



TESTING

Del 2+2 al fetch

ARIANE JURADO DE BILBAO

Soy Ari =)

ENFERMERA

Pediátrica

PSICOPEDAGOGA

Niños con diversidad funcional

PROGRAMADORA

En KairosDS

MAMÁ

De Sebas



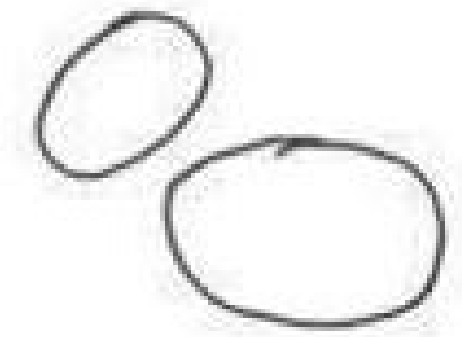
@Ari_Reinventada

DEL 2+2 AL FETCH

NO SOY UNA
EXPERTA

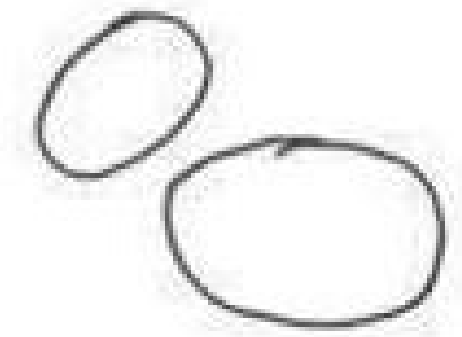
MI PROCESO

```
✓ test('two plus two is four', () => {  
  expect(2 + 2).toBe(4)  
})
```



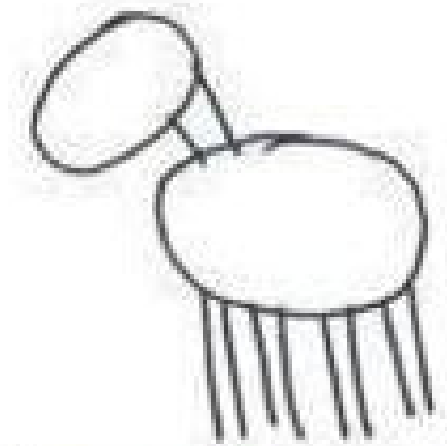
① Dibuja 2 círculos

```
✓ test('two plus two is four', () => {  
  expect(2 + 2).toBe(4)  
})
```



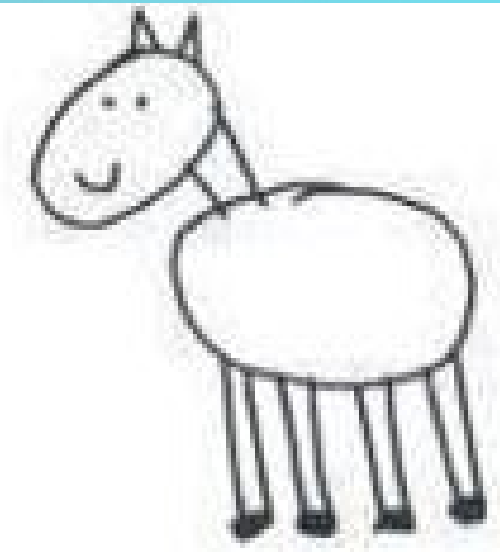
① Dibuja 2 círculos

```
test('adds 1 + 2 to equal 3', () => {  
  expect(sum(1,2)).toBe(3)  
})
```



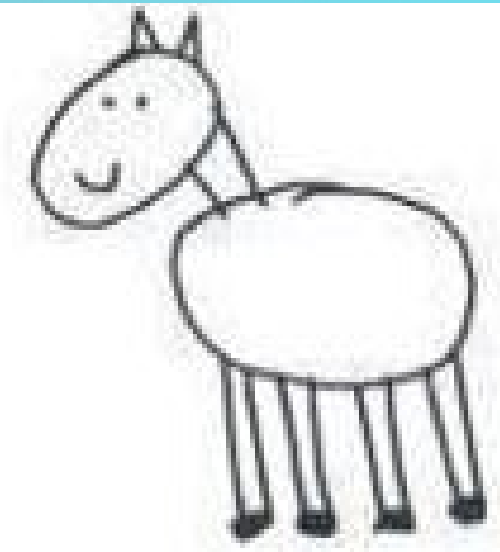
② dibuja las piernas


```
test('firstName + lastName should be John Doe', () => {  
  const firstName = 'John'  
  const lastName = 'Doe'  
  
  expect(firstName + lastName).toBe('John Doe')  
  
})
```



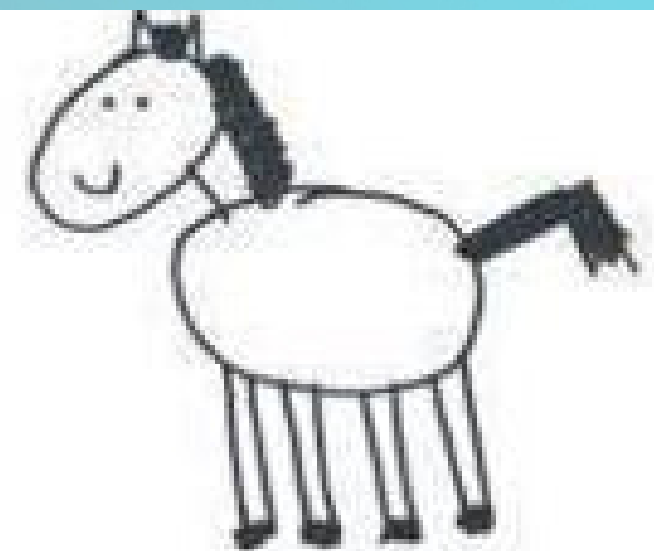
③ Dibuja la cara

```
test('firstName + lastName should be John Doe', () => {  
  const firstName = 'John'  
  const lastName = 'Doe'  
  
  expect(firstName + lastName).toBe('John Doe')  
  
})
```



③ Dibuja la cara

```
test('object assignment', () => {  
  const data = {one: 1};  
  data['two'] = 2;  
  expect(data).toEqual({one: 1, two: 2});  
});
```



