Guía Paso a Paso de Pruebas Unitarias - SchoolMe Portal

Introducción

Este documento explica paso a paso cómo se realizaron las pruebas unitarias en el proyecto SchoolMe Portal. Basado en las pruebas ejecutadas exitosamente (21 pruebas), se detalla el proceso de configuración, escritura y ejecución de pruebas unitarias utilizando Jasmine y Karma en Angular.

Configuración General de Pruebas

Antes de ejecutar cualquier prueba, se configuró el entorno de testing:

1. **Framework**: Jasmine para escribir y ejecutar pruebas.
2. **Ejecutor**: Karma para correr las pruebas en navegador (Chrome Headless).
3. **Módulos**: TestBed de Angular para configurar módulos de prueba.
4. **Dependencias**: ReactiveFormsModule para formularios, RxJS para observables.

Archivo karma.conf.js

```
browsers: ['ChromeHeadless'],
coverageReporter: {
  check: {
    global: {
      statements: 20,
      branches: 5,
      functions: 15,
      lines:  20
    }
  }
}
```

Paso a Paso para Configurar una Prueba Básica

1. Importar Dependencias

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

2. Definir el Bloque Describe

```
describe('AppComponent', () => {
  // Configuración y pruebas aquí
});
```


SCHOOLME ACADEMICS
Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

3. Configurar TestBed en beforeEach

```
beforeEach(async () => {
  await TestBed.configureTestingModule({
    imports: [AppComponent, RouterOutlet],
  }).compileComponents();
});
```

4. Crear Prueba Básica

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

5. Ejecutar Prueba con Cambios en Vista

```
it('should render router outlet', () => {
  const fixture = TestBed.createComponent(AppComponent);
  fixture.detectChanges();
  const compiled = fixture.nativeElement as HTMLElement;
  expect(compiled.querySelector('router-outlet')).toBeTruthy();
});
```

Paso a Paso para Pruebas con Servicios Mockeados

1. Crear Spies para Servicios

```
let mockUserService: jasmine.SpyObj<UserService>;
let mockPersonService: jasmine.SpyObj<PersonService>;

beforeEach(async () => {
  const userServiceSpy = jasmine.createSpyObj('UserService', ['createUserComplete', 'update
  const personServiceSpy = jasmine.createSpyObj('PersonService', ['obtenerTodos']);

  await TestBed.configureTestingModule({
    imports: [FormUserComponent, ReactiveFormsModule],
    providers: [
      { provide: UserService, useValue: userServiceSpy },
      { provide: PersonService, useValue: personServiceSpy }
    ]
  }).compileComponents();

  fixture = TestBed.createComponent(FormUserComponent);
  component = fixture.componentInstance;
  mockUserService = TestBed.inject(UserService) as jasmine.SpyObj<UserService>;
```


SCHOOLME ACADEMICS
Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
  mockPersonService = TestBed.inject(PersonService) as jasmine.SpyObj<PersonService>;
});
```

2. Configurar Retornos de Mocks

```
const mockPersons: PersonOrigin[] = [{
  id: 1,
  documentTypeId: 1,
  identification: '123',
  fisrtName: 'John',
  fullName: 'John Doe',
  // ... otros campos
}];
mockPersonService.obtenerTodos.and.returnValue(of(mockPersons));
```

3. Probar Inicialización y Llamadas

```
it('should load persons on init', () => {
  component.ngOnInit();
  expect(mockPersonService.obtenerTodos).toHaveBeenCalled();
  expect(component.personList).toEqual(mockPersons);
});
```

Paso a Paso para Pruebas de Formularios

1. Verificar Inicialización del Formulario

```
it('should initialize form', () => {
  expect(component.form).toBeDefined();
  expect(component.form.controls.email).toBeDefined();
  expect(component.form.controls.password).toBeDefined();
});
```

2. Probar Validaciones

