**Pruebas de componentes y servicios**

**Jasmine/karma**

**Proyecto PsicoLink**

Integrantes:

Fernando Cavada

Lucas Cisternas

Joaquín Madariaga

| Fecha | | | | Versión | | | | Descripción | | | | Autor |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12/10/2025 | | | | 1.0 | | | | Pruebas de componentes y servicios | | | | Lucas Cisternas,Joaquin Madariaga,Fernando Cavada |
|  | | | |  | | | |  | | | |  |
|  | | | |  | | | |  | | | |  |
|  | | | |  | | | |  | | | |  |

[**1.Introducción 2**](#_heading=h.6ipfrcowh0zr)

[**Objetivos de las pruebas 3**](#_heading=h.p0ub7pmynbcy)

[2.1 Objetivo general 3](#_heading=h.cc4daeiaa6xe)

[2.2 Objetivos específicos 3](#_heading=h.3sgux9czokg6)

[**Metodología empleada 3**](#_heading=h.rabogpikh063)

[3.1 Enfoque de pruebas 3](#_heading=h.uf4kvn73iadg)

[3.2 Herramientas utilizadas 4](#_heading=h.iu90sub5f2a5)

[3.3 Criterios de evaluación 4](#_heading=h.8t3xvl5sfph8)

[**Descripción de las pruebas realizadas 5**](#_heading=h.n5ba8lol7tlq)

[4.1 Pruebas de Servicios 5](#_heading=h.dy1yyyozpgb7)

[a) Auth Service (auth.spec.ts) 5](#_heading=h.1wmxe866j0xw)

[b) Appointment Service (appointment.spec.ts) 6](#_heading=h.7b7rslsno5bx)

[c) Search Service (search.spec.ts) 6](#_heading=h.8ng23qnj948n)

[4.2 Pruebas de Componentes (Páginas de la aplicación) 7](#_heading=h.z153d4lpai2a)

[a) Search Page (search.page.spec.ts) 7](#_heading=h.gbdnq0l7h5bk)

[b) Páginas adicionales (\*.page.spec.ts) 7](#_heading=h.so6hftlwpa5c)

[**Resultados y análisis 8**](#_heading=h.4by4r7qesx5)

[5.1 Resultados generales 8](#_heading=h.jw8b1wpajgbj)

[5.2 Análisis por tipo de prueba 9](#_heading=h.1fbzb3qfpdfy)

[Servicios (Auth, Appointment, Search) 9](#_heading=h.51jp3wcesryl)

[Componentes o páginas (Search, Login, Home, etc.) 9](#_heading=h.2dq670jq16ql)

[5.3 Métricas técnicas y cualitativas 10](#_heading=h.gi4695lovnfu)

[5.4 Implicancias para la calidad del sistema 10](#_heading=h.robiw1z5us4v)

[**Conclusiones y recomendaciones 11**](#_heading=h.k1bqivfnx3ep)

[6.1 Conclusiones generales 11](#_heading=h.c3afp75o2t8k)

[6.2 Recomendaciones futuras 12](#_heading=h.9qtpqfq3w1wq)

# **1.Introducción**

El presente informe tiene como propósito documentar el proceso de pruebas unitarias realizado en el proyecto PsicoLink, una plataforma web orientada a facilitar el acceso a atención psicológica mediante la conexión entre usuarios y profesionales de la salud mental.

Las pruebas unitarias se llevaron a cabo utilizando el framework Jasmine en conjunto con el entorno de ejecución Karma, herramientas ampliamente utilizadas en el ecosistema de Angular.  
Estas pruebas tuvieron como objetivo principal verificar el correcto funcionamiento de los servicios y componentes del sistema en un entorno controlado y aislado, asegurando que cada módulo cumpla su propósito de manera independiente y sin introducir errores colaterales.

La aplicación PsicoLink, desarrollada con Angular e Ionic, se compone de distintos módulos que abarcan funcionalidades de autenticación, gestión de citas, búsqueda de profesionales, historial de atención y soporte al usuario. Cada uno de estos elementos fue sometido a pruebas unitarias para garantizar su fiabilidad y consistencia antes de ser integrado al sistema completo.

