REVISTA PERSPECTIVAS

VOLUMEN 7, N°1 / ENERO - JULIO 2025 / e - ISSN: 2661

DOI: https://doi.org/10.47187/perspectivas.7.1.230

ESTRATEGIAS DE MANTENIMIENTO CORRECTIVO DE SOFTWARE: UN ESTUDIO DE CASO DE LA ORGANIZACIÓN COMUNITARIA CACTU

Software Corrective Maintenance Strategies: A Case Study of the Community Organization CACTU

Guerra Fiallos P. [1]	pablo.guerra@gmail.com
López Ríos J. [2]	joseg.lopez@gmail.com
Avila-Pesantez [3]	davila@espoch.edu.ec
Mena Reinoso A. [4]	angel.mena@espoch.edu.ec
Ávila L.M. [5]	miriam.avila@espoch.edu.ec

[1] Investigador Independiente, Ambato, Ecuador.

[2] Investigador Independiente, Quito, Ecuador.

[3] Grupo de Investigación en Innovación Científica y Tecnológica (GIICYT), Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador.

[4] Facultad de Informática y Electrónica, Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador.

[5] Facultad de Administración de Empresas, Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador.

RESUMEN

El sistema CACTU (Corporación de Asociaciones Comunitarias de Cotopaxi y Tungurahua), diseñado para gestionar la correspondencia entre niños afiliados y sus patrocinadores extranjeros, presentaba problemas operativos críticos. Para resolver esta situación, se identificaron las áreas problemáticas clave y se implementaron soluciones técnicas a través de un mantenimiento correctivo basado en la metodología Ágil MANTEMA. Posteriormente, se evaluó la fiabilidad del sistema conforme a la norma ISO/IEC 25010, utilizando herramientas como JMeter, Enlightn y R. Los resultados de la prueba U de Mann-Whitney revelaron mejoras significativas en los tiempos medios entre fallos, antes y después del mantenimiento. En particular, la fiabilidad del sistema aumentó de manera sustancial, pasando del 34.38% al 88.80%, lo que se tradujo en mejoras notables en la madurez, disponibilidad, tolerancia a fallos y capacidad de recuperación del sistema..

Palabras Clave: Mantenimiento, Metodología Ágil Mantema, Fiabilidad, Correspondencia.

ABSTRACT

The CACTU system (Corporación de Asociaciones Comunitarias de Cotopaxi y Tungurahua), designed to manage correspondence between affiliated children and their foreign sponsors, presented critical operational problems. Key problem areas were identified to resolve this situation, and technical solutions were implemented through corrective maintenance based on the Agile MANTEMA methodology. Subsequently, system reliability was evaluated according to ISO/IEC 25010 using JMeter, Enlightn and R tools. The results of the Mann-Whitney U test revealed significant improvements in mean times between failures before and after maintenance. In particular, system reliability increased substantially, from 34.38% to 88.80%, significantly improving system maturity, availability, fault tolerance and resilience.

Keywords: Maintenance, Agile Mantema Methodology, Reliability, Correspondence

Fecha de Recepción: 14/07/2024. **Fecha de Aceptación:** 25/10/2024 **Fecha de Publicación:** 20/01/2025

I. Introducción

El mantenimiento de software es crucial para resolver diversos problemas que surgen tras el despliegue inicial de un sistema. Este proceso permite modificar un producto software después de la entrega, para corregir defectos, mejorar el rendimiento o adaptarlo a una nueva o cambiante necesidad del entorno en el que operan [1]. Entre los problemas más comunes que aborda se encuentran los errores o "bugs" que afectan la funcionalidad del software (mantenimiento correctivo), la necesidad de adaptar el software a nuevos entornos o plataformas (mantenimiento adaptativo), la mejora del rendimiento y la optimización de recursos (mantenimiento perfectivo), y la prevención de futuras fallas a través de actualizaciones y mejoras de seguridad (mantenimiento preventivo) [2]. El mantenimiento asegura la estabilidad, fiabilidad y eficiencia del software a lo largo del tiempo, respondiendo a las demandas cambiantes del entorno tecnológico y del usuario final. Sin un mantenimiento adecuado, el software puede volverse obsoleto, inseguro o ineficiente, afectando negativamente su vida útil y la experiencia del usuario [3].

El mantenimiento correctivo asegura el correcto funcionamiento a lo largo del tiempo. No hacerlo puede ocasionar una serie de problemas que afectan tanto al software en sí mismo como a la organización que lo utiliza. Otros [4-5] definen el mantenimiento correctivo como un proceso de reparación de un problema en el software después de que se ha descubierto que no está funcionando correctamente.

En el ámbito del mantenimiento se utilizan términos como falla, defecto y error. Según [6], una falla se refiere a una disrupción o un desempeño deficiente en un sistema, proceso o producto que impide su capacidad para cumplir con su propósito o función. Por otro lado, [7] define un defecto es una imperfección o problema presente en un producto o servicio, lo cual provoca que no se ajuste a los requisitos o expectativas del usuario. Finalmente, según [8], un error está relacionado con una acción o decisión incorrecta que conduce a un resultado no deseado o incorrecto. Generalmente, un error es causado por un agente externo, que puede ser humano, otro proceso o software.

