



PRESENTA

JESÚS MARGARITO SANTOS GARCÍA

DOCENTE

NÉSTOR APOLO LÓPEZ GONZÁLEZ

TRABAJO

PORTAFOLIO DE EVIDENCIAS

GRUPO

2825IS

MATERIA

MANTENIMIENTO DE SOFTWARE

Fecha

FEBRERO 2025

INDICE

UNIVERSIDAD POLITÉCNICA DE TECÁMAC.....	1
INGENIERÍA EN SOFTWARE	1
INTRODUCCIÓN	3
Evidencia 1. Presentación electrónica sobre introducción al 14764.....	5
Evidencia 2. Reporte de los 8 casos de estudio.....	12
Evidencia 3.....	34
Evidencia 4.....	35
Evidencia 5.....	36
Evidencia 6.....	37
Evidencia 7 y 8.....	38
Evidencia 9.....	39
EVIDENCIA 10 EJEMPLO DE UN PLAN DE MANTENIMIENTO PREVENTIVO DE SOFTWARE	40
EVIDENCIA 11 EJEMPLO DE PLAN DE MANTENIMIENTO ADAPTATIVO DE SOFTWARE	44
EVIDENCIA 12 EJEMPLO PLAN DE MANTENIMIENTO PERFECTIVO	49
EVIDENCIA 13 EJEMPLO DE FORMATO PR.....	52
EVIDENCIA 14 EJEMPLO DE FORMATO MR.....	53
CONCLUSIÓN	55
REFERENCIAS.....	56

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 sobre introducción al 14764



Estándar de Mantenimiento

Descripción general: Este apartado proporciona una guía sobre el mantenimiento de software según el estándar ISO/IEC/IEEE 12207:2017, explicando los procesos, actividades y tipos de mantenimiento, excluyendo operaciones como copias de seguridad y recuperación.

Propósito: Define el propósito de guiar el proceso de mantenimiento, destacando la importancia de la mantenibilidad, los modelos de servicio y las estrategias de mantenimiento.

Campo de aplicación: Se enfoca en la planificación y mantenimiento de software, aplicable tanto a organizaciones internas como externas, y abarca diversos modelos de ciclo de vida del software.

Limitaciones: Describe el marco del proceso de mantenimiento sin especificar detalles de implementación, proporcionando listas ejemplares, pero no exhaustivas.

Mantenimiento adaptativo

Modificación de un producto de software realizada después de la entrega para mantener el producto utilizable en un entorno cambiado o en cambio.

Mantenimiento aditivo

Modificación realizada después de la entrega para agregar funcionalidad o características al producto.

Corrección

Cambios para abordar e implementar resoluciones de problemas, cerrando brechas para cumplir con los requisitos operativos definidos.

Mantenimiento correctivo

Modificación para reparar problemas descubiertos después de la entrega.

Mantenimiento de emergencia

Modificación no planificada para mantener temporalmente un sistema operativo hasta que se realice mantenimiento correctivo.

Mantenibilidad

Grado en que un producto puede ser modificado de manera efectiva y eficiente.

Mejora

Cambio que implementa un nuevo requisito, como mantenimiento adaptativo, perfectivo o aditivo.

Solicitud de modificación (MR)

Elemento informativo que identifica y describe cambios propuestos a un producto o servicio.

Mantenimiento perfectivo

Modificación para proporcionar mejoras en el rendimiento, mantenibilidad u otros atributos del software.

Mantenimiento preventivo

Modificación para corregir fallos latentes antes de que ocurran en el sistema en vivo.

Informe de problemas (PR)

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

Mantenimiento del software

Totalidad de actividades requeridas para dar soporte a un sistema de software.

Sostenimiento del software

Actividades para soportar, mantener y operar un sistema de software para garantizar que permanezca operativo.

Transición

Actividades involucradas en mover un servicio, sistema o componente nuevo o cambiado a un entorno.



Este documento ofrece orientación específica para la implementación del proceso de mantenimiento establecido en la norma **ISO/IEC/IEEE 12207**. La información contenida en el texto está diseñada para ser totalmente coherente con dicha norma, asegurando que todos los lineamientos y procesos sean compatibles y aplicables.

Eliminación de software

Hace referencia al proceso de desinstalar, retirar o desechar software de un sistema informático. Esto puede incluir la eliminación de programas obsoletos, no utilizados o que ya no cumplen con los requisitos de un usuario o una organización. También puede implicar asegurarse de que los datos asociados al software se borren de forma segura para proteger la privacidad o la seguridad de la información.



