



Documento de Pruebas Unitarias - SchoolMe Portal

Resumen Ejecutivo

Este documento detalla las pruebas unitarias del proyecto SchoolMe Portal, desarrollado con Angular. Se han ejecutado un total de 21 pruebas unitarias, todas con resultado exitoso (SUCCESS). Las pruebas cubren componentes clave de la aplicación, incluyendo formularios, navegación, autenticación y gestión de datos.

Configuración de Pruebas

- **Framework de Pruebas:** Jasmine
- **Ejecutor:** Karma
- **Navegador:** Chrome Headless (versión 142.0.0.0)
- **Cobertura de Código:**
 - Statements: 24.7% (105/425)
 - Branches: 7.05% (12/170)
 - Functions: 17.69% (20/113)
 - Lines: 23.52% (96/408)

Lista de Pruebas Unitarias

1. AppComponent (src/app/app.component.spec.ts)

- **should create the app:** Verifica que el componente principal se crea correctamente utilizando TestBed.configureTestingModuleTestingModule con RouterOutlet.
- **should render router outlet:** Confirma que el router outlet se renderiza en el template mediante querySelector.

2. SidebarComponent (src/app/components/sidebar/sidebar.component.spec.ts)

- **should create:** Verifica la creación del componente sidebar utilizando TestBed.
- **should fetch sidebar items on init:** Prueba la carga de elementos del sidebar al inicializar, mockeando SidebarService.

3. MenuComponent (src/app/components/nav/menu/menu.component.spec.ts)

- **should create:** Verifica la creación del componente menú.
- **should set idMenu on init:** Prueba la asignación del idMenu en la inicialización.
- **should accept dataSidebar input:** Verifica la recepción de datos de entrada dataSidebar mediante fixture.componentInstance.setInput.



4. DashboardComponent (src/app/layout/dashboard/dashboard.component.spec.ts)

- **should create:** Verifica la creación del componente dashboard.
- **should fetch sidebar items on init:** Prueba la obtención de items del sidebar al iniciar, utilizando spies en SidebarService.
- **should navigate to login on cerrarSession:** Confirma la navegación al login al cerrar sesión, mockeando Router.navigate.

5. EditarAcademicLoadComponent (src/app/page/business/academic-load/editar-academic-load)

- **should create:** Verifica la creación del componente de edición de carga académica utilizando TestBed.

6. FormFormComponent (src/app/page/forms/form-form/form-form.component.spec.ts)

- **should create:** Verifica la creación del componente de formulario genérico.

7. FormUserComponent (src/app/page/forms/form-user/form-user.component.spec.ts)

- **should create:** Verifica la creación del componente de formulario de usuario con mocks para UserService, PersonService y AlertApp.
- **should initialize form:** Prueba la inicialización del formulario con ReactiveFormsModuleModule.
- **should load persons on init:** Confirma la carga de personas al inicializar, utilizando of() de RxJS para mockear respuestas.
- **should emit form data on emitirValoresForm when valid:** Verifica la emisión de datos del formulario cuando es válido, utilizando spyOn en el EventEmitter.

8. LoginMainComponent (src/app/page/login/login-main/login-main.component.spec.ts)

- **should create:** Verifica la creación del componente de login principal con mocks para AuthMainService, UserService y Router.
- **should initialize forms:** Prueba la inicialización de los formularios (form, formRecuperar, formCodigo, formNuevaPassword).
- **should validate password requirements:** Confirma la validación de requisitos de contraseña mediante AbstractControl.
- **should call login on loguear with valid form:** Verifica la llamada al servicio de login con formulario válido, utilizando of() para respuesta mockeada.

9. EditarFormComponent (src/app/page/security/form/editar-form/editar-form.component.spec.ts)

- **should create:** Verifica la creación del componente de edición de formulario.



Resultados de Ejecución

Todas las 21 pruebas unitarias han pasado exitosamente. No se reportaron fallos ni errores durante la ejecución.

Cobertura de Código

La cobertura actual del código es baja, lo que indica oportunidades de mejora en la suite de pruebas. Se recomienda aumentar la cobertura mediante la adición de más pruebas unitarias para casos edge, integración y escenarios de error.

Estrategia de Pruebas

La estrategia de pruebas unitarias se centra en: - **Aislamiento**: Uso de mocks y spies para servicios externos (HttpClient, Router, etc.). - **Cobertura**: Enfoque en componentes críticos como formularios, autenticación y navegación. - **Patrones**: Utilización de Jasmine para describe/it, con beforeEach para configuración común. - **Herramientas**: TestBed para configuración de módulos, fixture para manipulación de componentes.

Diagramas de Flujo de Pruebas

Diagrama de Prueba de Formulario

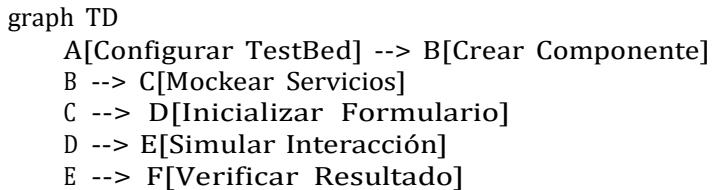
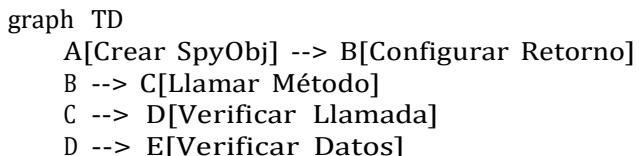


Diagrama de Prueba de Servicio



Mejores Prácticas Implementadas

- Uso de async en beforeEach para compilación de componentes.
- Mocks con jasmine.createSpyObj para servicios.
- Uso de of() de RxJS para observables mockeados.
- Verificación de EventEmitters con spyOn.
- Configuración de inputs con fixture.componentInstance.setInput.



Recomendaciones

1. Aumentar la cobertura de pruebas para alcanzar al menos 70% en todas las métricas.
2. Agregar pruebas de integración para servicios y componentes interdependientes.
3. Implementar pruebas end-to-end con Cypress o similar para flujos completos de usuario.
4. Configurar CI/CD para ejecutar pruebas automáticamente en cada push.
5. Agregar pruebas de error handling y casos edge.
6. Documentar mocks y fixtures para reutilización.

Configuración de Karma

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

Conclusión

Este documento proporciona una visión completa de las pruebas unitarias del proyecto SchoolMe Portal. Con 21 pruebas exitosas, se establece una base sólida para la calidad del código. Se recomienda expandir la suite de pruebas para mejorar la robustez y mantenibilidad del proyecto.

Este documento se generó y completado basado en los archivos de especificación (.spec.ts) del proyecto, con detalles técnicos y estrategias de testing.