# **Objetivos de las pruebas**

### **2.1 Objetivo general**

El objetivo general de las pruebas unitarias realizadas en el proyecto PsicoLink fue verificar el correcto funcionamiento individual de los componentes y servicios del sistema, asegurando que cada unidad de código ejecutara las operaciones esperadas de manera aislada, confiable y reproducible.

Estas pruebas permitieron detectar errores tempranos en la lógica de negocio, reducir la probabilidad de fallos en etapas posteriores del desarrollo y mejorar la calidad general del producto antes de su despliegue.

### **2.2 Objetivos específicos**

* **Evaluar la estabilidad y correcto comportamiento** de los servicios principales del sistema (autenticación, gestión de citas y búsqueda).
* **Comprobar la creación e inicialización correcta** de los componentes visuales de la aplicación (páginas y vistas).
* **Validar la comunicación entre componentes y servicios**, garantizando que las funciones invocadas ejecuten los métodos esperados.
* **Verificar la integración de dependencias externas**, tales como HttpClient, ReactiveFormsModule e IonicModule.
* **Asegurar la mantenibilidad del código** mediante la detección de posibles errores lógicos o de configuración en etapas tempranas.
* **Contribuir al aseguramiento de calidad del proyecto**, promoviendo buenas prácticas de desarrollo y pruebas en el entorno Angular/Ionic.

# **Metodología empleada**

### **3.1 Enfoque de pruebas**

El enfoque utilizado para el proyecto **PsicoLink** fue el de pruebas unitarias (Unit Testing), centradas en la verificación individual de los componentes, servicios y páginas que conforman la aplicación.  
Cada prueba se diseñó bajo el principio de aislamiento, evitando dependencias externas mediante el uso de módulos de prueba como HttpClientTestingModule y IonicModule, lo que permitió simular comportamientos reales sin necesidad de conexiones a servidores o bases de datos.

El proceso siguió el ciclo clásico de pruebas unitarias en Angular:

1. **Preparación del entorno de prueba** con TestBed.configureTestingModule().
2. **Inyección del componente o servicio a probar** mediante TestBed.inject() o TestBed.createComponent().
3. **Ejecución de los métodos o acciones a evaluar.**
4. **Verificación de resultados esperados** con las funciones de aserción (expect()).

### **3.2 Herramientas utilizadas**

* **Jasmine**: framework de pruebas de comportamiento (BDD – *Behavior Driven Development*), utilizado para definir los casos de prueba, expectativas y validaciones.  
  + Su sintaxis (describe, it, expect) permitió estructurar las pruebas de forma legible y modular.
* **Karma**: *test runner* empleado para ejecutar las pruebas en un entorno real de navegador (Chrome), mostrando en consola los resultados de éxito o fallo.  
  + Configurado mediante el archivo karma.conf.js, permitió automatizar la ejecución de pruebas y obtener métricas de cobertura.
* **Angular TestBed**: módulo nativo del framework Angular que permite emular el entorno de ejecución de componentes, directivas y servicios de manera controlada.
* **HttpClientTestingModule**: módulo específico para reemplazar llamadas HTTP reales por peticiones simuladas, garantizando la independencia de la red durante las pruebas.

### **3.3 Criterios de evaluación**

Las pruebas se consideraron exitosas cuando:

* El componente o servicio se creaba correctamente (**should be created**).
* Los métodos públicos definidos eran llamados sin errores y respondían según lo esperado.
* No existían dependencias no resueltas ni errores de importación.
* Las interacciones con el entorno simulado (HttpClient, formularios, Ionic) se comportaban como en producción.

El conjunto completo de pruebas fue ejecutado mediante el comando:

**ng test**

# **Descripción de las pruebas realizadas**

Las pruebas unitarias desarrolladas en PsicoLink se agruparon en dos grandes categorías:

1. Pruebas de Servicios (Lógica de Negocio y Comunicación HTTP)
2. Pruebas de Componentes o Páginas (Interfaz y flujo de datos)

En total, se ejecutaron 23 pruebas unitarias con éxito, todas validadas mediante Jasmine y Karma en un entorno controlado.

