

INGENIERÍA EN SOFTWARE

## REPORTE DE CASOS DE ESTUDIO DE MANTENIMIENTO DE SOFTWARE

PRESENTA

PATONI SOSA SHANNON GUIDO  
RODRÍGUEZ JAIME GEOVANI  
TORAL CARRERA JONATHAN YAIR  
CRISTOPHER EDMUNDO VIVEROS GERMÁN

S  
DOCENTE  
**NÉSTOR APOLO LÓPEZ GONZÁLEZ**

GRUPO  
**2825/S**

MATERIA  
**CALIDAD DE SOFTWARE**



## INDICE

INTRODUCCIÓN .....	3
EVIDENCIA 1. PRESENTACIÓN ELECTRÓNICA MANTENIMIENTO DE SOFTWARE 14764.4	
EVIDENCIA 2. REPORTE DE LOS 8 CASOS DE ESTUDIO. ....	30
EVIDENCIA 3. RESÚMEN DEL ISO 12207.....	45
EVIDENCIA 4. RESÚMEN DEL PROCESO DE ELIMINACIÓN 12207. ....	47
EVIDENCIA 5. GLOSARIO DE PROCESO ITERATIVO.....	49
EVIDENCIA 6. RESÚMEN DE LA NORMA ISO/IEC 14764:2006.....	51
EVIDENCIA 7. RESÚMEN DE MANTENIMIENTO ADAPTIVO, CORRECTIVO, MANTENIBILIDAD Y SOLICITUD DE MODIFICACIÓN.....	52
EVIDENCIA 8. RESÚMEN DE MANTENIMIENTO PERFECTIVO, PREVENTIVO, INFORME DE PROBLEMA Y TRANSICIÓN DE SOFTWARE. ....	53
EVIDENCIA 9. PROCESO DE MANTENIMIENTO. ....	54
EVIDENCIA 10. EJEMPLO DE MANTENIMIENTO PREVENTIVO. ....	55
EVIDENCIA 11. EJEMPLO DE PLAN DE MANTENIMIENTO ADAPTATIVO DE SOFTWARE. .....	63
EVIDENCIA 12. EJEMPLO DE PLAN DE MANTENIMIENTO PERFECTIVO. ....	69
EVIDENCIA 13. EJEMPLO DE PR (FORMATO) .....	73
EVIDENCIA 14. EJEMPLO DE MR (FORMATO) .....	74
CONCLUSIONES .....	78
REFERENCIAS BIBLIOGRÁFICAS.....	79

## INTRODUCCIÓN

El mantenimiento y la gestión de sistemas tecnológicos y operativos son pilares fundamentales para el éxito de las organizaciones en diversos sectores, desde la industria automotriz y hospitalaria, hasta el ámbito educativo y empresarial. A lo largo de los años, la evolución tecnológica ha permitido desarrollar estrategias y herramientas más sofisticadas para abordar los retos asociados al mantenimiento, la optimización de recursos y la mejora continua de procesos.

El mantenimiento de software es un proceso esencial para garantizar la continuidad, eficiencia y evolución de los sistemas tecnológicos en diferentes industrias. A medida que las organizaciones dependen cada vez más de soluciones digitales para sus operaciones, la necesidad de mantener y mejorar el software se ha vuelto una prioridad estratégica. Este documento recopila diversas evidencias y casos de estudio que reflejan la importancia del mantenimiento en diferentes contextos, desde el ámbito industrial y educativo hasta sectores críticos como la salud y la aviación.

A lo largo del reporte, se analizan las distintas categorías de mantenimiento de software, incluyendo el correctivo, adaptativo, perfectivo y preventivo, así como su impacto en la optimización de recursos, reducción de fallos y mejora en la experiencia del usuario. Mediante ejemplos concretos, se destacan estrategias y herramientas utilizadas para mejorar el rendimiento de los sistemas, la integración de nuevas tecnologías y la alineación con estándares internacionales como ISO/IEC 14764 e IEEE 1219.

El propósito de este informe es proporcionar un marco de referencia sobre la aplicación del mantenimiento de software en escenarios reales, evidenciando la necesidad de enfoques estructurados y metodologías eficientes para garantizar la sostenibilidad y evolución de los sistemas tecnológicos.

EVIDENCIA 1. PRESENTACIÓN ELECTRÓNICA MANTENIMIENTO DE SOFTWARE 14764.

# MANTENIMIENTO DE SOFTWARE

ESTÁNDAR INTERNACIONAL  
ISO/IEC/IEEE 14764



Patoni Sosa Shannon Guido  
Rodríguez Jaime Geovani  
Toral Carrera Jonathan Yair  
Viveros Germán Christopher Edmundo

# 1. Introducción



## 1.1

### Visión General

Guia el mantenimiento de software según la norma ISO/IEC/IEEE 12207:2017, cubriendo tipos de mantenimiento, procesos relacionados y aspectos de hardware.



## 1.2

### Propósito

Identificar cómo puede invocarse durante la adquisición y la explotación en el proceso de mantenimiento.



## 1.3

### Campo de aplicación

Se tiene como objeto proporcionar orientación para la planificación y el mantenimiento de productos o servicios de software tanto si se realiza interna como externamente a una organización.



## 1.4

### Limitaciones

Se describe el marco del proceso de mantenimiento, pero no se especifica los detalles de cómo implementar o realizar las actividades.



## 2. Referencias Normativas

Se refiere a los documentos, estándares, guías o normativas externas que son esenciales para su comprensión, aplicación o cumplimiento. Estas referencias proporcionan un marco de referencia técnico o legal que debe seguirse para asegurar la validez del contenido del documento.



### 3. Términos, definiciones y términos abreviados

#### 3.1.1 Mantenimiento adaptativo

Adaptación de un software tras su entrega para mantenerlo funcional en un entorno cambiante.

#### 3.1.2 Mantenimiento aditivo

Mejora de un software tras su entrega para añadir funciones o características.

#### 3.1.3 Corrección

Modificación de un software tras su entrega para corregir problemas detectados.

#### 3.1.4 Mantenimiento correctivo

Corrección de problemas en un software tras su entrega.

#### 3.1.5 Mantenimiento de emergencia

Modificación temporal para mantener un sistema operativo hasta su mantenimiento correctivo.

#### 3.1.6 Mantenibilidad

Grado de eficacia y eficiencia con que puede modificarse un producto.



#### 3.1 TÉRMINOS Y DEFINICIONES

### 3. Términos, definiciones y términos abreviados

#### 3.1.7 Mejoras

Cambio de software que aborda e implementa un nuevo requisito.

#### 3.1.8 Solicitud de modificación

Elemento que describe los cambios propuestos en un producto o servicio.

#### 3.1.9 Mantenimiento perfectivo

Modificación de un software para mejorar su rendimiento, mantenibilidad y atributos, y proporcionar mejoras e información a los usuarios.



### 3.1 TÉRMINOS Y DEFINICIONES

#### 3.1.10 Mantenimiento preventivo

Modificación de un software tras su entrega para corregir fallos latentes antes de que afecten al sistema.

#### 3.1.11 Reporte de problemas

Documento utilizado para identificar y describir los problemas detectados en un producto de software.

#### 3.1.12 Mantenimiento de Software

Conjunto de actividades necesarias para dar soporte a un sistema informático.

### 3. Términos, definiciones y términos abreviados

#### 3.1.13 Sostenimiento de Software

Conjunto de actividades necesarias para dar soporte a un sistema informático.



#### 3.1 TÉRMINOS Y DEFINICIONES

#### 3.1.14 Transición

Actividades para trasladar un servicio, sistema o componente entre entornos.



### 3. Términos, definiciones y términos abreviados

#### 3.2 Términos Abreviados

CM	Gestión de la configuración (Configuration Management).
COTS	Software comercial (Commercial-Off-The-Shelf Software).
MBSE	Ingeniería de sistemas basada en modelos (model-based systems engineering).
SEE	Entorno de Ingeniería de Software (Software Engineering Environment).
STE	Entornos de Prueba de Software (Software Test Environment).





## 4. Conformidad

Esto implica que el documento proporciona directrices para aplicar el proceso de mantenimiento según la norma ISO/IEC/IEEE 12207. Aunque no se puede afirmar que se cumple con este documento en sí, sí se puede hacer con el proceso de mantenimiento que describe y su adaptación. Las únicas cláusulas obligatorias provienen de la norma ISO/IEC/IEEE 12207, y los requisitos de esta norma que se repiten en el documento están destacados en recuadros. Además, se incluye el número de la cláusula correspondiente de la norma ISO/IEC/IEEE 12207 junto a cada requisito.



## 5. Aplicación del documento “Procesos del Ciclo – Mantenimiento”.

### 5. Aplicación de este documento

Describe cómo debe utilizarse el documento en el contexto específico para el que fue creado.

#### 5.1. Generalidades

Proporciona una visión general sobre la aplicabilidad del documento y su relevancia dentro del contexto específico.



#### 5.2. Proceso de mantenimiento

Detalla los pasos, actividades y tareas necesarias para implementar y gestionar el mantenimiento de manera efectiva.

#### 5.3. Organización del documento

Explica cómo está estructurado el documento, facilitando a los usuarios identificar y acceder a las secciones relevantes según sus necesidades específicas.



## 6. Proceso de mantenimiento

Es el conjunto estructurado de acciones, prácticas y procedimientos destinados a preservar o restaurar un sistema, software o equipo a su estado óptimo de funcionamiento. Su propósito es garantizar la continuidad operativa, prevenir fallos y mantener la eficiencia a lo largo del tiempo.

### 6.1. Actividades y tareas de mantenimiento

Acciones planificadas para garantizar la funcionalidad de un sistema o software.

#### 6.1.1. Generalidades

Describe principios clave del mantenimiento y su importancia operativa.

**Preventivo:** Inspecciones regulares y acciones para prevenir fallos.

**Correctivo:** Reparación de fallos una vez detectados.

**Predictivo:** Uso de datos y métricas para anticipar posibles fallos.



## 6. Proceso de mantenimiento

### 6.1.2. Estrategia de mantenimiento

Define el enfoque general a seguir para gestionar las actividades de mantenimiento.

### 6.1.3. Planificación del mantenimiento

Consiste en la organización temporal de las tareas de mantenimiento, considerando recursos, personal y tiempos de ejecución.

### 6.1.4. Planes y procedimientos de mantenimiento

Documentación formal de los pasos a seguir para cada tipo de mantenimiento. Incluye:

- Manuales de procedimientos.
- Protocolos de seguridad.
- Documentación técnica de los equipos o software.

