**Guia de Aprendizaje Pruebas de Software**

**Módulo Registro de Usuarios**

**Desarrollado por:**

**Jeiffer Harley Mosquera Abadia**

**Yeraldin Olaya Ríos**

**Septiembre 2025**

**Instructor: Jonathan Espitia**

**SENA**

**Centro de Gestión de Mercados, Logística y TIC (CGMLTI)**

**Análisis y Desarrollo de Software (ADSO)**

**Ficha: 3147246**

**Bogotá D.C**

**III. Formulación de las Actividades de Aprendizaje**

**• Actividades de Reflexión Inicial**

* Pregunta detonante: ¿Qué consecuencias puede tener un software que no ha sido adecuadamente probado?

RTA: Cuando las pruebas de software son insuficientes o están mal diseñadas, las consecuencias pueden ser mucho más graves de lo que parece a simple vista. Uno de los efectos más visibles es que el programa se bloquee o se cierre de forma inesperada, justo cuando el usuario más lo necesita. Esto no solo interrumpe la experiencia, sino que puede hacer que se pierda trabajo, tiempo o incluso dinero.

Otra consecuencia crítica son los errores de seguridad. Sin una validación rigurosa, los datos personales o financieros de los usuarios pueden quedar expuestos, abriendo la puerta a vulnerabilidades que podrían ser explotadas por terceros. A esto se suma la pérdida de información: documentos importantes, archivos o configuraciones pueden desaparecer sin previo aviso, generando frustración y desconfianza en la herramienta o en la empresa que la desarrolla.

* Discusión guiada sobre ejemplos reales de fallos de software.

## RTA: 1. Mariner 1: La omisión de un guión

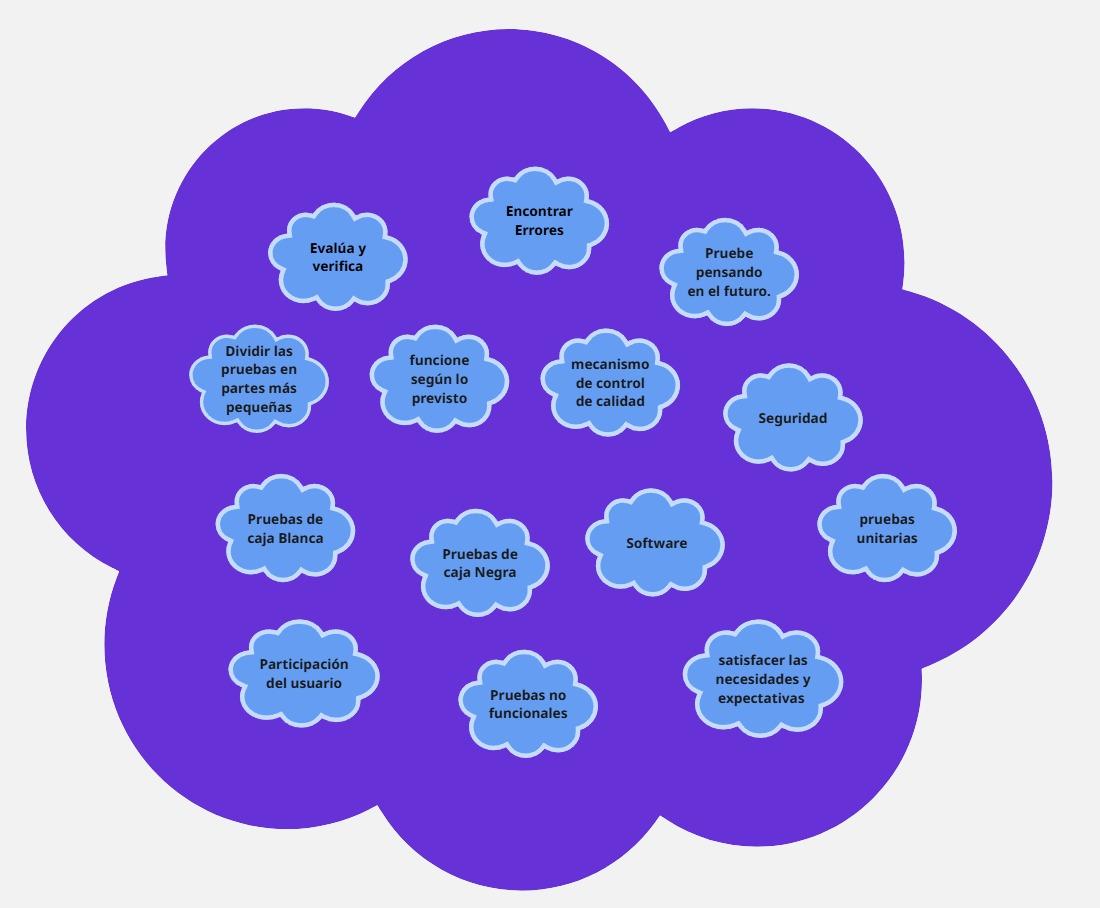
En el año 1962, la NASA se disponía realizar el lanzamiento al espacio de la misión Mariner 1. Con el fin de navegar la órbita de Venus. Sin embargo, un error de programación representó la diferencia entre el éxito y una catástrofe total.