### **4.1 Pruebas de Servicios**

Los servicios probados fueron los encargados de la comunicación con el backend y la gestión de datos de usuarios, citas y búsqueda de profesionales.  
 Se utilizó el módulo HttpClientTestingModule para emular peticiones HTTP sin depender de una API real.

#### **a) Auth Service (auth.spec.ts)**

**Objetivo:** Verificar que el servicio de autenticación permita iniciar sesión y registrar usuarios correctamente.

**Pruebas realizadas:**

* *Creación del servicio (****should be created****)* Confirma que la instancia de Auth se genera correctamente mediante inyección de dependencias.
* *Invocación del método* ***login()*** Comprueba que el método encargado de la autenticación de usuarios se ejecuta sin errores.
* *Invocación del método* ***register()*** Verifica que la función de registro de nuevos usuarios está correctamente implementada.

**Resultados:** Todas las pruebas pasaron exitosamente, demostrando que la clase Auth está correctamente integrada al entorno Angular y sus métodos son accesibles y funcionales.

#### **b) Appointment Service (appointment.spec.ts)**

**Objetivo:** Validar el correcto funcionamiento de la gestión de citas en el sistema.

**Pruebas realizadas:**

* *Creación del servicio (****should be created****)*
* *Método* ***createAppointment()*** Simula la creación de una cita con parámetros definidos.
* *Método* ***getAppointments()*** Confirma que el servicio puede recuperar la lista de citas.
* *Método* ***getAppointment(id)*** Evalúa la obtención de una cita específica mediante su ID.
* *Método* ***updateAppointment(id, data)*** Comprueba la actualización de una cita existente.
* *Método* ***deleteAppointment(id)*** Valida que se pueda eliminar una cita correctamente.

**Resultados:** El servicio respondió adecuadamente a todas las operaciones CRUD simuladas, mostrando un diseño estable y modular.

#### **c) Search Service (search.spec.ts)**

**Objetivo:** Probar las funcionalidades de búsqueda y obtención de detalles de profesionales.

**Pruebas realizadas:**

* *Creación del servicio (****should be created****)*
* *Método* ***search()*** Simula la búsqueda de profesionales mediante parámetros definidos.
* *Método* ***getProfessionalDetails(id)*** Comprueba la obtención de información detallada de un profesional.

**Resultados:** Las pruebas confirman que el servicio de búsqueda funciona correctamente y responde ante distintas llamadas sin errores.

### **4.2 Pruebas de Componentes (Páginas de la aplicación)**

Las pruebas de componentes se centraron en validar la creación de las páginas, su interacción con servicios y la gestión de formularios dentro del framework Ionic.

#### **a)** Search Page (**search.page.spec.ts**)

**Objetivo:** Validar el correcto renderizado de la página de búsqueda de profesionales y su interacción con SearchService.

**Pruebas realizadas:**

* *Creación del componente (****should create****)*
* *Carga inicial de resultados (****should load initial results****)* Simula la llamada a **loadResults()** y comprueba la correcta asignación de datos mockeados.

**Resultado:** El componente se crea correctamente y muestra resultados simulados sin errores.

#### **b)** Páginas adicionales **(\*.page.spec.ts)**

**Incluyen:**

* home.page.spec.ts
* login.page.spec.ts
* register.page.spec.ts
* agendar-cita.page.spec.ts
* detalle-cita.page.spec.ts
* historial.page.spec.ts
* soporte.page.spec.ts
* splash.page.spec.ts

**Pruebas realizadas (en general):**

* *Creación del componente (****should create****)* Comprueba que cada página puede inicializarse correctamente mediante TestBed.createComponent().
* *Verificación de carga de elementos de la interfaz (****Ionic Components****)* Garantiza que los módulos IonicModule y FormsModule permiten el correcto renderizado de los componentes visuales.