Puntos importantes

01. General **(Generalidades)**

Estrategia de eliminación
Este capítulo presenta un enfoque estructurado para gestionar la eliminación de software o sistemas, incluyendo tareas relacionadas, notificaciones y archivo de registros.

02. Disposal strategy **items (Elementos de la** **estrategia de eliminación)**

Detalla los componentes específicos que deben incluirse en la estrategia, como:

- Identificación del software o sistema a eliminar.
- Evaluación de riesgos asociados con la eliminación.
- Planificación de recursos necesarios para el proceso.

03. Disposal tasks **(Tareas de eliminación)**

Lista las tareas específicas que se deben realizar durante el proceso de eliminación, como:

- Desinstalación del software.
- Eliminación segura de datos asociados.
- Evaluación del cumplimiento normativo.

04. Disposal notification **(Notificación de eliminación)**

Se refiere al proceso de informar a las partes interesadas sobre la eliminación.

Incluye:

- Comunicación formal de las actividades de eliminación.
- Confirmación de que el software ya no está en uso.

05. Disposal notification **tasks (Tareas de notificación** **de eliminación)**

Describe las acciones específicas relacionadas con la notificación, como:

- Preparar informes detallados de eliminación.
- Asegurarse de que las partes relevantes (internas o externas) reciban la información.
- Registrar las notificaciones enviadas y recibidas.

06. Post-disposal tasks **(Tareas posteriores a la** **eliminación)**

Aborda las actividades necesarias después de completar la eliminación, como:

- Verificar que todos los datos relacionados se hayan eliminado por completo.
- Realizar auditorías para confirmar que se cumplieron los procedimientos.
- Liberar recursos asignados al software eliminado.

07. Archiving records **(Archivado de** **registros)**

Explica la importancia de conservar registros relacionados con el proceso de eliminación, como:

- Documentación del cumplimiento normativo.
- Registros de auditoría y notificaciones.
- Información archivada para futuras referencias o revisiones legales.

Consideraciones para la implementación

Se refiere a los factores, aspectos o elementos que deben tenerse en cuenta al llevar a cabo la implementación de un proyecto, sistema, proceso o solución. Estas consideraciones aseguran que la implementación sea efectiva, eficiente y cumpla con los objetivos establecidos.



Puntos importantes



El mantenimiento de software es esencial para garantizar su funcionalidad, eficiencia y alineación con las necesidades del usuario. Se clasifica en cuatro tipos principales: correctivo (solución de errores), adaptativo (ajustes al entorno), perfectivo (mejoras y nuevas funcionalidades) y preventivo (evitar problemas futuros).

Para su ejecución, se requieren acuerdos claros, como contratos de nivel de servicio (SLAs), y el uso de herramientas como sistemas de gestión de cambios y plataformas de seguimiento de errores. Además, la medición del mantenimiento mediante métricas como el tiempo de resolución y la frecuencia de solicitudes asegura su eficacia.

La participación temprana del equipo de mantenimiento en el desarrollo es clave para garantizar la mantenibilidad desde el diseño. También es crucial aplicar procesos del ciclo de vida enfocados en facilitar el mantenimiento y mantener registros detallados, como historial de cambios y documentación de incidentes, para una gestión efectiva y mejoras continuas.

Aplicación de este documento

El documento define el proceso de mantenimiento de software según el estándar ISO/IEC/IEEE 12207. Describe cómo se relaciona con otros procesos técnicos y administrativos, detalla las actividades necesarias, aborda



Proceso de mantenimiento

El proceso de mantenimiento de software asegura que el sistema continúe ofreciendo servicios confiables mediante correcciones, mejoras y actualizaciones. Se ocupa de defectos, obsolescencia y evolución del sistema, integrándose con procesos como gestión de configuración. Los resultados incluyen identificación de restricciones, disponibilidad de recursos, implementación de cambios y evaluación de costos y fallas.



General

Descripción general: Introducción al plan de mantenimiento, proporcionando un marco general para el proceso.

Identificación y control del plan: Definición de los métodos para identificar y controlar el plan de mantenimiento.

Alcance del mantenimiento: Explicación de los límites y alcance del mantenimiento de software.

Designación de la organización de mantenimiento: Asignación de responsabilidades a las organizaciones encargadas del mantenimiento.

