Nombre del instructor	Adolfo Centeno Tellez
Área temática	Ciencia de datos
Rol	Data Scientist
Competencia	Full Stack
Subcompetencia	Frontend
Objetivo Módulo Conocimiento	

Módulo Conocimiento (5h en total)			
	Duración	**Objetivos**	**Subtemas**
Tema 4 - Desarrollo dirigido por pruebas en Angular	**1 hora**	Definir y aplicar el modelo de Desarrollo Dirigido por pruebas de Angular para asegurar la calidad del Codigo, en sus 3 etapas de Pruebas unitarias, pruebas de integracion y pruebas End to End.	*1.- Concepto de desarrollo dirigido por pruebas y creacion de pruebas unitarias con Angular* *2.- Creacion de pruebas de integracion y Pruebas end to end*

Introducción al tema

En el tema anterior creo una base de datos no relacional en Firebase, se crearon las colecciones para cada seccion de nuestro proyecto de CV, asi mismo se crearon los services en Angular 18 que nos proporcionaron el acceso a los datos.

Finalmente se actualizaron los componentes para inyectar el servicio y visualizar la información en nuestro template HTML.

Al finalizar este módulo entenderás el desarrollo dirigido por pruebas en Angular 18, aprenderás cómo configurar tu proyecto para soportar pruebas, codificaras pruebas unitarias para probar módulos aislados de una aplicación, también crearás pruebas de integración para probar la interacción entre el código y la interfaz de tus componentes, finalmente instalamos y configuramos el framework Cypress para realizar pruebas globales de tu proyecto llamadas end to end, donde podrás garantizar que tu proyecto cumple con las expectativas del cliente desde el punto de vista de reglas de negocios.

SUBTEMA 1: Concepto de desarrollo dirigido por pruebas y pruebas unitarias con Angular

El Desarrollo Dirigido por Pruebas (TDD) es una metodología de desarrollo de software que implica escribir pruebas antes de escribir el código. En el caso de Angular, se puede aplicar TDD para garantizar la calidad y la confiabilidad del código.

Herramientas necesarias

Para implementar TDD en Angular, se necesitan las siguientes herramientas:

- ***Jasmine***: Un framework de testing para JavaScript que se utiliza para escribir pruebas unitarias y de integración. Jasmine ya viene integrado con Angular 18.
- ***Karma***: Un herramienta de testing que se encarga de ejecutar las pruebas escritas con Jasmine y mostrar el reporte de pruebas, ya integrado con Angular 18 no requiere instalación adicional.

Al implementar TDD en Angular, se pueden obtener varias ventajas, como:

Código más confiable: Las pruebas garantizan que el código funciona como se espera.

Menos bugs: Al escribir pruebas antes del código, se detectan errores más temprano en el proceso de desarrollo.

Mejora la documentación: Las pruebas proporcionan una documentación adicional sobre el comportamiento del código.

Aumenta la velocidad de desarrollo: Al tener pruebas escritas, se puede refactorizar el código con confianza, lo que reduce el tiempo de desarrollo.

El Desarrollo Dirigido por Pruebas (TDD) es una metodología efectiva para garantizar la calidad y la confiabilidad del código en Angular. Al utilizar herramientas como Jasmine y Karma, se pueden escribir pruebas unitarias y de integración para asegurarse de que el código funciona como se espera.

Configuración de Angular para soportar TDD

Crear karma.conf.js

En la raíz del proyecto:

$ pwd

$ ls -la

```
➜  mycv git:(master) ✗ pwd
/Users/adsoft/mycv
➜  mycv git:(master) ✗ ls -la
total 1104
drwxr-xr-x   17 adsoft  staff     544 Dec 26 01:33 .
drwxr-x---+  80 adsoft  staff    2560 Dec 31 01:26 ..
drwxr-xr-x    3 adsoft  staff      96 Dec 26 01:33 .angular
-rw-r--r--    1 adsoft  staff     274 Dec 26 01:30 .editorconfig
drwxr-xr-x   12 adsoft  staff     384 Dec 26 01:31 .git
-rw-r--r--    1 adsoft  staff     587 Dec 26 01:30 .gitignore
drwxr-xr-x    5 adsoft  staff     160 Dec 26 01:30 .vscode
-rw-r--r--    1 adsoft  staff    1065 Dec 26 01:30 README.md
-rw-r--r--    1 adsoft  staff    3060 Dec 26 17:04 angular.json
drwxr-xr-x  612 adsoft  staff   19584 Dec 26 17:03 node_modules
-rw-r--r--    1 adsoft  staff  530221 Dec 26 17:03 package-lock.json
-rw-r--r--    1 adsoft  staff    1067 Dec 26 17:03 package.json
drwxr-xr-x    3 adsoft  staff      96 Dec 26 01:30 public
drwxr-xr-x    7 adsoft  staff     224 Dec 26 17:04 src
-rw-r--r--    1 adsoft  staff     424 Dec 26 01:30 tsconfig.app.json
-rw-r--r--    1 adsoft  staff    1021 Dec 26 01:30 tsconfig.json
-rw-r--r--    1 adsoft  staff     434 Dec 26 01:30 tsconfig.spec.json
➜  mycv git:(master) ✗
```

$ ng generate config karma

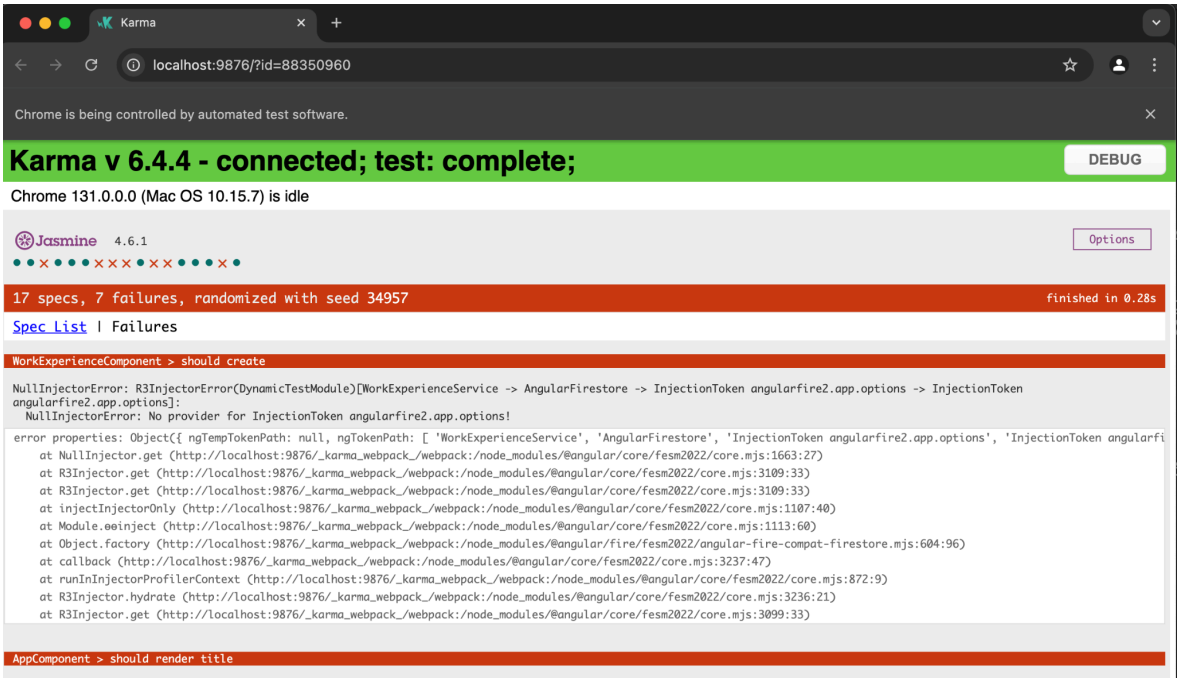
```
➜  mycv git:(master) ✗ ng generate config karma
CREATE karma.conf.js (1232 bytes)
UPDATE angular.json (3104 bytes)
➜  mycv git:(master) ✗
```

Ejecutamos nuestro proyecto en modo test:

$ ng test

```
→  mycv git:(master) ✗ ng test
✔ Browser application bundle generation complete.
31 12 2024 01:30:02.084:WARN [karma]: No captured browser, open http://localhost:9876/
31 12 2024 01:30:02.163:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
31 12 2024 01:30:02.164:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
31 12 2024 01:30:02.172:INFO [launcher]: Starting browser Chrome
31 12 2024 01:30:21.256:INFO [Chrome 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket uFur9U48ER6f8Rg5AAAB with id 88350960
        NullInjectorError: R3InjectorError(DynamicTestModule)[WorkExperienceService -> AngularFirestore -> InjectionToken angularfire2.app.options -> InjectionToken angularfire2.app.options]:
          NullInjectorError: No provider for InjectionToken angularfire2.app.options!
        error properties: Object({ ngTempTokenPath: null, ngTokenPath: [ 'WorkExperienceService', 'AngularFirestore', 'InjectionToken angularfire2.app.options', 'InjectionToken angularfire2.app.options' ] })
            at NullInjector.get (node_modules/@angular/core/fesm2022/core.mjs:1663:27)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at injectInjectorOnly (node_modules/@angular/core/fesm2022/core.mjs:1107:40)
            at Module.ɵɵinject (node_modules/@angular/core/fesm2022/core.mjs:1113:60)
            at Object.factory (node_modules/@angular/fire/fesm2022/angular-fire-compat-firestore.mjs:604:96)
            at callback (node_modules/@angular/core/fesm2022/core.mjs:3237:47)
            at runInInjectorProfilerContext (node_modules/@angular/core/fesm2022/core.mjs:872:9)
            at R3Injector.hydrate (node_modules/@angular/core/fesm2022/core.mjs:3236:21)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3099:33)
Chrome 131.0.0.0 (Mac OS 10.15.7): Executed 16 of 17 (7 FAILED) (0 secs / 0.187 Chrome 131.0.0.0 (Mac OS 10.15.7): Executed 17 of 17 (7 FAILED) (0 secs / 0.19 sChrome 131.0.0.0 (Mac OS 10.15.7): Executed 17 of 17 (7 FAILED) (0.276 secs / 0.19 secs)
TOTAL: 7 FAILED, 10 SUCCESS
```

Este proceso levanta una servidor llamado karma que ejecuta todas las pruebas que el proyecto trae por default, visualiza en Google Chrome el servidor karma



Actualizar karma.conf.js con el navegador ChromeHeadlessCI

Necesitamos realizar test, pero sin Google Chrome en modo web con gráficos, solo en consola, para después usarlo en procesos de CI/CD dentro de github que solo usa un Linux en consola, asi que debemos agregar **ChromeHeadlessCI** en karma.conf.js para que nos permita ejecutar las pruebas solo en modo texto.

Agregamos el browser **ChromeHeadlessCI** con su configuración, líneas 36-43.

$ nano karma.conf.js

```
35      browsers: ['Chrome'],
36      browsers: ['ChromeHeadlessCI'],
37       customLaunchers: {
38         ChromeHeadlessCI: {
39          base: 'ChromeHeadless',
40          flags: ['--no-sandbox']
41         }
42      },
43      singleRun: false,
44      restartOnFileChange: true
```

Probar ng test, solo en consola agregamos los parametros: ***--no-watch --no-progress --browsers=ChromeHeadlessCI***

$ ***ng test --no-watch --no-progress --browsers=ChromeHeadlessCI***

```
➜  mycv git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

31 12 2024 02:09:41.827:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
31 12 2024 02:09:41.838:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
31 12 2024 02:09:41.845:INFO [launcher]: Starting browser ChromeHeadless
31 12 2024 02:09:45.825:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket uqgPrKTpMn
jcDHggAAAB with id 84772530
Chrome Headless 131.0.0.0 (Mac OS 10.15.7) AppComponent should create the app FAILED
        Error: NG0304: 'app-header' is not a known element (used in the 'AppComponent' component template
):
        1. If 'app-header' is an Angular component, then verify that it is a part of an @NgModule where t
his component is declared.
        2. If 'app-header' is a Web Component then add 'CUSTOM_ELEMENTS_SCHEMA' to the '@NgModule.schemas
' of this component to suppress this message.
        error properties: Object({ code: 304 })
            at validateElementIsKnown (node_modules/@angular/core/fesm2022/core.mjs:10533:23)
            at ɵɵelementStart (node_modules/@angular/core/fesm2022/core.mjs:24343:9)
            at ɵɵelement (node_modules/@angular/core/fesm2022/core.mjs:24418:5)
            at templateFn (ng:///AppComponent.js:8:9)
            at executeTemplate (node_modules/@angular/core/fesm2022/core.mjs:12159:9)
            at renderView (node_modules/@angular/core/fesm2022/core.mjs:13366:13)
            at renderComponent (node_modules/@angular/core/fesm2022/core.mjs:13312:5)
            at renderChildComponents (node_modules/@angular/core/fesm2022/core.mjs:13414:9)
            at renderView (node_modules/@angular/core/fesm2022/core.mjs:13394:13)
            at ComponentFactory.create (node_modules/@angular/core/fesm2022/core.mjs:16768:17)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 17 of 17 (7 FAILED) (0.262 secs / 0.209 secs)
TOTAL: 7 FAILED, 10 SUCCESS
➜  mycv git:(master) ✗
```

Antes de continuar vamos a poner comentarios a todas las pruebas en cada archivo .spec.ts de cada servicio, cada prueba inicia con ***it (‘ should …***

Para poner en comentario una prueba se pone una x en cada it. ejemplo:
xit ('should be created', …

$ nano src/app/services/header-service/header.service.spec.ts

```
 1 import { TestBed } from '@angular/core/testing';
 2
 3 import { HeaderService } from './header.service';
 4
 5 describe('HeaderService', () => {
 6   let service: HeaderService;
 7
 8   beforeEach(() => {
 9     TestBed.configureTestingModule({});
10     service = TestBed.inject(HeaderService);
11   });
12
13   xit('should be created', () => {
14     expect(service).toBeTruthy();
15   });
16 });
```

Repetir para cada servicio.