④ Dibuja el pelo

```

const startGame = () => {

  const api = `https://raw.githubusercontent.com/Adalab/cards-data/master/${value}.json`;
  fetch(api)
    .then(response => response.json())
    .then(data => {
      list.innerHTML = '';
      for (let i = 0; i < data.length; i++) {
        const item = data[i];
        const { image } = item;
        const imgUp = document.createElement('img');
        imgUp.setAttribute('src', image);
        const showCards = e => {
          const triggerCards = e.currentTarget;

          if (triggerCards.classList.contains('face-down')) {
            imgUp.classList.remove('hidden');
            triggerCards.classList.remove('face-down');
            triggerCards.classList.add('face-up');
          } else {
            imgUp.classList.add('hidden');
            triggerCards.classList.add('face-down');
            triggerCards.classList.remove('face-up');
          }
        };

        imgUp.setAttribute('class', 'face-up');
        imgUp.setAttribute('class', 'hidden');
        const elementUp = document.createElement('li');
        elementUp.setAttribute('class', 'face-down');
        elementUp.appendChild(imgUp);
        list.appendChild(elementUp);

        elementUp.addEventListener('click', showCards);
      }
    });

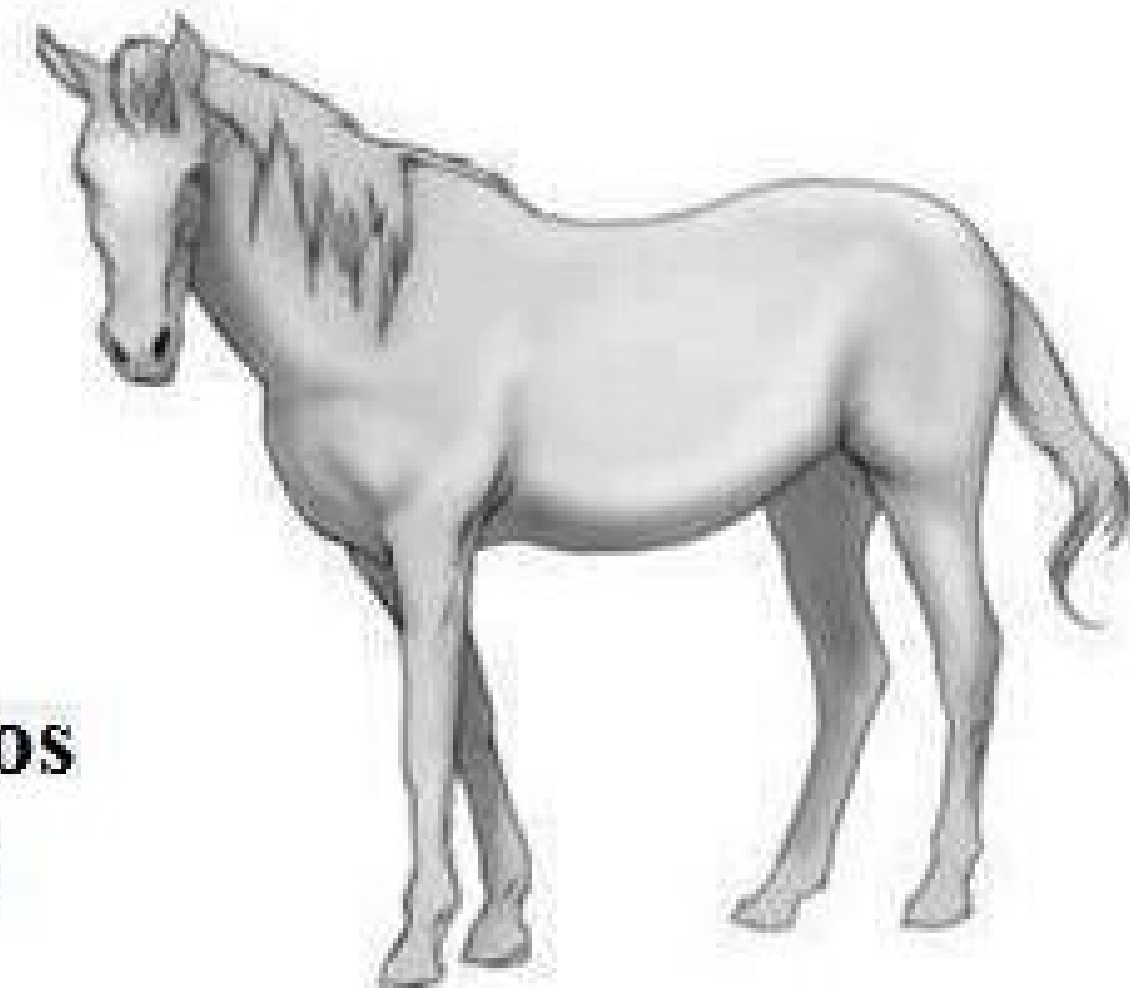
  btn.addEventListener('click', startGame);
};

items.addEventListener('click', chooseCards);

```

5

**Añade
pequeños
detalles**



Tipos de test



UNITARIOS

Una pequeña parte, un elemento básico del software.
Método, función, clase, módulo...

INTEGRACIÓN

Un conjunto de elementos interactuando entre sí.
Comunicación entre ellos

END TO END

Verificar si el software se comporta como se espera.
Suele ser el sistema completo

¿CÓMO SE HACE UN TEST?

```
describe('aquí debe haber una descripción', () => {  
  ✓ test('aquí debe ir una descripción de lo que hace el test', () => {  
    // Arrange - Given  
  
    // Act - When  
  
    // Assert - Then  
  })  
})
```

¿CÓMO SE HACE UN TEST?

```
✓ test("it should filter by a search term (sky)", () => {  
  // Arrange - Given  
  const input = [  
    'Castle in the Sky', 'Grave of the Fireflies', 'My Neighbor Totoro'  
  ];  
  const text = 'sky';  
  const output = ['Castle in the Sky'];  
  
  // Act - When  
  const filterByTitle = index.filterByTitle(input, text)  
  
  // Assert - Then  
  expect(filterByTitle).toEqual(output);  
});
```

