AlmavivA
digitale assoluto

Test

Fravezzi Mattia
m.fravezzi@almaviva.it

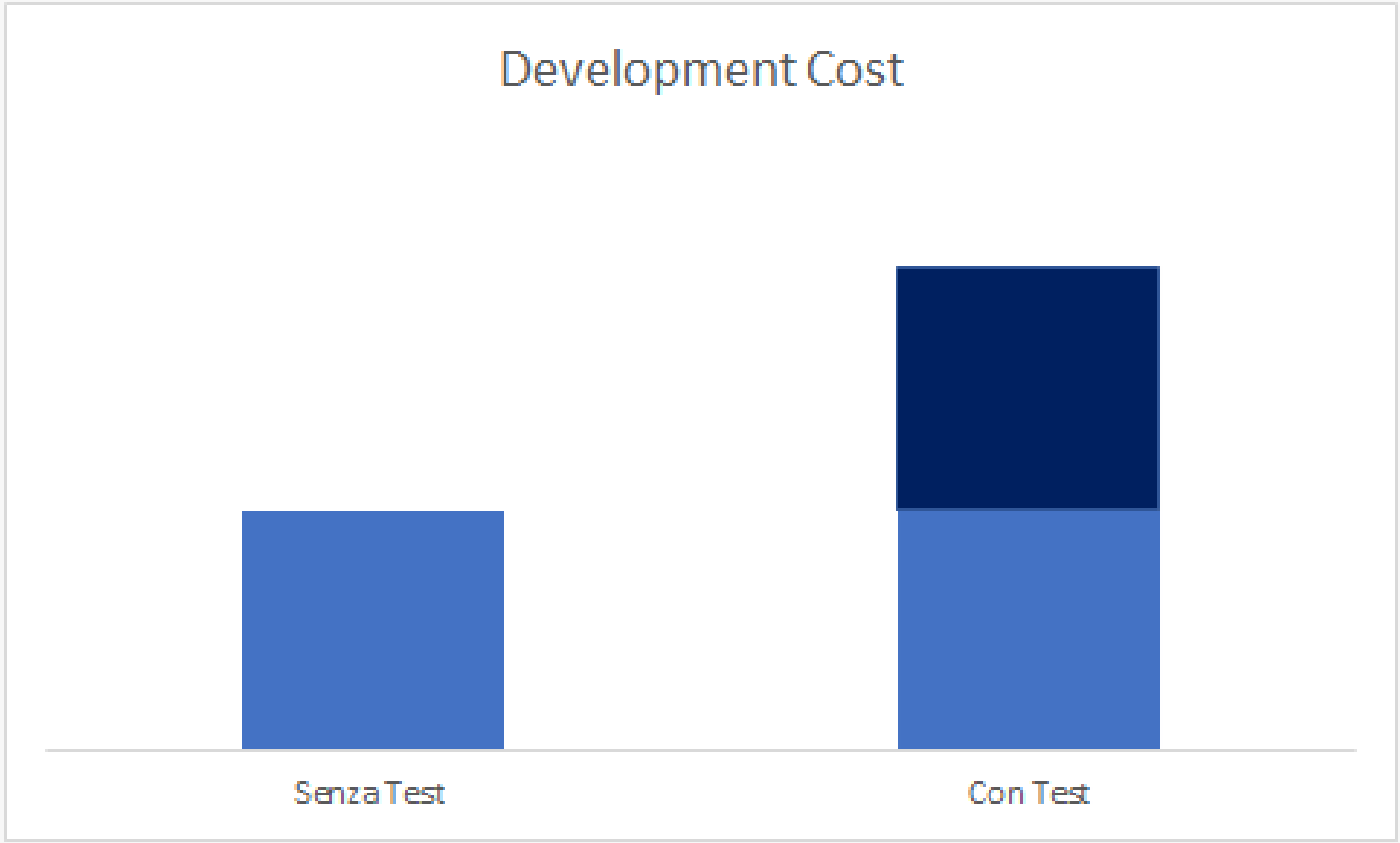
COSA SONO I TEST AUTOMATICI?

```
public calculate(inputs) {
    if (x) return ...
    else if (y) return ...
    return ...
}
```