$ nano src/app/services/work-experience-service/work-experience.service.spec.ts
$ nano src/app/services/education-service/education.service.spec.ts
$ nano src/app/services/skills-service/skills.service.spec.ts
$ nano src/app/services/certificates-service/certificates.service.spec.ts
$ nano src/app/services/languages-service/languages.service.spec.ts
$ nano src/app/services/interests-service/interests.service.spec.ts

Ahora ponemos en comentario, las pruebas que cada componente crea por default, incluyendo app.component.spec.ts

$ nano src/app/app.component.spec.ts

```
import { TestBed } from '@angular/core/testing';
import { RouterModule } from '@angular/router';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        RouterModule.forRoot([])
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  });

  xit('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  xit(`should have as title 'mycv'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('mycv');
  });

  xit('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('h1')?.textContent).toContain('Hello, mycv');
  });
});
```

Análogamente ponemos en comentario cada las pruebas de cada componente creado por nosotros:

$ nano src/app/header/header.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { HeaderComponent } from './header.component';

describe('HeaderComponent', () => {
  let component: HeaderComponent;
  let fixture: ComponentFixture<HeaderComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [HeaderComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(HeaderComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  xit('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

$ nano src/app/work-experience/work-experience.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { WorkExperienceComponent } from './work-experience.component';

describe('WorkExperienceComponent', () => {
  let component: WorkExperienceComponent;
  let fixture: ComponentFixture<WorkExperienceComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [WorkExperienceComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(WorkExperienceComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  xit('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Repetimos el proceso para el resto de los componentes:

$ nano src/app/education/education.component.spec.ts
$ nano src/app/skills/skills.component.spec.ts
$ nano src/app/certificates/certificates.component.spec.ts
$ nano src/app/languages/languages.component.spec.ts
$ nano src/app/interests/interests.component.spec.ts

No debe haber pruebas que fallen, si ejecutamos ng test.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  mycv git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

31 12 2024 02:38:15.254:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
31 12 2024 02:38:15.260:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
31 12 2024 02:38:15.271:INFO [launcher]: Starting browser ChromeHeadless
31 12 2024 02:38:17.638:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket v3oUkGXmO9
8DB7FVAAAB with id 10423933
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 5 of 17 (skipped 12) SUCCESS (0.092 secs / 0.069 sec
s)
TOTAL: 5 SUCCESS
➜  mycv git:(master) ✗
```

Pruebas unitarias en Angular

Las pruebas Angular son una parte integral del desarrollo de aplicaciones Angular. Estas pruebas permiten aislar y verificar el funcionamiento de bloques de código individuales, asegurando que funcionen correctamente antes de integrarlos en el sistema completo. Aquí hay algunos puntos clave sobre cómo realizar pruebas unitarias en Angular:

Estas pruebas unitarias son esenciales para garantizar la calidad y estabilidad del código en aplicaciones Angular.

Solo para efectos de aprendizaje crearemos algunos ejemplos de pruebas unitarias dentro de nuestro proyecto, aunque no aplican directamente a nuestro CV.

Probamos con una simple clase de contador llamada **compute**. Nos aseguramos de estar en la raíz del proyecto.

$ cd src/app

$ mkdir compute

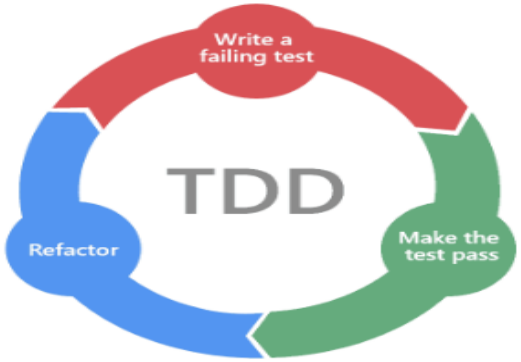
$ cd compute

$ touch compute.spec.ts

$ touch compute.ts

```
→  mycv git:(master) ✗ cd src/app
→  app git:(master) ✗ mkdir compute
→  app git:(master) ✗ cd compute
→  compute git:(master) ✗ touch compute.spec.ts
→  compute git:(master) ✗ touch compute.ts
→  compute git:(master) ✗ ls -la
total 0
drwxr-xr-x   4 adsoft  staff  128 Dec 31 18:02 .
drwxr-xr-x  18 adsoft  staff  576 Dec 31 18:02 ..
-rw-r--r--   1 adsoft  staff    0 Dec 31 18:02 compute.spec.ts
-rw-r--r--   1 adsoft  staff    0 Dec 31 18:02 compute.ts
→  compute git:(master) ✗
```

El desarrollo dirigido por pruebas se puede resumir en la siguiente figura



1.- Crear pruebas que fallen, ejemplo crearemos intencionalmente en el framework Jasmine pruebas unitarias que fallen en el archivo ***compute.spec.ts***, aplican sobre una clase llamada compute, que aún no existe y no tiene métodos.

$ nano compute.spec.ts

```
import { compute } from './compute';

describe('compute test suite', () => {
  it('Should return 0 if input is negative', () => {
          const result = compute(-1);
     expect(result).toBe(0);
  })

  it('Should increment the input if it is positive', () => {
          const result = compute(1);
          expect(result).toBe(2);
  })
})
```

Hacemos que fallen las pruebas unitarias con ng test.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  compute git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

Error: compute.spec.ts:1:25 - error TS2306: File '/Users/adsoft/mycv/src/app/compute/compute.ts' is not a
 module.

1 import { compute } from './compute';
                          ~~~~~~~~~~~


31 12 2024 19:20:39.243:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
31 12 2024 19:20:39.249:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
31 12 2024 19:20:39.250:ERROR [karma-server]: Error: Found 1 load error
    at Server.<anonymous> (/Users/adsoft/mycv/node_modules/karma/lib/server.js:243:26)
    at Object.onceWrapper (node:events:632:28)
    at Server.emit (node:events:530:35)
    at emitListeningNT (node:net:1906:10)
    at process.processTicksAndRejections (node:internal/process/task_queues:81:21)
➜  compute git:(master) ✗
```

2.- Crear código mínimo e indispensable en TypeScript en la clase ***compute***, para que pase las pruebas, aunque no el código no sea el optimo o tenga HardCode.

```
export function compute(x: number) {
  if (x < 0)
   return 0;

  if (x > 0)
   return x+1;

  return 0;
}
```

Probamos nuevamente que las pruebas pasen:

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  compute git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
02 01 2025 15:23:20.778:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 15:23:20.781:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 15:23:20.789:INFO [launcher]: Starting browser ChromeHeadless
02 01 2025 15:23:23.818:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket RkxvgmKh5O
xueQm0AAAB with id 52483518
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 2 of 19 (skipped 17) SUCCESS (0.03 secs / 0.005 secs
)
TOTAL: 2 SUCCESS
➜  compute git:(master) ✗
```

3.- Refactorizar el código existente en caso de ser necesario, para optimizarlo y hacerlo más elegante y eficiente

```
1 export function compute(x : number) {
2     if (x<0)
3       return 0
4
5     return x+1;
6 }
```

Probamos nuevamente que no se rompan las pruebas.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  compute git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
02 01 2025 15:23:20.778:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 15:23:20.781:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 15:23:20.789:INFO [launcher]: Starting browser ChromeHeadless
02 01 2025 15:23:23.818:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket RkxvgmKh5O
xueQm0AAAB with id 52483518
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 2 of 19 (skipped 17) SUCCESS (0.03 secs / 0.005 secs
)
TOTAL: 2 SUCCESS
➜  compute git:(master) ✗
```

Ejemplo Greet.

Regresamos un nivel en los directorios para llegar a ***src/app***, y creamos la carpeta greet.

$ cd ..
$ mkdir greet
$ cd greet
$ touch greet.spec.ts
$ touch greet.ts

```
➜  compute git:(master) ✗ cd ..
➜  app git:(master) ✗ mkdir greet
➜  app git:(master) ✗ cd greet
➜  greet git:(master) ✗ touch greet.spec.ts
➜  greet git:(master) ✗ touch greet.ts
➜  greet git:(master) ✗ ls -la
total 0
drwxr-xr-x   4 adsoft  staff  128 Jan  2 15:56 .
drwxr-xr-x  19 adsoft  staff  608 Jan  2 15:55 ..
-rw-r--r--   1 adsoft  staff    0 Jan  2 15:55 greet.spec.ts
-rw-r--r--   1 adsoft  staff    0 Jan  2 15:56 greet.ts
➜  greet git:(master) ✗ pwd
/Users/adsoft/mycv/src/app/greet
➜  greet git:(master) ✗
```

1.- Crear pruebas que fallen para greet

$ nano greet.spec.ts

```
1 import { greet} from './greet';
2
3 describe ('greet', () => {  // greet suit
4
5     it ('should include the name in the message ', () => {
6         expect(greet('adsoft')).toContain('adsoft');
7     } )
8
9 })
```

Hacemos que fallen las pruebas

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  greet git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI


Error: greet.spec.ts:1:22 - error TS2306: File '/Users/adsoft/mycv/src/app/greet/greet.ts' is not a modul
e.

1 import { greet} from './greet';
                      ~~~~~~~~~


02 01 2025 17:00:44.088:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 17:00:44.089:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 17:00:44.090:ERROR [karma-server]: Error: Found 1 load error
    at Server.<anonymous> (/Users/adsoft/mycv/node_modules/karma/lib/server.js:243:26)
    at Object.onceWrapper (node:events:632:28)
    at Server.emit (node:events:530:35)
    at emitListeningNT (node:net:1906:10)
    at process.processTicksAndRejections (node:internal/process/task_queues:81:21)
→  greet git:(master) ✗
```

2.- Crear código mínimo e indispensable en Typescript que pase las pruebas

```
1 export function greet(name : String) {
2   let message = 'Welcome';
3   message = message  + name
4   return message;
5 }
```

Probamos nuevamente que las pruebas pasen:

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  greet git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

02 01 2025 17:03:42.365:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 17:03:42.368:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 17:03:42.376:INFO [launcher]: Starting browser ChromeHeadless
02 01 2025 17:03:45.489:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket L2Oka9Otar
Y-S2eXAAAB with id 20110978
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 3 of 20 (skipped 17) SUCCESS (0.027 secs / 0.006 sec
s)
TOTAL: 3 SUCCESS
→  greet git:(master) ✗
```

3.- Refactorizar el código existente en caso de ser necesario.

Mejoramos el código, para ser más eficiente:

```
1 export function greet(name : String) {
2   return 'Welcome ' + name;
3 }
```

Probamos nuevamente que no se rompan las pruebas.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  greet git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

02 01 2025 17:07:44.548:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 17:07:44.551:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 17:07:44.560:INFO [launcher]: Starting browser ChromeHeadless
02 01 2025 17:07:47.262:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket QuRAChQJr2
nagXy7AAAB with id 99872862
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 3 of 20 (skipped 17) SUCCESS (0.031 secs / 0.011 sec
s)
TOTAL: 3 SUCCESS
➜  greet git:(master) ✗
```

Finalmente, realizamos la clase **Currencies**

$ cd ..
$ mkdir currencies
$ cd currencies
$ touch currencies.spec.ts
$ touch currencies.ts

```
➜  greet git:(master) ✗ cd ..
➜  app git:(master) ✗ mkdir currencies
➜  app git:(master) ✗ cd currencies
➜  currencies git:(master) ✗ touch currencies.spec.ts
➜  currencies git:(master) ✗ touch currencies.ts
➜  currencies git:(master) ✗ ls -la
total 0
drwxr-xr-x   4 adsoft  staff  128 Jan  2 17:31 .
drwxr-xr-x  20 adsoft  staff  640 Jan  2 17:30 ..
-rw-r--r--   1 adsoft  staff    0 Jan  2 17:31 currencies.spec.ts
-rw-r--r--   1 adsoft  staff    0 Jan  2 17:31 currencies.ts
➜  currencies git:(master) ✗
```

1.- Crear pruebas que fallen para **currencies**

$ nano currencies.spec.ts

```
1 import { getCurrencies } from './currencies';
2
3 describe('currencies', () => {  // currencies suite
4     it ('should return the supported currencies', () => {
5         const result = getCurrencies();
6
7         expect(result).toContain('USD');
8         expect(result).toContain('AUD');
9         expect(result).toContain('EUR');
10
11    })
12 })
```

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  currencies git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI


Error: currencies.spec.ts:1:31 - error TS2306: File '/Users/adsoft/mycv/src/app/currencies/currencies.ts'
 is not a module.

1 import { getCurrencies } from './currencies';
                               ~~~~~~~~~~~~~~~


02 01 2025 17:55:30.808:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 17:55:30.810:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 17:55:30.810:ERROR [karma-server]: Error: Found 1 load error
    at Server.<anonymous> (/Users/adsoft/mycv/node_modules/karma/lib/server.js:243:26)
    at Object.onceWrapper (node:events:632:28)
    at Server.emit (node:events:530:35)
    at emitListeningNT (node:net:1906:10)
    at process.processTicksAndRejections (node:internal/process/task_queues:81:21)
→  currencies git:(master) ✗
```

2.- Crear código mínimo e indispensable que pase las pruebas

```
1 export function getCurrencies() {
2   var currencies = [];
3   currencies.push('USD');
4   currencies.push('AUD');
5   currencies.push('EUR');
6
7   return currencies;
8 }
```

Probamos nuevamente que las pruebas no fallen:

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  currencies git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

02 01 2025 18:16:28.914:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 18:16:28.917:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 18:16:28.927:INFO [launcher]: Starting browser ChromeHeadless
02 01 2025 18:16:32.493:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket NQ46m22vsR
3o4Hp5AAAB with id 66255897
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 4 of 21 (skipped 17) SUCCESS (0.022 secs / 0.004 sec
s)
TOTAL: 4 SUCCESS
```

3.- Refactorizar el código existente para hacerlo ser más eficiente:

```
1 export function getCurrencies() {
2   return ['USD', 'AUD', 'EUR'];
3 }
```

Probamos nuevamente que no se rompan las pruebas.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  currencies git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

02 01 2025 18:20:56.697:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 18:20:56.699:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 18:20:56.707:INFO [launcher]: Starting browser ChromeHeadless
02 01 2025 18:20:59.459:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket sGWxW2Qpw5
RcZdbtAAAB with id 24023954
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 4 of 21 (skipped 17) SUCCESS (0.035 secs / 0.005 sec
s)
TOTAL: 4 SUCCESS
```

Ahora podemos probar la cobertura de código de nuestro proyecto, es decir que porcentaje de código Typescript está cubierto con pruebas en el framework Jasmine.