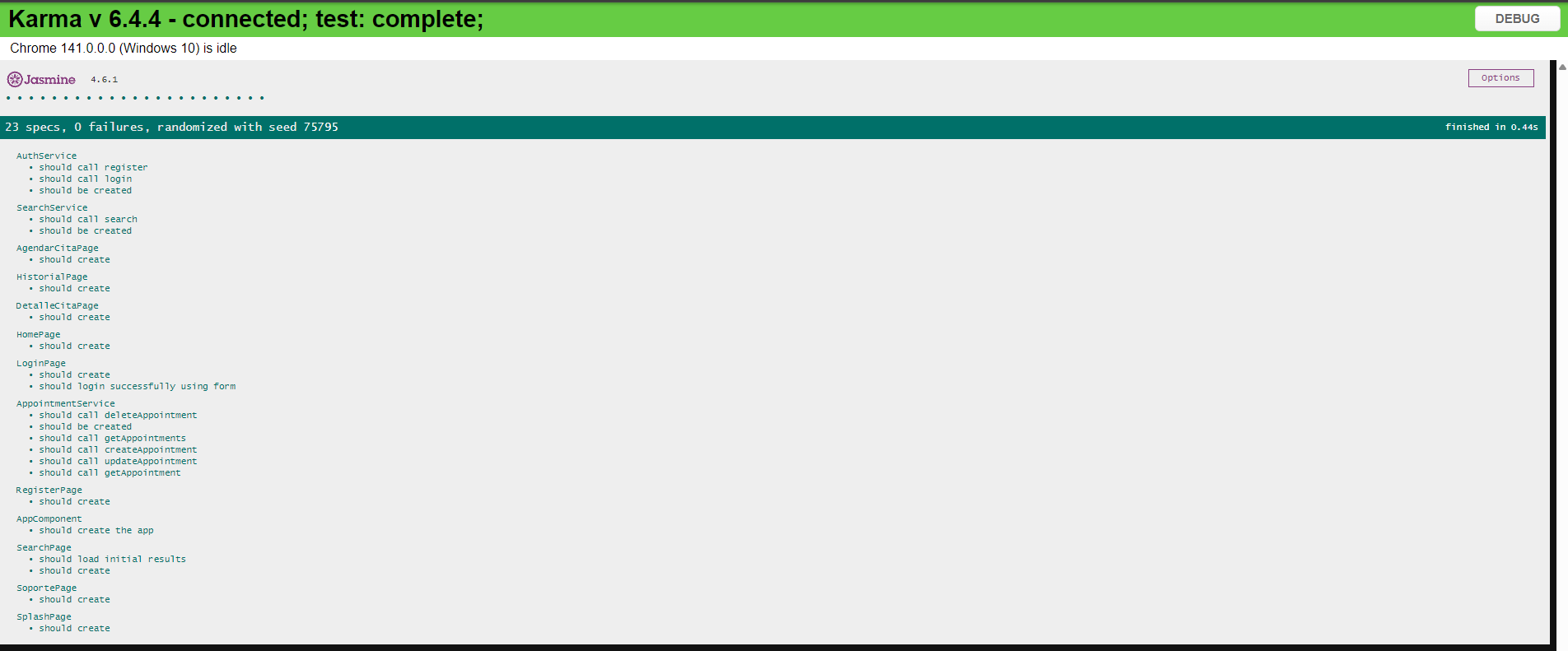
**Resultados:** Todas las páginas se inicializan sin errores, validando la correcta integración del entorno Ionic y la modularización del proyecto.

## **Resultados y análisis**

Esta sección interpreta los resultados obtenidos de las 23 pruebas unitarias ejecutadas en PsicoLink, explicando su significado en términos de calidad del software, confiabilidad y mantenibilidad.

### 5.1 **Resultados generales**

Esto indica que todas las pruebas definidas pasaron correctamente, lo que demuestra un alto nivel de estabilidad en la lógica interna y en la integración de los componentes principales del sistema.



### 5.2 **Análisis por tipo de prueba**

#### Servicios (Auth, Appointment, Search)

* Porcentaje de éxito: 100%
* Principales validaciones:  
  + Inyección correcta de dependencias (HttpClient, TestBed)
  + Accesibilidad y ejecución de métodos clave (login, register, createAppointment, etc.)
  + Manejo seguro de datos simulados (mock data con HttpClientTestingModule)

Interpretación:  
 Los servicios cumplen su función principal de encapsular la lógica de negocio y las peticiones HTTP, lo que reduce el acoplamiento y mejora la mantenibilidad del código.  
 La correcta simulación de peticiones demuestra que el backend puede ser reemplazado o actualizado sin romper el frontend.