La Mariner 1 fue creada para recabar datos sobre la temperatura y atmósfera de Venus. Esta no logró salir de la atmósfera de la tierra. La catástrofe se originó a solo 5 minutos del despegue debido a la mala transcripción de un código y la omisión de un guión “-“.

El fracaso de la Mariner 1 produjo la pérdida de 18,5 millones de dólares a la [NASA](https://www.nasa.gov/).

**• Actividades de Contextualización e Identificación de Saberes Previos**

* Lluvia de ideas: "¿Qué entienden por prueba de software?"



* Socialización de experiencias previas con herramientas de software y errores encontrados.

**• Actividades de Apropiación**

* Lectura dirigida sobre tipos de pruebas de software.

RTA**:**

### **Pruebas de integración**

Las pruebas de integración verifican que los distintos módulos o servicios utilizados por tu aplicación funcionan bien en conjunto.

### **Pruebas unitarias**

Las pruebas unitarias son de muy bajo nivel y se realizan cerca de la fuente de la aplicación. Consisten en probar métodos y funciones individuales de las clases, componentes o módulos que usa tu software.

### **Pruebas de extremo a extremo**

Las pruebas integrales replican el comportamiento de un usuario con el software en un entorno de aplicación completo. Además, verifican que diversos flujos de usuario funcionen según lo previsto

### **Pruebas funcionales**

Las pruebas funcionales se centran en los requisitos empresariales de una aplicación. Solo verifican el resultado de una acción y no comprueban los estados intermedios del sistema al realizar dicha acción.

### **Pruebas de aceptación**

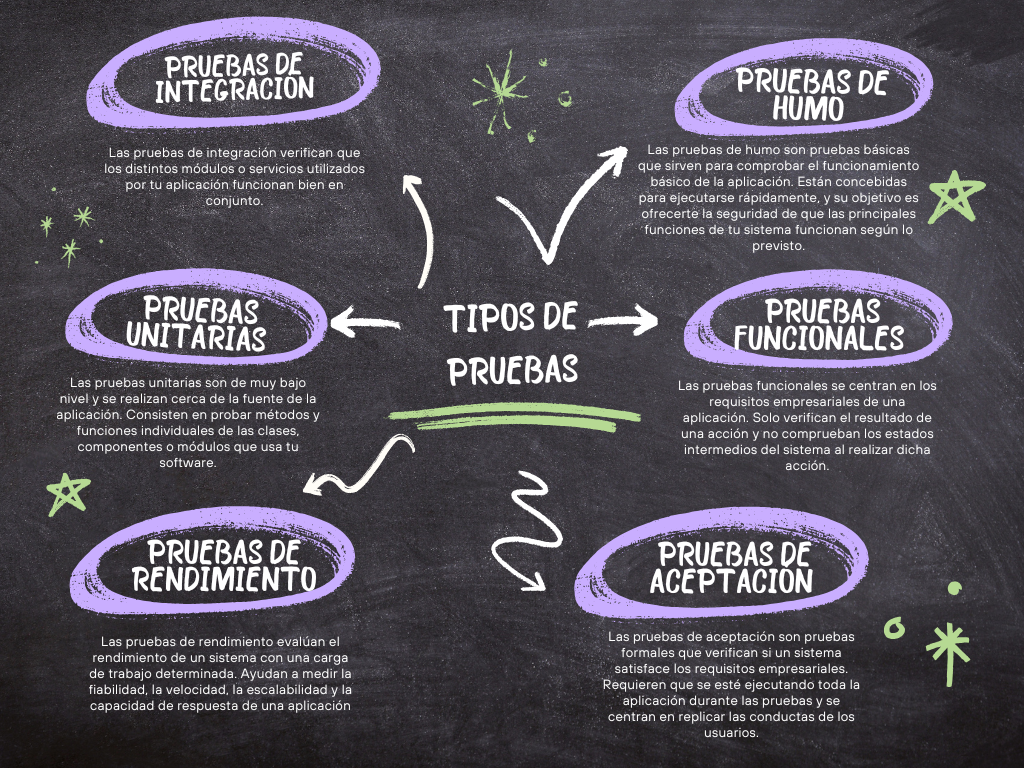
Las pruebas de aceptación son pruebas formales que verifican si un sistema satisface los requisitos empresariales. Requieren que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios.