### 6.1.5. Revisión de los requisitos de mantenimiento

Evaluación periódica de las necesidades y especificaciones del mantenimiento.

## 6. Proceso de mantenimiento

### 6.1.6. Análisis del impacto de los cambios de mantenimiento

Evaluación de las consecuencias que pueden surgir al modificar los planes de mantenimiento.

### 6.1.8. Monitoreo de la calidad y disponibilidad de elementos de reemplazo:

Garantizar que los repuestos y herramientas estén disponibles y sean de alta calidad para minimizar tiempos de inactividad.

### 6.1.7. Mediciones de mantenimiento

Definición e implementación de indicadores clave de rendimiento (KPIs) como:

- Tiempo medio entre fallos (MTBF).
- Tiempo medio de reparación (MTTR).
- Tasa de fallos.

### 6.1.9. Gestión de resultados de mantenimiento y logística

Implica la documentación y análisis de los resultados obtenidos tras la ejecución de tareas de mantenimiento.

## 6. Proceso de mantenimiento

### 6.2. Análisis de problemas y modificaciones

Proceso formal para identificar, documentar y resolver problemas o realizar modificaciones en un sistema.

#### 6.2.1. Tarea de análisis de problemas y modificaciones

Identificación y diagnóstico de problemas, evaluando su impacto y origen.

#### 6.2.2. Revisión de viabilidad de MR/PR:

Evaluación de la posibilidad de realizar una modificación o resolver un problema, considerando factores como:

- Costos.
- Recursos disponibles.
- Impacto en el sistema.



## 6. Proceso de mantenimiento

### 6.2.3. Replicación o verificación de MR/PR:

Intentar reproducir el fallo o confirmar la necesidad de una modificación antes de proceder con la implementación.

### 6.2.4. Implementación de MR/PR:

Ejecución de la solución o modificación propuesta.

### 6.2.5. Revisión y registro de MR/PR:

Documentación del proceso de implementación y validación.

### 6.2.6. Aprobación de MR/PR:

Validación formal de que la modificación es adecuada y cumple con los estándares.

## 6. Proceso de mantenimiento

### 6.2.7. Registro adicional de análisis de MR/PR:

Almacenamiento de registros detallados sobre el análisis y las decisiones tomadas.

### 6.2.9. Revisión del sistema modificado de MR/PR:

Evaluación del sistema después de la modificación para garantizar que funcione correctamente.

### 6.2.8. Implementación de MR/PR mediante procesos técnicos:

Uso de herramientas y metodologías técnicas específicas para aplicar la solución.

### 6.2.10. Aprobación e implementación final de MR/PR:

Validación y autorización final para la implementación completa de la modificación o solución al problema.



## 7. Desmantelamiento del software

Esta sección aborda cómo gestionar el final del ciclo de vida de un software, asegurando que su desconexión o retiro se realice de manera controlada, minimizando riesgos y preservando la información clave.

### 7.1. Generalidades

Proporciona una visión general del concepto de desmantelamiento, destacando su importancia para evitar problemas en el entorno operativo y liberar recursos. Se debe considerar cuando el software ya no es necesario, es reemplazado, o su mantenimiento ya no es viable.

## 7.2. Estrategia de desmantelamiento:

### 7.2.1. Generalidades

Introduce las estrategias clave para retirar software de manera efectiva, asegurando que el proceso sea bien planificado.

### 7.2.3 Tareas de desmantelamiento

Lista las actividades necesarias para la desconexión del software, como:

- Verificación de que el software ya no sea crítico.
- Migración de datos a sistemas alternativos.
- Desinstalación segura del software.

### 7.2.2 Elementos de la estrategia de desmantelamiento

Identifica los factores esenciales para planificar el desmantelamiento:

- Evaluación de riesgos.
- Recursos.
- Planificación.

### 7.2.4 Notificación de desmantelamiento

Detalla los requisitos para informar a las partes interesadas sobre la decisión de desmantelar el software, como los usuarios, el equipo técnico y los gestores.



## 7.2. Estrategia de desmantelamiento:

### 7.2.5 Tareas de notificación de desmantelamiento

Especifica cómo comunicar el desmantelamiento:

- Crear un plan de comunicación.
- Enviar notificaciones formales a los afectados.
- Ofrecer soporte técnico o alternativas si es necesario.

### 7.2.6 Tareas posteriores al desmantelamiento

Describe las acciones necesarias tras el retiro del software, como:

- Limpieza de datos residuales para cumplir con normativas de privacidad y seguridad.
- Evaluación del impacto del desmantelamiento en los procesos o sistemas restantes.
- Actualización de la documentación para reflejar el retiro.

### 7.2.7 Archivado de registros

Se centra en la importancia de almacenar la documentación relevante del software desmantelado, como:

- Registros de cambios.
- Manuales de usuario y técnicos.
- Información sobre el proceso de desmantelamiento.



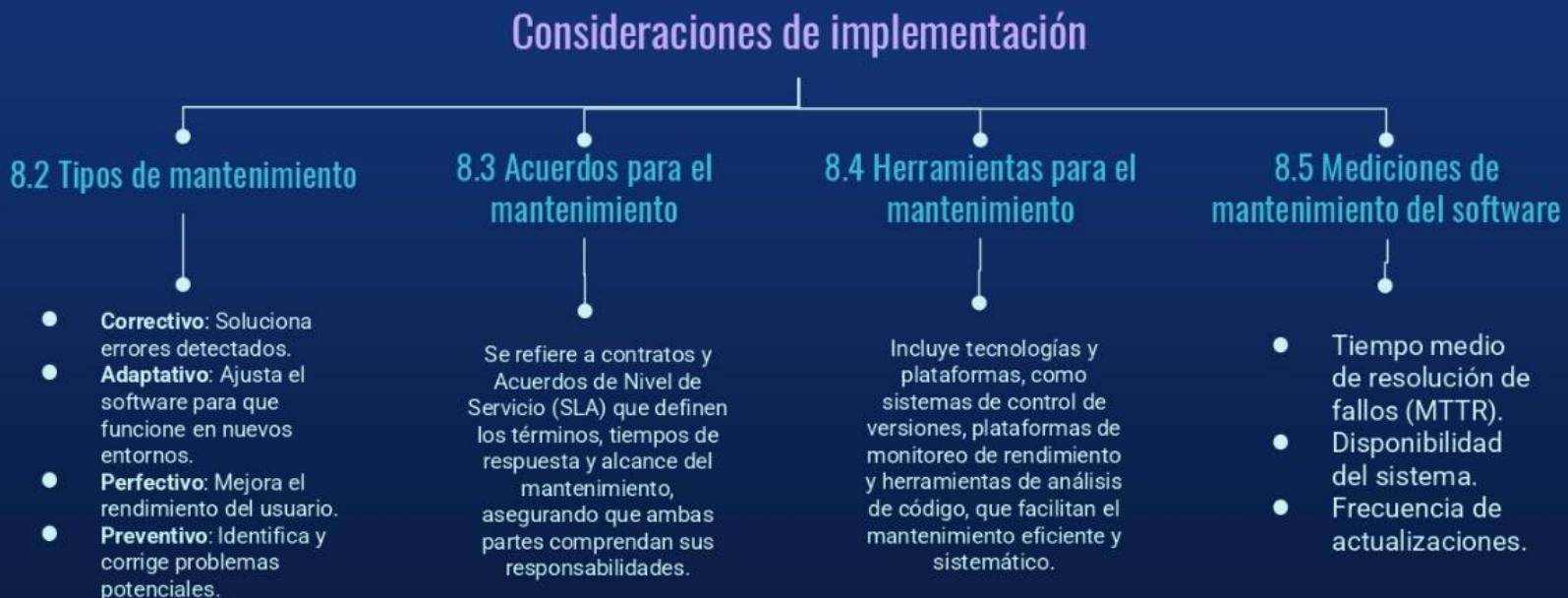
## 8. Consideraciones de implementación

Este tema abarca los aspectos necesarios para garantizar que el software sea mantenible y mejorable durante todo su ciclo de vida. La implementación efectiva del mantenimiento es clave para asegurar la calidad y continuidad operativa del sistema.

### 8.1 Generalidades

Introduce el concepto de mantenimiento de software, destacando su relevancia para prolongar la vida útil del software y asegurar su capacidad para adaptarse a cambios en el entorno, corregir errores y mejorar funcionalidades.

## 8. Consideraciones de implementación





## 8. Consideraciones de implementación

### Consideraciones de implementación

#### 8.6 Definición del proceso

Explica cómo establecer un proceso de mantenimiento efectivo, que incluye identificar necesidades, asignar recursos, definir roles y establecer procedimientos claros para gestionar y priorizar tareas de mantenimiento.

#### 8.7 Participación temprana en el desarrollo

##### 8.7.1 Generalidades

Involucrar al equipo de mantenimiento desde las primeras etapas del desarrollo mejora la calidad del software y facilita su mantenibilidad.

##### 8.7.2 Funciones de la organización de mantenimiento

Define los roles del equipo de mantenimiento, como proporcionar retroalimentación durante el desarrollo y garantizar que se sigan prácticas de diseño que faciliten el mantenimiento.

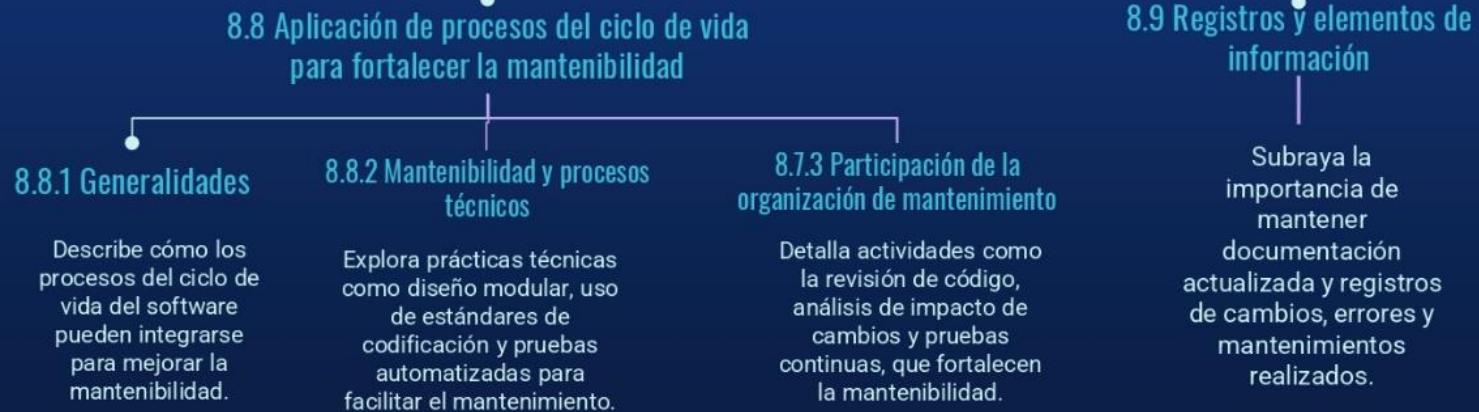
##### 8.7.3 Participación de la organización de mantenimiento

Incluye actividades como revisión de especificaciones, asesoramiento en decisiones técnicas y contribución a la documentación desde el inicio del proyecto.