Para obtener la cobertura de código agregamos el parámetro ***--code-coverage.***

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI --code-coverage

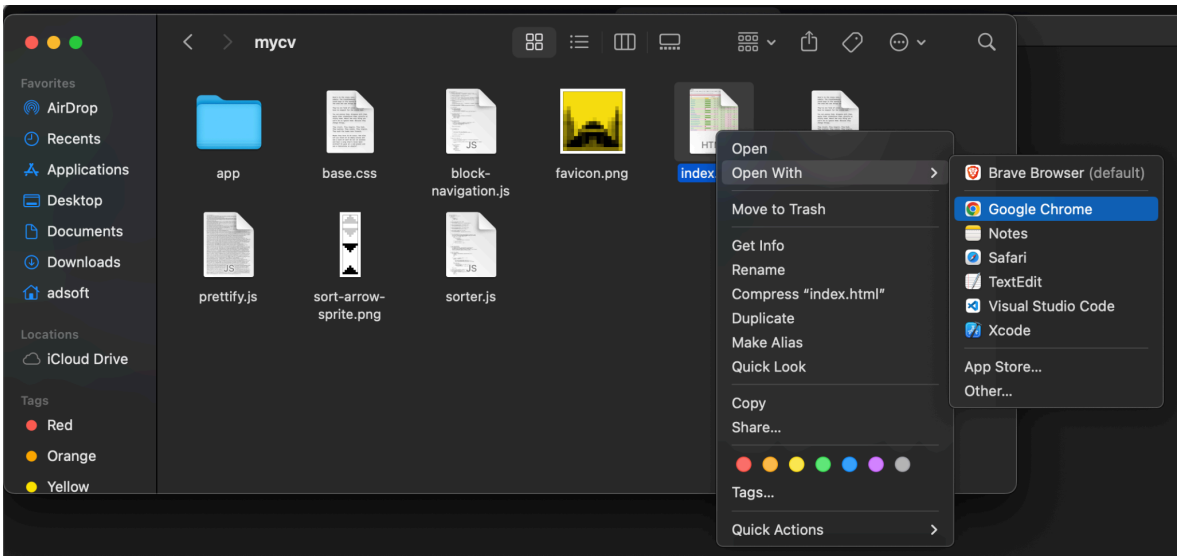
```
→  mycv git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI --code-coverage

02 01 2025 18:42:29.983:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
02 01 2025 18:42:29.989:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
02 01 2025 18:42:29.997:INFO [launcher]: Starting browser ChromeHeadless
02 01 2025 18:42:34.155:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket ytY0duGK9q
AmHwJtAAAB with id 19169901
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 4 of 21 (skipped 17) SUCCESS (0.069 secs / 0.006 sec
s)
TOTAL: 4 SUCCESS

============================== Coverage summary ==============================
Statements   : 43.33% ( 26/60 )
Branches     : 100% ( 1/1 )
Functions    : 13.63% ( 3/22 )
Lines        : 37.03% ( 20/54 )
================================================================================
→  mycv git:(master) ✗
```

Este proceso nos creará una carpeta en la raíz del proyecto llamada **coverage**, con un reporte en HTML con los detalles del reporte de cobertura de código. Ir a la raíz del proyecto

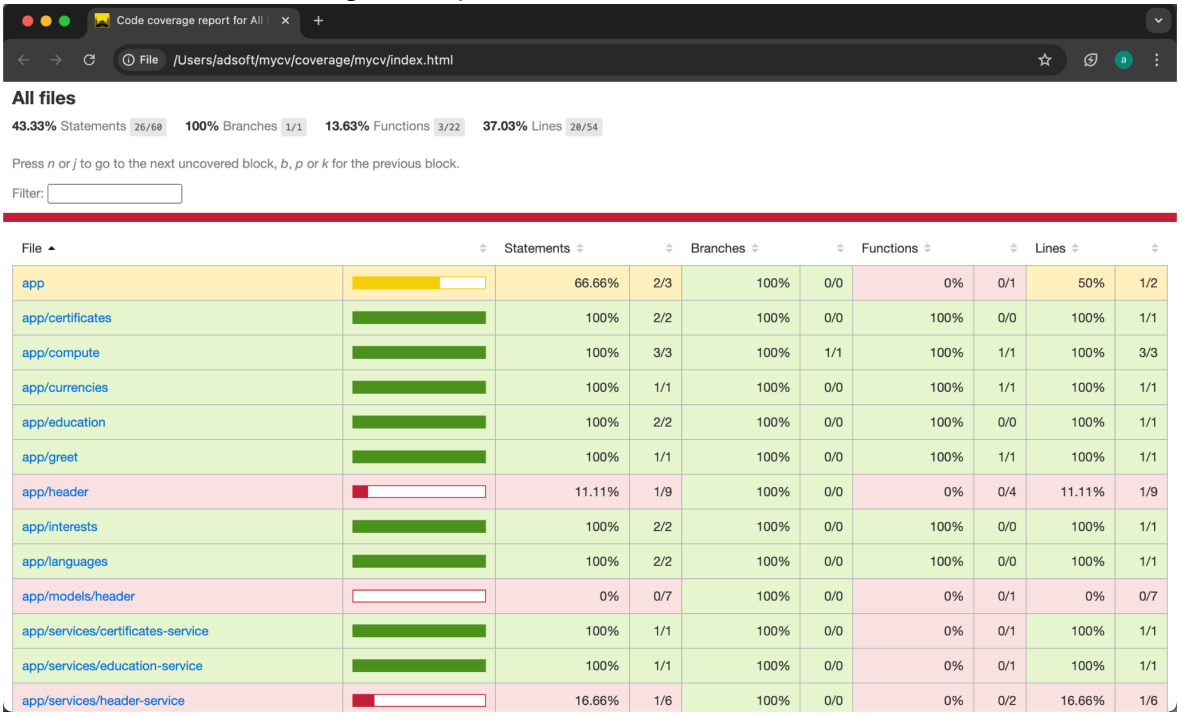
$ ls coverage
$ ls coverage/mycv

```
→  mycv git:(master) ✗ ls
README.md           karma.conf.js       package.json        tsconfig.app.json
angular.json        node_modules        public              tsconfig.json
coverage            package-lock.json   src                 tsconfig.spec.json
→  mycv git:(master) ✗ ls coverage/
mycv
→  mycv git:(master) ✗ ls coverage/mycv
app                 favicon.png         prettify.js
base.css            index.html          sort-arrow-sprite.png
block-navigation.js prettify.css        sorter.js
→  mycv git:(master) ✗
```

Al abrir el archivo ***index.html*** con el navegador de Google Chrome, podemos visualizar el reporte de código.

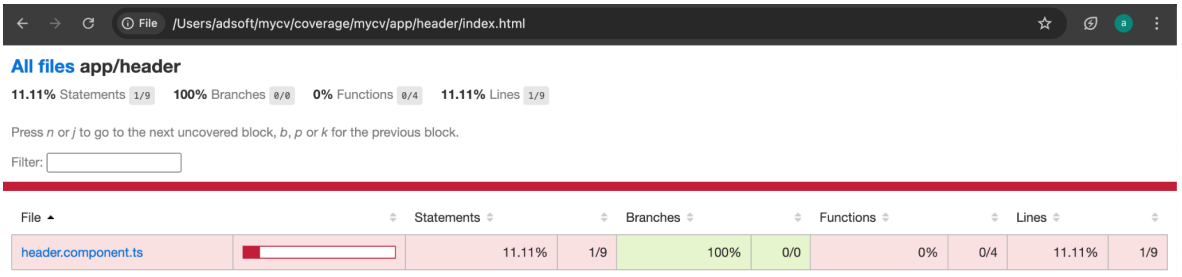


Podemos visualizar la siguiente pantalla con información:



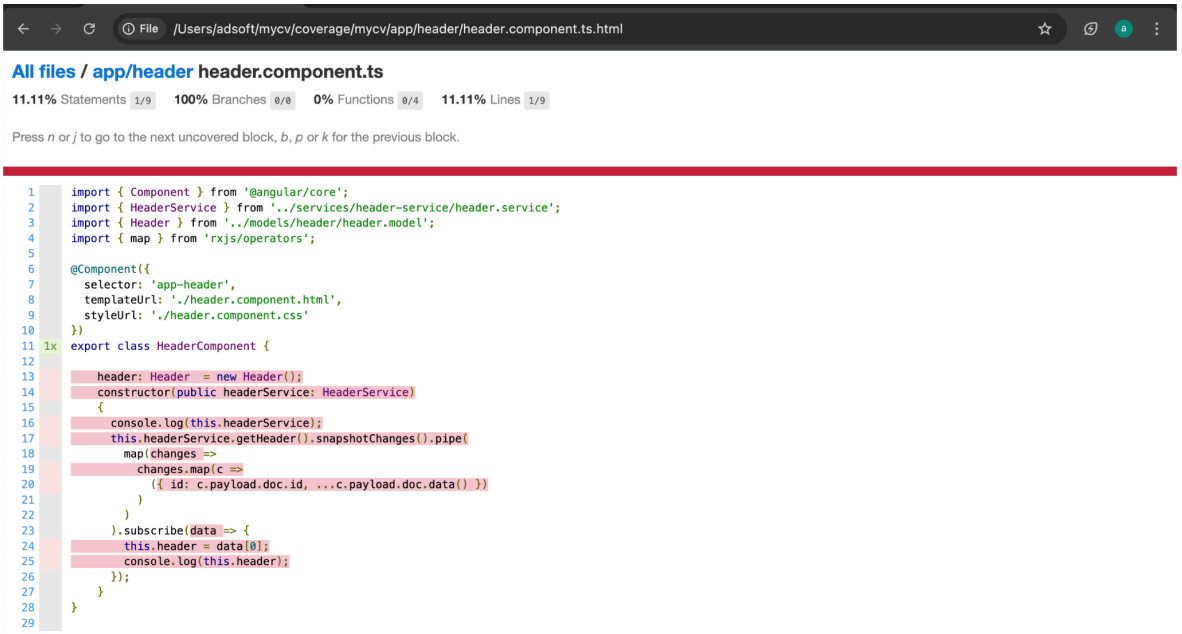
File	Statements		Branches		Functions		Lines	
app	66.66%	2/3	100%	0/0	0%	0/1	50%	1/2
app/certificates	100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/compute	100%	3/3	100%	1/1	100%	1/1	100%	3/3
app/currencies	100%	1/1	100%	0/0	100%	1/1	100%	1/1
app/education	100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/greet	100%	1/1	100%	0/0	100%	1/1	100%	1/1
app/header	11.11%	1/9	100%	0/0	0%	0/4	11.11%	1/9
app/interests	100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/languages	100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/models/header	0%	0/7	100%	0/0	0%	0/1	0%	0/7
app/services/certificates-service	100%	1/1	100%	0/0	0%	0/1	100%	1/1
app/services/education-service	100%	1/1	100%	0/0	0%	0/1	100%	1/1
app/services/header-service	16.66%	1/6	100%	0/0	0%	0/2	16.66%	1/6

Podemos navegar en los archivos .ts donde la cobertura de código sea menor del 100%, ejemplo: app/header.



File	Statements		Branches		Functions		Lines	
header.component.ts	11.11%	1/9	100%	0/0	0%	0/4	11.11%	1/9

Luego seleccionamos el archivo ***header.component.ts***



Nos muestras las líneas en rojo, aquellas líneas o métodos que no tienen pruebas asociadas. Puedes repetir el proceso para cada componente o servicio que no tenga el 100% de cobertura de código.

Hasta ahora ya tienes algunas herramientas que te permitirán construir pruebas unitarias básicas en Angular y aplicarlas a tu proyecto con el objetivo de mejorar la calidad del código.

Sala de inspiración

El conocimiento que has adquirido hasta ahora, te permite construir pruebas unitarias dentro un proyecto de Angular 18, podemos resumirlo en los siguientes pasos.

- Crear archivo de configuración karma.conf.js
- Agregar el navegador ChromeHeadlessCI
- Verificar que ng test funcione correctamente
- Crear pruebas unitarias que falle en Jasmine
- Crear código en Typescript que pase las pruebas unitarias
- Refactorizar el código existente.

