# 0 - Introducción al testing con angular

## Pruebas de Angular sin Angular

# 0 - Karma y Jasmine

- preconfigurado
- autogenerado
- scriptable
- describe it
- it should
- given when then

# snippets

```
{
  "Jasmine Should": {
    "prefix": "ab-jsm-is",
    "body": [
      "describe('$1', () => {",
      "  beforeEach(() => {});",
      "  it('SHOULD $2', () => {",
      "    const sut = null;",
      "    const actual = null;",
      "    const expected = null;",
      "    expect(actual).toEqual(expected);",
      "  });",
      "});",
    ],
    "description": "Esqueleto It Should con Jasmine"
  },
  "Jasmine Given When Then": {
    "prefix": "ab-jsm-gwt",
    "body": [
      "describe('GIVEN: $1', () => {",
      "  beforeEach(() => {});",
      "  it('WHEN $2 THEN $3', () => {",
      "    const sut = null;",
      "    const actual = null;",
      "    const expected = null;",
      "    expect(actual).toEqual(expected);",
      "  });",
      "});",
    ],
    "description": "Esqueleto GWT con Jasmine"
  },
}
```

# 1 - Probando un servicio como una clase

## Issue: Testing minimalista del LogicService

Probar un servicio con métodos de lógica de negocio.

Sin dependencias.

Métodos *puros*.

## S.U.T: LogicService

```
export class LogicService {
  public slugify(text: string): string {
    return text
      .toLowerCase()
      .trim()
      .replace(/[\s\W-]+/g, '-');
  }
}
```

## Test: LogicService - Test

```
describe('GIVEN: the slugify method', () => {
  it('WHEN receives Angular 10.1 THEN returns angular-10-1', () => {
    const sut = new LogicService();
    const actual = sut.slugify('Angular 10.1');
    const expected = 'angular-10-1';
    expect(actual).toEqual(expected);
  });
});
```

# 2 - Probando un componente como una clase

## Issue: Testing minimalista de un componente

Probar un componente con propiedades de datos.

Sin dependencias.

Sin importar *la presentación*.

## S.U.T: AboutPage

```
export class AboutPage implements OnInit {
  title = 'Angular Budget';
  constructor() {}

  ngOnInit(): void {}
}
```

## Test: AboutPage - Test

```
describe('GIVEN: the AboutComponent', () => {
  beforeEach(() => {});
  it('WHEN Ask for title THEN equals Angular Budget', () => {
    // Arrange
    const sut = new AboutPage();
    // Act
    const actual = sut.title;
    // Assert
    const expected = 'Angular Budget';
    expect(actual).toEqual(expected);
  });
});
```

# 3 - Probando unidades y espiando dependencias

## Issue: Pruebas de un servicio con dependencias usando espías

Probar un servicio con dependencias (Title).

Queremos hacer tests **unitarios**.

Usamos un doble en lugar de la dependencia original.

Con *Jasmine* lo aconsejable es usar un **spy**

## S.U.T: UtilService

```
export class UtilService {
  private siteTitle = 'Angular.Budget';

  constructor(private titleService: Title) {}

  public setDocumentTitle(title: string): void {
    const documentTitle = title ? `${title} | ${this.siteTitle}` : this.siteTitle;
    this.titleService.setTitle(documentTitle);
  }
}
```

## Test: UtilService - Test

```
describe('The UtilsService', () => {
  beforeEach(() => {});
  it('SHOULD set the correct title', () => {
    // Arrange
    const titleServiceSpy = jasmine.createSpyObj('TitleService', ['setTitle']);
    const setTitleSpy: jasmine.Spy = titleServiceSpy.setTitle;
    const stubTitle = 'Pruebas unitarias';
    setTitleSpy.and.returnValue(stubTitle);
    const sut = new UtilService(titleServiceSpy);
    // Act
    sut.setDocumentTitle('Pruebas unitarias');
    const actual = setTitleSpy.calls.mostRecent().returnValue;
    // Assert
    const expected = 'Pruebas unitarias';
    expect(actual).toEqual(expected);
  });
});
```

# 4 - Probando código asíncrono

## Issue: Prueba de un servicio asíncrono

Probar un servicio con dependencias asíncrona (HttpClient).

Los tests tienen ser **asíncronos**.

Podemos probar:

- la llamada
- la subscripción

Jasmine Asynchronous Work

## S.U.T: DataService

```
export class DataService {
  private rootUrl = `https://api-base.herokuapp.com/api/pub`;

  constructor(private httpClient: HttpClient) {}

  getProjects$(): Observable<Project[]> {
    return this.httpClient.get<Project[]>(`${this.rootUrl}/projects`);
  }
}
```

## Test: DataService - Test

```
describe('GIVEN: A dataService', () => {
  let httpClientSpy: any;
  let getSpy: jasmine.Spy;
  beforeEach(() => {
    // Arrange
    httpClientSpy = jasmine.createSpyObj('HttpClient', ['get']);
    getSpy = httpClientSpy.get;
    const stubPojects = [];
    getSpy.and.returnValue(of(stubPojects));
  });
```

```
it('WHEN call the getProjects THEN the url is the expected', () => {
  // Act
  const sut = new DataService(httpClientSpy);
  sut.getProjects$().subscribe();
  // Assert
  const actual = getSpy.calls.mostRecent().args[0];
  const expected = 'https://api-base.herokuapp.com/api/pub/projects';
  expect(actual).toEqual(expected);
});
```

```
it('WHEN call the getProjects THEN returns an observable of empty projects list', () => {
  // Act
  const sut = new DataService(httpClientSpy);
  let actual = null;
  sut.getProjects$().subscribe({
    next: data => (actual = data),
  });
  // Assert
  const expected = [];
  expect(actual).toEqual(expected);
});
});
```

> **Repositorio:** angularbuilders/angular-budget/test_0_intro
>
> > By Alberto Basalo