del test. Si la comprobación no es cierta el test falla. En el ejemplo anterior se comprueba si *true* es *true*, por lo que el test pasa.

El FSE invierte en tu futuro

```
describe('Hello world', () => {  
  
  let expected = "";  
  
  beforeEach(() => {  
    expected = "Hello World";  
  });  
  
  afterEach(() => {  
    expected = "";  
  });  
  
  it('says hello', () => {  
    expect(helloWorld())  
      .toEqual(expected);  
  });  
});
```

```
import { async, ComponentFixture, TestBed } from  
  '@angular/core/testing';  
  
import { NotesComponent } from './notes.component';  
  
describe('NotesComponent', () => {  
  let component: NotesComponent;  
  // Esta variable nos añadirá más información para que sea más fácil  
  el testeo.  
  let fixture: ComponentFixture<NotesComponent>;  
  
  beforeEach(async(() => {  
    TestBed.configureTestingModule({  
      declarations: [ NotesComponent ]  
      // Si tuviéramos algún servicio o dependencia debemos colocarla  
      aquí también como si de un módulo se tratara.  
    })  
    .compileComponents();  
  }));  
  
  beforeEach(() => {  
    fixture = TestBed.createComponent(NotesComponent);  
    component = fixture.componentInstance;  
    fixture.detectChanges();  
  });  
  
  it('should create', () => {  
    expect(component).toBeTruthy();  
  });  
});
```

```
import {LoginComponent} from './login.component';  
import {AuthService} from './auth.service';  
  
describe('Login component', () => {  
  
  let component: LoginComponent;  
  let service: AuthService;  
  
  beforeEach(() => {  
    service = new AuthService();  
    component = new LoginComponent(service);  
  });  
  
  afterEach(() => {  
    localStorage.removeItem('token');  
    service = null;  
    component = null;  
  });  
  
  it('canLogin returns true when the user is authenticated', () => {  
    localStorage.setItem('token', '12345');  
    expect(component.isLoggedIn()).toBeTruthy();  
  });  
  
});
```



red.es

PROFESIONALES
DIGITALES

Formación
Continua

El FSE invierte en tu futuro



UNIÓN EUROPEA

Spy de Jasmine

Jasmine ofrece también la posibilidad de coger una clase y devolver directamente lo que nos interese sin tener que ejecutar internamente sus métodos.

Con la función *spyOn* se puede hacer que el servicio devuelva directamente *true* en la llamada al nombre de función que le pasemos como parámetro al spy.

```
import {LoginComponent} from './login.component';
import {AuthService} from './auth.service';

describe('Component: Login', () => {

  let component: LoginComponent;
  let service: AuthService;
  let spy: any;

  beforeEach(() => {
    service = new AuthService();
    component = new LoginComponent(service);
  });

  afterEach(() => {
    service = null;
    component = null;
  });

  it('canLogin returns true when the user is authenticated', () => {
    spy = spyOn(service, 'isAuthenticated').and.returnValue(true);
    expect(component.isLogged()).toBeTruthy();
  });
});
```




red.es

PROFESIONALES
DIGITALES

Formación
Continua



UNIÓN EUROPEA

El FSE invierte en tu futuro

Accediendo a la vista

Para acceder a los elementos html de la vista de un componente, podemos usar su *fixture*.

```
describe('Component: Login', () => {  
  
  let component: LoginComponent;  
  let fixture: ComponentFixture<LoginComponent>;  
  let submitButton: DebugElement;  
  
  beforeEach(() => {  
  
    TestBed.configureTestingModule({  
      declarations: [LoginComponent]  
    });  
  
    // create component and test fixture  
    fixture = TestBed.createComponent(LoginComponent);  
  
    // get test component from the fixture  
    component = fixture.componentInstance;  
  
    submitButton = fixture.debugElement.query(By.css('button_submit'));  
  
  });  
});
```