El mantenimiento del software puede enfrentar múltiples desafíos, tales como la adaptación a los cambios continuos y el aumento en la complejidad. Estos obstáculos pueden surgir a raíz de diversos factores, como modificaciones en los requisitos del software, dificultades en la gestión de proyectos, carencia de documentación adecuada, y fallos de programación [8 - 9]. Asimismo, el autor [10] sostiene que a medida que el software evoluciona, su estructura tiende a volverse progresivamente más compleja, a menos que se realice un esfuerzo deliberado para prevenir este fenómeno. Esto implica que la complejidad en la construcción del software se incrementa cuando los programadores no pueden o no desean aplicar técnicas de ingeniería de software.

La figura 1 representa el flujo de actividades y tareas dentro de un proceso de mantenimiento de software, diferenciando entre tareas planificables y no planificables. Se inicia con tareas comunes, como la valoración de la petición y la documentación de posibles soluciones. Para el mantenimiento planificable (correctivo no urgente, perfectivo, adaptativo), se establecen etapas como la planificación del calendario, la ejecución de pruebas y la validación con el cliente, antes de pasar el software a producción. En el caso del mantenimiento no planificable (correctivo urgente), el flujo conduce directamente a la intervención y pruebas, sin etapas de planificación previas. Finalmente, se realiza una revisión, almacenamiento de datos del producto inicial y tareas finales comunes para cerrar el ciclo de mantenimiento.

Por otro lado, la falta de una metodología formal para el mantenimiento del software puede llevar a que este se realice de manera arbitraria, según lo determine el propio programador [11]. Esto puede generar problemas en la calidad y eficiencia del mantenimiento. Esta opinión también es compartida por [12], quien afirma que raramente existen procesos establecidos, de modo que el mantenimiento se lleva a cabo como se pueda. Según [13], el mantenimiento de software puede tener varios efectos secundarios, que se detallan

en la Tabla I, II y III.

A. Metodología Ágil MANTEMA

La metodología Ágil MANTEMA (proviene de la combinación de las palabras "MANTenimiento" y "MÉTodo") basada en SCRUM, se centra en la gestión eficiente de las etapas y actividades relacionadas con el mantenimiento de software, transformándolas en procesos supervisables y medibles de manera efectiva [12]. Ágil MANTEMA define varios niveles de servicio, destacando dos tipos de mantenimiento correctivo: urgente (no planificable) y no urgente (planificable). La principal diferencia reside en la prioridad asignada, con el mantenimiento correctivo urgente siendo de mayor prioridad en comparación con el no urgente [13].

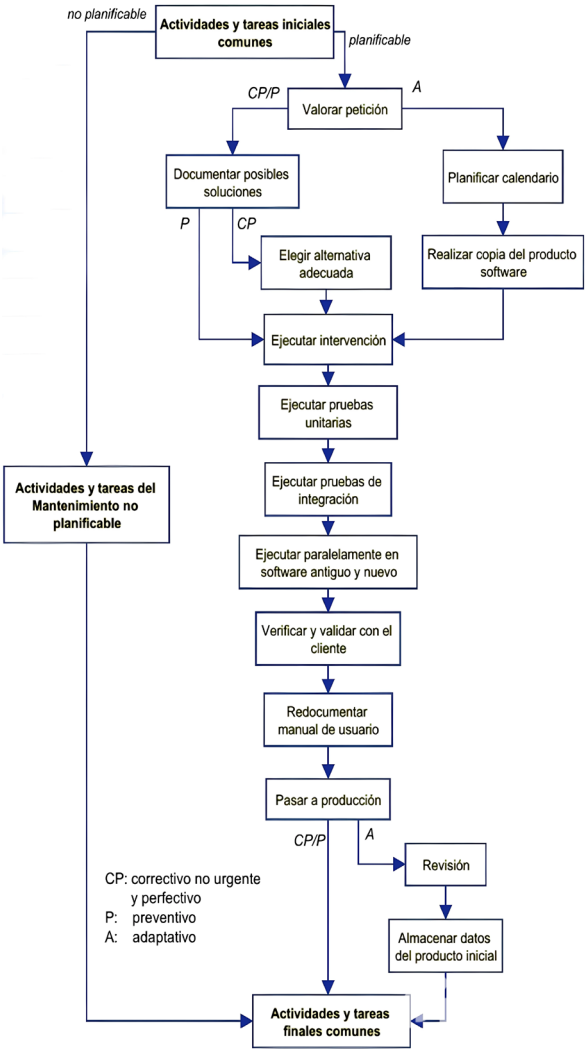

Fig. 1: *Estructura del mantenimiento de software.*

Tabla I
Efectos del mantenimiento sobre el código

Efecto	Descripción
Introducción de errores	Durante el mantenimiento del software, es posible que se introduzcan nuevos errores en el código existente. Estos errores pueden ser el resultado de una mala comprensión de los requisitos, una implementación incorrecta o una falta de pruebas adecuadas.
Aumento de la complejidad	A medida que se realizan cambios en el código, es posible que la complejidad del software aumente. Esto puede dificultar la comprensión y el mantenimiento del código en el futuro.
Impacto en la eficiencia	Algunas modificaciones en el código pueden afectar el rendimiento del software, lo que puede resultar en una disminución de la eficiencia.

Tabla II
Efectos del mantenimiento sobre la documentación

Efecto	Descripción
Desactualización	A medida que se realizan cambios en el software, es posible que la documentación existente se vuelva obsoleta. Esto puede dificultar la comprensión del software y el mantenimiento futuro.
Falta de documentación adecuada	Si la documentación existente no es clara o no refleja con precisión el código fuente, puede ser difícil comprender cómo un cambio en el código afecta a otras partes del sistema.