Referencias: Listado de documentos y estándares relacionados que sirven como base para el plan.

Definiciones: Clarificación de términos clave utilizados en el contexto del mantenimiento.

Procesos: Descripción de los procesos específicos que se llevarán a cabo durante el mantenimiento.

Organización y actividades de mantenimiento: Detalle de las actividades de mantenimiento y cómo se organizan.

Recursos: Identificación de los recursos necesarios para llevar a cabo el mantenimiento.

Estimación de costos de mantenimiento: Cálculo y proyección de los costos asociados al mantenimiento.

Capacitación: Formación necesaria para el personal involucrado en el mantenimiento.

Requisitos de control del mantenimiento de software: Normas y controles necesarios para gestionar el proceso de mantenimiento.

Registros e informes de mantenimiento: Documentación y reportes que deben mantenerse para el seguimiento del mantenimiento.

Análisis de recursos

General: Introducción al análisis de recursos necesarios para el mantenimiento.

Recursos de personal: Evaluación de los recursos humanos necesarios para realizar el mantenimiento.

Recursos del entorno: Identificación de los recursos ambientales necesarios, como infraestructura y herramientas.

Recursos financieros: Estimación de los recursos financieros necesarios para sostener las actividades de mantenimiento.

Evidencia 2. Reporte de los 8 casos de estudio

Caso de estudio 1

El Hospital Escuela San Juan de Dios, inaugurado en 1997, es un centro médico clave para la región. En la UCIN, la gestión del mantenimiento de equipos biomédicos es crítica para garantizar la continuidad de los servicios de salud. Previo a este proyecto, las actividades de mantenimiento se gestionaban manualmente mediante hojas de vida y órdenes de trabajo. Este sistema resultaba ineficiente y propenso a errores, poniendo en riesgo la disponibilidad de los equipos y, en consecuencia, la vida de los pacientes.

En este contexto, se identificó la necesidad de un sistema automatizado que optimizara la planificación, programación y control de las actividades de mantenimiento preventivo y correctivo. Este sistema también debía ser capaz de almacenar y organizar información relevante sobre los equipos, como inventarios y registros históricos.

Problemática

Entre los principales problemas detectados se encontraban:

1. **Gestiones manuales ineficientes:** La utilización de formatos impresos generaba una acumulación de información dispersa y de difícil acceso.
2. **Falta de accesibilidad:** Un 62% del personal reportó dificultades para acceder a la información.
3. **Capacitación insuficiente del personal:** La carencia de conocimientos sobre la gestión moderna dificultaba la implementación de mejoras.
4. **Falta de sistematización:** Los procesos no seguían una estructura clara ni se alineaban con un plan programado de mantenimiento.

Soluciones Implementadas

La propuesta incluyó el diseño de un software de gestión de mantenimiento asistido por ordenador (GMAO). Este sistema cumple con funciones clave como la planificación de actividades, registro de inventarios, generación de reportes y control de órdenes de trabajo. El desarrollo se llevó a cabo en Visual Studio 2010 y utilizó MySQL como gestor de bases de datos. Además, se implementó una interfaz intuitiva para facilitar su adopción por parte del personal.

Etapas del Proyecto

1. Levantamiento de Inventario: Se registraron todos los equipos existentes en la UCIN, recopilando información técnica relevante.
2. Entrevistas y Encuestas: Se recabaron opiniones y sugerencias del personal para adaptar el software a sus necesidades.
3. Diseño del Sistema: Se estructuraron las actividades de mantenimiento y se desarrolló una propuesta funcional del sistema.
4. Pruebas y Validación: El sistema fue probado exhaustivamente para garantizar su funcionalidad y facilidad de uso.

Personas Involucradas

- Desarrolladores: Escarlet Sarahy Arvizú, Isamara Francisca Bravo y Dania Lisseth Alaniz, quienes lideraron el diseño y desarrollo del software.

- Personal del Hospital: Incluyendo el jefe del departamento de mantenimiento y los técnicos de la UCIN, quienes proporcionaron información crítica y participaron en la validación del sistema.
- Usuarios Finales: Enfermeros y otros trabajadores del hospital que interactúan directamente con el software y los equipos biomédicos.

Resultados y Beneficios

El software GMAO permitió una mejora significativa en la gestión de mantenimiento al automatizar tareas, reducir el tiempo de acceso a la información y facilitar la generación de reportes. Además, su implementación incentivó un cambio cultural hacia la sistematización y mejora continua dentro del área de mantenimiento.