¿CÓMO SE HACE UN TEST?

```
describe('LoginService', () => {  
  let service: LoginService;  
  const config = {  
    api: 'http://localhost:8765/api/sso'  
  };  
  beforeEach(() => {  
    TestBed.configureTestingModule({  
      imports: [HttpClientTestingModule],  
      providers: [{ provide: 'config', useValue: config }]  
    });  
    service = TestBed.inject(LoginService);  
  });  
  
  it('should be created', () => {  
    expect(service).toBeTruthy();  
  });  
});
```

DOBBLES

FAKE

```
✓ test("it should filter by a search term (sky)", () => {  
  const input = [  
    'Castle in the Sky', 'Grave of the Fireflies', 'My Neighbor Totoro'  
  ];  
  const text = 'sky';  
  const output = ['Castle in the Sky'];  
  
  const filterByTitle = index.filterByTitle(input, text)  
  
  expect(filterByTitle).toEqual(output);  
});
```

DOBBLES

MOCK

```
const GestorNotas = require('./gestorNotas');  
  
test('Cálculo nota media', () => {  
  const getNotasAlumno = jest.fn();  
  
  getNotasAlumno.mockReturnValue([5, 6, 8, 9]);  
  
  let alumnos = { getNotasAlumno };  
  
  let gestorNotas = new GestorNotas(alumnos);  
  
  expect(gestorNotas.calculaNotaMedia(1)).toBeCloseTo(7);  
  
  expect(alumnos.getNotasAlumno).toBeCalledWith(1);  
});
```



DOBLES

MOCK FUNCIONES

```
const GestorNotas = require('./gestorNotas');  
  
test('Cálculo nota media', () => {  
  const getNotasAlumno = jest.fn();  
  getNotasAlumno.mockReturnValue([5, 6, 8, 9]);  
  let alumnos = { getNotasAlumno };  
  let gestorNotas = new GestorNotas(alumnos);  
  expect(gestorNotas.calculaNotaMedia(1)).toBeCloseTo(7);  
  expect(alumnos.getNotasAlumno).toBeCalledWith(1);  
});
```

DOBLES

MOCK MODULOS

```
const axios = require('axios');  
const Users = require('./users');  
  
jest.mock('axios');  
  
test('should fetch users', async () => {  
  const users = [{ name: 'Bob' }];  
  const resp = { data: users };  
  
  axios.get.mockResolvedValue(resp);  
  
  expect(await Users.all()).toEqual(users);  
});
```



DOBLES

STUB

```
const GestorNotas = require('./gestorNotas');

test('Cálculo nota media', ()=>{

  const getNotasAlumno = jest.fn();

  getNotasAlumno.mockReturnValue([5, 6, 8, 9]);

  let alumnos = { getNotasAlumno };

  let gestorNotas = new GestorNotas(alumnos);

  expect(gestorNotas.calculaNotaMedia(1)).toBeCloseTo(7);

});
```

DOBLES

SPY

```
goToCreatePolicy() {  
  this.router.navigate(['policies/create-policy']);  
}
```

```
✓ it('goToCreatePolicy should call to Router and go to policies/  
create-policy path', () => {  
  const router = TestBed.inject(Router);  
  const spy = spyOn(router, 'navigate');  
  component.goToCreatePolicy();  
  expect(spy).toHaveBeenCalledWith(['policies/create-policy']);  
});
```

DOBBLES

DOM

DOM

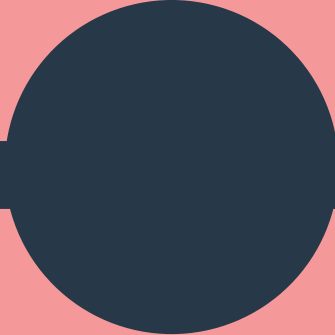
```
function createRestaurantCard(restaurant) {  
  const list = document.querySelector('.restaurant_list');  
  if (list) {  
    const li = document.createElement('li');  
    const name = document.createElement('h2');  
    name.textContent = restaurant.restaurantName;  
    const img = document.createElement('img');  
    img.src = restaurant.restaurantImg;  
    const typeOfFood = document.createElement('p');  
    typeOfFood.textContent = restaurant.kindOfFood;  
    const imgRating = document.createElement('img');  
    imgRating.src = restaurant.rating;  
    li.setAttribute("class", "restaurant_card");  
    name.setAttribute("class", "restaurant_name");  
    img.setAttribute("class", "restaurant_img");  
    typeOfFood.setAttribute("class", "restaurant_kindOfFood");  
    imgRating.setAttribute("class", "rating_img");  
    li.appendChild(img);  
    li.appendChild(name);  
    li.appendChild(typeOfFood);  
    li.appendChild(imgRating);  
    list.appendChild(li);  
  }  
}
```

DOM

```
describe('createRestaurantCard', () => {  
  ✓ test('should create a card for each restaurant', () => {  
    const contextHTML = `

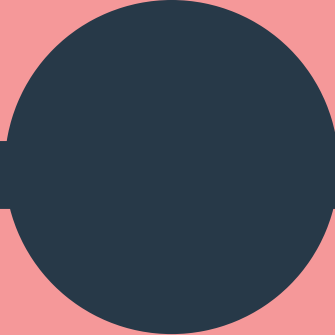

      class="restaurant_card"><h2  
class="restaurant_name">Foster Hollywood</h2><p  
class="restaurant_kindOfFood">Americana</p></li></ul>`  
    document.body.innerHTML = contextHTML;  
    const card = document.querySelector('.restaurant_card');  
    const act = createRestaurantCard(restaurant).innerHTML  
    expect(act).toEqual(card.innerHTML)  
  });  
})
```

TESTS ASÍNCRONOS



```
test('the data is peanut butter', async () => {
  const data = await fetchData();
  expect(data).toBe('peanut butter');
});

test('the fetch fails with an error', async () => {
  expect.assertions(1);
  try {
    await fetchData();
  } catch (e) {
    expect(e).toMatch('error');
  }
});
```



```
function getNameFromServer(callback) {  
  setTimeout(() => callback('pepe'), 5000);  
}  
  
test('Callback Name is pepe', done => {  
  getNameFromServer(name => {  
    expect(name).toBe('pepe');  
    done();  
  });  
});
```

API REST

```
const app = require('./server')
const supertest = require('supertest')

const request = supertest(app)

test('when get all ads then get test adds', async () => {
  const { body } = await request.get('/ads')
    .expect('Content-type', /json/)
    .expect(200)

  expect(body[0].message).toBeTruthy();
  expect(body[0].author).toBeTruthy();
  expect(body[0].id).toBeTruthy();
})
```