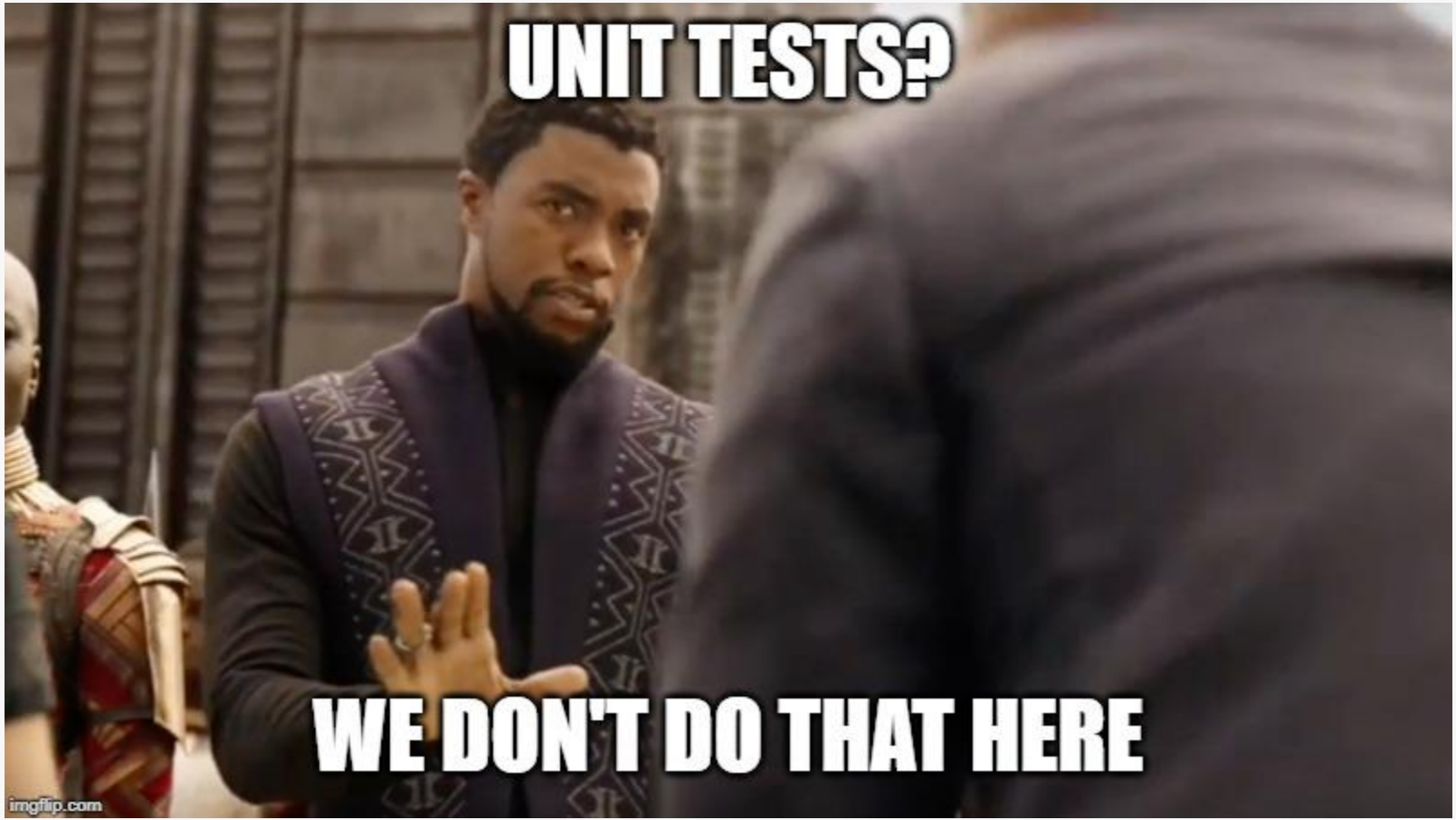
```
const result1 = calculate(x);

const result2 = calculate(y);

assert(result1 === 'pippo');
assert(result2 === 'pluto');
```