Tabla III
Efectos del mantenimiento sobre los datos

Efecto	Descripción
Pérdida de datos	Durante el mantenimiento del software, es posible que se produzca una pérdida de datos. Esto puede ocurrir debido a errores en la implementación de cambios o a problemas con la integridad de los datos.
Inconsistencia de datos	Al realizar cambios en el software, es posible que se produzcan inconsistencias en los datos. Esto puede dificultar el uso del software y la toma de decisiones basadas en los datos.
Problemas de seguridad	Al realizar cambios en el software, es posible que se introduzcan nuevos problemas de seguridad. Esto puede poner en riesgo la integridad y la confidencialidad de los datos.

La concepción de Ágil MANTEMA se centra en proporcionar orientación metodológica a las pequeñas y medianas empresas que buscan llevar a cabo eficientemente el mantenimiento de software. Su estructura se ha diseñado tomando como base las directrices establecidas, que se visualiza en la figura 2.

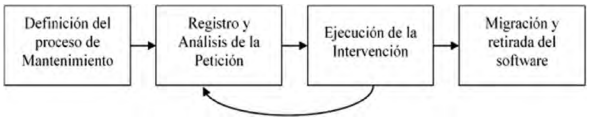

Fig. 2: *Diagrama de proceso de Ágil MANTEMA*

II. desarrollo del mantenimiento del software

En esta sección se describe el proceso de mantenimiento del sistema implementado mediante la metodología Ágil MANTEMA, el cual se estructura en cuatro fases principales: (1) Definición del proceso de mantenimiento, donde se establecen las estrategias y procedimientos a seguir; (2) Registro y análisis de la petición, que implica la identificación y evaluación de las solicitudes de mantenimiento recibidas; (3) Ejecución de la intervención, enfocada en la resolución efectiva de los problemas detectados; y (4) Migración y retirada del software, asegurando la correcta implementación de cambios y la eliminación de versiones obsoletas.

A. Definición del proceso de mantenimiento

En este proceso se definió claramente qué partes del sistema serán objeto de mantenimiento y se estableció los límites, incluyendo las áreas que no serán cubiertas. Además, se estableció claramente los roles de los equipos de mantenimiento y desarrollo, asignando responsabilidades específicas para cada etapa del proceso de mantenimiento. Para la planificación y gestión del trabajo. Se incluyó un cronograma detallado con hitos y fechas límite para las actividades de mantenimiento y la gestión del trabajo en sprints.

B. Registro y análisis de la petición

Para la identificación de problemas, se llevaron a cabo entrevistas con dos gestores de Cotopaxi y Tungurahua en varias reuniones independientes, respectivamente. Durante estos encuentros, se abordaron los inconvenientes detectados por los mismos al utilizar el sistema. Los problemas se documentan con el artefacto Informe de Problema (IP) según las directrices de Ágil MANTEMA, con sus respectivas pruebas de aceptación (PA) y actividades de mantenimiento y gestión (AMG), como se visualiza en la tabla IV.

C. Ejecución de la intervención

En este apartado se definió la especificación de nuevos requisitos y modificaciones a los ya existentes, se emplea el artefacto Solicitud de Cambio (SC). Al igual que los IP, requieren de sus respetivas PA y AMG (ver tabla V).

Tabla IV
Ejemplo del formato de informe de problema

<table>
<tr><td colspan="3">INFORME DE PROBLEMA</td></tr>
<tr><td>ID: IP_01</td><td colspan="2">Nombre del problema: Error en el enlace de redirección hacia la página principal del sistema.</td></tr>
<tr><td colspan="2">Usuario: Usuarios del sistema</td><td>Sprint: 2</td></tr>
<tr><td colspan="2">Prioridad en el negocio: 2</td><td>Puntos estimados: 7</td></tr>
<tr><td colspan="2">Riesgo en el desarrollo: 3</td><td>Puntos reales: 7</td></tr>
<tr><td colspan="3">Descripción: Yo, como usuario, al tratar de regresar a la página principal con el enlace destinado ("/home"), en lugar de redirigir a la ruta raíz, lleva a la ruta "/". Este comportamiento no coincide con la expectativa del usuario y genera confusión en la navegación.</td></tr>
<tr><td colspan="3">Observación: Ninguna</td></tr>
<tr><td colspan="3">Paso para reproducción:
1. Iniciar sesión en el sistema con credenciales válidas.
2. Clic sobre el botón home o ingresar a la ruta “/”.
3. Observar la falla generada.</td></tr>
<tr><td colspan="3">ACTIVIDADES DE MANTENIMIENTO Y GESTIÓN</td></tr>
<tr><td rowspan="3">ID: AMG_09_IP_01</td><td colspan="2">Nombre de la actividad: Análisis del error al intentar acceder a la ruta principal del sistema.</td></tr>
<tr><td>Responsable: Pablo Guerra</td><td>Tipo de actividad: Análisis</td></tr>
<tr><td colspan="2">Descripción: Observar el error generado al intentar redirigirse a la ruta principal del sistema.</td></tr>
<tr><td rowspan="3">ID: AMG_10_IP_01</td><td colspan="2">Nombre de la actividad: Diseño de la solución técnica para la correcta redirección del enlace.</td></tr>
<tr><td>Responsable: Pablo Guerra</td><td>Tipo de actividad: Diseño</td></tr>
<tr><td colspan="2">Descripción: Diseñar la solución que corregirá el error cuando el usuario desee redirigirse a la ruta principal del sistema.</td></tr>
<tr><td rowspan="3">ID: AMG_10_IP_01</td><td colspan="2">Nombre de la actividad: Implementación de la solución técnica</td></tr>
<tr><td>Responsable: Pablo Guerra</td><td>Tipo de actividad: Implementar solución</td></tr>
<tr><td colspan="2">Descripción: Implementar la modificación necesaria dentro del código para redirigir al usuario a la ruta principal del sistema sin ningún error.</td></tr>
<tr><td colspan="3">PRUEBAS DE ACEPTACIÓN</td></tr>
<tr><td rowspan="3">ID: PA_01_ IP_01</td><td colspan="2">Nombre de la prueba: Prueba de redirección a "/" correcta.</td></tr>
<tr><td colspan="2">Criterio: Verificar que el usuario ahora pueda redirigirse sin problema a la página principal del sistema a través del botón “Escritorio”</td></tr>
<tr><td>Responsable: José López</td><td>Evaluación de la prueba: Prueba Exitosa.</td></tr>
<tr><td rowspan="3">ID: PA_02_IP_01</td><td colspan="2">Nombre de la prueba: Prueba de comportamiento al acceder directamente a "/”.</td></tr>
<tr><td colspan="2">Criterio: Verificar que el usuario ahora pueda acceder sin problema a la página principal del sistema a través del enlace directo en la barra de búsqueda.</td></tr>
<tr><td>Responsable: José López</td><td>Evaluación de la prueba: Prueba Exitosa.</td></tr>
</table>