GET

```
test('when create new ad then this ad can be obtained', async () => {  
  var newAd = { message: 'Compro lavadora', author: 'Manolo' };  
  
  const { body } = await request.post('/ads')  
    .send(newAd)  
    .expect('Content-type', /json/)   
    .expect(200)  
  
  expect(body.id).toBeTruthy();  
  
  const { body: getBody } = await request.get('/ads/' + body.id)  
    .expect('Content-type', /json/)   
    .expect(200)  
  
  getBody.message = newAd.message;  
  getBody.author = newAd.author;  
})
```



POST


```
test('when delete ad then is effectively deleted', async () => {  
  await request.delete('/ads/1')  
    .expect('Content-type', /json/)   
    .expect(200)  
  
  await request.get('/ads/1')  
    .expect(404)  
  
})
```

DELETE

```
test('when update ad then is effectively updated', async () => {  
  const updatedAd = { id: '2', message: 'new message', author: 'new author' };  
  
  await request.put('/ads/2')  
    .send(updatedAd)  
    .expect('Content-type', /json/)  
    .expect(200)  
  
  var { body } = await request.get('/ads/2')  
    .expect('Content-type', /json/)  
    .expect(200)  
  
  expect(body).toEqual(updatedAd);  
})
```

PUT

END TO END

```
describe('Visits de home page and', function () {  
  it('should go to /login and log in', function () {  
    cy.visit('http://localhost:8080/')  
    cy.contains('Entrar').click({ force: true })  
    cy.url().should('include', '/login')  
    cy.get('.input_username')  
      .type('ari')  
    cy.get('.input_password')  
      .type('ane')  
    cy.contains('Enviar').click({ force: true })  
    cy.contains('Ver comentarios').click({ force: true })  
  })  
  it('should write a post with offensive words', function () {  
    cy.get('.input_text')  
      .type('cerdo')  
    cy.contains('Enviar').click({ force: true })  
    cy.contains('.error_valid', 'No puedes incluir palabras ofensivas')  
  })  
})  
})
```



ENTENDER POR QUÉ
EL TEST FALLA

Summary of all failing tests

FAIL src/app/auth/auth-interceptor.service.spec.ts

● AuthInterceptorService › should be created

NullInjectorError: StaticInjectorError(DynamicTestModule)[HttpClient]:
StaticInjectorError(Platform: core)[HttpClient]:
NullInjectorError: No provider for HttpClient!

```
at NullInjector.Object.<anonymous>.NullInjector.get (../packages/core/src/di/injector_compatibility.ts:229:21)
at resolveToken (../packages/core/src/di/injector.ts:357:20)
at tryResolveToken (../packages/core/src/di/injector.ts:299:12)
at StaticInjector.Object.<anonymous>.StaticInjector.get (../packages/core/src/di/injector.ts:180:14)
at resolveToken (../packages/core/src/di/injector.ts:357:20)
at tryResolveToken (../packages/core/src/di/injector.ts:299:12)
at StaticInjector.Object.<anonymous>.StaticInjector.get (../packages/core/src/di/injector.ts:180:14)
at resolveNgModuleDep (../packages/core/src/view/ng_module.ts:126:25)
at NgModuleRef_.Object.<anonymous>.NgModuleRef_.get (../packages/core/src/view/refs.ts:390:12)
at injectInjectorOnly (../packages/core/src/di/injector_compatibility.ts:97:29)
at ɵɵinject (../packages/core/src/di/injector_compatibility.ts:116:54)
at injectArgs (../packages/core/src/di/injector_compatibility.ts:213:17)
at ../packages/core/src/di/util.ts:54:30
at _callFactory (../packages/core/src/view/ng_module.ts:196:14)
at _createProviderInstance (../packages/core/src/view/ng_module.ts:149:20)
at resolveNgModuleDep (../packages/core/src/view/ng_module.ts:122:15)
at NgModuleRef_.Object.<anonymous>.NgModuleRef_.get (../packages/core/src/view/refs.ts:390:12)
at injectInjectorOnly (../packages/core/src/di/injector_compatibility.ts:97:29)
at ɵɵinject (../packages/core/src/di/injector_compatibility.ts:116:54)
at injectArgs (../packages/core/src/di/injector_compatibility.ts:213:17)
at ../packages/core/src/di/util.ts:54:30
at _callFactory (../packages/core/src/view/ng_module.ts:196:14)
at _createProviderInstance (../packages/core/src/view/ng_module.ts:149:20)
at resolveNgModuleDep (../packages/core/src/view/ng_module.ts:122:15)
at NgModuleRef_.Object.<anonymous>.NgModuleRef_.get (../packages/core/src/view/refs.ts:390:12)
at injectInjectorOnly (../packages/core/src/di/injector_compatibility.ts:97:29)
at ɵɵinject (../packages/core/src/di/injector_compatibility.ts:116:54)
at injectArgs (../packages/core/src/di/injector_compatibility.ts:213:17)
at ../packages/core/src/di/util.ts:54:30
at _callFactory (../packages/core/src/view/ng_module.ts:196:14)
at _createProviderInstance (../packages/core/src/view/ng_module.ts:149:20)
at resolveNgModuleDep (../packages/core/src/view/ng_module.ts:122:15)
at NgModuleRef_.Object.<anonymous>.NgModuleRef_.get (../packages/core/src/view/refs.ts:390:12)
at TestBedViewEngine.Object.<anonymous>.TestBedViewEngine.inject (../..../packages/core/testing/src/test_bed.ts:481:45)
at Function.Object.<anonymous>.TestBedViewEngine.inject (../..../packages/core/testing/src/test_bed.ts:223:36)
at src/app/auth/auth-interceptor.service.spec.ts:10:23
at ZoneDelegate.invoke (node_modules/zone.js/dist/zone.js:396:30)
at ProxyZoneSpec.onInvoke (node_modules/zone.js/dist/proxy.js:117:43)
at ZoneDelegate.invoke (node_modules/zone.js/dist/zone.js:395:36)
at Zone.run (node_modules/zone.js/dist/zone.js:153:47)
```