## 8. Consideraciones de implementación

### Consideraciones de implementación





## 9. Plan de mantenimiento del software

El plan de mantenimiento del software es un documento formal que define cómo se gestionará y controlará el mantenimiento de un sistema de software a lo largo de su ciclo de vida.

### 9.1. Generalidades

Proporciona una visión general del concepto de desmantelamiento, destacando su importancia para evitar problemas en el entorno operativo y liberar recursos. Se debe considerar cuando el software ya no es necesario, es reemplazado, o su mantenimiento ya no es viable.

# 9. Plan de mantenimiento del software

## 9.1.1. Visión general

Introducción al propósito y objetivos del plan.

## 9.1.2. Identificación y control del plan

Gestión de versiones y control documental para asegurar la trazabilidad.

## 9.1.3. Alcance del mantenimiento

Definición clara de lo que se incluirá y lo que no en el plan de mantenimiento.



## 9.1. GENERALIDADES

## 9.1.4. Designación de la organización de mantenimiento

Asignación formal de equipos o personal responsables.

## 9.1.5. Referencias

Citas a normas, estándares y documentos de referencia.

## 9.1.6. Definiciones:

Clarificación de términos técnicos utilizados en el plan.

# 9. Plan de mantenimiento del software

## 9.1.7. Procesos

Descripción de las actividades específicas a seguir durante el mantenimiento.

**9.1.8. Organización y actividades de mantenimiento**  
Detalle sobre cómo se estructurará el equipo y sus responsabilidades.

## 9.1.9. Recursos

Definición de herramientas, licencias y personal necesario.



## 9.1. GENERALIDADES

### 9.1.10. Estimación de costos de mantenimiento

Análisis financiero del mantenimiento a lo largo del tiempo.

### 9.1.11. Capacitación

Citas a normas, estándares y documentos de referencia.

### 9.1.12. Requisitos de control de mantenimiento del software:

Normas de auditoría y control de calidad.

## 9. Plan de mantenimiento del software

### 9.2. Análisis de recursos

Evaluación de los recursos necesarios para implementar un plan de mantenimiento exitoso.

#### 9.2.1. Generalidades:

Explicación general sobre los tipos de recursos evaluados.



#### 9.2.2. Recursos de personal:

Evaluación de la capacidad y disponibilidad de los equipos humanos.

#### 9.2.3. Recursos del entorno:

Análisis de herramientas tecnológicas, licencias, hardware y software necesario.

## **EVIDENCIA 2. REPORTE DE LOS 8 CASOS DE ESTUDIO.**

### **CASO DE ESTUDIO REDUCCIÓN DEL TIEMPO DE INTERACTIVIDAD Y EL MANTENIMIENTO EN TORC ROBOTICS**

#### **Antecedentes**

Torc Robotics es un fabricante estadounidense de camiones autónomos y filial de Daimler Truck AG. La empresa, con sedes en diversos países, lidera el desarrollo de tecnologías de transporte de próxima generación. Con el crecimiento de la compañía, también aumentaron las complejidades operativas, haciendo necesaria una solución eficiente para gestionar activos, optimizar el mantenimiento y estandarizar procesos.

#### **Problemática/Necesidad**

- **Falta de estandarización:** Los flujos de trabajo y procesos dependían de conocimiento no documentado, dificultando la coherencia y la formación de nuevos empleados.
- **Complejidades tecnológicas:** Los sistemas existentes no se adaptaban a los requisitos únicos de sensores y hardware propios del transporte autónomo.
- **Gestión ineficiente de activos:** No había un sistema centralizado para coordinar el mantenimiento ni para implementar estrategias preventivas y predictivas.

#### **Efectos/Consecuencias**

- 1. Ineficiencia operativa:** Incremento de tiempos de inactividad y limitaciones en la formación de personal.
- 2. Problemas no resueltos:** Fallos recurrentes por falta de análisis detallado e implementación de mantenimiento predictivo.

#### **Solución ¿Cómo?**

Torc adoptó **eMaint X5**, un software de gestión del mantenimiento informatizado (GMAO), que permitió:

- 1. Personalización:** Adaptación del sistema a las necesidades específicas de TORC.
- 2. Centralización de datos:** Integración de información mediante API, conectando sensores, inventario y procedimientos de mantenimiento en un solo sistema.
- 3. Mantenimiento estandarizado:** Uso de procedimientos detallados y organizados jerárquicamente en eMaint para inspecciones y reparaciones.
- 4. Herramientas móviles:** Los técnicos utilizaron la aplicación X5 para reportar problemas en tiempo real, incluyendo imágenes y notas, agilizando la resolución.
- 5. Análisis avanzado:** Integración de inteligencia empresarial (BI) para identificar patrones de fallos y mejorar el mantenimiento predictivo.

## Resultados

- 1. Reducción del tiempo de inactividad:** Se redujo un 50% la mayoría de los meses mediante flujos de trabajo optimizados.
- 2. Estandarización y formación:** Flujos documentados facilitaron la incorporación de nuevo personal, abordando el problema de la escasez de técnicos especializados.
- 3. Mantenimiento predictivo efectivo:** Se establecieron desencadenantes automáticos para órdenes de trabajo basadas en patrones de fallo.
- 4. Resolución ágil:** La aplicación móvil permitió capturar datos y resolver problemas con mayor rapidez.
- 5. Mejora continua:** La capacidad de análisis granular de los datos permitió identificar y abordar causas raíz de fallos recurrentes, mejorando la fiabilidad general

## **CASO DE ESTUDIO 2 MANTENIMIENTO DE CENTRO DE REFERENCIAS PARA LA EDUCACIÓN AVANZADA (CREA)**

### **Antecedentes**

El sistema de gestión GSP es utilizado en el Centro de Referencia para la Educación Avanzada (CREA) en Cuba para gestionar actividades de superación pedagógica. Sin embargo, presentaba limitaciones en su funcionamiento debido a deficiencias en el mantenimiento, como la falta de documentación adecuada y planificación insuficiente. Estas carencias generaban un manejo improvisado y poco eficiente de las modificaciones necesarias.

### **Problemática/Necesidad**

La ausencia de un procedimiento formal y estandarizado para el mantenimiento de software en pequeñas organizaciones, específicamente en el sistema GSP, afectaba su capacidad de adaptación a nuevos requisitos y su rendimiento general. Esto se agravó con las demandas organizacionales cambiantes, como la creación de proyectos productivos, la vinculación de temas de investigación y la gestión de grandes volúmenes de información.

### **Efectos/Consecuencias**

- Incapacidad del sistema para adaptarse de manera eficiente a los nuevos procesos organizacionales.
- Insatisfacción de los usuarios debido a la baja calidad y funcionalidad del software.
- Compromiso de la vida útil del sistema y su sostenibilidad.
- Falta de trazabilidad y documentación en las actividades de mantenimiento.

### **Solución ¿Cómo?**

Se elaboró y aplicó un procedimiento específico para el mantenimiento de software, basado en estándares internacionales como ISO/IEC 14764 e IEEE 1219. Este procedimiento incluyó:

- Evaluación inicial del estado del software.
- Definición de modificaciones necesarias.

- Desarrollo de herramientas para facilitar la trazabilidad y documentación.
- Prácticas para evaluar la calidad del software antes y después de las modificaciones.

## **Resultados**

- Mejoras significativas en la calidad y mantenibilidad del sistema GSP.
- Extensión de la vida útil del sistema y mejor adaptación a cambios organizacionales.
- Incremento en la trazabilidad y transparencia del proceso de mantenimiento.
- Aumento en la satisfacción y confianza de los usuarios en el sistema.
- Creación de una base sólida para futuros trabajos de mantenimiento sostenible.

## **CASO DE ESTUDIO 3 MANTENIMIENTO DEL KERNEL DE LINUX**

### **Antecedentes**

El kernel de Linux comenzó en 1991 como un proyecto personal de Linus Torvalds para desarrollar un núcleo similar a Unix. Con el tiempo, se convirtió en el núcleo de sistemas operativos y dispositivos diversos gracias al modelo de software libre y de código abierto. Desde sus inicios, enfrentó desafíos como crear una base de código modular y gestionar las contribuciones de miles de desarrolladores.

### **Problemática/Necesidad**

El mantenimiento del kernel de Linux enfrenta desafíos constantes, incluyendo:

- Adaptarse a avances tecnológicos, como soportar nuevas arquitecturas de hardware.
- Gestionar y corregir bugs y vulnerabilidades de seguridad de manera oportuna.
- Escalar eficientemente sin comprometer el rendimiento.
- Coordinar y revisar contribuciones de miles de desarrolladores para evitar conflictos y errores.

### **Efectos/Consecuencias**

- Riesgo de problemas de seguridad y vulnerabilidades sin corregir.
- Posibilidad de afectar el rendimiento general debido a nuevas funcionalidades mal integradas.
- Complejidad en la coordinación de una comunidad tan grande, lo que puede generar retrasos y conflictos en el desarrollo.

### **Solución ¿Cómo?**

Se implementaron estrategias específicas para abordar estos desafíos:

- **Organización Jerárquica:** Jerarquía clara de mantenedores (Torvalds, mantenedores de subsistemas, y contribuidores) para revisión y control.
- **Control de Versiones con Git:** Creación de Git para gestionar cambios a gran escala, resolver conflictos y rastrear modificaciones eficientemente.
- **Liberaciones Estables y LTS:** Modelo de lanzamientos regulares y versiones con soporte a largo plazo para estabilidad en sistemas críticos.
- **Comunicación y Transparencia:** Uso de listas de correo públicas para discutir y revisar cambios, fomentando colaboración abierta.