ID: PA_03_IP_01	Nombre de la prueba: Prueba de retroceso en la navegación.	
	Criterio: Verificar que el usuario ahora pueda redirigirse sin problema a la página principal del sistema a través del botón “atrás”.	
	Responsable: José López	Evaluación de la prueba: Prueba Exitosa.

Para llevar a cabo la medición de los tiempos de estimación, se implementa el método T-Shirt que consta de 5 tallas con sus correspondientes puntos estimados y horas de trabajo. Es fundamental considerar que 1 punto estimado equivale a 1 hora de desarrollo utilizada por el equipo de trabajo, y que 1 día de trabajo se corresponde con 7 horas de desarrollo. Se desplegó el producto backlog, que consiste en una lista priorizada de cada cambio que se hizo al sistema de correspondencia de CACTU. Esta lista está organizada a través de su complejidad y puntos estimados, como se observa en la tabla VI.

Tabla V
Formato de solicitud de cambio.

SOLICITUD DE CAMBIO		
ID: SC_01	**Nombre de la solicitud:** Expansión del módulo de registro de afiliados.	
Usuario: Usuarios del sistema	**Sprint:** 8	
Prioridad en el negocio: 2	**Puntos estimados:** 14	
Riesgo en el desarrollo: 3	**Puntos reales:** 14	
Descripción: Yo, como usuario del sistema, solicito que dentro del módulo de registro de afiliados se agreguen los campos adicionales solicitados en las respectivas reuniones. Esta adición facilitara un completo y ágil registro de todos los afiliados dentro del sistema.		
Módulos intervenidos: Registro de afiliados		
Cambios en la petición: Ninguno		
Observación: Ninguna		
ACTIVIDADES DE MANTENIMIENTO Y GESTIÓN		
ID: AMG_54_ SC_01	**Nombre de la actividad:** Análisis de los requisitos para la expansión del módulo de registro de afiliados.	
	Responsable: José López	**Tipo de actividad:** Análisis
	Descripción: Identificar los nuevos campos y funcionalidades que se requieren en el módulo.	
ID: AMG_55_ SC_01	**Nombre de la actividad:** Diseño de la arquitectura de la expansión del módulo de registro de afiliados.	
	Responsable: José López	**Tipo de actividad:** Diseño
	Descripción: Diseñar la interfaz de usuario para los nuevos campos y funcionalidades.	
ID: AMG_56_ SC_01	**Nombre de la actividad:** Implementación de la expansión del módulo de registro de afiliados.	
	Responsable:	**Tipo de actividad:** Implementar solución
	Descripción: Implementar los nuevos campos y funcionalidades en el código fuente del módulo.	

PRUEBAS DE ACEPTACIÓN		
ID: PA_28_ SC_01	Nombre de la prueba: Prueba de registro exitoso dentro de los nuevos campos añadidos.	
	Criterio: Verificar que los usuarios puedan guardar información en los nuevos campos agregados al módulo.	
	Responsable: José López	Evaluación de la prueba: Prueba Exitosa.

El estándar PSR-12, conocido como "Extended Coding Style", estableció directrices específicas para la codificación en PHP, mejorando la legibilidad, comprensión, depuración y modificación del código fuente. Esto contribuye a reducir la complejidad del sistema y facilita su mantenimiento. Para el diseño de las nuevas interfaces de usuario requeridas durante el proceso de mantenimiento, se utilizó la herramienta Balsamiq en su versión gratuita, permitiendo el desarrollo de prototipos claros y comprensibles, adecuados para usuarios sin experiencia técnica. La Figura 3 presenta un ejemplo de estos prototipos, destacando su diseño intuitivo y accesible.