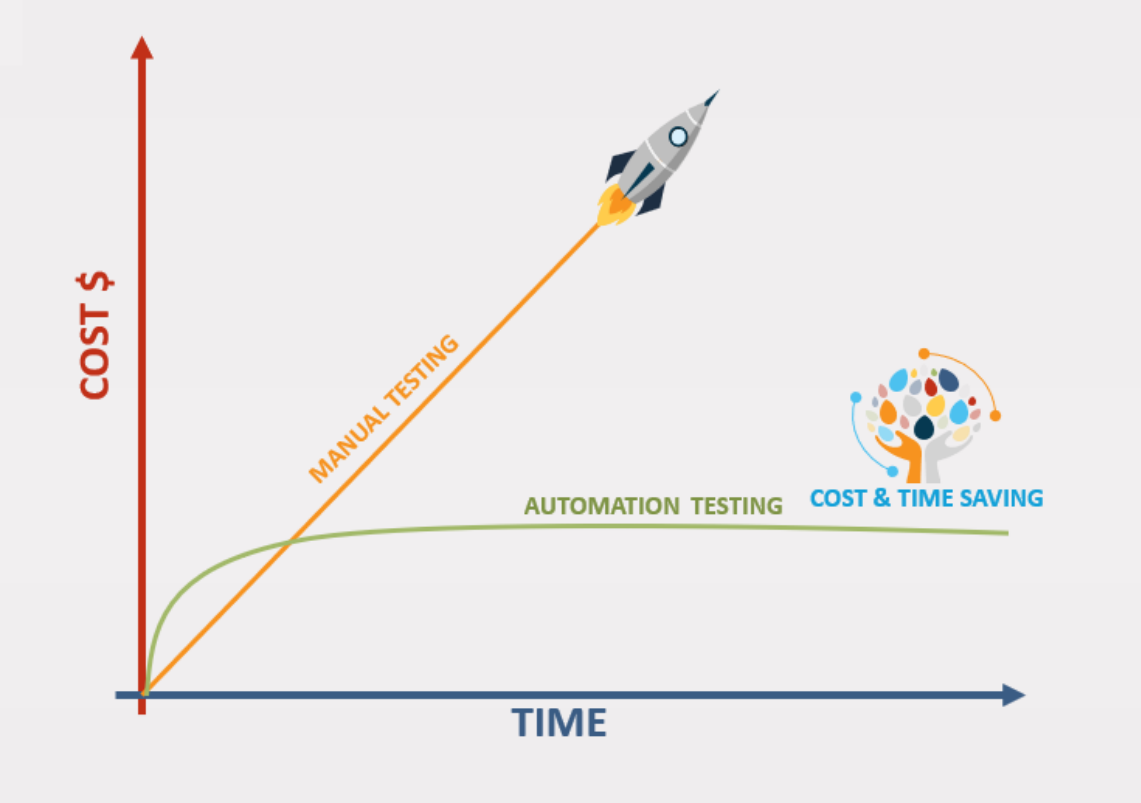

Development Cost
Senza Test