### **Pruebas de humo**

Las pruebas de humo son pruebas básicas que sirven para comprobar el funcionamiento básico de la aplicación. Están concebidas para ejecutarse rápidamente, y su objetivo es ofrecerte la seguridad de que las principales funciones de tu sistema funcionan según lo previsto.

**Pruebas de rendimiento**

Las pruebas de rendimiento evalúan el rendimiento de un sistema con una carga de trabajo determinada. Ayudan a medir la fiabilidad, la velocidad, la escalabilidad y la capacidad de respuesta de una aplicación

* Investigación individual y grupal de conceptos clave.
* Elaboración de un mapa mental o diagrama conceptual sobre los tipos de pruebas.

**IV. Planteamiento de Evidencias de Aprendizaje**

* **Fase del Proyecto Formativo:** Construcción del software
* **Actividad del Proyecto Formativo:** Aplicación de pruebas funcionales al módulo de gestión del sistema desarrollado

1. **Nombre del Módulo:** Registro de usuarios para Iniciar Sesión en la plataforma WaveSound.

2.1 Desarrollador: Yeraldin Olaya

2.2 Cliente: Jeiffer Mosquera

1. **Enfoque:** Se debe de implementar el Registro de forma que :

**3.1 Register:**

* En el campoNombre solo se admiten letras mayusculas y minusculas,con espacios y es obligatorio
* En el campo Usuario se admiten letras, números, . y \_ (sin espacios) es obligatorio
* En el campoCorreo Electrónico Se verifica que tenga el @ para poder ser admitido y es obligatorio .
* En el campoRol se puede seleccionar solo uno y es obligatorio
* En el campoContraseña debe tener mínimo 8 a 10 caracteres, se admite todo símbolos ,letras, números etc,no se muestra la contraseña solo unos puntos y es obligatorio.

**3.3 Requisitos Funcionales:**

### **3.3.1 Registro de Usuario:**

**RF-01. Campo Nombre**

* El sistema debe permitir ingresar únicamente letras mayúsculas y minúsculas, además de espacios.
* El campo es obligatorio.
* Si el usuario ingresa caracteres no válidos (números o símbolos), el sistema debe mostrar un mensaje de error.

**RF-02. Campo Usuario (nickname)**

* El sistema debe permitir ingresar letras, números, punto (.) y guión bajo (\_).
* No se permiten espacios.
* El campo es obligatorio.
* Si se ingresan caracteres diferentes a los permitidos, el sistema debe mostrar un mensaje de error.

**RF-03. Campo Correo Electrónico**

* El sistema debe validar que el correo tenga el formato correcto (incluyendo @).
* El campo es obligatorio.
* Si no cumple con el formato, se debe mostrar un mensaje de error.

**RF-04. Campo Rol**

* El sistema debe mostrar una lista de roles disponibles.
* El usuario debe poder seleccionar solo un rol.
* El campo es obligatorio.
* Si no se selecciona un rol, se debe mostrar un mensaje de error.

**RF-05. Campo Contraseña**

* La contraseña debe tener entre 8 y 10 caracteres.
* Puede contener letras, números y símbolos.
* El campo es obligatorio.
* La contraseña no debe mostrarse en texto plano, sino enmascarada (puntos o asteriscos).
* Si no cumple con los requisitos, el sistema debe mostrar un mensaje de error.

1. **Casos de Uso o Historias de Usuario:**

**4.1HU02-**Como productor musical, quiero crear mi cuenta ingresando mi nombre, contraseña y rol (productor, artista, músico, oyente), para tener acceso a las funciones específicas según mi perfil.

1. **Descripción:**

**5.1**. **Pruebas funcionales/manuales (Pruebas Unitarias):** Se realizaron las pruebas para el módulo de Registro,donde se probaron y verificaron que los datos que se insertan en el formulario de registro son correctamente almacenados en la Base de Datos por medio de unas APIS , utilizando pytest para probar.