### **Resultados**

- **Crecimiento Sostenido:** El kernel creció a millones de líneas de código, soportando arquitecturas como x86, ARM, y RISC-V, y distribuciones populares como Ubuntu y Android.
- **Resolución Rápida de Vulnerabilidades:** Ejemplo destacado en 2018 con las vulnerabilidades Spectre y Meltdown, donde la comunidad generó parches en pocos días.
- **Adaptación Continua:** Incorporación de tecnologías emergentes como contenedores (Docker), nuevos sistemas de archivos (Btrfs), y computación en la nube.

## **CASO DE ESTUDIO 4 MANTENIMIENTO DEL SISTEMA PARA LA TRAMITACIÓN DE INTERRUPCIONES EN LOS SISTEMAS DE INFORMÁTICA Y COMUNICACIONES (TI)**

### **Antecedentes**

El sistema TI fue desarrollado para gestionar interrupciones en los sistemas de informática y comunicaciones de la Zona Oriente Norte de ECASA, en Cuba. Es crucial para garantizar la calidad de los servicios aeronáuticos, donde la comunicación desempeña un papel esencial. Se utilizó tecnología de software libre y se alineó con la normativa ISO 9001:2000. El mantenimiento se planteó para corregir fallos, adaptarlo a nuevos entornos y agregar funcionalidades solicitadas por los usuarios.

### **Problemática/Necesidad**

El sistema TI presentaba diversas carencias y deficiencias:

- **Carencias Funcionales:** Falta de métodos de seguridad (respaldos), un sistema de alertas, informes avanzados y la integración con otros sistemas como SIMA.
- **Deficiencias Técnicas:** Errores en la manipulación e integridad de la base de datos, y la falta de alineación con el Procedimiento Específico PE-2806-02.
- **Costos de Mantenimiento:** Elevado costo comparado con el desarrollo inicial, agravado por la complejidad del código.

### **Efectos/Consecuencias**

- Vulnerabilidad del sistema ante fallos en la base de datos.
- Ineficiencia en la tramitación de interrupciones debido a la falta de un sistema de alertas.
- Dificultad para mantener el sistema y agregar nuevas funcionalidades.
- Incremento en los tiempos y costos de mantenimiento debido a la falta de estándares y documentación en el código.

## **Solución ¿Cómo?**

Se implementaron diversas estrategias de mantenimiento:

- **Mantenimiento Correctivo:** Corrección de errores en la base de datos y cumplimiento del Procedimiento PE-2806-02.
- **Mantenimiento Adaptativo:** Uso de AJAX para mejorar velocidad y usabilidad, y adaptación del sistema para manejar nuevas entidades.
- **Mantenimiento Perfectivo:** Adición de 23 nuevos requisitos funcionales, rediseño visual, y mejoras en los menús y filtros.
- **Mantenimiento Preventivo:** Estándares de codificación, comentarios en el código, y creación de un sistema de ayuda para usuarios.

## **Tecnologías:**

PHP, PostgreSQL, Rational Unified Process (RUP), y la técnica AJAX.

## **Resultados**

- Incremento en la seguridad y confiabilidad del sistema.
- Reducción en tiempos de respuesta y errores en la manipulación de datos.
- Mejor experiencia de usuario gracias a una interfaz más intuitiva.
- Cumplimiento del Procedimiento PE-2806-02.
- Inclusión de nuevas funcionalidades como un sistema de alertas, optimizando la tramitación de interrupciones.

## **CASO DE ESTUDIO 5**

### **Antecedentes**

Con la llegada de nuevas tecnologías en las últimas décadas, el mantenimiento ha pasado de ser manual a sistemas automatizados basados en computación, lo que permite predicciones confiables. Estas tecnologías han facilitado la inspección predictiva y el control remoto en tiempo

real, optimizando los procesos de mantenimiento. La aplicación de estas herramientas se ha visto impulsada por la industria 4.0, incluyendo inteligencia artificial y aprendizaje automatizado.

### **Problemática/Necesidad**

A pesar de los avances tecnológicos, las empresas enfrentan desafíos al adoptar herramientas de mantenimiento predictivo. Algunas limitaciones incluyen la falta de cultura tecnológica, personal capacitado y adaptación a tecnologías avanzadas. Esto genera una necesidad de mejorar los procesos de recopilación y análisis de datos para optimizar recursos y costos.

### **Efectos/Consecuencias**

La falta de implementación adecuada lleva a una gestión ineficiente de recursos y mayores costos operativos. Por ejemplo, sin un sistema predictivo, las empresas pueden enfrentar gastos energéticos y de mantenimiento más elevados debido a fallas imprevistas en maquinaria crítica.

### **Solución ¿Cómo?**

- **Identificación de empresas clave:** Se realizó un análisis cualitativo basado en entrevistas para recolectar información sobre las prácticas actuales.
- **Desarrollo de sistemas propios:** Uso de software como Python para crear herramientas que permitan analizar y monitorear datos en tiempo real, categorizando información según variables críticas.
- **Implementación tecnológica:** Instalación de sensores y sistemas de automatización (CPO) para monitorear y controlar procesos, como el consumo energético y la gestión de recursos.

### **Resultados**

**Eficiencia energética:** Se logró una mejora significativa en el consumo energético. Por ejemplo, el costo de enfriar 500 toneladas de refrigeración se redujo de COP 1.825.000 por kWh a 1.500 toneladas por COP 4.320.000 por kWh.

**Optimización de recursos:** A través del análisis y automatización, se redujeron costos y tiempos de operación, maximizando la productividad en las empresas participantes.

## CASO DE ESTUDIO 6

### Antecedentes

El Hospital Escuela San Juan de Dios, inaugurado en 1997, es un centro médico clave para la región. En la Unidad de Cuidados Intensivos Neonatales (UCIN), la gestión de mantenimiento de equipos biomédicos es crítica para garantizar la continuidad de los servicios de salud. Previo al proyecto, la gestión de mantenimiento se realizaba manualmente, utilizando hojas de vida y órdenes de trabajo impresas, lo que resultaba ineficiente y propenso a errores. Esto ponía en riesgo la disponibilidad de los equipos y la vida de los pacientes.

### Problemática/Necesidad

Se identificaron los siguientes problemas:

1. **Gestiones manuales ineficientes:** Información dispersa y difícil de acceder debido al uso de formatos impresos.
2. **Falta de accesibilidad:** Un 62% del personal reportó dificultades para obtener información necesaria.
3. **Capacitación insuficiente:** El personal carecía de conocimientos sobre gestión moderna, limitando la posibilidad de mejoras.
4. **Falta de sistematización:** Los procesos no seguían una estructura clara ni un plan programado de mantenimiento.

### Efectos/Consecuencias

- Riesgo elevado de fallos en equipos biomédicos críticos.
- Pérdida de tiempo y recursos en la gestión de mantenimiento.

- Dificultades para implementar mejoras en los procesos.
- Impacto negativo en la calidad del servicio de salud, afectando directamente a los pacientes.

### **Solución ¿Cómo?**

**Se diseñó un software de gestión de mantenimiento asistido por ordenador (GMAO) que incluye las siguientes características:**

- **Planificación de actividades:** Programación de mantenimientos preventivos y correctivos.
- **Gestión de inventarios:** Registro y organización de información técnica de los equipos.
- **Generación de reportes:** Automatización en la creación de informes y órdenes de trabajo.
- **Interfaz intuitiva:** Para facilitar el uso y adopción del sistema por parte del personal.

### **Resultados**

- Automatización de tareas de mantenimiento, mejorando la eficiencia.
- Reducción en los tiempos de acceso a la información crítica.
- Generación rápida y precisa de reportes.
- Cambio cultural hacia la sistematización y la mejora continua en la gestión de mantenimiento.

### **Beneficios adicionales:**

- Mayor confiabilidad y disponibilidad de los equipos biomédicos.
- Mejora en la calidad de los servicios de salud para los pacientes.

## **CASO DE ESTUDIO 7 MANTENIMIENTO DE UN SISTEMA DE GESTIÓN DE INVENTARIOS PARA UNA TIENDA DE ROPA.**

### **Antecedentes**

La tienda "Moda y Estilo" implementó un sistema de gestión de inventarios diseñado para registrar y controlar las existencias de productos. Este sistema utiliza una base de datos para almacenar información como descripción, precio y cantidad en stock. Durante sus primeros años, el sistema cumplió con los requerimientos, pero con el crecimiento de la tienda y el aumento en el volumen de datos, comenzaron a surgir problemas.

### **Problemática/Necesidad**

Se identificaron los siguientes problemas:

1. **Uso de tecnologías obsoletas:** El sistema está basado en estándares que ya no son compatibles ni eficientes.
2. **Falta de mantenimiento:** No se han realizado actualizaciones ni mejoras, lo que ha causado la acumulación de errores.
3. **Interfaz poco intuitiva:** El diseño dificulta su uso, lo que afecta la productividad de los empleados.
4. **Ausencia de respaldo de datos:** No hay mecanismos para prevenir la pérdida de información en caso de fallos.

### **Efectos/Consecuencias**

- Lentitud en el sistema, lo que retrasa las operaciones diarias.
- Pérdida de datos críticos debido a la falta de respaldo.
- Baja productividad del personal por la dificultad de uso del sistema.
- Vulnerabilidades en la seguridad y autenticación de usuarios.

## **Solución ¿Cómo?**

Se proponen las siguientes acciones para mejorar el sistema:

1. **Actualización tecnológica:** Migrar a tecnologías más modernas y compatibles con estándares actuales.
2. **Mejora de la interfaz de usuario:** Rediseñar la experiencia para que sea más intuitiva y eficiente.
3. **Implementación de respaldos:** Establecer un proceso regular de copias de seguridad para proteger los datos.
4. **Depuración del sistema:** Realizar pruebas exhaustivas para identificar y solucionar errores.

## **Resultados**

La implementación de mejoras en el sistema de gestión de inventarios tendría los siguientes impactos positivos:

1. **Mayor eficiencia operativa:**
  - Reducción del tiempo necesario para registrar y consultar información sobre productos.
  - Aceleración de procesos de gestión de inventarios, como actualizaciones de stock y generación de reportes.
2. **Incremento en la productividad del personal:**
  - Una interfaz más intuitiva facilitaría el uso del sistema y reduciría los errores humanos.
3. **Mejor protección de datos:**

- El establecimiento de procesos de respaldo de datos garantizaría la recuperación rápida en caso de fallos o incidentes.