Summary of all failing tests

FAIL src/app/auth/auth-interceptor.service.spec.ts

● AuthInterceptorService › should be created

```
NullInjectorError: StaticInjectorError(DynamicTestModule)[HttpClient]:  
  StaticInjectorError(Platform: core)[HttpClient]:  
    NullInjectorError: No provider for HttpClient!
```

● AuthInterceptorService › should be created

expect(received).toBeTruthy()

Received: undefined

```
12 |  
13 |     it('should be created', () => {  
> 14 |         expect(service).toBeTruthy();  
    |                           ^  
15 |     });  
16 | });  
17 |
```

```
at src/app/auth/auth-interceptor.service.spec.ts:14:21  
at ZoneDelegate.invoke (node_modules/zone.js/dist/zone.js:396:30)  
at ProxyZoneSpec.onInvoke (node_modules/zone.js/dist/proxy.js:117:43)  
at ZoneDelegate.invoke (node_modules/zone.js/dist/zone.js:395:36)  
at Zone.run (node_modules/zone.js/dist/zone.js:153:47)
```


FAIL src/app/policies/edit-policy/edit-policy.component.spec.ts

- Test suite failed to run

Cannot find module 'primeng/radiobutton/public_api' from 'edit-policy.component.spec.ts'

```
7 | import { CheckboxModule } from 'primeng/checkbox';
8 | import { InputTextModule } from 'primeng/inputtext';
> 9 | import { RadioButtonModule } from 'primeng/radiobutton/public_api';
    |      ^
10 | import { ValidateOnblurDirective } from 'src/app/sharedModule/validate-on-blur.dir
11 | import { ValidationMessagesComponent } from 'src/app/sharedModule/validation-messa
12 | import { CreatePolicyComponent } from '../create-policy/create-policy.component';
```

at Resolver.resolveModule (node_modules/jest-resolve/build/index.js:296:11)

at Object.<anonymous> (src/app/policies/edit-policy/edit-policy.component.spec.ts:9:1)

Test Suites: **1 failed**, **22 passed**, 23 total

Tests: **43 passed**, 43 total

Snapshots: 0 total

Time: **7.746s**

Ran all test suites.

```
export const mappingAfterFilter = () => {

  const byTitle = filterByTitle(listOfMovies, input.value)
  byTitle.map(item => {showListOfMovies(item)})

}
```

```
Debug
x test('calls show list of movies method', () => {
  document.body.innerHTML = `<input class='input' type="text" id='input_id' />`;
  const inputHTML = document.querySelector('.input');

  index.mappingAfterFilter(inputHTML);
  expect(showListOfMoviesSpy).toHaveBeenCalled();
})
```

● printFilteredItems › calls clearlist, filteredByTitle and showListsOfMovies

TypeError: Cannot read property 'value' of null

```
79 | export const mappingAfterFilter = () => {
80 |
> 81 |     const byTitle = filterByTitle(listOfMovies, input.value)
    |                                           ^
82 |     byTitle.map(item => {showListOfMovies(item)})
83 |
84 | }
```

at mappingAfterFilter (index.js:81:55)
 at Object.printFilteredItems (index.js:75:9)
 at Object.<anonymous> (index.test.js:118:11)

```
export const mappingAfterFilter = () => {  
  if(input){  
    const byTitle = filterByTitle(listOfMovies, input.value)  
    byTitle.map(item => {showListOfMovies(item)})  
  }  
}
```

```
✓ test('calls show list of movies method', () => {  
  document.body.innerHTML = `<input class='input' type="text" id='input_id' />`;  
  const inputHTML = document.querySelector('.input');  
  
  index.mappingAfterFilter(inputHTML);  
  expect(showListOfMoviesSpy).toHaveBeenCalled();  
})
```

FAIL src/app/auth/login-proxy.service.spec.ts

LoginProxyService

- ✓ should be created (28ms)
- ✗ should call to obfuscated method (76ms)
- ✓ should call to doRefresh Method (11ms)

● LoginProxyService › should call to obfuscated method

Expected one matching request for criteria "Match URL: http://localhost:8765/api/sso/utilities/obfuscatedpwd?pwd=", found none. Requests received are: GET http://localhost:8765/api/sso/utilities/obfuscatedpwd?pwd=PT1RTX1NRE4=.



COBERTURA

100% FAKE

PASS src/app/app.component.spec.ts

AppComponent

- ✓ should create the app (108ms)
- ✓ should have as title ' ' (1ms)
- ✓ should render title (45ms)

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	61.29	100	30.77	55.56	
app	70.59	100	40	66.67	
app.component.html	100	100	100	100	
app.component.ts	68.75	100	40	64.29	22-30
app/services	50	100	25	41.67	
notifications-bus-service.service.ts	50	100	25	41.67	19-40

Test Suites: 1 passed, 1 total

Tests: 3 passed, 3 total

Snapshots: 0 total

Time: 2.296s, estimated 7s

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	57.41	17.86	37.69	54.2	
app	68.55	0	56.36	64.55	
app-proxy.service.ts	64	100	53.85	59.09	27-32,58-70
app.component.html	100	100	100	100	
app.component.ts	68.75	100	40	64.29	22-30
app.service.ts	69.39	100	60	68.09	43-46,111-138
data-resolve.service.ts	80	100	50	71.43	14-15
interceptor.service.ts	44.44	0	50	28.57	16-23
notifications-bus-service.service.ts	78.57	100	62.5	75	19,26-29
app/auth	61.11	0	35.71	55.56	
auth-interceptor.service.ts	28.57	0	0	23.08	13-51
auth.module.ts	100	100	100	100	
login-proxy.service.ts	70.59	100	40	64.29	21-31
login.service.ts	78.57	100	75	75	24-26
app/auth/login	81.08	16.67	80	80	
fake-login.ts	100	100	100	100	
fake-offuscated.ts	100	100	100	100	
fake-token.ts	100	100	100	100	
login.component.html	100	100	100	100	
login.component.ts	78.13	16.67	80	76.67	63-69
app/fixtures	100	100	100	100	
fake-actions.ts	100	100	100	100	
fake-applications.ts	100	100	100	100	
fake-payload-policy.ts	100	100	100	100	
fake-policies.ts	100	100	100	100	
fake-user-list.ts	100	100	100	100	
app/home/home	100	100	100	100	
home.component.html	100	100	100	100	
home.component.ts	100	100	100	100	
app/layouts/app-layout	100	100	0	100	
app-layout.component.html	100	100	100	100	
app-layout.component.ts	100	100	0	100	
app/layouts/simple-layout	100	100	100	100	
simple-layout.component.html	100	100	100	100	
simple-layout.component.ts	100	100	100	100	
app/navBar/navBar	57.89	50	33.33	52.94	
navBar.component.html	100	100	100	100	
navBar.component.ts	55.56	50	33.33	50	20-38
app/policies/create-policy	46.27	12.5	17.65	43.75	
create-policy.component.html	100	100	100	100	
create-policy.component.ts	45.45	12.5	17.65	42.86	50-52,72-129,137-138
app/policies/edit-policy	22.22	0	0	19.35	
edit-policy.component.html	100	100	100	100	
edit-policy.component.ts	20	0	0	16.67	20-60
app/policies/policies-list	46.67	100	0	41.67	
policies-list.component.html	100	100	100	100	
policies-list.component.ts	42.86	100	0	36.36	17-28
app/policies/policy-detail	25.81	0	0	22.22	
policy-detail.component.html	100	100	100	100	
policy-detail.component.ts	23.33	0	0	19.23	16-57
app/sharedModule	88.89	100	33.33	85.71	