Tabla VI
Product Backlog

Épica	Peticiones de modificación	Complejidad	Puntos estimación
AMG_01	Análisis de la base de datos	Media	14
AMG_02	Análisis de los módulos afectados	Baja	7
AMG_03	Identificación de problemas de codificación	Alta	**28**
AMG_04	Rediseño de base de datos	Alta	**28**
AMG_05	Migración de rediseño de la base de datos	Baja	7
AMG_06	Implementación de nuevas interfaces	Baja	7
AMG_07	Pruebas de fiabilidad utilizando Enlightn	Baja	7
AMG_08	Pruebas de tolerancia a fallos utilizando JMeter	Muy Alta	**56**
Horas			**154**
IP_01	Error en el enlace de redirección hacia la página principal del sistema.	Baja	7
IP_02	Error al enviar el enlace por correo al niño afiliado	Baja	7
IP_03	Vista defectuosa al momento de escribir la ruta raíz en el navegador.	Baja	7
IP_04	Problemas al realizar cambio de contraseña de la cuenta.	Baja	7
IP_05	Lentitud en la Ejecución de Acciones del Sistema	Media	14
IP_06	Error al hacer búsquedas numéricas en el módulo de afiliados	Media	14
IP_07	Error al intentar entrar al módulo gestión de perfil de usuario	Baja	7

IP_08	Error al entrar al manual de usuario	Baja	7
IP_09	Error al intentar descargar listado PDFs de niños afiliados	Baja	7
IP_10	La firma del usuario solo puede ser cambiada por el administrador	Baja	7
IP_11	Clase de Google Maps expuesta en Código	Baja	7
IP_12	Error al revertir migraciones de la base de datos	Media	14
IP_13	Error al ejecutar seeders	Baja	7
IP_14	Problemas de rendimiento en consultas SQL	Media	14
IP_15	Variables de entorno expuestas en Código usando método env()	Baja	7
IP_16	Rutas inaccesibles	Baja	7
IP_17	Secciones de módulos con funcionalidad repetidas	Baja	7
IP_18	Código muerto y Código comentado	Baja	7
Horas			147
SC_01	Expansión del módulo de registro de niños	Alta	28
SC_02	Envío de Múltiples Cartas en un Correo	Media	14
SC_03	Registro de tutores	Media	14
SC_04	Gestión de Respuestas de Cartas	Baja	7
SC_05	Autenticación en Dos Factores	Media	14
SC_06	Integración de Modo Oscuro	Baja	7
SC_07	Creación de Nuevos Roles	Baja	7
SC_08	Generación de PDF de la Ficha de Afiliación/Actualización de datos	Baja	7
SC_09	Monitoreo de Gestores por Comunidades	Media	14
SC_10	Registro de Cantones por Provincia, Cantones y Parroquias	Baja	7
SC_11	Creación de Registro de Unidades Educativas	Media	14
SC_12	Registro del domicilio del niño	Baja	7
SC_13	Adición de Miembros Familiares	Media	14
SC_14	Actualización Única de Información Familiar Compartida	Muy Alta	56
SC_15	Buzón Activo para Niños Afiliados	Baja	7
SC_16	Registro de Cuenta Bancaria Propia del afiliado y tutor	Media	14
SC_17	Notificaciones por WhatsApp	Muy Alta	84
SC_18	Creación de Cartas Respuesta por WhatsApp	Alta	28
SC_19	Correcciones Ortográficas Automáticas en Cartas	Muy Alta	56
Horas			**399**
Total Horas			**700**

Adicionalmente, se implementaron diversas modificaciones en la base de datos del sistema con el fin de mejorar su eficiencia y fiabilidad. Estas modificaciones incluyeron la normalización de tablas, optimización de consultas y la creación de índices, lo que incrementó el rendimiento de las operaciones de búsqueda y recuperación de datos. Asimismo, se añadieron nuevas tablas y campos para soportar las funcionalidades recientemente incorporadas en el sistema de correspondencia (ver Figura 4).

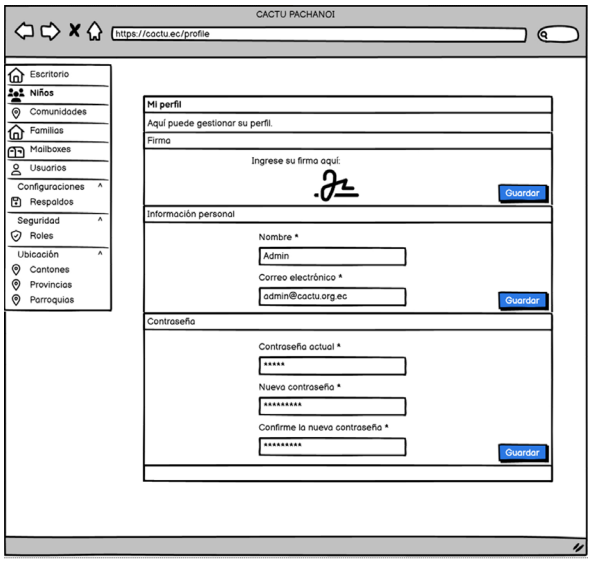

Fig. 3: *Diseño del prototipado de la pantalla*

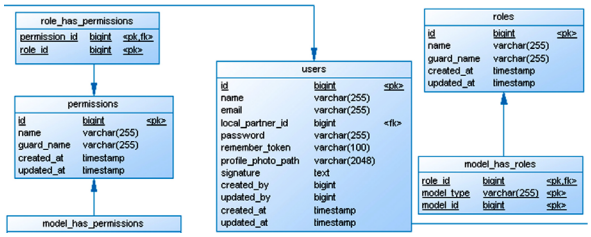

Fig. 4: *Base de datos con nuevas tablas y campos*

D. Migración y retirada del software

Durante esta etapa, se llevó a cabo la migración de los datos existentes a la nueva base de datos y la actualización del sistema, asegurando la integridad y consistencia de la información a lo largo del proceso. La migración fue cuidadosamente planificada para minimizar la interrupción del servicio, implementando procedimientos de respaldo y validación que evitaron la pérdida de datos críticos. Paralelamente, la retirada del software anterior se ejecutó de manera gradual, garantizando que todas las funcionalidades del nuevo sistema estuvieran completamente