Con Test

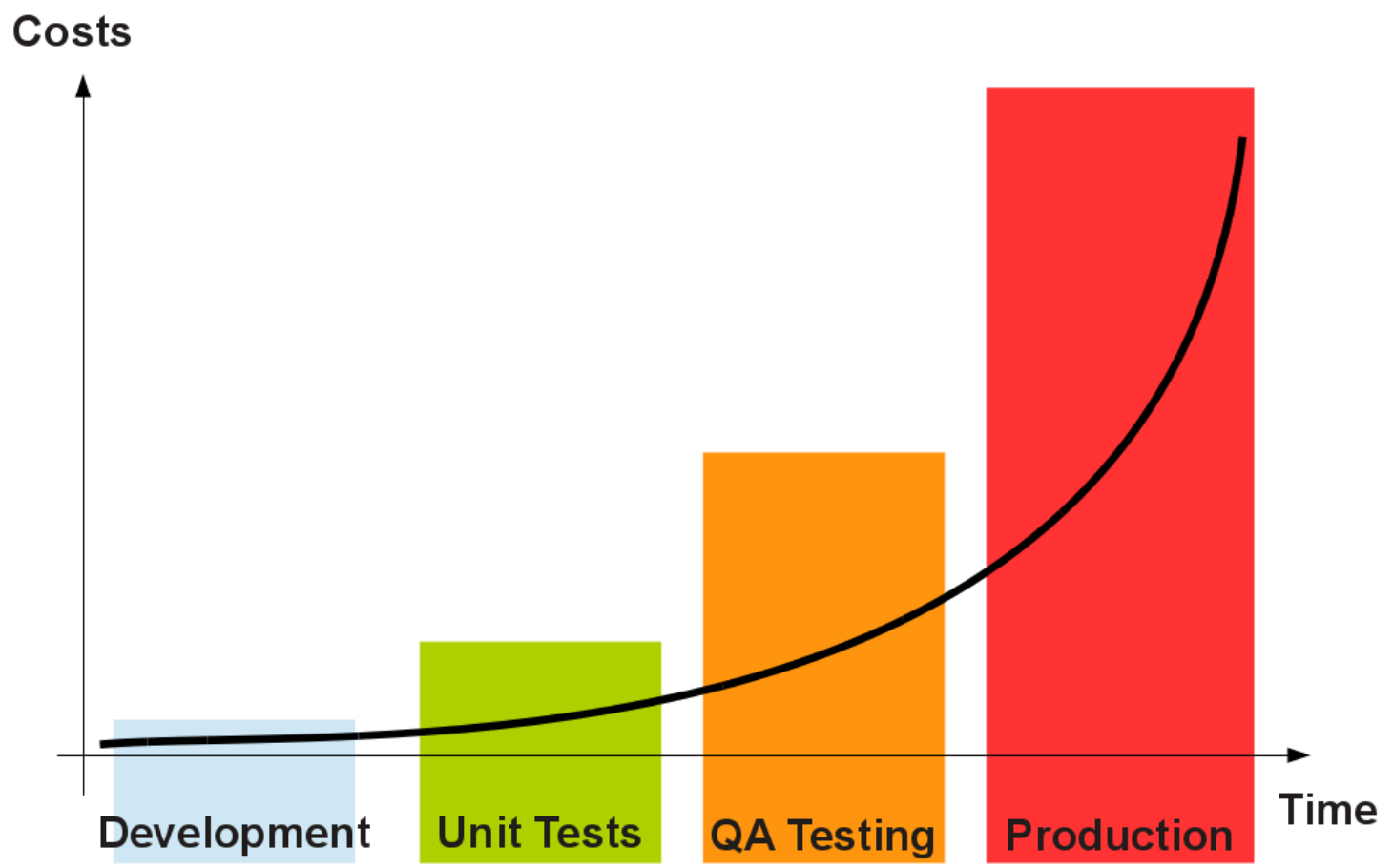
DEVO VERAMENTE FARLI?