All files app/auth

68.92% Statements 51/74 **0%** Branches 0/10 **57.14%** Functions 8/14 **64.62%** Lines 42/65

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

File ▲		Statements ⇅		Branches ⇅		Functions ⇅		Lines ⇅	
auth-interceptor.service.ts	<div><div></div></div>	33.33%	10/30	0%	0/10	20%	1/5	28.57%	8/28
auth.module.ts	<div><div></div></div>	100%	13/13	100%	0/0	100%	0/0	100%	11/11
login-proxy.service.ts	<div><div></div></div>	82.35%	14/17	100%	0/0	60%	3/5	78.57%	11/14
login.service.ts	<div><div></div></div>	100%	14/14	100%	0/0	100%	4/4	100%	12/12

All files / app/auth login-proxy.service.ts

82.35% Statements 14/17 100% Branches 0/0 60% Functions 3/5 78.57% Lines 11/14

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

```
1  5x import { HttpClient, HttpResponse } from '@angular/common/http';
2  5x import { Inject, Injectable } from '@angular/core';
3      import { Observable } from 'rxjs';
4  5x import { mergeMap } from 'rxjs/operators';
5      import { OffuscatedPwdDTO } from '../models/offuscatedPwd-dto.model';
6      @Injectable({
7          providedIn: 'root'
8      })
9  5x export class LoginProxyService {
10
11 12x     constructor(private http: HttpClient, @Inject('config') private config) { }
12
13         offuscate(password: string): Observable<OffuscatedPwdDTO> {
14             1x         const reversePassword = password.split('').reverse().join('');
15             1x         const newPassword = btoa(reversePassword);
16             1x         console.log('OFFUSCATED PROXY');
17
18             1x         return this.http.get<OffuscatedPwdDTO>(this.config.api + '/utilities/obfuscatedpwd?pwd=' + newPassword);
19         }
20         login(username: string, password: string): Observable<HttpResponse<any>> {
21             console.log('LOGIN PROXY');
22
23             return this.offuscate(password)
24                 .pipe(mergeMap(data =>
25                     this.http.post(this.config.api + '/tokens/credentials', {
26                         email: username,
27                         password: data.obfuscated,
28                         applicationName: 'policies'
29                     }, {observe: 'response'})
30                 ));
31     }
32
33     doRefresh(refreshToken: string): Observable<HttpResponse<any>> {
34         1x         console.log('DO REFRESH PROXY', refreshToken);
35         1x         return this.http.get(this.config.api + '/tokens/refresh/' + refreshToken, {observe: 'response'});
36     }
37 }
38 }
```

```
import { images } from './images.js';

export default function randomNumber() {
  return Math.floor(Math.random() * images.length);
}
```

```
import randomNumber from './randomNumber.js';

✓ test('test random number function', () => {
  const result = randomNumber();

  expect(result).toEqual(expect.any(Number));
});
```

```
PASS ./randomNumber.test.js
  ✓ test random number function (2ms)
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	100	100	100	100	
images.js	100	100	100	100	
randomNumber.js	100	100	100	100	

```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.978s, estimated 1s
Ran all test suites matching /randomNumber.test.js/i.
```

```
export const fakeList = `<li class="list_item"><h2>Castle in the Sky</h2></li>`
```

```
✓ test('print one item of the list', () => {  
  document.body.innerHTML = `<ul class="list"></ul>`;   
  const input = `<li class="list_item"><h2>Castle in the Sky</h2></li>`  
  index.showListOfMovies(input);  
  expect(input).toBe(fakeList)  
})
```

```
1x export const showListOfMovies = (title = 'Title', list) => {  
  
1x   list = document.querySelector('.list');  
1x   if(list){  
1x     const li = document.createElement('li')  
1x     li.classList.add('list_item')  
1x     const movie_title = document.createElement('h2')  
1x     const liText = document.createTextNode(title)  
1x     const img = document.createElement('img')  
1x     img.setAttribute('src', getRandomImage());  
1x     img.setAttribute('class', 'movie_img');  
1x     li.appendChild(img);  
1x     movie_title.appendChild(liText)  
1x     li.appendChild(movie_title)  
1x     list.appendChild(li)  
  