operativas antes de desactivar la versión previa. Este proceso incluyó la desactivación de módulos obsoletos y la eliminación de componentes no esenciales, con el fin de mitigar riesgos de seguridad y optimizar la eficiencia del sistema. La fase concluyó con pruebas para verificar la funcionalidad del sistema tras la migración, asegurando la continuidad del registro de niños y la mensajería con patrocinadores, sin interrupciones dentro del sistema CACTU.

III. Metodología de investigación

La norma ISO/IEC 25010 establece un marco estándar para evaluar la fiabilidad del software, definiéndola como la capacidad del sistema para mantener un desempeño especificado bajo condiciones operativas establecidas. Para medir la fiabilidad en el contexto del mantenimiento correctivo siguiendo esta norma, el proceso incluye varios pasos clave: a) Identificación de problemas ocultos de código mediante herramientas como Enlightn, para detectar vulnerabilidades que podrían comprometer la estabilidad del software; b) Cálculo de métricas de fiabilidad, utilizando el análisis del Tiempo Medio entre Fallos (MTBF) y el Tiempo Medio de Reparación (MTTR), lo que permite una evaluación precisa de la frecuencia y duración de las interrupciones del servicio; c) Pruebas de estrés y carga, se puede utilizar herramientas como JMeter, simulando condiciones extremas de uso para evaluar la capacidad del sistema de mantener su funcionalidad después de la implementación de correcciones y d) Evaluación de atributos de fiabilidad como la madurez, disponibilidad, tolerancia a fallos y capacidad de recuperación, para garantizar que el sistema cumpla con los requisitos de calidad definidos en la norma ISO/IEC 25010.

A. Identificación de problemas ocultos de código

Para identificar problemas ocultos en el código del sistema, se utilizó la herramienta Enlightn, la cual ejecutó 27 pruebas centradas en evaluar la fiabilidad de aplicaciones desarrolladas en Laravel. Este análisis permitió detectar 18 problemas que afectaban a 9 de los 14 módulos del sistema, proporcionando una visión completa de las deficiencias existentes y una base sólida para el desarrollo de soluciones efectivas. Las mejoras implementadas incluyeron técnicas de refactorización, ingeniería inversa, reingeniería y modernización del sistema, distribuidas a lo largo de 13 sprints.

B. Cálculo de métricas MTBF y MTTR

Para este proceso se estableció un período de un mes que permitió registrar los tiempos de operación y contabilizar el número de fallos. El análisis de los datos se logró codificando un script empleando el lenguaje R para calcular el Tiempo de operación (TO), Tiempo Medio de Reparación (MTTR), Numero de fallas (NE), y Tiempo Medio entre Fallas (MTBF). Los resultados obtenidos para antes y después del mantenimiento se detallan a continuación en la Tabla VII.

Tabla VII
Medias estadísticas del grupo de datos pre y post mantenimiento

Variable	**Pre-Mantenimiento**	**Post-Mantenimiento**
Tiempo medio entre fallos (MTBF)	3.48 minutos / falla	16.42 minutos / falla
Tiempo medio de reparación (MTTR)	2.62 minutos	1.25 minutos
Numero promedio de fallos (NE)	4.47 fallos	1.46 fallos
Tiempo Promedio de Operación (TO)	17.43 minutos	18.22 minutos
Lambda (1/MTBF)	0.28 fallas	0.06 fallas

C. Pruebas de estrés y carga

Esta evaluación se centró en medir y analizar la capacidad del sistema para mantener un rendimiento constante bajo diferentes condiciones de carga. Se diseñó un plan de pruebas utilizando Apache JMeter, que recopiló datos sobre los tiempos de respuesta y el estado de las solicitudes HTTP (GET, POST, PUT y DELETE), simulando un entorno de estrés continuo. Este enfoque permitió obtener una visión precisa del comportamiento del sistema a lo largo del tiempo. La configuración del entorno de hardware y los detalles del plan de pruebas se presentan en las tablas VIII y IX.

Tabla VIII
Configuración del entorno hardware.

Plan de Prueba JMeter	Entorno Hardware
• Cantidad de usuarios: 100 • Intervalo: 10 segundos • Duración: 20 minutos	• Procesador Core i7 12th generación • 32 GB RAM

Tabla IX
Rutas del sistema pre-mantenimiento y post-mantenimiento

No	Descripción	Tipo	Pre-Mantenimiento	Post-Mantenimiento
1	Obtener Página Principal	GET	/home	/
2	Iniciar Sesión	GET	/login	/login
3	Iniciar Sesión (Enviar Datos)	POST	/login	/login
4	Consultar Usuarios	GET	/usuarios	/users
5	Crear Nuevo Usuario	GET	/usuarios-nuevo	/user/create
6	Guardar Usuario	POST	/usuarios-guardar	/api/users
7	Editar Usuario	GET	/usuarios-editar/{id}	/users/{id}/edit
8	Eliminar Usuario	DELETE	/usuarios-eliminar/{id}	/api/users/{id}
9	Consultar Niños	GET	/ninios	/children
10	Consultar Buzones	GET	/buzones	/mailbox
11	Consultar Buzón de Niño	GET	/niño-buzon/{id}	/mailbox/{id_child}
12	Crear Cartas Nuevas	POST	/crear-cartas-nuevo	/api/mails/answer
13	Cerrar Sesión	POST	/ logout	/logout