**4. Reducción de errores técnicos:**

- La depuración y el mantenimiento regular minimizarían los problemas de conexión y visualización de datos.

**5. Escalabilidad y sostenibilidad:**

- El uso de tecnologías modernas permitiría al sistema adaptarse al crecimiento de la tienda y manejar mayores volúmenes de datos en el futuro.

**6. Mayor satisfacción del personal:**

- Los empleados contarían con una herramienta más confiable y fácil de usar, mejorando su experiencia laboral.

## CASO DE ESTUDIO 8

### Antecedentes

Netflix es una plataforma líder de streaming con una arquitectura distribuida que soporta millones de usuarios globalmente. En 2012, sufrió una interrupción masiva en la víspera de Navidad debido a un fallo en Amazon Web Services (AWS), su proveedor de infraestructura en la nube.

### Problemática/Necesidad

El fallo ocurrió en el sistema de Elastic Load Balancer (ELB) de AWS, lo que provocó:

- Interrupciones en la capacidad de los usuarios para reproducir contenido.
- Dependencia excesiva de un solo proveedor de servicios en la nube.
- Insuficiente redundancia en las regiones afectadas.
- Falta de pruebas para escenarios críticos como alta demanda en días festivos.

### Efectos/Consecuencias

- Millones de usuarios quedaron sin acceso al servicio durante horas en una fecha clave.
- Riesgo significativo para la reputación de Netflix como proveedor confiable.
- Incremento en la presión para garantizar la resiliencia de la infraestructura.

### Solución ¿Cómo?

Netflix implementó varias estrategias para mejorar la resiliencia y mantenibilidad del sistema:

- **Arquitectura de Alta Disponibilidad:**
  - Adopción de un enfoque multi-región dentro de AWS.
  - Incremento de redundancias y replicación automática de servicios.
- **Pruebas de Resiliencia (Chaos Engineering):**

- Desarrollo de herramientas como Chaos Monkey para simular fallos en producción y fortalecer la infraestructura.
- **Minimización de Dependencias:**
  - Diversificación de proveedores con soluciones híbridas.
  - Estrategias "fallback" para mantener funciones básicas como la navegación durante fallos.

## Resultados

- **Mayor Resiliencia:** Netflix mantiene un tiempo de actividad cercano al 100%, incluso durante picos de demanda.
- **Cultura de Mantenibilidad:** Adopción de Chaos Engineering como estándar en la industria.
- **Reducción de Impactos:** Sistemas de respaldo que garantizan acceso básico durante interrupciones.
- **Consolidación Técnica:** Fortalecimiento de su reputación como líder técnico en el sector de streaming.

### EVIDENCIA 3. RESÚMEN DEL ISO 12207.

NOMBRE:	FOLIO		
TEMA: ISO-12207	DIA	MES	AÑO
MATERIA: 6.4.13 Proceso de Mantenimiento			

El propósito de este proceso es mantener la capacidad del sistema para proporcionar sus servicios, realizar acciones correctivas, adaptativas, perfectivas y preventivas. También se integra mantenimiento con gestión de configuración y gestión de activos de software.

Resultados:

- Identificar restricciones de mantenimiento en requisitos, diseño o arquitectura.
- Garantizar disponibilidad de sistemas o servicios habilitadores para mantenimiento.
- Proveer elementos reemplazadas, reparados o revisados.
- Informar necesidades de mantenimiento correctivo, adaptativo o perfectivo.
- Documentar datos de fallos y costos asociados.

Actividades y Tareas:

- a) Preparar el mantenimiento
  1. Definir estrategia de mantenimiento.
    - Priorizar y programar cambios.
    - Monitorizar necesidades de mantenimiento.
    - Evaluar diseño y arquitectura periódicamente.
    - Prever obsolescencia de componentes.
    - Medir rendimiento y efectividad del mantenimiento.
  2. Desarrollar estrategia logística.
    - Considerar almacenamiento, tasas de reemplazo y renovación.
  3. Identificar restricciones del mantenimiento.
    - Incorporarlas en los requisitos, arquitectura y diseño.
  4. Realizar análisis de compromisos:
    - Garantizar soluciones sostenibles y operables.

NOMBRE:	FOLIO		
TEMA:	DIA	MES	ANO
MATERIA:			

5. Planificar sistemas habilitadores ~~secundarios~~ para soporte.

6. Adquirir acceso a sistemas habilitadores.

~~8)~~

b) Mantenimiento:

- Identificar necesidades y evaluar cambios.
- Restaurar operación y aplicar correcciones.
- Mantenimiento preventivo y adaptativo.

c) Soporte logístico:

- Asegurar recursos y calidad de reemplazos.
- Gestionar distribución y almacenamiento.
- Verificar cumplimiento de soporte.

Ejemplo:

- Correctivo: Sistema de gestión de inventarios que da error al calcular el total de los productos. Debido a una mala implementación. El error se corrige para arreglar ese problema.
- Adaptativo: Una aplicación de Windows 10 se actualiza para garantizar su compatibilidad con Windows 11.
- Perfectivo: En un sistema de ventas en linea, se agrega una nueva funcionalidad para añadir criptomonedas.
- Preventivo: En un sistema con bibliotecas obsoletas, se debe actualizar para introducir bibliotecas actualizadas y compatibles.

## EVIDENCIA 4. RESÚMEN DEL PROCESO DE ELIMINACIÓN 12207.

NOMBRE:	FOLIO
TEMA:	DIA / MES / AÑO
MATERIA: <b>Proceso de eliminación</b>	

En este proceso, se desactiva y elimina el sistema o cualquiera de sus elementos de uso específico. Aborda los productos, archivos residuales y los asigna a una condición aceptable.

**Éxito del Proceso**

1. Requisitos de eliminación integrados en los elementos de diseño, arquitectura e implementación.
2. Disponibilidad de sistemas o servicios habilitadores para la eliminación.
3. Elementos del sistema o archivos residuales destruidos, almacenados, reciclados o recuperados según los requisitos de seguridad y protección.
4. Restauración del entorno o su estado original o aceptado.
5. Registro de las acciones y análisis de eliminación disponibles para auditorías y revisiones.

**Preparación para la eliminación:**

1. Definir una estrategia que contemple:
  - Terminación permanente de funciones.
  - Propiedad y responsabilidad de datos y propiedad intelectual.
  - Transformación del producto para minimizar riesgos.
  - Consideraciones de salud, seguridad, privacidad y medio ambiente.
  - Notificación a las partes interesadas.
  - Cronogramas y recursos para la eliminación.
2. Identificar restricciones relacionadas con los requisitos, diseño y técnicas de implementación.
3. Planificar sistemas habilitadores necesarios para soportar la eliminación.

NOMBRE:  
TEMA:  
MATERIA:

FOLIO /  
DIA MES AÑO

### Realizar la eliminación:

- Desactivar el sistema o elemento de software para su remoción.
- Eliminar el sistema y materiales no reutilizables de acuerdo con las normas.
- Retirar al personal operativo y registrar su conocimiento relevante.
- Reutilizar, reciclar o destruir elementos, según corresponda.
- Destruir elementos no reutilizables para evitar su regreso.

### Finalizar la eliminación:

- Confirma b. Identificación y tratamiento de riesgos para la seguridad.
- Archivar información recolectada para auditorías y revisiones futuras.
- Restaurar el entorno al estado original o acordado.

USUAL

## EVIDENCIA 5. GLOSARIO DE PROCESO ITERATIVO.

NOMBRE:  
TEMA: ISO/IEC 19764  
MATERIA:

FOLIO /  
DIA / MES / AÑO

Círculo de Mantenimiento de Software

Evaluar las Necesidades del Sistema

Implementar Actualizaciones

Planificar Mejoras Futuras

Monitorear el Rendimiento

Recopilar Retroalimentación

Visión General

Este estándar detalla la gestión del Proceso de Mantenimiento según la norma ISO/IEC 12207, incluyendo enmiendas, y define los tipos de mantenimiento. Ofrece orientación para planificar, ejecutar, controlar, revisar actividades.

Glosario:

Elabora un glosario que incluya las siguientes términos:

- Proceso iterativo. Un ciclo de trabajo repetitivo que se utiliza para mejorar un producto, servicio o iniciativa.
- Modelo. Se refiere a la representación abstracta del software que incluye su arquitectura, componentes y relaciones.
- Ejecución. Es la fase en la que se implementan los cambios y mejoras en el software.

Fas

USUAL

NOMBRE	FOLIO		
TEMA:	DIA	MES	AÑO
MATERIA			

- Fase de planificación: Es la etapa inicial del mantenimiento de software, en la que se identifican los objetivos, se analizan los requisitos y se establecen los planes para el mantenimiento.

- Herramientas: Son las tecnologías y aplicaciones utilizadas para apoyar el mantenimiento de software, como editores de código, herramientas de prueba y sistemas de control de versiones.

- Técnicas: Son los métodos y enfoques utilizados para realizar el mantenimiento de software, como la refactorización, la ingeniería y la prueba de regresión.

- Métodos: Son los enfoques sistemáticos y estructurados para realizar el mantenimiento de software, como el modelo IEEE.

- Recuperación: Se refiere a la capacidad del software para recuperarse de errores o fallas, y para minimizar el impacto de estos eventos en el funcionamiento del sistema.

### 1.3 Ámbito de aplicación (ISO/IEC/IEEE 14764)

- Esta norma proporciona orientación para el mantenimiento de software, complementando a la norma ISO/IEC 12207 (Ciclo de vida de software.)
- Es aplicable en situaciones entre dos partes o para autoimpresión.
- No está destinada a software desechable o soluciones a corto plazo, ni a productos personalizados por usuarios singulares.
- Se enfoca en el mantenimiento de productos de software listos para usar por parte de sus desarrolladores.
- Aplica a programas, código, datos y documentación, incluyendo elementos como software de prueba y entornos de prueba.

## EVIDENCIA 6. RESÚMEN DE LA NORMA ISO/IEC 14764:2006.

NOMBRE:	FOLIO /		
TEMA:	DIA	MES	ANO
MATERIA:			

### 1.4 Limitaciones

- Proceso de mantenimiento: La norma ISO/IEC 14764:2006 no está diseñada para que se pueda redamar conformidad directa con ella. En su lugar, se espera que los usuarios sigan la norma ISO/IEC 12207 en lo que respecta al proceso de mantenimiento del software.