1x   }  
}
```

¿QUÉ DICE

LA GENTE

SOBRE TESTING?



QUIENES NO HACEN
TEST Y LO HAN
INTENTADO:

"ES MUY FALSEABLE A VECES"

"Me cuesta encontrar el momento"

"EN MI TRABAJO NO SE USA"

"En mis proyectos personales me hace sentir
que no es de gran valor agregado"

"NO SE QUÉ SE TESTEA Y QUÉ NO"

"No se qué cantidad de tests unitarios hacer"

"LO INTENTÉ Y ME QUEDÉ TRABADO"

"Me confunde mockear cosas y siento que no
testeo mi app de verdad"

"NO SE POR DONDE ABORDAR LA
FEATURE DESDE LAS PRUEBAS"

"Tuve la sensación de que era algo muy difícil,
frustrante y que no me gustaría"



QUIENES NO HACEN TEST Y LO HAN INTENTADO:

"AL PRINCIPIO NO LE VEIA EL SENTIDO SI
YO SE QUE MI APP FUNCIONA!"

"A veces tienes otros problemas de diseño e
implementación y adiós tests"

"AL PRINCIPIO SENTÍA INCREDULIDAD Y
DESCONCIERTO"

"Debo aceptar que es super tedioso y que en
momentos me da fastidio hacer"

"LA CURVA DE APRENDIZAJE ES ABRUPTA"

"Hay que leer mucho y pocos lo hacen"

"ME RESULTABA MUY DIFÍCIL PORQUE NO
CONOCÍA LOS PRINCIPIOS DE DISEÑO"

"Alguien que domine diseño de software puede
aprender rápido, al revés: imposible"



CUANDO NO TE PIDEN HACER TESTS:

"SIN TESTS NO HAY CÓDIGO"

"El cliente no es quién para decidir hacer tests"

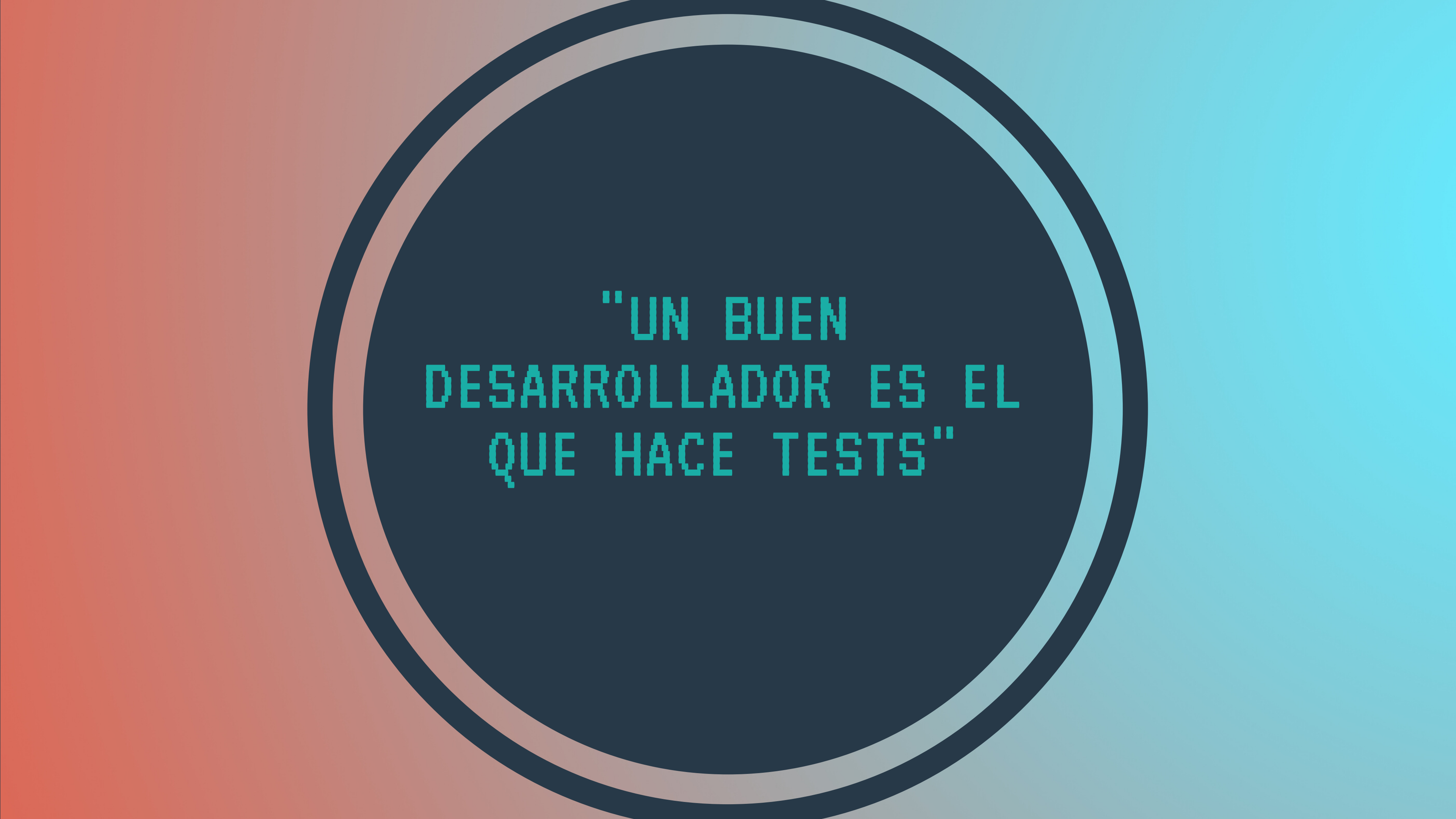
"INCLUYE TESTS SIEMPRE"

"Tu obligación es entregar código de calidad y el test es parte de ello"

"YO SIN TESTS NO TRABAJO"

"El testing debe estar integrado en el ADN de todo programador y forma parte del ciclo de vida del producto"

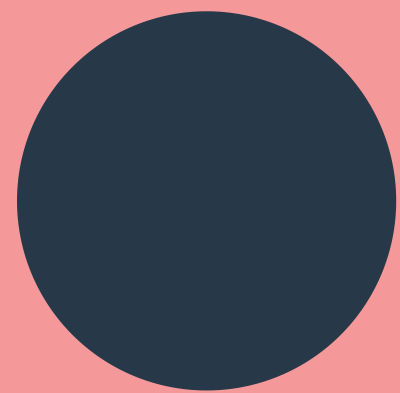
"EL TESTING ES RESPONSABILIDAD TUYA"



"UN BUEN
DESARROLLADOR ES EL
QUE HACE TESTS"

"UN BUEN
DESARROLLADOR ES EL
QUE HACE TESTS"





ALGUNAS REALIDADES:

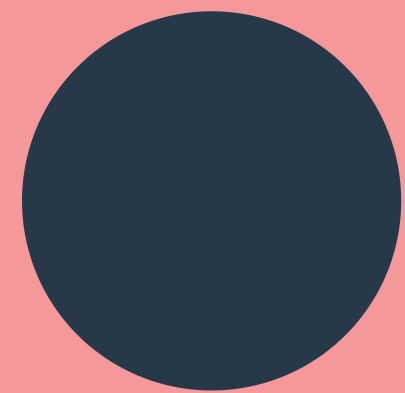
"LO USO SOLO EN PROYECTOS CON BASTANTE TIEMPO PARA EL DESARROLLO".

"Me resultó complicado y pensé que tardaría más en aprenderlo que en resolver mi problema manualmente"

"HAY PROYECTOS QUE TE COME EL TIEMPO"

"Si quien paga te exige tener X en unos tiempos super ajustados, habrá sacrificios por el camino"

"SI HAY IMPREVISTOS, SE ME DICE QUE DEJE LOS TESTS PARA MAS ADELANTE Y HACIENDO QUE ASÍ QUE ME OLVIDE"



ALGUNAS REALIDADES:

"ESTO SOLO LO VEMOS NOSOTROS, LOS JEFES NO LO ENTIENDEN"

"Me han llegado a decir que si necesito

"gastar" tanto tiempo en prevenir errores es porque no se programar y no confío en lo que hago"

"SI EL EQUIPO NO SABE HACER TESTS, SE PEDIRÁ NO EMPLEAR TIEMPO (DINERO) EN QUE APRENDAN"

"Es muy importante para los proyectos, aunque a veces las empresas no lo vean así y no quieran invertir en ésta área"



ALGUNAS REALIDADES:

"GRACIAS A FULANO QUE HIZO EL
ESFUERZO DE SENTARSE Y EXPLICARME"
"Alguien se sentó conmigo a enseñarme cómo
se aplicaba eso del testing, cómo se hacía y el
valor que aporta tenerlo automatizado, el saber
que funciona"

"Me hace sentir más segura sobre el código, me da sensación de solidez"

"No fue fácil, pero fue un descubrimiento. Un cambio de paradigma que permite ver la programación desde otro enfoque"