UNIT TESTS?
WE DON'T DO THAT HERE
imgflip.com

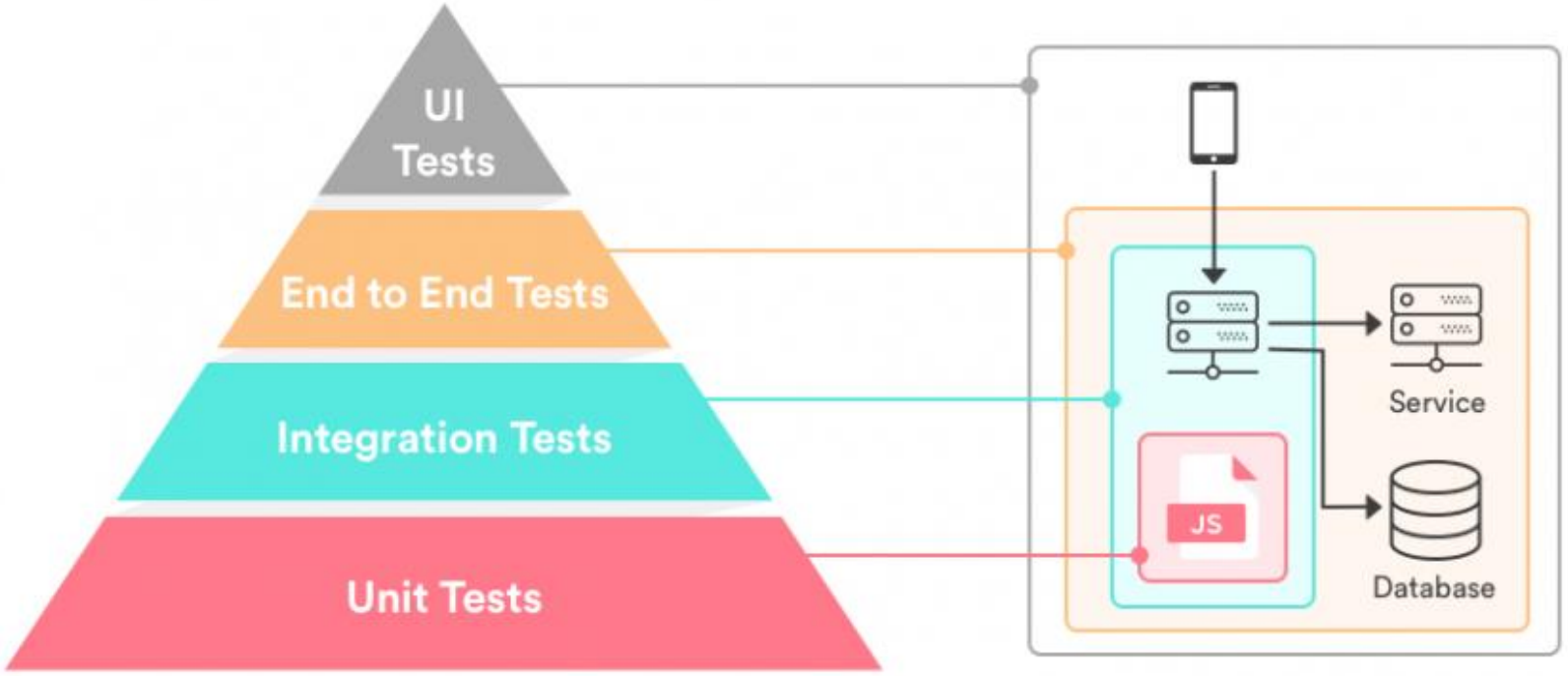
Vantaggi

- Riduce i bug in una funzionalità nuova o esistente
- Incrementa la qualità del codice
- I test documentano il codice
- Si può aggiornare il sistemare senza paura di romperlo
- Riduce il costo del cambiamento
- Facilità il refactoring
- Rimuove le regressioni
- Aiuta ad aggiornare librerie di terze parti
- **SVILUPPATORI PIU' FELICI**