### • Referencias Normativas

- ISO - IEC 9126-1: 2001: Este documento describe el modelo de calidad del producto de software. Es una guía fundamental para evaluar la calidad de los productos de software durante su ciclo de vida.
- ISO - IEC 12207:1995: Establece los procesos del ciclo de vida del software. La norma ISO/IEC 14764 se basa en este proceso de mantenimiento dentro de la ISO/IEC 12207

### 3. Términos y Definiciones

#### 3.1 Mantenimiento Adaptativo

La modificación de un producto de software, realizada después de su entrega, para mantenerlo utilizable en un entorno modificado o en constante cambio.

- Proporciona mejoras necesarias para adaptarse a los cambios en el entorno en el que el software debe operar.

#### 3.2 Mantenimiento Correctivo

La modificación reactiva de un producto de software realizada después de su entrega para corregir problemas detectados.

- Esta modificación repara el software para satisfacer los requisitos.

USUAL

## EVIDENCIA 7. RESÚMEN DE MANTENIMIENTO ADAPTIVO, CORRECTIVO, MANTENIBILIDAD Y SOLICITUD DE MODIFICACIÓN.

NOMBRE	FOLIO
TEMA	DIA / MES / AÑO
MATERIA	

3.7 Mantenimiento perfectivo  
Es la modificación de un producto de software después de la entrega para detectar y corregir fallos latentes antes que se manifiesten como fallos operacionales.

Objetivos principales:

- Mejoras para los usuarios: Cambios que mejoran la experiencia del usuario final.
- Mejora de la documentación del programa: Actualización y enriquecimiento de la documentación técnica.
- Recodificación: Modificaciones en el código para mejorar el rendimiento, la mantenibilidad y otros atributos de software.

3.8 Mantenimiento preventivo  
Modificación de un producto de software después de la entrega para detectar y corregir fallos latentes antes de que se conviertan en fallos operacionales.

Objetivos:

- Prevención:  
Evitar que los fallos latentes se conviertan en problemas.

3.9 Informe del problema  
Término utilizado para identificar y describir problemas detectados en un producto de software.

Típos de PR:

- Directos: Presentados directamente para señalar un fallo en el software.
- Indirectos: Establecidos después de un análisis de impacto realizado a partir de solicitudes de modificación.

## EVIDENCIA 8. RESÚMEN DE MANTENIMIENTO PERFECTIVO, PREVENTIVO, INFORME DE PROBLEMA Y TRANSICIÓN DE SOFTWARE.

NOMBRE	FOLIO		
TEMA	DÍA	MES	AÑO
MATERIA			
<p>3.10 Mantenimiento de Software</p> <p>Conjunto de actividades necesarias para proporcionar soporte rentable y efectivo a un sistema de software durante su ciclo de vida.</p> <p>Objetivos:</p> <ul style="list-style-type: none"><li>• Sostenibilidad a largo plazo.</li><li>• Soporte eficiente</li></ul> <p>3.11 Transición de software</p> <p>Sucesión controlada y coordinada de acciones donde el desarrollo de software pasa de la organización que realizó el desarrollo inicial a la organización encargada del mantenimiento.</p> <p>Objetivos:</p> <p>Asegurar que el software esté listo para ser mantenido y operado eficientemente por el equipo de mantenimiento.</p>			

## EVIDENCIA 9. PROCESO DE MANTENIMIENTO.

NOMBRE:	FOLIO		
TEMA: <b>Proceso de Mantenimiento</b>	DIA	MES	ATO
MATERIA:			

a) Proceso de Implementación (MR/PR)  
Es cuando se reciben los datos de MR y PR, sin estos, no se puede iniciar el mantenimiento. Es fundamental para identificar que tipo de Mantenimiento usar.

b) Análisis del problema y modificación  
Se busca y analiza si el problema o el conflicto es verídico y conforme al tipo de mantenimiento, buscar posibles soluciones.

c) Implementación de la modificación  
Una vez se haya hecho el análisis y la planeación, se empieza a poner en funcionamiento y en ejecución la solución llevada a cabo.

d) Revisión / aceptación del mantenimiento  
Si la solución ya se implementó, se realizar pruebas de funcionamiento para comprobar que sirva la implementación sirva y haya solucionado el problema.

e) Migración  
Si la implementación ya se probó y ~~ta~~ el problema fué solucionado, se empieza a migrar al nuevo sistema.

f) Retiro  
Si el sistema ya llegó a su final de ciclo, el software se retira y ya no se le daí mas mantenimiento.

## EVIDENCIA 10. EJEMPLO DE MANTENIMIENTO PREVENTIVO.

Evidencia 6. Ejemplo de plan de mantenimiento preventivo de software

### 1. Objetivo:

Asegurar la continuidad, el correcto funcionamiento y el rendimiento óptimo del software a través de actividades planificadas que prevengan errores y reduzcan fallos críticos.

### 2. Alcance:

Este plan cubre todas las actualizaciones y revisiones de software necesarias para los sistemas operativos, aplicaciones críticas y sistemas de seguridad. Se enfocará en los sistemas que requieran intervención para prevenir problemas y mejorar su rendimiento.

### 3. Actividades y tareas del mantenimiento preventivo:

#### Actividad 1: Revisión de Actualizaciones de Seguridad

- Descripción:

Verificar y aplicar las actualizaciones de seguridad de software (antivirus, firewall, sistema operativo) para prevenir vulnerabilidades de seguridad.

- Tiempo Estimado: 2 horas

- Aplicaciones que se ocupan: Antivirus

(ej. Norton, Kaspersky). Firewall (ej.  
ZoneAlarm).

- Sistema Operativo (Windows Update, Linux Updates).

- Responsable: Administrador de sistemas.

- Parámetros Previos:

El sistema debe estar actualizado a su última versión disponible, sin parches pendientes.

- Parámetros Posteriores:

El software debe indicar que todas las actualizaciones de seguridad han sido aplicadas correctamente.

## Actividad 2: Optimización del Sistema

- **Descripción:**

Ejecutar herramientas de optimización para limpiar el sistema de archivos temporales, eliminar registros obsoletos y mejorar el rendimiento general del sistema.

- **Tiempo Estimado:** 1 hora.

- **Aplicaciones que se ocupan:**

CCleaner.

Disk Cleanup (Herramienta nativa de Windows). Herramientas del sistema operativo.

- **Responsable:** Técnico de TI.

- **Parámetros Previos:**

El sistema debe funcionar sin errores graves y sin archivos innecesarios.

- **Parámetros Posteriores:**

El sistema debe mostrar un rendimiento mejorado y estar libre de archivos temporales.

## Actividad 3: Revisión de Copias de Seguridad

- **Descripción:**

Verificar que las copias de seguridad (backup) se realicen de manera regular, asegurando que los datos críticos se almacenen adecuadamente.

- **Tiempo Estimado:** 3 horas.

- **Aplicaciones que se ocupan:**

Software de respaldo como Acronis, Veeam, Windows Backup.

- **Responsable:** Responsable de respaldo de datos.

- **Parámetros Previos:**

El sistema de copias de seguridad debe estar funcionando correctamente, con copias recientes y válidas.

- **Parámetros Posteriores:**

Las copias de seguridad deben ser verificadas, y cualquier error o inconsistencia debe ser solucionado.

#### Actividad 4: Prueba de Integridad del Sistema

- **Descripción:**

Ejecutar diagnósticos para verificar la integridad de los archivos del sistema operativo, la memoria y el disco duro, asegurando que no haya fallos ni errores.

- **Tiempo Estimado:** 4 horas.

- **Aplicaciones que se ocupan:**

Herramientas de diagnóstico del sistema como chkdsk, Memtest.

Herramientas del sistema operativo para verificar discos y memoria.

- **Responsable:** Ingeniero de sistemas.

- **Parámetros Previos:**

El sistema debe estar funcionando de manera estable sin errores graves.

- **Parámetros Posteriores:**

El sistema debe estar libre de errores y fallos críticos, y debe mostrar un funcionamiento óptimo.

#### Actividad 5: Actualización de Software y Aplicaciones

- **Descripción:**

Verificar y aplicar actualizaciones en todas las aplicaciones del sistema, asegurando que estén actualizadas a su versión más reciente.

- **Tiempo Estimado:** 2 horas.
- **Aplicaciones que se ocupan:**  
Gestores de actualizaciones (ej. WSUS, gestor de actualizaciones de Linux). Aplicaciones de terceros (ej. Adobe, Office Suite).
- **Responsable:** Administrador de software.
- **Parámetros Previos:**  
Las aplicaciones deben estar en versiones anteriores que requieran actualizaciones.
- **Parámetros Posteriores:**  
Las aplicaciones deben estar actualizadas a su última versión estable.

## 5.1 Cronograma de Mantenimiento Preventivo

Actividad	Tiempo Estimado	Frecuencia	Responsable
Revisión de Actualizaciones de Seguridad	2 horas	Mensual	Administrador de sistemas
Optimización del Sistema	1 hora	Trimestral	Técnico de TI
Revisión de Copias de Seguridad	3 horas	Mensual	Responsable de respaldo
Prueba de Integridad del Sistema	4 horas	Semestral	Ingeniero de sistemas
Actualización de Software y Aplicaciones	2 horas	Trimestral	Administrador de software

## 5.2 Responsables del Mantenimiento Preventivo

- **Administrador de Sistemas:** Se encargará de la revisión de actualizaciones de seguridad y la

actualización del software.

- **Técnico de TI:** Será responsable de la optimización del sistema y la ejecución de las pruebas de diagnóstico.
- **Responsable de Respaldo de Datos:** Se encargará de verificar las copias de seguridad, asegurando que estén completas y sin errores.
- **Ingeniero de Sistemas:** Ejecutará las pruebas de integridad y asegurará que el software funcione correctamente.

#### Actividad 6. Objetivos del Mantenimiento Preventivo

- Optimizar el rendimiento del software asegurando que el sistema funcione de manera eficiente, libre de errores y con recursos adecuados.
- Prevenir problemas de seguridad aplicando actualizaciones y parches de seguridad de forma regular.
- Garantizar la integridad de los datos mediante copias de seguridad confiables y pruebas de recuperación.
- Mantener la estabilidad del sistema al asegurar que el software esté actualizado, sin fallos críticos y funcionando según lo esperado.