"Al principio me resultó bastante frustrante porque no me salía nada, pero luego ya fue como una maravilla"

SI HACES
TESTING...

"No solo ayuda a comprobar que el desarrollo funciona, sino que ayuda a hacer un mejor diseño de este, a modularizarlo más e incluso a codificar mucho mejor y de una manera más clara"

"Me ayuda muchísimo a debuggear"

"Duermo más tranquila"

SI HACES
TESTING...

COSAS A TENER EN CUENTA

CONCLUSIONES



EL TEST NO SABE.

Tú lo tienes que
poner en contexto
igual al de tu
aplicación para
que haga esa
prueba.

LIBRE DE BUGS, NO

Las circunstancias
las definimos
nosotros y pueden
escaparse errores.
tests > bugs

VERDE GUAY, NO SIEMPRE

Recuerda que hay
falsos positivos.
Rompe tu código
para comprobar
el test.
Pide code review.

PIERDE MIEDO AL ROJO

Busca el error,
entiéndelo y
convértelo en
verde.



UNA SOLA ASERCIÓN

¡Pero que exista!

Un test sin
aserción, pasa
aunque no esté
haciendo nada.

TIENES PERMISO DE DUPLICAR

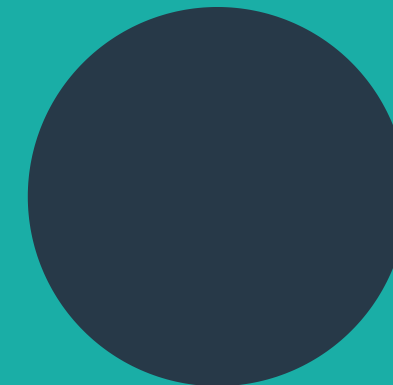
No pasa nada si
eso favorece a
que sean más
legibles.

COBERTURA

Nunca pongas la
cobertura como tu
objetivo. Nunca.

REFACTOR EN VERDE

No refactorices
con los tests
fallando, que
luego no sabes
qué se rompe.



REFACTORIZA

Principio de
responsabilidad
única.

PRUEBA VALOR

Haz tests que
prueben
funcionalidad de
valor de negcio,
pero practica con
todo.

NO TODO ES HAPPY

No solo pruebes el
"happy path".
Pruebas otras
alternativas.

NO PUSH SIN TESTS

Es mejor ir
haciendo tests de
lo que vas
desarrollando a
que se te
acumulen.



VUELVE A LEER

Vuelve a todo lo que leíste. Lo verás con otros ojos y entenderás nuevas cosas.

ASUME LA CURVA

El testing es como un "nuevo lenguaje" que tienes que aprender y no será fácil ni rápido.

PRACTICA

Mucho. Haz tests de proyectos que ya tienes hechos. De todo lo que puedas.

CUANDO TE SIENTAS COMODO:

Comienza a meterlos en tu sprint.

A large teal circle with a thick pink border is centered on a dark blue background. Inside the teal circle, the text "Muchas gracias =)" is written in a bold, dark blue font.

Muchas gracias =)

ESPERO LES HAYA GUSTADO Y
MOTIVADO A APRENDER

Recursos:

Charla: 'Testing práctico con JavaScript', Ramón Guijarro

Charla: "it('should be easier'): Testing automatizado en el mundo real, Paqui Calabria

Charla: "El testing ya no es para gurús", Ricardo Borillo

Charla: "Mocks & Stubs, Ken Scambler

Charla: "Testing, CI and CD in the real world", Roc Boronat

Introducción al testing, Jaime Barrio (openWebinars)

Desarrollo dirigido por test, Carlos Herrera (openWebinars)

Testing: Introducción y buenas prácticas, CodelyTV

Testing con Javascript, Jorge Baumann

100% Code Coverage is Useless, Jorge Baumann

Recursos:

Learn the smart, efficient way to test any Javascript application, Kent Dodds.

Write tests. Not too many. Mostly integration, Kent Dodds.

The Merits of Mocking, Kent Dodds.

Testing JavaScript with Jest, Flavio Copes

Libro: Clean code, solid y testing aplicado a JavaScript, Miguel A. Gómez (Software crafters)

Documentación oficial de Jest

Jest cheat sheet

Documentación oficial de Cypress

SuperTest