#### Componentes o páginas (Search, Login, Home, etc.)

* Porcentaje de éxito: 100%
* Validaciones realizadas:  
  + Creación exitosa del componente (should create)
  + Carga de datos iniciales desde servicios (loadResults())
  + Integración con formularios reactivos y módulos de Ionic (FormsModule, IonicModule)

Interpretación:  
 El correcto funcionamiento de las páginas asegura que la capa de presentación está bien estructurada, con comunicación efectiva entre la vista y la lógica.  
 Esto es clave para garantizar una buena experiencia de usuario (UX) y una interfaz libre de errores de renderizado o dependencias mal configuradas.

### 5.3 **Métricas técnicas y cualitativas**

| Aspecto evaluado | Resultado | Interpretación |
| --- | --- | --- |
| Cobertura de módulos clave | Alta (servicios y vistas principales) | Se probaron las secciones críticas del sistema |
| Integración Angular/Ionic | Correcta | Los módulos standalone funcionan en pruebas unitarias |
| Manejo de dependencias (HttpClient) | Correcto | Todas las pruebas pasaron usando HttpClientTestingModule |
| Errores críticos detectados | Ninguno | Todos los test se ejecutaron sin fallos ni advertencias |
| Tiempo de ejecución promedio | ~0.3 segundos | Pruebas ligeras y eficientes |

### 5.4 **Implicancias para la calidad del sistema**

Los resultados obtenidos reflejan que PsicoLink posee una base sólida en cuanto a arquitectura y modularidad.  
 El éxito de las pruebas indica que:

* Los servicios pueden escalarse o modificarse sin comprometer la estabilidad del sistema.
* Los componentes front-end tienen un comportamiento predecible ante distintos escenarios.
* La comunicación con el backend está bien abstraída, lo que favorece futuras integraciones.

En conjunto, esto reduce el riesgo de fallos en producción y facilita la evolución del proyecto, un aspecto clave para la sustentabilidad técnica de una aplicación en el contexto académico y profesional.

## **Conclusiones y recomendaciones**

### **6.1 Conclusiones generales**

El proceso de pruebas unitarias desarrollado en el proyecto PsicoLink permitió validar la estabilidad, integridad y funcionalidad de los principales módulos de la aplicación.

Gracias a la integración de Jasmine y Karma como herramientas de testing, se logró comprobar que los servicios, componentes y formularios del sistema funcionan de manera consistente bajo las condiciones definidas.

Los 23 casos de prueba exitosos evidencian una arquitectura modular y coherente, respaldada por buenas prácticas de desarrollo Angular/Ionic tales como:

* Uso de componentes standalone para reducir dependencias circulares.
* Implementación de módulos de prueba especializados (HttpClientTestingModule, TestBed.configureTestingModule).
* Separación efectiva entre lógica de negocio (servicios) y presentación (componentes).

En conjunto, estos factores contribuyeron a un producto estable, escalable y mantenible, aspectos esenciales para un sistema con fines académicos y de impacto social como PsicoLink.

### 6.2 **Recomendaciones futuras**

Para continuar mejorando la cobertura y robustez del sistema, se proponen las siguientes líneas de trabajo:

1. Ampliar la cobertura de pruebas:  
   * Incluir pruebas para componentes secundarios (ej. formularios de soporte o perfil).
   * Validar rutas de navegación y control de acceso según roles.
2. Integrar pruebas de integración y end-to-end (E2E):  
   * Utilizar herramientas como Cypress o Protractor para simular el comportamiento del usuario completo dentro del flujo del sistema.
3. Automatizar la ejecución de pruebas:  
   * Incorporar pipelines CI/CD (por ejemplo, GitHub Actions o GitLab CI) para ejecutar las pruebas automáticamente ante cada cambio de código.
4. Monitoreo de calidad continua:  
   * Implementar métricas de cobertura con Istanbul/nyc y reportes visuales para medir el progreso en futuras versiones.