Las empresas en la actualidad usan Angular porque además de permitir la creación de plataformas web de manera muy práctica y sencilla, es un framework orientado hacia el desarrollo dirigido por pruebas e incluye todas las herramientas necesarias para trabajar en este enfoque.

Sala de pruebas

Lee cada una de las preguntas y selecciona la respuesta que consideres correcta de acuerdo a tu experiencia hasta ahora.

1. *Comando para crear el archivo karma.conf.js*

Opción a: ng generate karma	Incorrecta: *El comando de Angular CLI correcto es:* ng generate config karma
Opción b: ng config karma	Incorrecta: *El comando de Angular CLI correcto es:* ng generate config karma
Opción c: ng create config karma	Incorrecta: *El comando de Angular CLI correcto es:* ng generate config karma
Opción d: ng generate config karma	*CORRECTA:* *El comando de Angular CLI correcto es:* ng generate config karma

2. *Cual de las siguientes es una prueba que karma debe ignorar*

Opción a:
 	
#it('Should add 2 + 2 = 4', () => {
let result = 0;
result = addition(2, 2);
expect(result).toBe(4);
})
	Incorrecta
	
	La sintaxis correcta en Jasmine para ignorar una prueba es colocar una x antes de it(...

	xit('Should add 2 + 2 = 4', () => {

Opción b:
 	
//it('Should add 2 + 2 = 4', () => {
let result = 0;
result = addition(2, 2);
expect(result).toBe(4);
})
	Incorrecta
	
	La sintaxis correcta en Jasmine para ignorar una prueba es colocar una x antes de it(...
	
	xit('Should add 2 + 2 = 4', () => {

Opción c:
 	
xit('Should add 2 + 2 = 4', () => {
let result = 0;
result = addition(2, 2);
expect(result).toBe(4);
})
	CORRECTA
	
	La sintaxis correcta en Jasmine para ignorar una prueba es colocar una x antes de it(...
	
	xit('Should add 2 + 2 = 4', () => {

Opción d:
 	
--it('Should add 2 + 2 = 4', () => {
let result = 0;
result = addition(2, 2);
expect(result).toBe(4);
})
	Incorrecta:
	
	La sintaxis correcta en Jasmine para ignorar una prueba es colocar una x antes de it(...
	
	xit('Should add 2 + 2 = 4', () => {

3.- *Comando para correr las pruebas y generar reporte de cobertura de codigo*

Opción a: ng --no-watch --no-progress --browsers=ChromeHeadlessCI **--code-coverage**	Incorrecta: El comando de Angular CLI correcto para generar reporte de cobertura de codigo es: ng test --no-watch --no-progress --browsers=ChromeHeadlessCI **--code-coverage**
Opción b:	Incorrecta:

ng test --no-watch --no-progress --browsers=ChromeHeadlessCI	El comando de Angular CLI correcto para generar reporte de cobertura de codigo es: ng test --no-watch --no-progress --browsers=ChromeHeadlessCI **--code-coverage**
Opción c: ng test --no-watch --no-progress --browsers=ChromeHeadlessCI **--code-coverage**	*CORRECTA* El comando de Angular CLI correcto para generar reporte de cobertura de codigo es: ng test --no-watch --no-progress --browsers=ChromeHeadlessCI **--code-coverage**
Opción d: ng test	Incorrecta: El comando de Angular CLI correcto para generar reporte de cobertura de codigo es: ng test --no-watch --no-progress --browsers=ChromeHeadlessCI **--code-coverage**

SUBTEMA 2: *Creación de pruebas de integración y pruebas end to end*

Las pruebas unitarias permiten probar partes aisladas del código como cálculos matemáticos, algoritmos, estructuras de datos, entre otros. Son rápidas, fáciles de crear y mantener. Sin embargo no son suficientes para probar un proyecto Angular a nivel de componentes o nivel funcionalidad global.

Por otro lado, las pruebas de integración nos permiten probar un componente a nivel de interacción del código TypeScript con su HTML, o bien la integración de un componente con un servicio. Asimismo las pruebas de extremo a extremo (end to end) son lentas y comprueban el sistema entero en términos de reglas de negocio o funcionalidad del cliente.

Primero, vamos a trabajar con las pruebas de integración. Crearemos pruebas de integración en Jasmine para nuestros servicios, empezamos quitando el comentario a las pruebas del servicio Header.

Nos aseguramos de estar en la raíz del proyecto, y nos movemos a src/app

$ ls -la

$ cd src/app

$ pwd

```
➜  mycv git:(master) ✗ ls -la
total 1128
drwxr-xr-x   20 adsoft  staff     640 Jan  2 18:55 .
drwxr-x---+  80 adsoft  staff    2560 Jan  3 00:14 ..
-rw-r--r--@   1 adsoft  staff    6148 Jan  2 18:55 .DS_Store
drwxr-xr-x    3 adsoft  staff      96 Dec 26 01:33 .angular
-rw-r--r--    1 adsoft  staff     274 Dec 26 01:30 .editorconfig
drwxr-xr-x   12 adsoft  staff     384 Dec 26 01:31 .git
-rw-r--r--    1 adsoft  staff     587 Dec 26 01:30 .gitignore
drwxr-xr-x    5 adsoft  staff     160 Dec 26 01:30 .vscode
-rw-r--r--    1 adsoft  staff    1065 Dec 26 01:30 README.md
-rw-r--r--    1 adsoft  staff    3104 Dec 31 01:28 angular.json
drwxr-xr-x    4 adsoft  staff     128 Jan  2 18:55 coverage
-rw-r--r--    1 adsoft  staff    1421 Dec 31 02:09 karma.conf.js
drwxr-xr-x  612 adsoft  staff   19584 Dec 26 17:03 node_modules
-rw-r--r--    1 adsoft  staff  530221 Dec 26 17:03 package-lock.json
-rw-r--r--    1 adsoft  staff    1067 Dec 26 17:03 package.json
drwxr-xr-x    3 adsoft  staff      96 Dec 26 01:30 public
drwxr-xr-x    7 adsoft  staff     224 Dec 26 17:04 src
-rw-r--r--    1 adsoft  staff     424 Dec 26 01:30 tsconfig.app.json
-rw-r--r--    1 adsoft  staff    1021 Dec 26 01:30 tsconfig.json
-rw-r--r--    1 adsoft  staff     434 Dec 26 01:30 tsconfig.spec.json
➜  mycv git:(master) ✗ cd src/app
➜  app git:(master) ✗ pwd
/Users/adsoft/mycv/src/app
➜  app git:(master) ✗
```

Pruebas de integración del Service header.

Abrimos ***header.service.spec.ts*** y quitamos el comentario a la prueba por default.

$ nano services/header-service/header.service.spec.ts

```
17      service = TestBed.inject(HeaderService);
18    });
19
20
21
22    it('should be created', () => {
23      expect(service).toBeTruthy();
24    });
```

Si probamos con ng test, esta prueba deberá fallar.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

03 01 2025 00:40:17.697:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
03 01 2025 00:40:17.712:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
03 01 2025 00:40:17.718:INFO [launcher]: Starting browser ChromeHeadless
03 01 2025 00:40:19.593:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket bPEvKy97N2Qb057cAAAB
 with id 73351626
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 2 of 23 SUCCESS (0 secs / 0.083 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 4 of 23 (skipped 2) SUCCESS (0 secs / 0.083 Chrome Headless 13
1.0.0.0 (Mac OS 10.15.7): Executed 5 of 23 (skipped 2) SUCCESS (0 secs / 0.084 Chrome Headless 131.0.0.0 (Mac OS 10
.15.7): Executed 6 of 23 (skipped 7) SUCCESS (0 secs / 0.085 Chrome Headless 131.0.0.0 (Mac OS 10.15.7) HeaderServi
ce should be created FAILED
        NullInjectorError: R3InjectorError(DynamicTestModule)[HeaderService -> AngularFirestore -> InjectionToken a
ngularfire2.app.options -> InjectionToken angularfire2.app.options]:
          NullInjectorError: No provider for InjectionToken angularfire2.app.options!
        error properties: Object({ ngTempTokenPath: null, ngTokenPath: [ 'HeaderService', 'AngularFirestore', 'Inje
ctionToken angularfire2.app.options', 'InjectionToken angularfire2.app.options' ] })
            at NullInjector.get (node_modules/@angular/core/fesm2022/core.mjs:1663:27)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at injectInjectorOnly (node_modules/@angular/core/fesm2022/core.mjs:1107:40)
            at Module.ɵɵinject (node_modules/@angular/core/fesm2022/core.mjs:1113:60)
            at Object.factory (node_modules/@angular/fire/fesm2022/angular-fire-compat-firestore.mjs:604:96)
            at callback (node_modules/@angular/core/fesm2022/core.mjs:3237:47)
            at runInInjectorProfilerContext (node_modules/@angular/core/fesm2022/core.mjs:872:9)
            at R3Injector.hydrate (node_modules/@angular/core/fesm2022/core.mjs:3236:21)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3099:33)
```

Ahora, configuramos ***header.service.spec.ts***, para soportar el servicio de Firebase.

Importamos las keys de acceso de Firebase en la línea (3), y el módulo de ***AngularFireModule*** en la línea 4.

En la línea 7, definimos una instancia de ***HeaderService***. En las líneas 11 a 13, inicializamos el módulo de ***AngularFireModule***.

```
1  import { TestBed } from '@angular/core/testing';
2  import { HeaderService } from './header.service';
3  import { environment } from '../../../environments/environment';
4  import { AngularFireModule } from '@angular/fire/compat';
5
6  describe('HeaderService', () => {
7    let service: HeaderService;
8
9    beforeEach(() => {
10     TestBed.configureTestingModule({
11       imports: [
12         AngularFireModule.initializeApp(environment.firebaseConfig),
13       ],
14     });
15     service = TestBed.inject(HeaderService);
16   });
17
18
19
20   it('should be created', () => {
21     expect(service).toBeTruthy();
22   });
```

Si probamos nuevamente con ***ng test***, la prueba del servicio ***'should be created'*** debe pasar.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

03 01 2025 01:01:07.326:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
03 01 2025 01:01:07.340:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
03 01 2025 01:01:07.399:INFO [launcher]: Starting browser ChromeHeadless
03 01 2025 01:01:09.220:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket muc1OLixMwmiKyTOAAAB
 with id 48911461
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 5 of 22 (skipped 7) SUCCESS (0 secs / 0.093 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 6 of 22 (skipped 7) SUCCESS (0 secs / 0.1 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 7 of 22 (skipped 15) SUCCESS (0.15 secs / 0.103 secs)
TOTAL: 7 SUCCESS
```

Agregamos una nueva prueba que permita validar que el método ***getHeader()*** esté funcionando correctamente (líneas 24 a 26), en nuestro caso que no retorna null si no cualquier documento de la colección ***Header***.

```
19
20    it('should be created', () => {
21      expect(service).toBeTruthy();
22    });
23
24    it('should get Header', () => {
25      expect(service.getHeader()).not.toBeNull();
26    });
27
28  });
```

Probamos que la prueba de getHeader, pase correctamente:

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  app git:(master) × ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

03 01 2025 01:23:33.376:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
03 01 2025 01:23:33.385:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
03 01 2025 01:23:33.392:INFO [launcher]: Starting browser ChromeHeadless
03 01 2025 01:23:35.178:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket eJD58C92No7UrLZcAAAB
 with id 46157436
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 4 of 23 (skipped 14) SUCCESS (0 secs / 0.094 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 8 of 23 (skipped 15) SUCCESS (0.139 secs / 0.101 secs)
TOTAL: 8 SUCCESS
```

Pruebas de integración del Componente Header

En el componente Header probaremos la integración del servicio ***Header*** en el componente a través de su constructor, es turno de modificar el componente **Header**, vamos a quitar el comentario de ***header.component.spect.ts***

$ nano header/header.component.spec.ts

```
26
27    it('should create', () => {
28      expect(component).toBeTruthy();
29    });
```

Probamos con ng test, la prueba ***'should create'*** deberá fallar

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
03 01 2025 01:47:07.471:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
03 01 2025 01:47:07.473:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
03 01 2025 01:47:07.483:INFO [launcher]: Starting browser ChromeHeadless
03 01 2025 01:47:09.304:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket B8Fz05TQf6
kUd1KOAAAB with id 63749541
Chrome Headless 131.0.0.0 (Mac OS 10.15.7) HeaderComponent should create FAILED
        NullInjectorError: R3InjectorError(DynamicTestModule)[HeaderService -> AngularFirestore -> Inject
ionToken angularfire2.app.options -> InjectionToken angularfire2.app.options]:
          NullInjectorError: No provider for InjectionToken angularfire2.app.options!
        error properties: Object({ ngTempTokenPath: null, ngTokenPath: [ 'HeaderService', 'AngularFiresto
re', 'InjectionToken angularfire2.app.options', 'InjectionToken angularfire2.app.options' ] })
            at NullInjector.get (node_modules/@angular/core/fesm2022/core.mjs:1663:27)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at injectInjectorOnly (node_modules/@angular/core/fesm2022/core.mjs:1107:40)
            at Module.ɵɵinject (node_modules/@angular/core/fesm2022/core.mjs:1113:60)
            at Object.factory (node_modules/@angular/fire/fesm2022/angular-fire-compat-firestore.mjs:604:
96)
            at callback (node_modules/@angular/core/fesm2022/core.mjs:3237:47)
            at runInInjectorProfilerContext (node_modules/@angular/core/fesm2022/core.mjs:872:9)
            at R3Injector.hydrate (node_modules/@angular/core/fesm2022/core.mjs:3236:21)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3099:33)
```

Ahora, configuramos ***header.component.spec.ts***, para soportar el servicio de Firebase.

