

1 - Pruebas usando TestBed

Pruebas de Angular *con Angular*

5 - Probando componentes aparentemente simples

Issue: Componentes aparentemente simples

Testbed

Cualquier componente simple, suele requerir integración.

Al pedirle a Angular que compile un componente...

... necesita sus dependencias de código y vista.

Es decir su **módulo**

S.U.T: AboutComponent

```
export class AboutComponent implements OnInit {  
  title = 'Angular Budget';  
  constructor() {}  
  
  ngOnInit(): void {}  
}
```

Test: AboutComponent - spec

```
describe('GIVEN: the AboutComponent in a TesBed', () => {
  let component: AboutComponent;
  let fixture: ComponentFixture<AboutComponent>;
  beforeEach(async () => {
    // Arrange
    await TestBed.configureTestingModule({
      imports: [SharedModule], // lo necesitamos para la vista
      declarations: [AboutComponent],
    }).compileComponents();
  });
  beforeEach(() => {
    fixture = TestBed.createComponent(AboutComponent);
    component = fixture.componentInstance;
    fixture.detectChanges(); // simulación del comportamiento
  });
  it('WHEN ask for title THEN equals Angular Budget', () => {
    // Act
    const actual = component.title;
    // Assert
    const expected = 'Angular Budget';
    expect(actual).toEqual(expected);
  });
});
```

Ejercicio

corregir `app.component.spec.ts`

GIVEN: the AppComponent

WHEN: starts

THEN: should be created

6 - Pipes con dependencias integradas

Issue: Prueba de integración de un pipe con TestBed

🧪 Pruebas no invasivas

En ciertos casos las pruebas de integración son las más prácticas

S.U.T: TimeAgoPipe

```
export class TimeAgoPipe implements PipeTransform {  
    constructor(private util: UtilService) {}  
  
    transform(fecha: Date, ...args: unknown[]): string {  
        return this.util.getFechaColoquial(fecha);  
    }  
}
```

Test: TimeAgoPipe - spec

```
describe('GIVEN the TimeAgoPipe', () => {  
  let timeAgoPipeSUT: TimeAgoPipe;  
  beforeEach(() => {  
    // Arrange  
    TestBed.configureTestingModule({ providers: [UtilService] });  
    const utilService = TestBed.inject<UtilService>(UtilService);  
    // es una prueba de integración, porque usa el servicio real como colaborador  
    timeAgoPipeSUT = new TimeAgoPipe(utilService);  
  });  
  it('WHEN called with an ancient date SHOULD generate a long time ago message', () => {  
    // Act  
    const inputAncientDate = new Date(2000, 1, 1);  
    const actual = timeAgoPipeSUT.transform(inputAncientDate);  
    // Assert  
    const expected = 'hace mucho tiempo';  
    // ⚠ dependemos de que el servicio esté bien programado  
    expect(actual).toBe(expected);  
  });  
});
```


Ejercicio

Refactorizar usando inyección de dependencias

GIVEN: the TimeAgoPipe
WHEN: called with an ancient date
THEN: generate a long time ago message

PISTA: providers: [{ provide: UtilService, useClass: UtilService }]

7 - Componentes complejos aislados

Issue: Componentes complejo aislado

 TestBed para configurar y compilar

 Cero dependencias mediante inversión del control 🙄

 CUSTOM_ELEMENTS_SCHEMA   NO_ERRORS_SCHEMA

Usando capacidades y trucos propios de Angular A

S.U.T: HomeComponent

```
export class HomeComponent implements OnInit {  
  loaded = false;  
  constructor(private dataService: DataService, private logicService: LogicService) {}  
  ngOnInit(): void {  
    this.loadData(); // async...  
  }  
}
```

Test: HomeComponent - spec

```
fdescribe('GIVEN the HomeComponent', () => {
  let homeComponentSUT: HomeComponent;
  let fixture: ComponentFixture<HomeComponent>;
  beforeEach(async () => {
    // Arrange
    await TestBed.configureTestingModule({
      schemas: [CUSTOM_ELEMENTS_SCHEMA],
      declarations: [HomeComponent],
      providers: [ // Inversión del control
        {
          provide: DataService,
          useValue: jasmine.createSpyObj('DataService', {
            getProjects$: of([]),
            getTasks$: of([]),
            getTransactions$: of([]),
          }),
        },
        {
          provide: LogicService, // Aislado de cualquier dependencia
          useValue: jasmine.createSpyObj('LogicService', {
            composeProjectViews: [], // Demostrar lanzando error en origen
            composeTasksView: {},
          }),
        },
      ],
    }).compileComponents();
  });
  ...
}
```



```
...
beforeEach(() => {
  // Arrange
  fixture = TestBed.createComponent(HomeComponent);
});
it('WHEN instantiated THEN should knows it is not loaded', () => {
  // Act : Implícito, objeto construido pero no inicializado
  // Assert -> prueba de estado indirecta
  homeComponentSUT = fixture.componentInstance;
  const actual = homeComponentSUT.loaded;
  expect(actual).toBeFalsy();
});
it('WHEN initialized THEN should knows it is loaded', () => {
  // Act
  fixture.detectChanges(); // Llama a ngOnInit()
  // Assert -> prueba de estado indirecta
  homeComponentSUT = fixture.componentInstance;
  const actual = homeComponentSUT.loaded;
  expect(actual).toBeTruthy();
});
});
```

Ejercicio



```
GIVEN: the ProjectsComponent  
WHEN: instantiated  
THEN: should knows it is not loaded  
WHEN: instantiated  
THEN: should knows it is loaded
```

Recomendaciones

Usar patrón Container Presenter

-  Container : probar **lógica**
-  Presenter : probar **vista**

Arquitectura de Servicios

-  Adapters : dependencias **controladas**
-  Facades : dependencias **unificadas**

👁️ Valorar librerías de terceros

- [Spectator](#)
- [Auto-Spies](#)
- [Observer-Spy](#)

Repositorio: [angularbuilders/angular-budget/test_1_test-bed](#)

By [Alberto Basalo](#)