COST $
MANUAL TESTING
AUTOMATION TESTING
COST & TIME SAVING
TIME


Costs
Development
Unit Tests
QA Testing
Production
Time

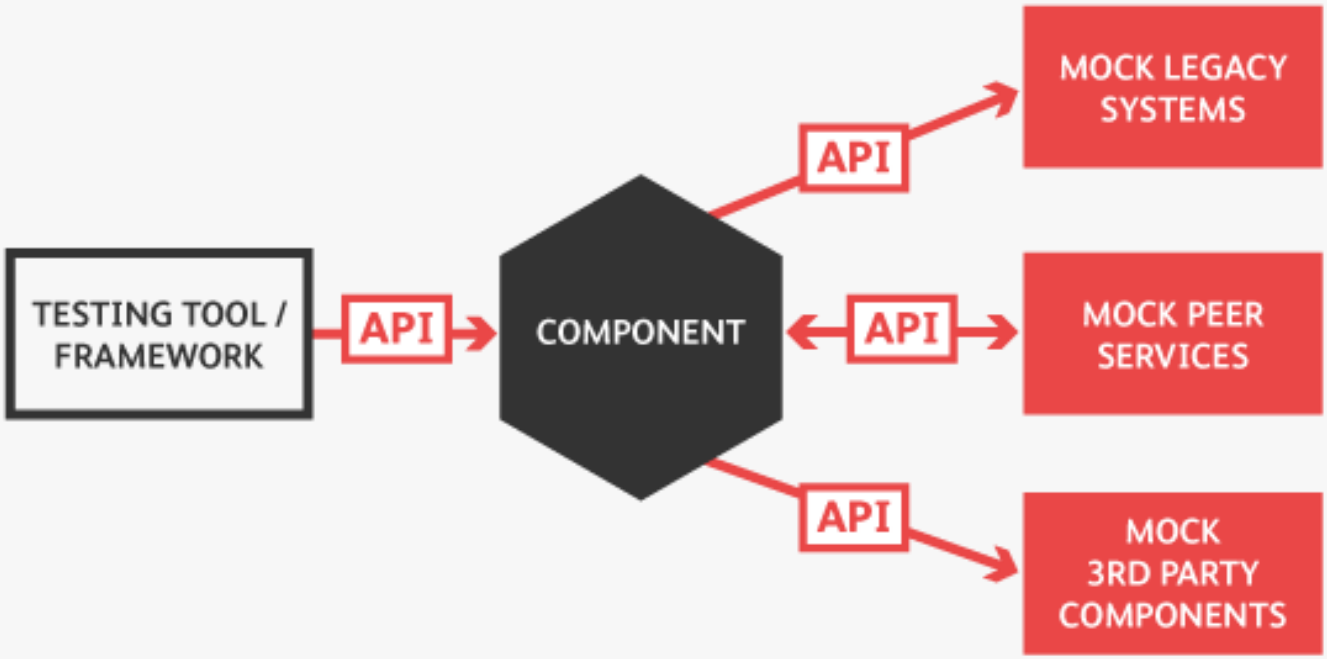

TEST...
EVERYWHERE
makeameme.org


UI Tests
End to End Tests
Integration Tests
Unit Tests
Service
JS
Database

Tests - Unit Test

- Testa la più piccola unità di codice possibile
- Obbliga a scrivere codice flessibile, manutenibile, riusabile
- Non importa come funzionano diverse unità tra di loro
- Verifica solo poche assertions
- Facili da fare
- Veloce
- Affidabile

Tests - Unit Test

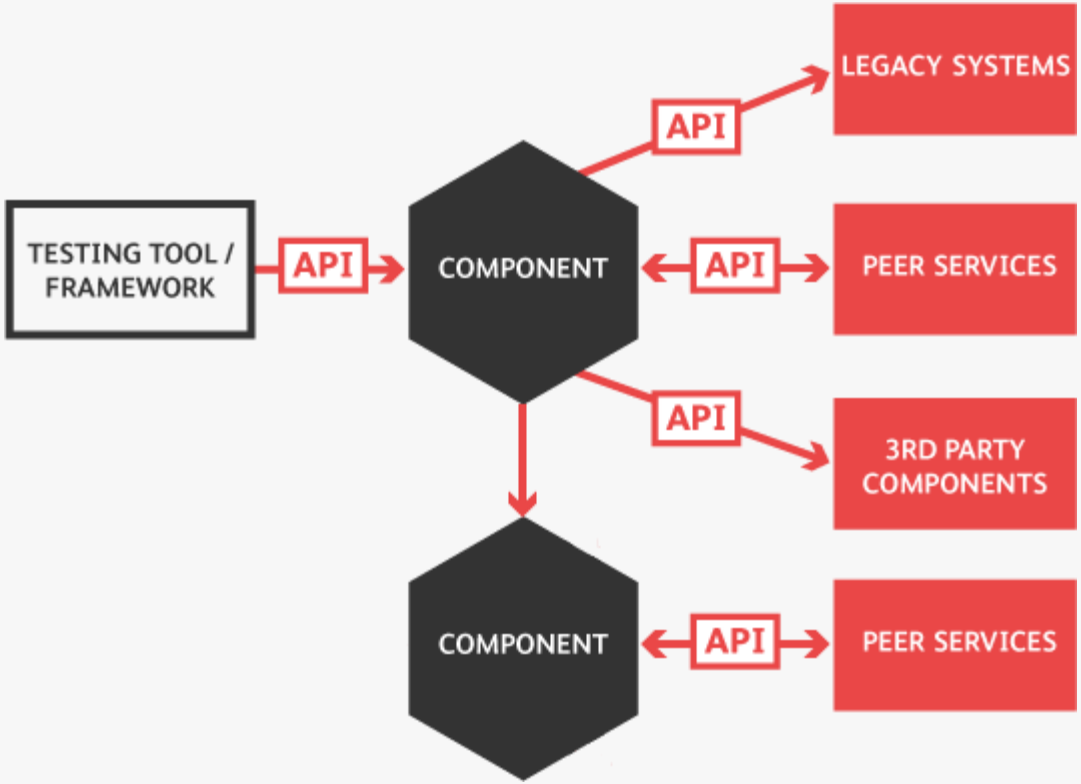



Unit test vs. Integration test

Tests - Integration Test

- Copre diverse unità di codice
- Non mocka servizi o altri componenti di terze parti
- Previene modifiche dei dati nel trasferimento da un’unità all’altra
- Non verifica un flusso di lavoro completo (mock api).
- Facili da integrare su build giornaliere e da testare in locale
- Aumenta il code coverage

Tests - Integration Test



Tests - End-to-end Test

- Simula il flusso completo
- Automatizza test utente, interazione utente-browser
- Possibilità di testare su diversi browser (IE)
- Aumenta il test coverage
- Riduce il time-to-market
- Rileva bug associati al sottosistema (api etc)
- Fragili e Lenti