Importamos el módulo de ***AngularFireModule*** en la línea 3, las keys de acceso de Firebase en la línea (4), asimismo se importa el servicio ***HeaderService*** en la línea 5.

En la línea 10, definimos una instancia de ***HeaderService*** llamada **service**. En las líneas 13 a 18, inicializamos el módulo de ***AngularFireModule*** con las keys de acceso de ***environment***.

En la línea 24 inyectamos el ***HeaderService*** en nuestra instancia **service**.

$ nano header/header.component.spec.ts

```
 2 import { HeaderComponent } from './header.component';
 3 import { AngularFireModule } from '@angular/fire/compat';
 4 import { environment } from '../../environments/environment';
 5 import { HeaderService } from '../services/header-service/header.service';
 6
 7 describe('HeaderComponent', () => {
 8   let component: HeaderComponent;
 9   let fixture: ComponentFixture<HeaderComponent>;
10   let service: HeaderService;
11
12   beforeEach(async () => {
13     await TestBed.configureTestingModule({
14       declarations: [HeaderComponent],
15       imports: [
16         AngularFireModule.initializeApp(environment.firebaseConfig),
17       ],
18     })
19     .compileComponents();
20
21     fixture = TestBed.createComponent(HeaderComponent);
22     component = fixture.componentInstance;
23     fixture.detectChanges();
24     service = TestBed.inject(HeaderService);
25   });
26
27   it('should create', () => {
28     expect(component).toBeTruthy();
29   });
```

Probamos nuevamente, la prueba ‘should create’ deberá pasar.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
03 01 2025 01:58:16.203:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
03 01 2025 01:58:16.209:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
03 01 2025 01:58:16.216:INFO [launcher]: Starting browser ChromeHeadless
03 01 2025 01:58:18.209:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket O78pSI2Deyt3xuZEAAAB
 with id 97298130
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 1 of 23 (skipped 3) SUCCESS (0 secs / 0.08 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 7 of 23 (skipped 16) SUCCESS (0.138 secs / 0.091 secs)
TOTAL: 7 SUCCESS
```

Agregamos una prueba nueva, para probar que el servicio ejecute correctamente el método ***getHeader()*** del servicio (líneas 31 a 33)

$ nano header/header.component.spec.ts

```
    fixture = TestBed.createComponent(HeaderComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
    service = TestBed.inject(HeaderService);
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  it('should get Header documents from Firestore', () => {
    expect(service.getHeader()).not.toBeNull();
  });

});
```

Si ejecutamos nuevamente ng test, la prueba nueva deberá pasar

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
03 01 2025 02:03:06.621:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
03 01 2025 02:03:06.624:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
03 01 2025 02:03:06.630:INFO [launcher]: Starting browser ChromeHeadless
03 01 2025 02:03:08.356:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket IiWJRzUulydwR3WmAAAB
 with id 74850458
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 7 of 23 (skipped 14) SUCCESS (0 secs / 0.092 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 8 of 23 (skipped 15) SUCCESS (0.124 secs / 0.098 secs)
TOTAL: 8 SUCCESS
```

Repetimos el proceso para el servicio ***work-experience***, y el componente ***work-experience***.

Pruebas de integración del Service Work-Experience.

Asegúrate de continuar en la carpeta ***src/app/***

```
➜  app git:(master) ✗ pwd
/Users/adsoft/mycv/src/app
➜  app git:(master) ✗ ls -la
total 40
drwxr-xr-x  20 adsoft  staff   640 Jan  2 20:10 .
drwxr-xr-x   7 adsoft  staff   224 Dec 26 17:04 ..
-rw-r--r--   1 adsoft  staff   245 Dec 26 01:30 app-routing.module.ts
-rw-r--r--   1 adsoft  staff     0 Dec 26 01:30 app.component.css
-rw-r--r--   1 adsoft  staff   422 Dec 26 01:57 app.component.html
-rw-r--r--   1 adsoft  staff  1040 Dec 31 16:39 app.component.spec.ts
-rw-r--r--   1 adsoft  staff   205 Dec 26 01:30 app.component.ts
-rw-r--r--   1 adsoft  staff  1245 Jan  2 20:10 app.module.ts
drwxr-xr-x   6 adsoft  staff   192 Dec 31 17:49 certificates
drwxr-xr-x   4 adsoft  staff   128 Jan  2 15:22 compute
drwxr-xr-x   4 adsoft  staff   128 Jan  2 18:20 currencies
drwxr-xr-x   6 adsoft  staff   192 Dec 26 01:46 education
drwxr-xr-x   4 adsoft  staff   128 Jan  2 17:07 greet
drwxr-xr-x   6 adsoft  staff   192 Jan  3 02:02 header
drwxr-xr-x   6 adsoft  staff   192 Dec 31 17:50 interests
drwxr-xr-x   6 adsoft  staff   192 Dec 31 17:48 languages
drwxr-xr-x   4 adsoft  staff   128 Dec 26 21:11 models
drwxr-xr-x   9 adsoft  staff   288 Dec 26 02:07 services
drwxr-xr-x   6 adsoft  staff   192 Dec 31 17:47 skills
drwxr-xr-x   6 adsoft  staff   192 Dec 31 02:37 work-experience
```

Abrimos ***work-experience.service.spec.ts*** y quitamos el comentario a la prueba por default ***'should be created'.***

$ nano services/work-experience-service/work-experience.service.spec.ts

```
 1 import { TestBed } from '@angular/core/testing';
 2
 3 import { WorkExperienceService } from './work-experience.service';
 4
 5 describe('WorkExperienceService', () => {
 6   let service: WorkExperienceService;
 7
 8   beforeEach(() => {
 9     TestBed.configureTestingModule({});
10     service = TestBed.inject(WorkExperienceService);
11   });
12
13   it('should be created', () => {
14     expect(service).toBeTruthy();
15   });
16 });
```

Si probamos con ***ng test***, esta prueba deberá fallar.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
05 01 2025 21:37:41.001:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876
/
05 01 2025 21:37:41.015:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unl
imited
05 01 2025 21:37:41.026:INFO [launcher]: Starting browser ChromeHeadless
05 01 2025 21:37:45.334:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket If
9wFRdVjnen2d7iAAAB with id 13132481
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 1 of 23 (skipped 4) SUCCESS (0 secs / 0.092
secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 2 of 23 (skipped 4) SUCCESS (0 secs / 0.098
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 3 of 23 (skipped 7) SUCCESS (0 secs / 0.099
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 4 of 23 (skipped 8) SUCCESS (0 secs / 0.1 se
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 5 of 23 (skipped 8) SUCCESS (0 secs / 0.1 se
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 6 of 23 (skipped 12) SUCCESS (0 secs / 0.1 s
Chrome Headless 131.0.0.0 (Mac OS 10.15.7) WorkExperienceService should be created FAILED
        NullInjectorError: R3InjectorError(DynamicTestModule)[WorkExperienceService -> AngularFir
estore -> InjectionToken angularfire2.app.options -> InjectionToken angularfire2.app.options]:
          NullInjectorError: No provider for InjectionToken angularfire2.app.options!
        error properties: Object({ ngTempTokenPath: null, ngTokenPath: [ 'WorkExperienceService',
 'AngularFirestore', 'InjectionToken angularfire2.app.options', 'InjectionToken angularfire2.app.
options' ] })
            at NullInjector.get (node_modules/@angular/core/fesm2022/core.mjs:1663:27)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at injectInjectorOnly (node_modules/@angular/core/fesm2022/core.mjs:1107:40)
            at Module.ɵɵinject (node_modules/@angular/core/fesm2022/core.mjs:1113:60)
            at Object.factory (node_modules/@angular/fire/fesm2022/angular-fire-compat-firestore.
```

Ahora, configuramos ***work-experience.service.spec.ts***, para soportar el servicio de Firebase.

Importamos las keys de acceso de Firebase en la línea (3), y el módulo de ***AngularFireModule*** en la línea 4.

En la línea 7, definimos una instancia de ***WorkExperienceService***. En las líneas 11 a 13, inicializamos el módulo de ***AngularFireModule***.

```
 1 import { TestBed } from '@angular/core/testing';
 2 import { WorkExperienceService } from './work-experience.service';
 3 import { environment } from '../../../environments/environment';
 4 import { AngularFireModule } from '@angular/fire/compat';
 5
 6 describe('WorkExperienceService', () => {
 7   let service: WorkExperienceService;
 8
 9   beforeEach(() => {
10     TestBed.configureTestingModule({
11       imports: [
12         AngularFireModule.initializeApp(environment.firebaseConfig),
13       ],
14     });
15     service = TestBed.inject(WorkExperienceService);
16   });
17
18   it('should be created', () => {
19     expect(service).toBeTruthy();
20   });
21 });
```

Si probamos nuevamente con ***ng test***, la prueba del servicio

WorkExperienceService 'should be created' debe pasar.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
05 01 2025 23:53:32.333:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
05 01 2025 23:53:32.336:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
05 01 2025 23:53:32.368:INFO [launcher]: Starting browser ChromeHeadless
05 01 2025 23:53:35.485:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket 7as0EYtvwn
HGj2hqAAAB with id 32296232
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 1 of 23 (skipped 4) SUCCESS (0 secs / 0.107 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 2 of 23 (skipped 4) SUCCESS (0 secs / 0.117 Chrome H
eadless 131.0.0.0 (Mac OS 10.15.7): Executed 3 of 23 (skipped 4) SUCCESS (0 secs / 0.119 Chrome Headless
131.0.0.0 (Mac OS 10.15.7): Executed 4 of 23 (skipped 8) SUCCESS (0 secs / 0.124 Chrome Headless 131.0.0.
0 (Mac OS 10.15.7): Executed 5 of 23 (skipped 8) SUCCESS (0 secs / 0.129 Chrome Headless 131.0.0.0 (Mac O
S 10.15.7): Executed 6 of 23 (skipped 9) SUCCESS (0 secs / 0.13 sChrome Headless 131.0.0.0 (Mac OS 10.15.
7): Executed 7 of 23 (skipped 9) SUCCESS (0 secs / 0.131 Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Exec
uted 8 of 23 (skipped 9) SUCCESS (0 secs / 0.132 Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 9 o
f 23 (skipped 11) SUCCESS (0 secs / 0.139Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 9 of 23 (sk
ipped 14) SUCCESS (0.299 secs / 0.139 secs)
TOTAL: 9 SUCCESS
```

Agregamos una nueva prueba que permita validar que el método ***getWorkExperience()*** esté funcionando correctamente (líneas 22 a 24), en nuestro caso que no retorna null si no cualquier documento de la colección ***work-experience***.

```
21
22   it('should get work-experience collection', () => {
23     expect(service.getWorkExperience()).not.toBeNull();
24   });
25
26 });
```

Probamos que la prueba de ***getHeader***, pase correctamente:

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
05 01 2025 23:58:30.774:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
05 01 2025 23:58:30.778:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
05 01 2025 23:58:30.786:INFO [launcher]: Starting browser ChromeHeadless
05 01 2025 23:58:33.980:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket WJEXRcQ5T5
BB-ly_AAAB with id 92517969
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 5 of 24 (skipped 2) SUCCESS (0 secs / 0.141 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 10 of 24 (skipped 14) SUCCESS (0.26 secs / 0.166 sec
s)
TOTAL: 10 SUCCESS
```

Pruebas de integración del Componente WorkExperience

En el componente ***WorkExperience*** probaremos la integración del servicio ***WorkExperience*** en el componente a través de su constructor, es turno de modificar el componente **WorkExperience**, vamos a quitar el comentario de ***work-experience.component.spect.ts***

$ nano work-experience/work-experience.component.spec.ts

```
15      fixture = TestBed.createComponent(WorkExperienceComponent);
16      component = fixture.componentInstance;
17      fixture.detectChanges();
18    });
19
20    it('should create', () => {
21      expect(component).toBeTruthy();
22    });
23  });
```

Probamos con ng test, la prueba ***'should create'*** deberá fallar

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
06 01 2025 00:12:22.185:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
06 01 2025 00:12:22.188:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
06 01 2025 00:12:22.198:INFO [launcher]: Starting browser ChromeHeadless
06 01 2025 00:12:26.155:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket 8A4onGYKX2
f1ptj4AAAB with id 51096869
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 4 of 24 SUCCESS (0 secs / 0.147 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7) WorkExperienceComponent should create FAILED
        NullInjectorError: R3InjectorError(DynamicTestModule)[WorkExperienceService -> AngularFirestore -
> InjectionToken angularfire2.app.options -> InjectionToken angularfire2.app.options]:
          NullInjectorError: No provider for InjectionToken angularfire2.app.options!
        error properties: Object({ ngTempTokenPath: null, ngTokenPath: [ 'WorkExperienceService', 'Angula
rFirestore', 'InjectionToken angularfire2.app.options', 'InjectionToken angularfire2.app.options' ] })
            at NullInjector.get (node_modules/@angular/core/fesm2022/core.mjs:1663:27)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3109:33)
            at injectInjectorOnly (node_modules/@angular/core/fesm2022/core.mjs:1107:40)
            at Module.ɵɵinject (node_modules/@angular/core/fesm2022/core.mjs:1113:60)
            at Object.factory (node_modules/@angular/fire/fesm2022/angular-fire-compat-firestore.mjs:604:
96)
            at callback (node_modules/@angular/core/fesm2022/core.mjs:3237:47)
            at runInInjectorProfilerContext (node_modules/@angular/core/fesm2022/core.mjs:872:9)
            at R3Injector.hydrate (node_modules/@angular/core/fesm2022/core.mjs:3236:21)
            at R3Injector.get (node_modules/@angular/core/fesm2022/core.mjs:3099:33)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 11 of 24 (1 FAILED) (skipped 10) (0 secs / 0.189 sec
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 11 of 24 (1 FAILED) (skipped 13) (0.282 secs / 0.189
 secs)
TOTAL: 1 FAILED, 10 SUCCESS
```

Ahora, configuramos **work-experience.component.spec.ts**, para soportar el servicio de Firebase.