Tabla X
Resultados de la subcaracterísticas de la fiabilidad

Subcaracterística	Pre-Mantenimiento		Post-Mantenimiento	
	Alcanzado	Ponderación	Alcanzado	Ponderación
Madurez (25%)				
Eficacia de eliminación de defectos	0.00%	0.00%	100.00%	25.00%
Disponibilidad (25%)				
Tiempo medio disponible	84.96%	21.24%	93.13%	23.28%
Tolerancia a fallos (25%)				
Tiempo medio entre fallas (MTBF)	17.40%	4.35%	82.10%	20.52%
Capacidad de recuperación (25%)				
Tiempo de recuperación promedio	38.16%	8.79%	80.00%	20.00%
TOTAL 100%		34.38 %		88.80%

D. Evaluación de atributos de fiabilidad

Para medir las subcaracterísticas de la fiabilidad, se utilizó JMeter mediante pruebas de madurez, disponibilidad, tolerancia a fallos y capacidad de recuperación. Se ejecutaron múltiples solicitudes simultáneas y se monitoreó el comportamiento del sistema en condiciones de uso continuo. Los resultados obtenidos con JMeter permitieron evaluar el desempeño del sistema en cada subcaracterística. La Tabla X presenta los porcentajes de cumplimiento y sus valores ponderados.

IV. Resultados y Discusión

El análisis con la herramienta Enlightn identificó varios problemas ocultos en el código del sistema, proporcionando información clave para mejorar su fiabilidad. Este proceso resultó en la generación de 14 informes de problemas y 19 solicitudes de cambio. La Figura 5 muestra los resultados de las pruebas de fiabilidad, destacando que el 85% de los casos fueron aprobados y el 15% fueron fallidos.

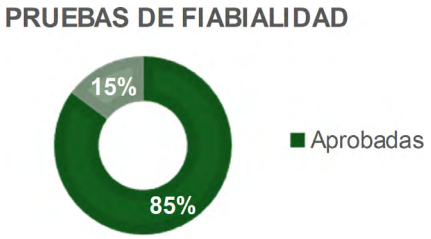

Fig. 5: *Resultados de pruebas de fiabilidad de Enlightn.*

El cálculo de las métricas de fiabilidad, como el Tiempo Medio entre Fallos (MTBF) y el Tiempo Medio de Reparación (MTTR), mostró mejoras significativas tras el mantenimiento. El MTBF incrementó de 3.48 a 16.42 minutos, lo que sugiere una mayor estabilidad del sistema debido a intervalos más prolongados entre fallos. Simultáneamente, el MTTR disminuyó de 2.62 a 1.25 minutos, indicando una recuperación más rápida del sistema ante fallos. Además, el número promedio de fallos (NE) se redujo de 4.47 a 1.46, y el tiempo promedio de operación (TO) aumentó ligeramente, reflejando un rendimiento más consistente con menos interrupciones. La tasa de fallos (Lambda) también disminuyó de 0.28 a 0.06, lo que reafirma una mejora en la fiabilidad general del sistema y evidencia la efectividad de las acciones correctivas implementadas.

El análisis de resultados muestra una mejora significativa en la fiabilidad del sistema tras el mantenimiento, evaluando las subcaracterísticas de madurez, disponibilidad, tolerancia a fallos

y capacidad de recuperación. La madurez, medida a través de la eficacia en la eliminación de defectos, pasó del 0% al 100%, reflejando una corrección completa de los errores detectados, lo que contribuyó a una mejora ponderada del 0% al 25%. En cuanto a la disponibilidad, el tiempo medio disponible aumentó de 84.96% a 93.13%, incrementando su ponderación de 21.24% a 23.28%, lo que indica una mayor accesibilidad del sistema tras las correcciones. La tolerancia a fallos, representada por el Tiempo Medio entre Fallos (MTBF), mejoró de un 17.40% a un 82.10%, con una ponderación que subió de 4.35% a 20.52%, lo que sugiere intervalos más largos sin fallos y una mayor resistencia del sistema. Por último, la capacidad de recuperación, medida por el tiempo de recuperación promedio, avanzó de un 38.16% a un 80%, aumentando su ponderación de 8.79% a 20.00%, evidenciando una recuperación más rápida ante fallos. En conjunto, la evaluación global de fiabilidad subió de 34.38% a 88.80%, lo que confirma la efectividad del mantenimiento correctivo en mejorar la estabilidad y el desempeño del sistema en todas las subcaracterísticas evaluadas. Estos resultados demuestran la eficacia de las intervenciones realizadas y subrayan la importancia de un mantenimiento bien planificado y ejecutado en sistemas críticos que gestionan flujos de trabajo esenciales. La figura 6 muestra los valores de la fiabilidad antes descrita.