Tool Angular Testing

- Jasmine
- Karma
- Protractor

Jasmine

Karma

Protractor

Protractor

end to end testing for Angular

Code Coverage

```
"coverage": "ng test --no-watch --code-coverage"
```

All files

82.22% Statements 37/45 **33.33%** Branches 3/9 **100%** Functions 14/14 **80%** Lines 32/40

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

File	Statements		Branches		Functions		Lines	
src	100%	3/3	100%	0/0	100%	0/0	100%	3/3
src/app	61.9%	13/21	20%	1/5	100%	4/4	55.56%	10/18
src/app/pipes	100%	6/6	50%	1/2	100%	3/3	100%	5/5
src/app/services	100%	15/15	50%	1/2	100%	7/7	100%	14/14

AlmavivA

AAA Pattern

```
it('transform should return faked correct value from a mock math service', () => {
        //Arrange
        const pipe = new SumPipe(new MockMathService());

        //Act
        const value = pipe.transform(5, 2);

        //Assert
        expect(value).toBe(25);
});
```

AlmavivA

Unit Test File

```
export class MathService {
    public sum(x: number, y: number) {
        return x + y;
    }
    public difference(x: number, y: number) {
        return x - y;
    }
    public product(x: number, y: number) {
        return x * y;
    }
    public quotient(x: number, y: number) {
        return x / y;
    }
}
```

```
describe('MathService', () => {
    let service: MathService;
    beforeEach(() => {
        TestBed.configureTestingModule({});
        service = TestBed.get(MathService);
    });
    it('should be created', () => {
        expect(service).toBeTruthy();
    });
    it('sum should return correct value', () => {
        expect(service.sum(2, 3)).toBe(5);
    });
    ...
});
```

AlmavivA

Unit Test File

```
beforeEach(async(() => {
        TestBed.configureTestingModule({
            imports: [
                BrowserModule,
                AppRoutingModule,
                FormsModule,
                MatInputModule
            ],
            declarations: [ AppComponent, SumPipe ],
            providers: [ { provide: MathService, useClass: MockMathService }]
        }).compileComponents();
        fixture = TestBed.createComponent(AppComponent);
        mathService = TestBed.get(MathService);
        component = fixture.debugElement.componentInstance;
    }));
```

AlmavivA

Unit Test File

```
it('should render title in a h1 tag', () => {
        expect(nativeElement.querySelector('h1').textContent).toContain('La mia calcolatrice');
});

it('should 4 in component variable num2', () => {
        const num2Input = nativeElement.querySelector('#num2');
        num2Input.value = 4;
        num2Input.dispatchEvent(new Event('input'));

        expect(+component.num2).toEqual(4);
});
```

AlmavivA

Unit Test File

```
it('should render result', () => {
        component.result = 10;
        fixture.detectChanges();
        fixture.whenStable()
            .then(() => {
                const resultInput = nativeElement.querySelector('#result');
                expect(+resultInput.value).toEqual(10);
            });
    });
    it('should result evaluted', () => {
        component.num1 = 4;
        component.num2 = 7;
        component.operator = 1;
        component.calculate();
        expect(+component.result).toEqual(11);
    });
```

File spec vicino al file da testare

- Sono facili da trovare.
- Vedi a colpo d'occhio se una parte della applicazione manca di test.
- Possono rivelare come funziona una parte nel contesto.
- Quando sposti la sorgente, ti ricordi di spostare il test
- Quando rinomini il file sorgente, ti ricordi di rinominare il file di test.

Unit Test con dipendenze

- Creare un oggetto e utilizzarlo come se fosse il servizio reale
- Creare un servizio mock
- Creare un servizio con Jasmine Spies

Unit Test - mock object

```
it('transform should return faked correct value from a fake object', () => {
        const fake: any = { pow: () => 8 };
        const pipe = new SumPipe(fake as MathService);

        expect(pipe.transform(2, 3)).toBe(8);
 });
```

AlmavivA

Unit Test - mock service

```
it('transform should return faked correct value from a mock math service', () => {
        const pipe = new SumPipe(new MockMathService());

        expect(pipe.transform(5, 2)).toBe(25);
});
```