Importamos el módulo de **AngularFireModule** en la línea 3, las keys de acceso de ***Firebase*** en la línea (4), asimismo se importa el servicio

WorkExperienceService en la línea 5.

En la línea 10, definimos una instancia de ***WorkExperienceService*** llamada **service**. En las líneas 15 a 17, inicializamos el módulo de ***AngularFireModule*** con las keys de acceso de ***environment***.

En la línea 24 inyectamos el ***WorkExperienceService*** en nuestra instancia **service**.

$ nano work-experience/work-experience.component.spec.ts

```
1  import { ComponentFixture, TestBed } from '@angular/core/testing';
2  import { WorkExperienceComponent } from './work-experience.component';
3  import { AngularFireModule } from '@angular/fire/compat';
4  import { environment } from '../../environments/environment';
5  import { WorkExperienceService } from '../services/work-experience-service/work-experience.service';
6
7  describe('WorkExperienceComponent', () => {
8    let component: WorkExperienceComponent;
9    let fixture: ComponentFixture<WorkExperienceComponent>;
10   let service: WorkExperienceService;
11
12   beforeEach(async () => {
13     await TestBed.configureTestingModule({
14       declarations: [WorkExperienceComponent],
15       imports: [
16         AngularFireModule.initializeApp(environment.firebaseConfig),
17       ],
18     })
19     .compileComponents();
20
21     fixture = TestBed.createComponent(WorkExperienceComponent);
22     component = fixture.componentInstance;
23     fixture.detectChanges();
24     service = TestBed.inject(WorkExperienceService);
25   });
26
27   it('should create', () => {
28     expect(component).toBeTruthy();
29   });
30 });
```

Probamos nuevamente, la prueba ‘should create’ deberá pasar.

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
→  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
06 01 2025 00:22:25.648:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
06 01 2025 00:22:25.651:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
06 01 2025 00:22:25.662:INFO [launcher]: Starting browser ChromeHeadless
06 01 2025 00:22:28.505:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket DHK860TKkhvsJdl9AAAB
 with id 53623876
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 0 of 24 SUCCESS (Chrome Headless 131.0.0.0 (Mac OS 10.15.7): E
xecuted 1 of 24 SUCCESS (0 secs / 0.129 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 2 of 24 SUCCESS (0 secs / 0.135 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 3 of 24 (skipped Chrome Headless 131.0.0.0 (Mac OS 10.15.7): E
xecuted 4 of 24 (skipped Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 5 of 24 (skipped Chrome Headless 131.
0.0.0 (Mac OS 10.15.7): Executed 6 of 24 (skipped LOG: WorkExperienceService{db: AngularFirestore{schedulers: ɵAngu
larFireSchedulers{ngZone: ..., outsideAngular: ..., insideAngular: ...}, firestore: Firestore{_delegate: ..., _pers
istenceProvider: ..., INTERNAL: ..., _appCompat: ...}, persistenceEnabled$: Observable{_subscribe: ...}}, accesoWor
kExperience: 'work experience running...', dbPath: '/work-experience', workExperienceRef: AngularFirestoreCollectio
n{ref: CollectionReference{firestore: ..., _delegate: ..., _userDataWriter: ...}, query: CollectionReference{firest
ore: ..., _delegate: ..., _userDataWriter: ...}, afs: AngularFirestore{schedulers: ..., firestore: ..., persistence
Enabled$: ...}}}
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 6 of 24 (skipped 8) SUCCESS (0 secs / 0.146 secs)
LOG: WorkExperienceService{db: AngularFirestore{schedulers: ɵAngularFireSchedulers{ngZone: ..., outsideAngular: ...
, insideAngular: ...}, firestore: Firestore{_delegate: ..., _persistenceProvider: ..., INTERNAL: ..., _appCompat: .
..}, persistenceEnabled$: Observable{_subscribe: ...}}, accesoWorkExperience: 'work experience running...', dbPath:
 '/work-experience', workExperienceRef: AngularFirestoreCollection{ref: CollectionReference{firestore: ..., _delega
te: ..., _userDataWriter: ...}, query: CollectionReference{firestore: ..., _delegate: ..., _userDataWriter: ...}, a
fs: AngularFirestore{schedulers: ..., firestore: ..., pChrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 7 of 24
 (skipped Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 8 of 24 (skipped Chrome Headless 131.0.0.0 (Mac OS 1
0.15.7): Executed 9 of 24 (skipped Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 10 of 24 (skippedChrome Hea
dless 131.0.0.0 (Mac OS 10.15.7): Executed 11 of 24 (skippedChrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 11
 of 24 (skipped 13) SUCCESS (0.238 secs / 0.174 secs)
TOTAL: 11 SUCCESS
```

Agregamos una prueba nueva, para probar que el servicio ejecute correctamente el método ***getWorkExperience()*** del servicio (líneas 31 a 33)

$ nano work-experience/work-experience.component.spec.ts

```
27   it('should create', () => {
28     expect(component).toBeTruthy();
29   });
30
31   it('should get work-experience documents from Firestore', () => {
32     expect(service.getWorkExperience()).not.toBeNull();
33   });
34
35 });
```

Si ejecutamos nuevamente ***ng test***, la prueba nueva deberá pasar

$ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI

```
➜  app git:(master) ✗ ng test --no-watch --no-progress --browsers=ChromeHeadlessCI
06 01 2025 00:35:05.421:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
06 01 2025 00:35:05.430:INFO [launcher]: Launching browsers ChromeHeadlessCI with concurrency unlimited
06 01 2025 00:35:05.441:INFO [launcher]: Starting browser ChromeHeadless
06 01 2025 00:35:08.542:INFO [Chrome Headless 131.0.0.0 (Mac OS 10.15.7)]: Connected on socket N05ZzHuKNecJ--9bAAAB with id 35204101
LOG: WorkExperienceService{db: AngularFirestore{schedulers: ɵAngularFireSchedulers{ngZone: ..., outsideAngular: ..., insideAngular: ...}, fir
estore: Firestore{_delegate: ..., _persistenceProvider: ..., INTERNAL: ..., _appCompat: ...}, persistenceEnabled$: Observable{_subscribe: ...
}}, accesoWorkExperience: 'work experience running...', dbPath: '/work-experience', workExperienceRef: AngularFirestoreCollection{ref: Collec
tionReference{firestore: ..., _delegate: ..., _userDataWriter: ...}, query: CollectionReference{firestore: ..., _delegate: ..., _userDataWrit
er: ...}, afs: AngularFirestore{schedulers: ..., firestore: ..., persistenceEnabled$: ...}}}
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 2 of 25 (skipped 6) SUCCESS (0 secs / 0.049 secs)
LOG: WorkExperienceService{db: AngularFirestore{schedulers: ɵAngularFireSchedulers{ngZone: ..., outsideAngular: ..., insideAngular: ...}, fir
estore: Firestore{_delegate: ..., _persistenceProvider: ..., INTERNAL: ..., _appCompat: ...}, persistenceEnabled$: Observable{_subscribe: ...
}}, accesoWorkExperience: 'work experience running...', dbPath: '/work-experience', workExperienceRef: AngularFirestoreCollection{ref: Collec
tionReference{firestore: ..., _delegate: ..., _userDataWriter: ...}, query: CollectionReference{firestore: ..., _delegate: ..., _userDataWrit
er: ...}, aLOG: WorkExperienceService{db: AngularFirestore{schedulers: ɵAngularFireSchedulers{ngZone: ..., outsideAngular: ..., insideAngular
: ...}, firestore: Firestore{_delegate: ..., _persistenceProvider: ..., INTERNAL: ..., _appCompat: ...}, persistenceEnabled$: Observable{_sub
scribe: ...}}, accesoWorkExperience: 'work experience running...', dbPath: '/work-experience', workExperienceRef: AngularFirestoreCollection{
ref: CollectionReference{firestore: ..., _delegate: ..., _userDataWriter: ...}, query: CollectionReference{firestore: ..., _delegate: ..., _u
serDataWriter: ...}, afs: AngularFirestore{schedulers: ..., firestore: ..., persistenceEnabled$: ...}}}
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 3 of 25 (skipped 6) SUCCESS (0 secs / 0.133 secs)
LOG: WorkExperienceService{db: AngularFirestore{schedulers: ɵAngularFireSchedulers{ngZone: ..., outsideAngular: ..., insideAngular: ...}, fir
estore: Firestore{_delegate: ..., _persistenceProvider: ..., INTERNAL: ..., _appCompat: ...}, persistenceEnabled$: Observable{_subscribe: ...
}}, accesoWorkExperience: 'work experience running...', dbPath: '/work-experience', workExperienceRef: AngularFirestoreCollection{ref: Collec
tionReference{firestore: ..., _delegate: ..., _userDataWriter: ...}, query: CollectionReference{firestore: ..., _delegate: ..., _userDataWrit
er: ...}, aChrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 8 of 25 (skipped 9) SUCCESS (0 secs / 0.178 secs)
Chrome Headless 131.0.0.0 (Mac OS 10.15.7): Executed 12 of 25 (skipped 13) SUCCESS (0.287 secs / 0.19 secs)
TOTAL: 12 SUCCESS
```

Como reto, tendríamos que repetir este proceso para el resto de los servicios y componentes.

Pruebas End-to-End en Angular

Angular ofrece varias herramientas y técnicas para realizar pruebas de extremo a extremo (E2E). Estas pruebas validan que la aplicación funcione correctamente desde el inicio hasta el final, imitando la interacción de un usuario real con la aplicación.

El Angular CLI facilita la configuración y ejecución de pruebas E2E. Para comenzar, se puede ejecutar el comando ng e2e que descarga e instala todo lo necesario para ejecutar pruebas E2E. Si no se encuentra la configuración de pruebas E2E en el proyecto, el CLI solicitará que se elija un paquete de pruebas E2E y guiará al usuario a través de la configuración.

Las herramientas que pueden ser utilizadas para pruebas E2E, son **Cypress**, ***Nightwatch, WebdriverIO, Playwright y Puppeteer***. Estas herramientas pueden ser instaladas manualmente utilizando el comando ng e2e.

Nos aseguramos de estar en la raíz del proyecto.

```
➜  mycv git:(master) ✗ pwd
/Users/adsoft/mycv
➜  mycv git:(master) ✗ ls
README.md           karma.conf.js       package.json        tsconfig.app.json
angular.json        node_modules        public              tsconfig.json
coverage            package-lock.json   src                 tsconfig.spec.json
➜  mycv git:(master) ✗
```

Ejecutamos ng e2e y seleccionamos cypress

$ ng e2e

```
[→  mycv git:(master) ✗ ng e2e
Cannot find "e2e" target for the specified project.
You can add a package that implements these capabilities.

For example:
  Playwright: ng add playwright-ng-schematics
  Cypress: ng add @cypress/schematic
  Nightwatch: ng add @nightwatch/schematics
  WebdriverIO: ng add @wdio/schematics
  Puppeteer: ng add @puppeteer/ng-schematics

Would you like to add a package with "e2e" capabilities now?
  No
  Playwright
❯ Cypress
  Nightwatch
  WebdriverIO
  Puppeteer
```

Posteriormente tecleamos ***Y*** para proceder.

```
→  mycv git:(master) ✗ ng e2e
Cannot find "e2e" target for the specified project.
You can add a package that implements these capabilities.

For example:
  Playwright: ng add playwright-ng-schematics
  Cypress: ng add @cypress/schematic
  Nightwatch: ng add @nightwatch/schematics
  WebdriverIO: ng add @wdio/schematics
  Puppeteer: ng add @puppeteer/ng-schematics

 Would you like to add a package with "e2e" capabilities now? Cypress
✔ Determining Package Manager
  › Using package manager: npm
✔ Searching for compatible package version
  › Found compatible package version: @cypress/schematic@2.5.2.
✔ Loading package information from registry
∴ Confirming installation

The package @cypress/schematic@2.5.2 will be installed and executed.
Would you like to proceed? (Y/n) Y
```

Seleccionamos ***cypress*** como la herramienta por default para ***ng e2e***

```
→  mycv git:(master) ✗ ng e2e
Cannot find "e2e" target for the specified project.
You can add a package that implements these capabilities.

For example:
  Playwright: ng add playwright-ng-schematics
  Cypress: ng add @cypress/schematic
  Nightwatch: ng add @nightwatch/schematics
  WebdriverIO: ng add @wdio/schematics
  Puppeteer: ng add @puppeteer/ng-schematics

 Would you like to add a package with "e2e" capabilities now? Cypress
✔ Determining Package Manager
  › Using package manager: npm
✔ Searching for compatible package version
  › Found compatible package version: @cypress/schematic@2.5.2.
✔ Loading package information from registry
✔ Confirming installation
✔ Installing package
? Would you like the default `ng e2e` command to use Cypress? [ Protractor to Cypress Migration Guide:
https://on.cypress.io/protractor-to-cypress?cli=true ] (Y/n)Y
```

Tecleamos **Y** para soportar pruebas de componentes en ***cypress***.