### PLAN DE MANTENIMIENTO PREVENTIVO DE SOFTWARE

<b>Organización:</b>		
<b>Departamento:</b>		
<b>Sección:</b>		<b>Hojas:</b>

#### INFORMACIÓN GENERAL

<b>Fecha de publicación:</b>	<b>Prioridad del mantenimiento:</b>
<b>Emitido por:</b>	<b>Puesto:</b>

## ACTIVIDADES DEL MANTENIMIENTO PREVENTIVO

### Revisión de Actualizaciones de Seguridad

Descripción:

Tiempo Estimado:

Aplicaciones que se ocupan:

Responsable:

Parámetros Previos:

Parámetros Posteriores:

### Optimización del Sistema

Descripción:

Tiempo Estimado:

Aplicaciones que se ocupan:

Responsable:

Parámetros Previos:

Parámetros Posteriores:

### Revisión de Copias de Seguridad

Descripción:

Tiempo Estimado:

Aplicaciones que se ocupan:

Responsable:

Parámetros Previos:
Parámetros Posteriores:

<b>Prueba de Integridad del Sistema</b>
Descripción:
Tiempo Estimado:
Aplicaciones que se ocupan:
Responsable:
Parámetros Previos:
Parámetros Posteriores:

<b>Actualización de Software y Aplicaciones</b>
Descripción:
Tiempo Estimado:
Aplicaciones que se ocupan:
Responsable:
Parámetros Previos:
Parámetros Posteriores:

## **EVIDENCIA 11. EJEMPLO DE PLAN DE MANTENIMIENTO ADAPTATIVO DE SOFTWARE.**

### **Plan de Mantenimiento Adaptativo**

#### **1. Propósito del Plan**

Ajustar el software a nuevos requisitos o cambios tecnológicos para mantener su compatibilidad y eficiencia.

#### **2. Alcance**

- Adaptación del software SCADA a cambios en hardware o sistemas operativos.
- Modificación de configuraciones para la integración de nuevos dispositivos.
- Ajuste de protocolos de comunicación según nuevas normativas o tecnologías.

#### **3. Fases del Mantenimiento Adaptativo**

##### **Fase 1: Análisis de Cambios y Nuevas Necesidades**

- Identificación de actualizaciones de hardware o software.
- Evaluación de impacto en la infraestructura y el rendimiento del sistema.
- Recopilación de requisitos técnicos para la adaptación.

##### **Fase 2: Diseño de Ajustes y Configuraciones**

- Desarrollo de modificaciones en el software para compatibilidad con los cambios.
- Ajuste de interfaces y protocolos de comunicación.
- Elaboración de planes de transición para minimizar interrupciones.

##### **Fase 3: Implementación de Adaptaciones**

- Configuración del software para soportar nuevos dispositivos o cambios en la infraestructura.
- Ajustes en bases de datos, parámetros de red y autenticación.
- Pruebas de compatibilidad antes de la implementación en producción.

##### **Fase 4: Validación y Ajustes Finales**

- Evaluación del desempeño tras la implementación.
- Corrección de problemas derivados de la adaptación.
- Documentación de los cambios y actualización de manuales técnicos.

#### **4. Responsables**

Actividad	Responsable	Recursos Necesarios	Tiempo Estimado
<b>Análisis de cambios</b>	Equipo de desarrollo	Herramientas de monitoreo	2 semanas
<b>Diseño de ajustes</b>	Ingenieros de software	Entornos de prueba	3 semanas
<b>Implementación</b>	Equipo de DevOps	Infraestructura actualizada	4 semanas
<b>Validación y ajustes</b>	QA/Testers	Herramientas de prueba	2 semanas

#### **5. Documentación**

- Actualización de manuales de usuario y guías técnicas.
- Registro de cambios en el sistema.
- Capacitación del equipo de soporte sobre las nuevas adaptaciones.

#### **6. Cierre del Mantenimiento**

- Informe final con análisis de impacto.
- Evaluación de resultados y retroalimentación de usuarios.
- Planificación de futuras adaptaciones.

### 1. Información General del Proyecto

Campo	Descripción:
Nombre del Proyecto	
Fecha de Inicio	Fecha prevista para el inicio del proyecto.
Fecha de Finalización	Fecha prevista para la finalización del proyecto.
Responsable Principal	Persona o rol encargado de liderar el proyecto.
Stakeholders	Lista de personas o departamentos involucrados o afectados por el proyecto.

### 2. Identificación del Cambio

Campo	Descripción
Motivo del Cambio	Razón por la cual se requiere el mantenimiento adaptativo (por ejemplo, nueva normativa, actualización de tecnología, etc.).
Impacto Esperado	Descripción del impacto que tendrá el cambio en el software y en los usuarios.
Requisitos Asociados	Lista de requisitos funcionales y no funcionales que deben cumplirse.

### 3. Análisis de Impacto

Campo	Descripción
Módulos Afectados	Lista de módulos o componentes del software que serán modificados.
Riesgos Identificados	Posibles riesgos asociados con el cambio y plan de mitigación.
Estimación de Recursos	Recursos necesarios (humanos, técnicos, financieros).
Estimación de Tiempo	Tiempo estimado para completar el mantenimiento adaptativo.

#### 4. Planificación

Campo	Descripción
Cronograma	Detalle de las fases y fechas clave (por ejemplo, diseño, desarrollo, pruebas, despliegue).
Asignación de Recursos	Lista de personas y roles asignados a cada tarea.
Criterios de Aceptación	Condiciones que deben cumplirse para considerar el proyecto como exitoso.

#### 5. Diseño de Cambios

Campo	Descripción
Descripción Técnica	Detalle de los cambios técnicos que se realizarán (por ejemplo, nuevos campos, integraciones, etc.).
Diagramas o Esquemas	Diagramas de flujo, arquitectura o esquemas que ilustren los cambios.
Aprobación	Firma o aprobación de los stakeholders para proceder con el diseño.

#### 6. Implementación

Campo	Descripción
Entorno de Desarrollo	Descripción del entorno donde se realizarán los cambios.
Código y Configuraciones	Lista de archivos, módulos o configuraciones que serán modificados.
Pruebas Unitarias	Resultados de las pruebas unitarias realizadas durante la implementación.

## 7. Pruebas

Campo	Descripción
Pruebas de Integración	Resultados de las pruebas de integración para asegurar que los cambios no afectan otras partes del sistema.
Pruebas de Regresión	Resultados de las pruebas de regresión para verificar que la funcionalidad existente sigue funcionando.
Validación con Usuarios	Feedback de los usuarios finales sobre los cambios realizados.

## 8. Despliegue

Campo	Descripción
Plan de Despliegue	Detalle del plan para implementar los cambios en producción (fecha, ventana de tiempo, etc.).
Copia de Seguridad	Confirmación de que se realizó una copia de seguridad antes del despliegue.
Monitoreo Inicial	Resultados del monitoreo después del despliegue para detectar problemas.

## 9. Monitoreo y Soporte Post-Implementación

Campo	Descripción
Incidentes Reportados	Lista de incidencias reportadas por los usuarios después del despliegue.
Solución de Problemas	Detalle de las soluciones implementadas para resolver incidencias.
Soporte Continuo	Plan de soporte continuo para los usuarios durante la transición.

## 10. Documentación y Cierre

Campo	Descripción
Documentación Actualizada	Lista de documentos actualizados (manuales, especificaciones técnicas, etc.).
Revisión Final	Revisión y aprobación final del proyecto por parte de los stakeholders.
Lecciones Aprendidas	Resumen de lecciones aprendidas durante el proceso de mantenimiento adaptativo.

## 11. Aprobación y Firma

Campo	Descripción
Rol	Nombre   Firma   Fecha
Responsable del Proyecto	[Nombre]   [Firma]   [Fecha]
Stakeholder Principal	[Nombre]   [Firma]   [Fecha]
Equipo de Desarrollo	[Nombre]   [Firma]   [Fecha]

## EVIDENCIA 12. EJEMPLO DE PLAN DE MANTENIMIENTO PERFECTIVO.

### Información general

- **Propósito del plan:** Mejorar la velocidad de carga y la experiencia del usuario en el sistema de gestión de inventarios "StockControl".
- **Alcance del plan:** Se optimizará la base de datos y la interfaz gráfica para agilizar la consulta de productos y reducir el tiempo de carga en un 30%.

### Identificación de Software

- **Nombre del software:** StockControl
- **Versión actual:** 2.5
- **Encargados de desarrollo:** Equipo de TI - Departamento de Software

### Análisis de necesidades

- **Diagnóstico del software actual:** El sistema presenta lentitud en la consulta de productos debido a una base de datos mal indexada y una interfaz poco optimizada.
- **Requerimientos de optimización:**
  - Indexar la base de datos para mejorar las consultas SQL.
  - Optimizar el código de la interfaz para mejorar tiempos de respuesta.
  - Reducir el uso de imágenes pesadas en el frontend.

### Plan de trabajo

- **Actividades a realizar:**
  1. Analizar estructura de la base de datos y aplicar optimizaciones.
  2. Revisar y refactorizar código de frontend.
  3. Implementar carga diferida (lazy loading) en imágenes.
  4. Realizar pruebas de rendimiento antes y después de los cambios.
- **Recursos de trabajo:** SQL Server, React.js, herramientas de análisis de rendimiento.
- **Responsables de actividades:**
  - Base de datos: Juan Pérez

- Optimización de interfaz: María López
- Pruebas de rendimiento: Carlos Sánchez
- **Tiempo estimado en actividades:** 3 semanas

## Pruebas

- **Estrategia de pruebas:** Se realizarán pruebas de carga y estrés para medir mejoras en el tiempo de respuesta del sistema antes y después de las optimizaciones.

## Documentación

- **Actualizaciones en la documentación:** Manual del sistema y guías de optimización.
- **Capacitación:** Se impartirá una sesión de capacitación para el equipo de soporte técnico sobre las nuevas optimizaciones.

## Instrucciones para completar la plantilla de mantenimiento perfectivo de software

El mantenimiento **perfectivo** se enfoca en mejorar el rendimiento, usabilidad o eficiencia del software sin alterar su funcionalidad principal. Sigue estas instrucciones para completar la plantilla correctamente.

### 1. Información general

- **Propósito del plan:** Describe el objetivo principal del mantenimiento perfectivo. Explica qué mejoras se van a realizar y por qué.
- **Alcance del plan:** Define qué partes del software serán mejoradas. Especifica si el mantenimiento afectará la interfaz de usuario, el rendimiento del sistema, la seguridad, entre otros aspectos.