AlmavivA

Unit Test - Jasmine Spies

```
it('transform should return stubbed value from a spy', () => {
        const mathServiceSpy = jasmine
            .createSpyObj('MathService', ['pow']);

        const stubValue = 9;
        mathServiceSpy.pow.and.returnValue(stubValue);

        const pipe = new SumPipe(mathServiceSpy);

        expect(pipe.transform(3, 2)).toBe(stubValue);
    });
```

Test e2e

```
export class AppPage {
    navigateTo() {
        return browser.get(browser.baseUrl) as Promise<any>;
    }

    getTitleText() {
        return element(by.css('app-root h1')).getText() as Promise<string>;
    }

    getNum1Input(): ElementFinder {
        return element(by.id("num1"));
    }
```

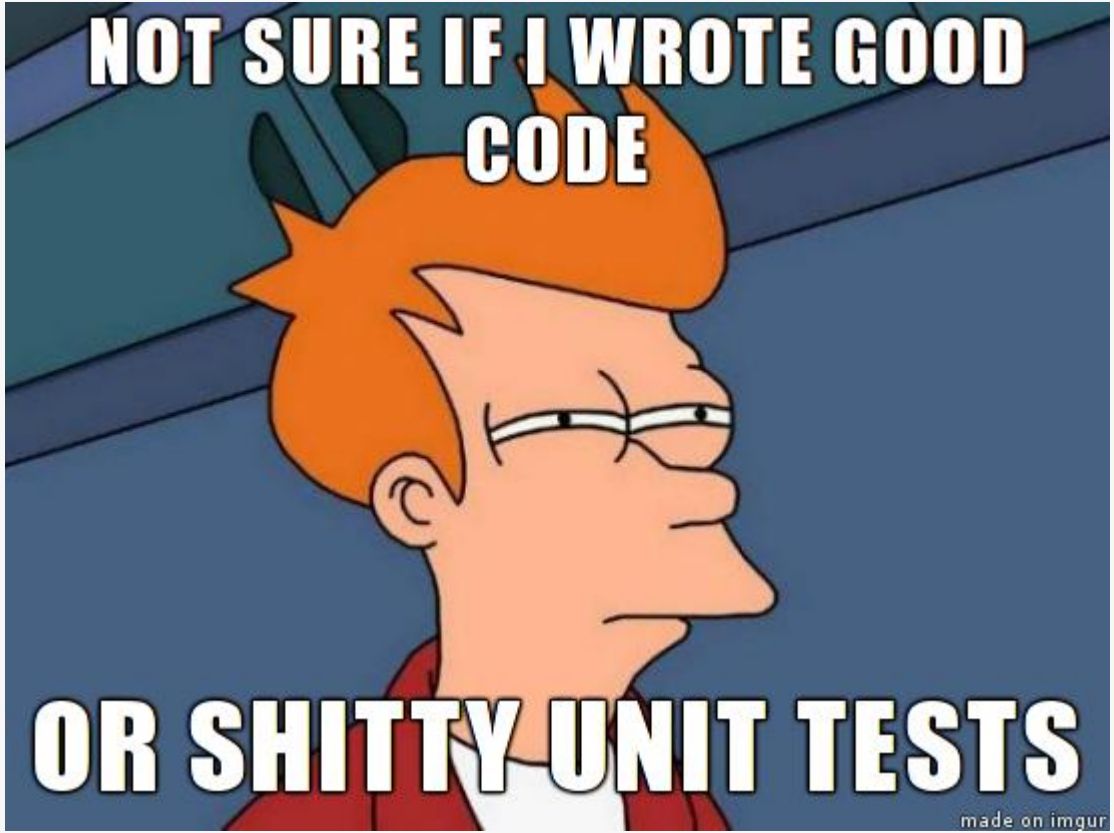
Almaviva

Test e2e

```
it('should display result value', () => {
        page.navigateTo();
        page.getNum1Input().clear()
            .then(() => page.getNum1Input().sendKeys('52') )
        page.getNum2Input().clear()
            .then(() => page.getNum2Input().sendKeys('3') )

        page.getOperatorOptionsProduct().click();
        page.getCalculateButton().click();

        expect(page.getResultValue()).toEqual('156');
    });
```


NOT SURE IF I WROTE GOOD CODE
OR SHITTY UNIT TESTS
made on imgur

DEMO

AlmavivA
digitale assoluto

almaviva.it