```
→  mycv git:(master) × ng e2e
Cannot find "e2e" target for the specified project.
You can add a package that implements these capabilities.

For example:
  Playwright: ng add playwright-ng-schematics
  Cypress: ng add @cypress/schematic
  Nightwatch: ng add @nightwatch/schematics
  WebdriverIO: ng add @wdio/schematics
  Puppeteer: ng add @puppeteer/ng-schematics

 Would you like to add a package with "e2e" capabilities now? Cypress
✔ Determining Package Manager
  › Using package manager: npm
✔ Searching for compatible package version
  › Found compatible package version: @cypress/schematic@2.5.2.
✔ Loading package information from registry
✔ Confirming installation
✔ Installing package
? Would you like the default `ng e2e` command to use Cypress? [ Protractor to Cypress Migration Guide:
https://on.cypress.io/protractor-to-cypress?cli=true ] yes
? Would you like to add Cypress component testing?  This will add all files needed for Cypress component
testing. (Y/n) Y
```

Esperamos que la instalación de ***cypress*** termine exitosamente.

```
 Would you like to add a package with "e2e" capabilities now? Cypress
✔ Determining Package Manager
  › Using package manager: npm
✔ Searching for compatible package version
  › Found compatible package version: @cypress/schematic@2.5.2.
✔ Loading package information from registry
✔ Confirming installation
✔ Installing package
? Would you like the default `ng e2e` command to use Cypress? [ Protractor to Cypress Migration Guide:
https://on.cypress.io/protractor-to-cypress?cli=true ] yes
? Would you like to add Cypress component testing?  This will add all files needed for Cypress component
testing. yes
CREATE cypress.config.ts (264 bytes)
CREATE cypress/tsconfig.json (139 bytes)
CREATE cypress/e2e/spec.cy.ts (142 bytes)
CREATE cypress/fixtures/example.json (85 bytes)
CREATE cypress/support/commands.ts (1377 bytes)
CREATE cypress/support/e2e.ts (649 bytes)
CREATE cypress/support/component-index.html (290 bytes)
CREATE cypress/support/component.ts (1123 bytes)
UPDATE package.json (1219 bytes)
UPDATE angular.json (4456 bytes)
✔ Packages installed successfully.
→  mycv git:(master) ×
```

Una vez instalado, ***cypress*** nos crea pruebas end to end de ejemplo en la ruta ***cypress/e2e/spec.cy.ts***

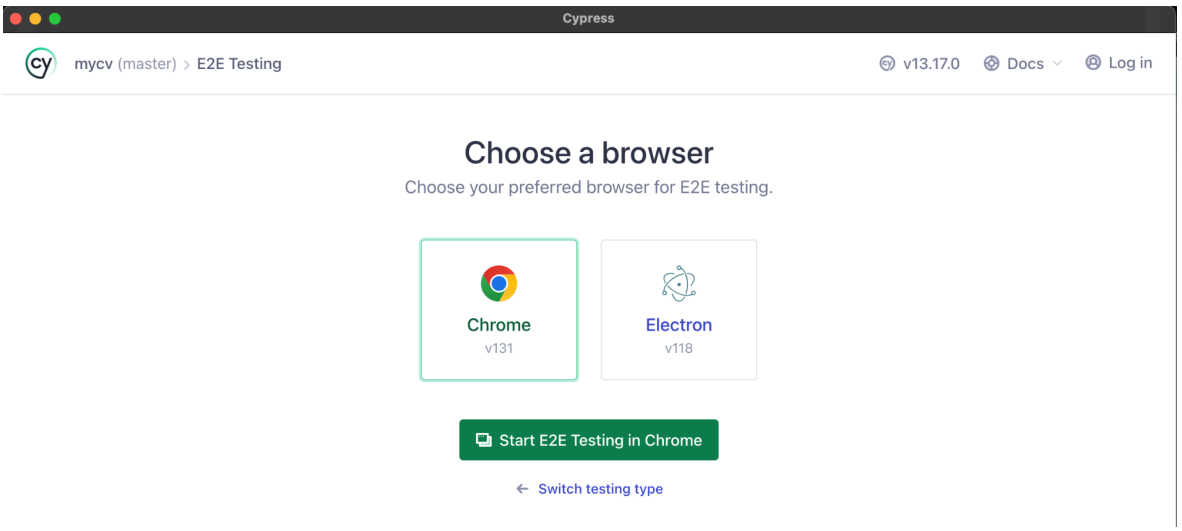
```
→  mycv git:(master) ✗ ls
README.md          cypress            node_modules       public             tsconfig.json
angular.json       cypress.config.ts  package-lock.json  src                tsconfig.spec.json
coverage           karma.conf.js      package.json       tsconfig.app.json
→  mycv git:(master) ✗ ls cypress
downloads     e2e           fixtures      support       tsconfig.json
→  mycv git:(master) ✗ ls cypress/e2e
spec.cy.ts
→  mycv git:(master) ✗ cat cypress/e2e/spec.cy.ts
describe('My First Test', () => {
  it('Visits the initial project page', () => {
    cy.visit('/')
    cy.contains('app is running')
  })
})
```

Podemos verificar que la instalación de ***cypress*** este correcta, ejecutamos ***ng e2e***

$ ng e2e

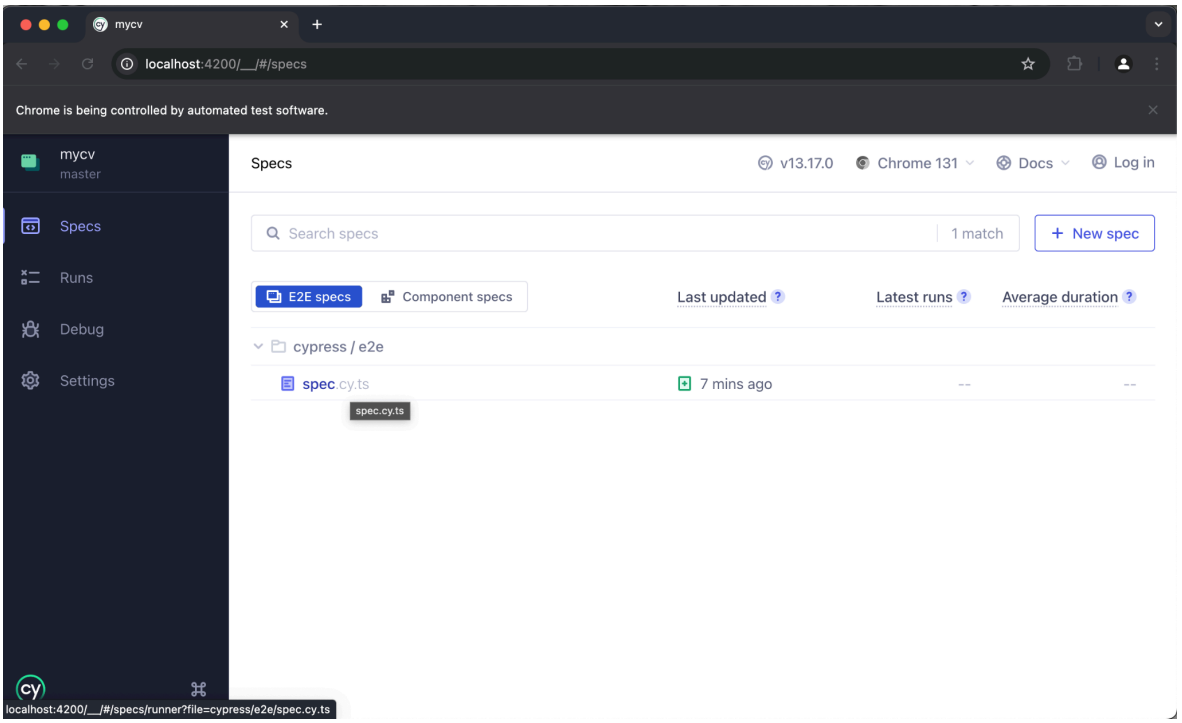
```
→  mycv git:(master) ✗ ng e2e
Passing watch mode to DevServer - watch mode is true
Initial chunk files | Names         |  Raw size
polyfills.js        | polyfills     |  90.20 kB |
main.js             | main          |  17.30 kB |
styles.css          | styles        |  95 bytes |

                    | Initial total | 107.59 kB

Application bundle generation complete. [3.226 seconds]

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
Re-optimizing dependencies because lockfile has changed
  ➜  Local:   http://localhost:4200/
  ➜  press h + enter to show help
It looks like this is your first time using Cypress: 13.17.0

✔  Verified Cypress! /Users/adsoft/Library/Caches/Cypress/13.17.0/Cypress.app

Opening Cypress...

DevTools listening on ws://127.0.0.1:50033/devtools/browser/85730613-b9c7-4503-bed6-c6146d4be738
GraphQL server is running at http://localhost:50034/__launchpad/graphql
Missing baseUrl in compilerOptions. tsconfig-paths will be skipped
```

Inmediatamente abre una ventana del navegador.



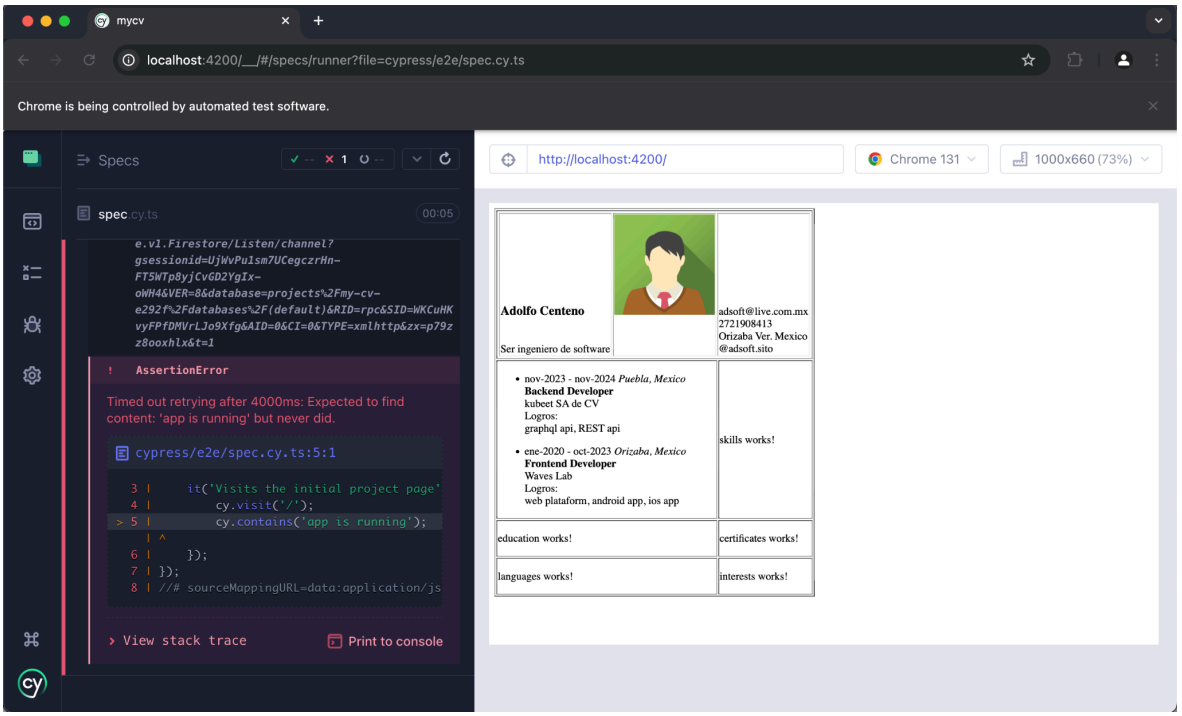
Posteriormente aparece, escogemos Chrome. y click **Start E2E Testing in Chrome**.

Nos abre el entorno de cypress, ahora ejecutamos las pruebas de ***cypress/e2e/spec.cy.ts***, ***click en spec.cy.ts***



Ejecutara la prueba por default que existe en ***spec.cy.ts***