```
it('should validate password requirements', () => {
  const control1 = { value: 'Abc123' } as AbstractControl;
  const control2 = { value: 'abc123' } as AbstractControl;
  expect(component.validarPasswordSegura(control1)).toBeNull(); // Pasa
  expect(component.validarPasswordSegura(control2)).toEqual({ passwordDebil: true }); // Fa
});
```

3. Probar Emisión de Datos

```
it('should emit form data on emitirValoresForm when valid', () => {
  spyOn(component.posteoForm, 'emit');
```


SCHOOLME ACADEMICS
Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
  component.form.setValue({
    personId: 1,
    email: 'test@example.com',
    password: 'password123',
    status: true,
    photo: null
  });

  component.emitirValoresForm();

  expect(component.posteoForm.emit).toHaveBeenCalled();
});
```

Paso a Paso para Pruebas de Navegación

1. Mockear Router

```
let mockRouter: jasmine.SpyObj<Router>;

beforeEach(async () => {
  const routerSpy = jasmine.createSpyObj('Router', ['navigate']);

  await TestBed.configureTestingModule({
    // ...
    providers: [
      { provide: Router, useValue: routerSpy }
    ]
  }).compileComponents();

  mockRouter = TestBed.inject(Router) as jasmine.SpyObj<Router>;
});
```

2. Probar Navegación

```
it('should navigate to login on cerrarSession', () => {
  component.cerrarSession();
  expect(mockRouter.navigate).toHaveBeenCalledWith(['/login']);
});
```

Diagrama de Flujo General de una Prueba

```
graph TD
    A[Importar dependencias] --> B[Crear spies para servicios]
    B --> C[Configurar TestBed]
    C --> D[Compilar componentes]
    D --> E[Crear fixture y componente]
    E --> F[Configurar mocks con retornos]
```


SCHOOLME ACADEMICS
Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
F --> G[Ejecutar método a probar]
G --> H[Verificar expectativas]
```

Ejecución de Pruebas

Para ejecutar las pruebas: 1. Abrir terminal en el directorio del proyecto. 2. Ejecutar ng test para correr todas las pruebas. 3. Karma abre navegador y ejecuta pruebas automáticamente. 4. Resultados se muestran en consola y navegador. ## Debugging de Pruebas

1. Ejecutar Pruebas en Modo Debug

- Ejecutar ng test --browsers=Chrome para abrir navegador y ver resultados en tiempo real.
- Usar DevTools del navegador para inspeccionar elementos y logs.

2. Agregar Logs en Pruebas

```
it('should debug form validation', () => {
  console.log('Form value:', component.form.value);
  console.log('Form valid:', component.form.valid);
  // ... resto de la prueba
});
```

3. Verificar Mocks y Spies

- Usar console.log(mockService.method.calls.all()) para ver llamadas a spies.
- Verificar que los mocks retornen los datos esperados.

4. Errores Comunes y Soluciones

- **Error: "Can't bind to 'ngModel' since it isn't a known property"**: Importar FormsModule en TestBed.
- **Error: "Expected spy to have been called"**: Verificar que el método se llame correctamente y que el spy esté configurado.
- **Error: "fixture.detectChanges() throws error"**: Asegurar que todos los inputs requeridos estén seteados con fixture.componentRef.setInput.
- **Cobertura baja**: Agregar más pruebas para branches y funciones no cubiertas.

5. Usar Karma para Depuración

- En el navegador, hacer clic en "Debug" en una prueba fallida para pausar ejecución.
- Inspeccionar variables en el scope de la prueba.

SCHOOLME ACADEMICS
Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

Conclusión

Este proceso paso a paso permite crear pruebas unitarias robustas que verifican la funcionalidad de componentes, servicios y navegación en Angular. Las pruebas realizadas en SchoolMe Portal siguen estos patrones, asegurando aislamiento y fiabilidad.

SCHOOLME ACADEMICS
Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com