**5.2 Pruebas Pytest para Python:**

Se configuro un cliente de prueba (Test Client)

Esto sirve para simular peticiones HTTP (GET, POST, etc.) a la API de Fast API, sin necesidad de tener el servidor corriendo en un navegador.

**se usó fixtures**

Por ejemplo, mock db → un objeto falso (MagicJack) que simula la base de datos.

Esto evita tocar la base real y permite controlar lo que devuelve las funciones.

**se usó monkey patch**

Con esto reemplazamos funciones reales por versiones falsas.

**Ejemplo**: usuarios.registrar usuario fue reemplazado por una función que devuelve un usuario simulado.

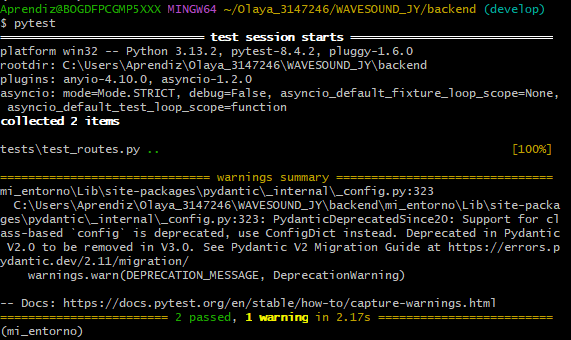
Así se controla exactamente el resultado y se puede comprobar la respuesta de la API.

**Se probó 2 endpoints principales**

/usuarios/register: simula el registro de un usuario y verifica que la API devolviera el nombre correcto.

/usuarios/login: simulas un inicio de sesión, forzando que devuelva un fake token y comprueba que aparezca en la respuesta.

1. **Nombre de las Pruebas:** Pruebas funcionales/manuales(Pruebas Unitarias) y Pruebas de Integración Pytest para Python.
2. **Resultado:** Se realizaron pruebas sobre el módulo de Registro, verificando que los datos ingresados en el formulario se almacenarán correctamente en la Base de Datos a través de las API. Usando pytest, se simuló el envío de los datos al endpoint de registro y se confirmó que el sistema responde correctamente, cumpliendo los criterios de aceptación y asegurando que la creación de usuarios funciona de manera confiable.
3. **Capturas de las pruebas:**

****

**IV. Planteamiento de Evidencias de Aprendizaje**

* **Fase del Proyecto Formativo:** Construcción del software
* **Actividad del Proyecto Formativo:** Aplicación de pruebas funcionales al módulo de gestión del sistema desarrollado
* **Actividad de Aprendizaje:** Investigación en grupo y/o individual que permita socializar y adquirir conocimientos de la temática propuesta, usando herramientas como mapas mentales, diagramas, etc.

**Evidencias:**

* **De Conocimiento:** Identificación y definición de los tipos de pruebas de software.
* **De Desempeño:** Participación en actividades grupales e individuales.
* **De Producto:** Mapa mental, diagrama conceptual u otro producto visual sobre tipos de pruebas.

**Evaluación:**

* **Criterios de Evaluación:** Claridad conceptual, participación activa, calidad del producto entregado.
* **Técnicas e Instrumentos:** Lista de chequeo, observación directa, retroalimentación grupal.

**V. Glosario de Términos**

* **Prueba de software:** Proceso para evaluar la funcionalidad de una aplicación.
* **Verificación:** Comprobación de que el producto cumple con los requisitos especificados.
* **Validación:** Aseguramiento de que el producto cumple su propósito.
* **Mapa mental:** Herramienta visual para organizar información.

**VI. Referentes Bibliográficos**

* Sommerville, I. (2011). *Ingeniería del software*.
* ISTQB Foundation Level Syllabus.
* Recursos digitales: blogs especializados, YouTube, documentación de herramientas de testing.

**VII. Control del Documento**

* **Nombre del autor:** Jonathan David Espitia Rivera
* **Cargo:** Instructor
* **Dependencia:** SENA
* **Fecha de elaboración:** Mayo de 2025

**Referencias Bibliográficas**

<https://infseg.com/informatica/el-arte-de-probar-software-calidad-cofianza-errores/>

<https://fyccorp.com/10-grandes-errores-de-software/>

<https://www.testdevlab.com/blog/software-testing-101-definition-types-everything-else>