### 2. Identificación de Software

- **Nombre del software:** Indica el nombre completo del software a mejorar.
- **Versión actual:** Especifica la versión sobre la que se aplicarán las mejoras.
- **Encargados de desarrollo:** Lista los desarrolladores o equipos responsables de la implementación de las mejoras.

### 3. Análisis de necesidades

- **Diagnóstico del software actual:** Describe el estado actual del software y las áreas que requieren mejoras.
- **Requerimientos de optimización:** Indica los cambios o mejoras específicas que se deben implementar.

#### **4. Plan de trabajo**

- **Actividades a realizar:** Detalla cada tarea necesaria para implementar las mejoras.
- **Recursos de trabajo:** Lista las herramientas, tecnologías y recursos necesarios.
- **Responsables de actividades:** Asigna a cada actividad un encargado específico.
- **Tiempo estimado en actividades:** Define el tiempo que tomará cada tarea.

#### **5. Pruebas**

- **Estrategia de pruebas:** Explica cómo se verificará que las mejoras funcionan correctamente, qué pruebas se realizarán y en qué entorno.

#### **6. Documentación**

- **Actualizaciones en la documentación:** Especifica qué documentos deben actualizarse tras la implementación (manuales, especificaciones, etc.).
- **Capacitación:** Indica si el personal necesitará capacitación para utilizar las mejoras implementadas.

Información general

<b>Propósito del plan:</b>	
<b>Alcance del plan:</b>	

Identificación de Software

<b>Nombre del software:</b>	
<b>Versión actual:</b>	

**Encargados de desarrollo:**

Análisis de necesidades

**Diagnóstico del software actual:**

**Requerimientos de optimización:**

Plan de trabajo

**Actividades a realizar:**

**Recursos de trabajo:**

**Responsables de actividades:**

**Tiempo estimado en actividades:**

Pruebas

**Estrategia de pruebas:**

Documentación

**Actualizaciones en la documentación:**

**Capacitación:**

## EVIDENCIA 13. EJEMPLO DE PR (FORMATO)

### INSTRUCCIONES:

Llena los campos correspondientes acorde a las siguientes indicaciones:

1. **Organización:** Nombre de la empresa, institución o equipo responsable del software.
2. **Departamento:** Área o división dentro de la organización que está reportando el problema.
3. **Sección:** Subdivisión del departamento que está involucrada en el problema (si aplica).
4. **Hojas:** Número total de páginas del informe (ejemplo: 1 de 3).
5. **Fecha de publicación:** Día, mes y año en que se genera el informe.
6. **Prioridad del sistema:** Nivel de importancia del problema en el sistema (Ejemplo: Alta, Media, Baja).
7. **Emitido por:** Nombre de la persona que reporta el problema.
8. **Puesto:** Cargo o función dentro de la organización de la persona que emite el informe.
9. **Problemas reportados:** Breve lista o resumen de los principales inconvenientes detectados.
10. **Descripción del problema:** Explicación detallada del problema, incluyendo pasos para reproducirlo, versiones del software afectadas y cualquier otro dato relevante.
11. **Solución propuesta:** Si el usuario tiene una posible solución o recomendación para mitigar el problema, debe incluirse aquí.
12. **Resultados esperados:** Descripción del comportamiento esperado del software si el problema estuviera solucionado.
13. **Lecciones aprendidas:** Información útil derivada del problema, como errores a evitar en el futuro o mejoras en procesos de desarrollo.
14. **Frecuencia de uso:** Indicación de cuán seguido se usa la funcionalidad afectada (Ejemplo: Diario, Semanal, Ocasional, Raro).
15. **Categoría del problema:** Tipo de error reportado (Ejemplo: Errores de código, Problemas de interfaz, Fallos de rendimiento, Seguridad, Integración).

INFORME DE PROBLEMAS		
Organización:		
Departamento:		
Sección:		Hojas:

INFORMACIÓN BÁSICA SOBRE EL PROBLEMA			
Fecha de publicación:		Prioridad del problema:	
Emitido por:		Puesto:	
Problemas reportados:			
Descripción del problema:			
Solución propuesta:			
Resultados esperados:			
Lecciones aprendidas:			
Frecuencia de uso:			
Categoría del problema:			

#### **EVIDENCIA 14. EJEMPLO DE MR (FORMATO)**

##### **INSTRUCCIONES:**

Llena los campos correspondientes acorde a las siguientes indicaciones:

##### **Información General**

- **Folio:** Número único de identificación de la solicitud de modificación.
- **Fecha de recepción:** Día, mes y año en que se recibe la solicitud.
- **Producto:** Nombre del software o sistema afectado por la solicitud.
- **Autor:** Nombre de la persona o equipo que solicita la modificación.

##### **Descripción del Problema y Prioridad**

- **Nombre corto del problema:** Breve título o descripción que resuma el problema.
- **Prioridad usuario:** Nivel de urgencia asignado por el usuario (Ejemplo: **Alta, Media, Baja**).

### **Detalles de la Modificación**

- **Descripción:** Explicación detallada del problema y la modificación requerida. Incluir el impacto y el contexto del cambio solicitado.
- **Número MR:** Identificador único de la solicitud de modificación dentro del sistema.
- **Ingeniero asignado:** Nombre del ingeniero encargado de analizar y ejecutar la modificación.

### **Pruebas y Evaluación**

- **Versión SW:** Versión actual del software en la que se presenta el problema.
- **Nivel de prueba:** Tipo de prueba en la que se identificó el problema o en la que se validará la modificación (Ejemplo: **Unitarias, Integración, Aceptación**).
- **ID prueba:** Identificador de la prueba específica donde se detectó el problema.
- **Categoría:** Clasificación del problema (Ejemplo: **Error de software, Mejora de funcionalidad, Rendimiento, Seguridad**).
- **Prioridad:** Nivel de urgencia determinado por el equipo técnico.

### **Impacto en el Sistema**

- **CSU afectados:** Casos de uso específicos que se ven afectados por el problema.
- **Documentos afectados:** Manuales, especificaciones o cualquier documentación técnica impactada por la modificación.

### **Solución y Análisis**

- **Solución alternativa:** Si existe una manera temporal de mitigar el problema, se debe especificar aquí.
- **Resultado del análisis:** Determina si la solicitud es una **Mejora** o una **Corrección** de error.

- **Ingeniero de Software:** Nombre del ingeniero que realizó el análisis.
- **Fecha:** Fecha en la que se realizó el análisis.
- **Clasificación de la solución:**
  - **Rentable:** Si la solución es viable en términos de costo y esfuerzo.
  - **Cambios de diseño:** Si la modificación requiere ajustes estructurales en el software.
- **Comentarios del ingeniero de Software:** Observaciones y detalles adicionales sobre el análisis.
- **Estimación de recursos:** Cantidad de tiempo, esfuerzo y recursos necesarios para implementar la modificación.

### **Revisión y Aprobación**

- **Administrador de Software:** Persona responsable de revisar y aprobar la solicitud.
- **Fecha:** Día en que se revisa la solicitud.
- **Comentarios del administrador:** Opiniones, restricciones o condiciones para la aprobación de la solicitud.
- **Seguro de Calidad:** Responsable de validar que la modificación cumpla con los estándares de calidad.
- **Fecha:** Fecha de revisión por el equipo de calidad.
- **Comentarios de garantía de Calidad:** Observaciones finales sobre el impacto y cumplimiento de la modificación.

Folio	Fecha recepcion	Producto	Autor
Nombre corto del Problema			Prioridad Usuario
Descripción			

Número MR		Ingeniero asignado				
Versión SW	Nivel de Prueba	ID prueba	Categoría	Prioridad		
CSU afectados		Documentos afectados				
Solución alternativa						
Resultado del análisis			Mejora <ul style="list-style-type: none"> <li><input type="radio"/> Correctiva</li> <li><input type="radio"/> Rentable</li> <li><input type="radio"/> Cambios de Diseño</li> </ul>			
Ingeniero de Software	Fecha					
Comentarios del ingeniero de Software			Estimación recursos			
Administrador de Software	Fecha					
Comentarios del administrador						
Seguro de Calidad	Fecha					
Comentarios de garantía de Calidad						

## **CONCLUSIONES**

El mantenimiento de software es un aspecto fundamental para la sostenibilidad y evolución de los sistemas tecnológicos en diversas industrias. A lo largo del documento, se han presentado estudios de caso que evidencian la importancia de estrategias eficientes de mantenimiento, desde enfoques correctivos y adaptativos hasta modelos predictivos basados en tecnologías de la Industria 4.0. Empresas como Torc Robotics han demostrado que la implementación de sistemas de gestión del mantenimiento informatizados (GMAO) permite una mayor optimización de recursos, reducción de tiempos de inactividad y estandarización de procesos.

Asimismo, en entornos académicos y hospitalarios, la correcta gestión del mantenimiento mejora la trazabilidad, calidad y funcionalidad de los sistemas informáticos, impactando positivamente en la operatividad y satisfacción de los usuarios. En el caso del kernel de Linux, la comunidad ha logrado mantener un ecosistema robusto mediante un modelo colaborativo de desarrollo y mantenimiento, evidenciando que la gestión adecuada del software es clave para su escalabilidad y seguridad.

El mantenimiento preventivo y correctivo, aplicado en empresas y organizaciones con altos niveles de criticidad, ha demostrado su relevancia para garantizar la continuidad operativa y mitigar riesgos de fallas imprevistas. La adopción de estándares internacionales como ISO/IEC 14764 ha permitido a diversas entidades estructurar procesos más eficientes, alineados con las mejores prácticas del sector.

En conclusión, la evolución del mantenimiento de software ha pasado de ser una actividad reactiva a una estrategia proactiva e integral. La combinación de metodologías estructuradas, herramientas digitales avanzadas y una cultura organizacional orientada a la mejora continua es esencial para garantizar sistemas tecnológicos confiables, seguros y alineados con los objetivos de negocio.

## REFERENCIAS BIBLIOGRÁFICAS

- ISO/IEC 14764. (2006). *Software Engineering—Software Life Cycle Processes—Maintenance*. International Organization for Standardization.
- IEEE 1219. (1998). *Standard for Software Maintenance*. Institute of Electrical and Electronics Engineers.
- Sommerville, I. (2020). *Software Engineering* (10th ed.). Pearson Education.
- Pressman, R. S., & Maxim, B. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
- Van Wingen, S., & Lenstra, J. (2022). *Modern Software Maintenance: Best Practices for Agile and DevOps Teams*. Springer.