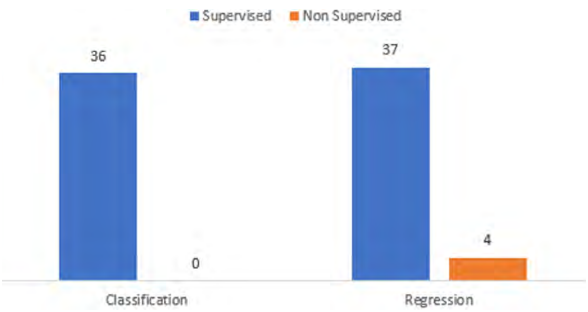

Fig. 6: *Nivel de fiabilidad por subcaracterísticas*

VI. Conclusiones

El mantenimiento correctivo implementado en el sistema de correspondencia de la Organización Comunitaria CACTU, utilizando la metodología Ágil MANTEMA, demostró ser altamente eficaz para mejorar la fiabilidad y funcionalidad del sistema. Los resultados reflejaron un aumento de la fiabilidad del 34.38% al 88.80%, con mejoras significativas en el Tiempo Medio entre Fallos (MTBF), que pasó de 3.48 a 16.42 minutos, y una reducción del Tiempo Medio de Reparación (MTTR), de 2.62 a 1.25 minutos. Además, el análisis de las subcaracterísticas de fiabilidad mostró incrementos considerables en la madurez, disponibilidad, tolerancia a fallos y capacidad de recuperación, lo que confirma la efectividad de la metodología utilizada. Estos hallazgos subrayan la importancia de un enfoque sistemático y planificado para el mantenimiento de software en entornos críticos, mejorando tanto la estabilidad operativa como la capacidad de respuesta ante fallos.

VII. Referencias

[1] U. Kaur and G. Singh, “A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts,” Int J Comput Appl, vol. 118, no. 1, pp. 6–7, 2015, Accessed: Nov. 08, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:55976322.

[2] A. Kamei, S. Kim, and Y. Tanaka, "A Software Maintenance Methodology: An Approach Applied to Mitigate Software Aging," in Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), Adelaide, Australia, Sept. 28 - Oct. 2, 2020, pp. 255-262. doi: 10.1109/ICSME46990.2020.00032.

[3] A. Gokhale, S. Gallagher, and T. Tremblay, "A Software Maintenance-Focused Process and Supporting Toolset for Academic Environments," 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), Adelaide, Australia, pp. 357-362, Sept. 2020. DOI: 10.1109/ICSME46990.2020.00042.

[4] E. Manotas, S. Yáñez, C. Lopera, and M. Jaramillo, “Estudio del efecto de la de pendencia en la estimación de la confiabilidad de un sistema con dos modos de falla concurrentes,” Dyna (Medellin), p. 30, 2008.

[5] S. A. R. Lopez, “Definición de una

secuencia de refactorización que permite mejorar la mantenibilidad minimizando el impacto negativo a la eficiencia basados en la medición de atributos de calidad internos.," Working papers Maestría en Ingeniería de Sistemas, vol. 4, no. 1, Aug. 2019, doi: 10.15765/WPMIS.V4I1.1241.

[6] T. C. Gutiérrez, R. Martínez, K. Soto, and J. R. Z. Morales, "REVISIÓN DE LITERATURA DEL ERROR HUMANO Y SU ESTUDIO EN LA INDUSTRIA Y LOS SERVICIOS," 2015.

[7] R. Pérez, F. Ruiz, I. Rodríguez, M. Polo, and M. Piattini, Mantenimiento y Evolución de Sistemas de información, 1ra ed. Madrid: Grupo Editorial RA-MA, 2019.

[8] P. Pisco, M. Marcillo, J. Gutiérrez, and J. Cedeño, "CONTROL DE MANTENIMIENTO PREVENTIVO EN COMPUTADORES A NIVEL DE SOFTWARE," UNESUM-Ciencias. Revista Científica Multidisciplinaria. ISSN 2602-8166, vol. 4, no. 1, pp. 143–154, May 2020, doi: 10.47230/UNESUM-CIENCIAS.V4.N1.2020.213.

[9] M. F. Fernández, J. G. Ruiz, and R. Acosta, "Analizador de Ensamblados orientados a objetos para el mantenimiento de software," CULCyT, pp. 29–34, 2015.

[10] M. Polo, "MANTEMA: una metodología paramantenimiento de software," Universidad de Castilla, La Mancha, 2000.

[11] M. M. Manhães, M. C. F. P. Emer, and L. Bastos, "Classifying defects in software maintenance to support decisions using hierarchical ODC," Anais do Computer on the Beach, p. 290, 2014.

[12] F. J. Pino, F. Ruiz, J. Triñanes, and F. Garcia, "Agil_MANTEMA: Una Metodología de Mantenimiento de Software para Pequeñas Organizaciones," 2008, Accessed: Jun. 08, 2024. [Online]. Available: https://www.researchgate.net/publication/221595280

[13] F. Pino, F. Ruiz, and S. Salas, "Ágil Mantema (Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica)," p. 13, 2008.

[14] ISO/IEC, "Systems and software engineering - Software life cycle processes," Art. no. ISO/IEC/IEEE 12207:2017, 2017.

[15] ISO/IEC, "System and Software Quality Requirements and Evaluation," 2011, Accessed: Feb. 27, 2024. [Online]. Available: https://iso25000.com/index.php/normas-iso-25000/iso-25010

[16] PHP Framework Intergroup, "Extended Coding Style." 2019. Accessed: Jun. 29, 2024. [Online]. Available: https://www.php-fig.org/psr/psr-12/.