```
describe('My First Test', () => {
  it('Visits the initial project page', () => {
    cy.visit('/')
    cy.contains('app is running')
  })
})
```



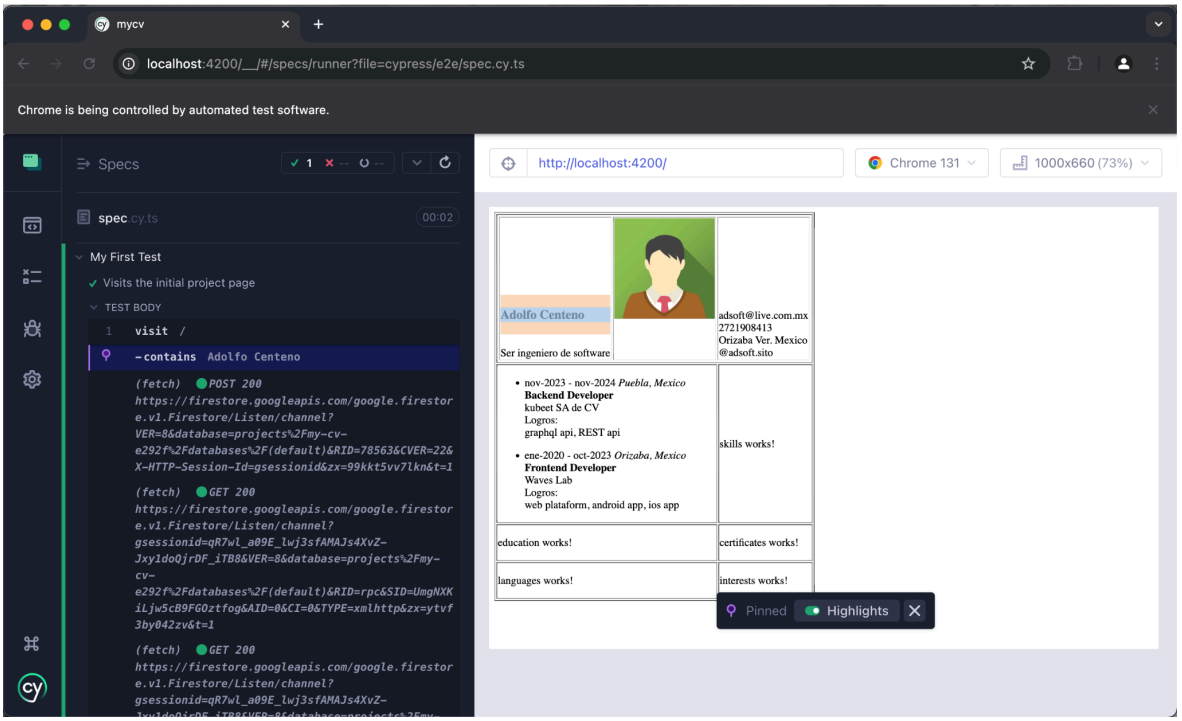
En este framework, las pruebas se ejecutan línea a línea, la primera línea pasa correctamente, la segunda no, ya que no se encuentra el texto ***'app is running'*** en nuestro html.

Modifiquemos la prueba para que encuentre un texto que sí existe, como nuestro nombre.

$ nano cypress/e2e/spec.cy.ts

```
describe('My First Test', () => {
  it('Visits the initial project page', () => {
    cy.visit('/')
    cy.contains('Adolfo Centeno')
  })
})
```

La prueba deberá ejecutarse con éxito.

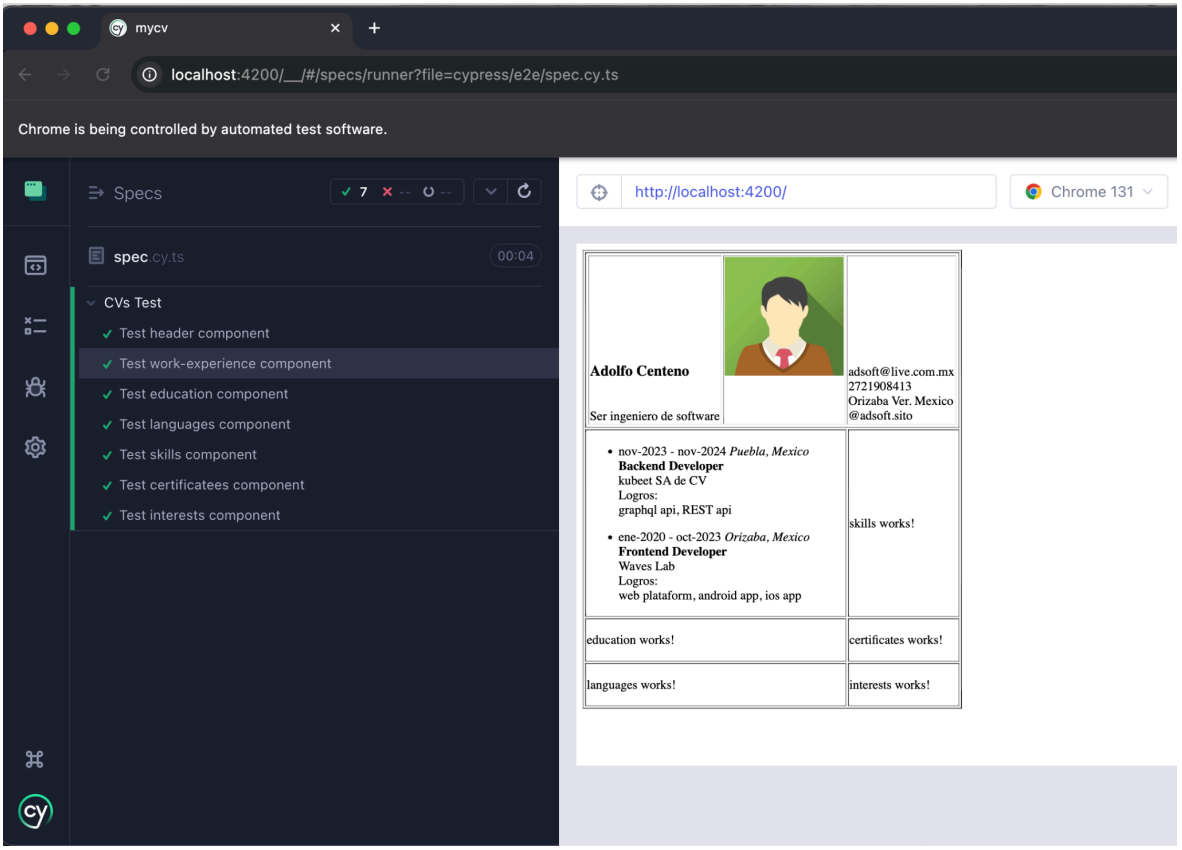


Finalmente, podemos modificar el código para probar que el contenido de cada componente sea el correcto de acuerdo a las colecciones de Firestore.

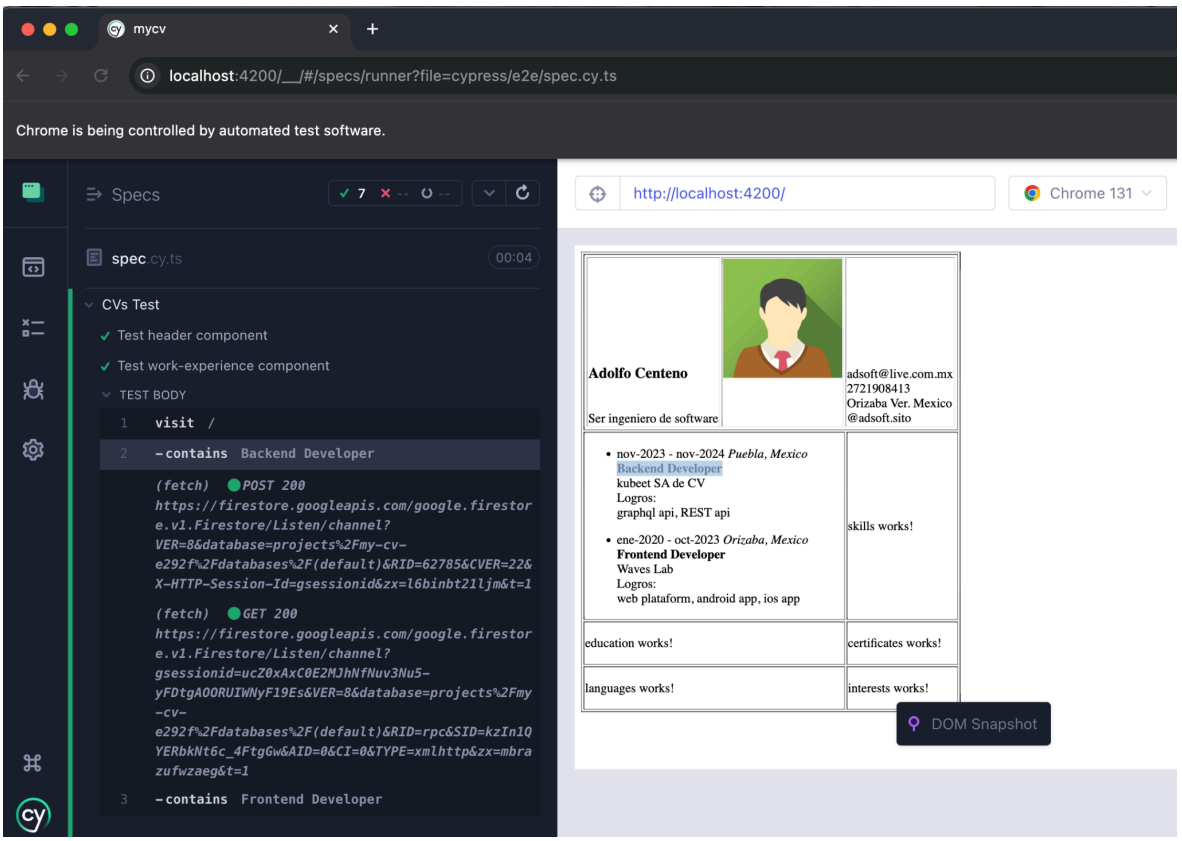
$ nano cypress/e2e/spec.cy.ts

```
describe('CVs Test', () => {
  it('Test header component', () => {
    cy.visit('/')
    cy.contains('Adolfo Centeno')
  })
  it('Test work-experience component', () => {
    cy.visit('/')
    cy.contains('Backend Developer')
    cy.contains('Frontend Developer')
  })
  it('Test education component', () => {
    cy.visit('/')
    cy.contains('education works!')
  })
  it('Test languages component', () => {
    cy.visit('/')
    cy.contains('languages works!')
  })
  it('Test skills component', () => {
    cy.visit('/')
    cy.contains('skills works!')
  })
  it('Test certificatees component', () => {
    cy.visit('/')
    cy.contains('certificates works!')
  })
  it('Test interests component', () => {
    cy.visit('/')
    cy.contains('interests works!')
  })
})
```

Si volvemos a correr las pruebas de ***cypress***, deberán ejecutarse correctamente.



Podemos inspeccionar cada prueba para verificar el código que ejecuta y el contenido en HTML que verifica.



Las pruebas ***e2e*** en Angular 18 son una parte esencial para asegurar que la aplicación funcione correctamente en un entorno real, imitando la interacción del usuario y validando la integración con interfaces externas.

Sala de inspiración

Todas los proyectos globales requieren de las pruebas unitarias, de integración y end to end, para garantizar la calidad del código y soportar actualizaciones periódicas que no destruyan las versiones anteriores.

Sala de pruebas

Lee cada una de las preguntas y selecciona la respuesta que consideres correcta

1.- *Comando de angular CLI para ejecutar pruebas end to end.*

Opción a:	Incorrecta:

ng serve	*El comando de Angular CLI para ejecutar pruebas end to end es:* *ng e2e*
Opción b: ng test	Incorrecta: *El comando de Angular CLI para ejecutar pruebas end to end es:* *ng e2e*
Opción c: *ng e2e*	*CORRECTA:* *El comando de Angular CLI para ejecutar pruebas end to end es:* *ng e2e*
Opción d: *ng test --code-coverage*	Incorrecta: *El comando de Angular CLI para ejecutar pruebas end to end es:* *ng e2e*

2.- **cypress** genera de forma automática el siguiente archivo con pruebas

Opción a: ***cypress/e2e/spec.spec.ts***	Incorrecta: La ruta y archivo que cypress crea es: cypress/e2e/spec.cy.ts
Opción b: ***cypress/e2e/spec.ts***	Incorrecta: La ruta y archivo que cypress crea es: cypress/e2e/spec.cy.ts

Opción c: ***cypress/spec.cy.ts***	Incorrecta: La ruta y archivo que cypress crea es: cypress/e2e/spec.cy.ts
Opción d: ***cypress/e2e/spec.cy.ts***	CORRECTA: La ruta y archivo que cypress crea es: ***cypress/e2e/spec.cy.ts***

3.- *Ademas de cypress, es una herramienta para pruebas end to end*

Opción a: puppeteer	*CORRECTA* puppeteer es una herramienta que puede ser usada en Angular para realizar pruebas end to end
Opción b: jasmine	Incorrecta: puppeteer es la herramienta que puede ser usada en Angular para realizar pruebas end to end
Opción c: *karma*	Incorrecta: puppeteer es la herramienta que puede ser usada en Angular para realizar pruebas end to end
Opción d: node	Incorrecta: puppeteer es la herramienta que puede ser usada en Angular para realizar pruebas end to end

Pruébate (de los 3 temas)

Este es el momento para realizar el reto de realizar las pruebas de integración para los siguientes componentes:

1. Repetir el mismo proceso que se realizó header y work-experience en los componentes education, skills, certificates, languages e interests,
2. Se deberá asegurar que no se rompan las pruebas anteriores.
3. Posteriormente deberá modificar las pruebas end to end en cypress, ya que se romperán con las pruebas de integración recientes.

Ideas para llevar

1. El desarrollo dirigido por pruebas es una tecnología ampliamente utilizada en proyectos medianos y grandes.
2. Es necesario implementar los 3 niveles de pruebas: pruebas unitarias, pruebas de integración y pruebas end to end.
3. La cobertura de código al 100% garantiza la calidad del proyecto
4. Ademas de cypress existen otros frameworkss populares como selenium (https://www.selenium.dev/)

Referencias

Callaghan, M.D. (2024). Test-Driven Development with Angular. In: Angular for Business . Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-9609-7_13

Żurawski, Paweł (2024-05-09). "Angular SSR: Your server-side rendering implementation guide". https://pretius.com/blog/angular-ssr/ *Pretius*. Retrieved 2025-01-11.

Release 19.0.5. https://github.com/angular/angular/releases/tag/19.0.5 18 December 2024. Retrieved 30 December 2024.

Material de consulta

https://anamartinezaguilar.medium.com/tipos-de-testing-en-js-89cd1c4507bf

https://angular.dev/guide/testing

https://testing-angular.com/

https://angular.dev/tools/cli/end-to-end

https://www.cypress.io/

https://www.selenium.dev/