En conclusión, este proyecto demostró cómo la aplicación de tecnologías modernas puede transformar procesos clave en el ámbito de la salud, beneficiando tanto a los pacientes como al personal técnico.

Caso de estudio 2

Sistema para la Tramitación de Interrupciones en los Sistemas de Informática y Comunicaciones (TI)

El sistema de información TI se desarrolló para gestionar interrupciones en los sistemas de informática y comunicaciones de la Zona Oriente Norte de ECASA, en Cuba. Este sistema es crucial para garantizar la calidad de los servicios aeronáuticos, donde la comunicación es fundamental. El mantenimiento del sistema fue necesario para abordar fallos, adaptarlo a nuevos entornos y agregar funcionalidades requeridas por los usuarios. Este proceso se alineó con la normativa ISO 9001:2000 y utilizó software libre como tecnología base.

Problemáticas Identificadas

1. Carencias Funcionales:

Ausencia de métodos de seguridad como respaldos de la base de datos.

Falta de un sistema de alertas que notificara al personal sobre nuevos reportes.

Limitaciones en los informes, restringidos al formato HTML tradicional.

No existía integración con otros sistemas como el Sistema Internacional de Mensajería Aeronáutica (SIMA).

2. Deficiencias Técnicas:

Mala manipulación de información en la base de datos.

Problemas en la integridad referencial y errores en la inserción de datos.

Falta de alineación con el Procedimiento Específico PE280602.

3. Costos de Mantenimiento:

Elevado costo del mantenimiento en comparación con el desarrollo inicial del software.

La complejidad en la comprensión del código aumentaba los costos.

Soluciones Implementadas

El proceso de mantenimiento se dividió en varias categorías:

1. Mantenimiento Correctivo:

Corrección de errores en la manipulación de la base de datos y en la integridad referencial.

Ajustes para cumplir con el Procedimiento Específico PE280602.

2. Mantenimiento Adaptativo:

Incorporación de la técnica AJAX, mejorando la velocidad y usabilidad del sistema.

Adaptación de funcionalidades para incluir nuevas entidades como equipos, departamentos y observaciones.

3. Mantenimiento Perfectivo:

Agregó 23 nuevos requisitos funcionales, como menús dinámicos y filtros en la información resultante.

Mejora del diseño visual mediante nuevos estilos.

4. Mantenimiento Preventivo:

Comentarios en el código fuente y adopción de estándares de codificación.

Creación de un sistema de ayuda para guiar a los usuarios en el uso del software.

Tecnologías Utilizadas

Lenguaje de Programación: PHP, un lenguaje interpretado ampliamente usado en el desarrollo web.

Base de Datos: PostgreSQL, reconocido por su robustez y características avanzadas.

Metodología de Desarrollo: Rational Unified Process (RUP), basada en iteraciones y uso de UML.

Técnica AJAX: Para mejorar la experiencia del usuario mediante peticiones asincrónicas.

Personas Involucradas

Líder del Proyecto: Ing. Oscar Gabriel Reyes Pupo, responsable del diseño y ejecución del mantenimiento.

Equipo Técnico: Desarrolladores que implementaron y probaron las nuevas funcionalidades.

Usuarios Finales: Personal de ECASA que proporcionó retroalimentación clave para adaptar el sistema a sus necesidades.

Resultados y Beneficios

El proceso de mantenimiento resultó en una nueva versión del sistema TI que:

Aumentó la seguridad y confiabilidad del sistema.

Redujo los tiempos de respuesta y errores en la manipulación de datos.

Mejoró la usabilidad gracias a una interfaz más intuitiva.

Cumplió con el Procedimiento Específico PE280602.

Incluyó funcionalidades como un sistema de alertas, que redujo el tiempo de tramitación de interrupciones.

Caso de estudio 3

Caso de Estudio: Mantenimiento del Kernel de Linux

El kernel de Linux es uno de los proyectos de software más exitosos en términos de mantenimiento, escalabilidad y longevidad. Desde su creación en 1991 por Linus Torvalds, el kernel ha evolucionado para ser el núcleo de miles de sistemas operativos y dispositivos, desde supercomputadoras hasta teléfonos inteligentes. Este caso de estudio explora cómo se ha gestionado el mantenimiento del kernel de Linux a lo largo de más de tres décadas.

1. Contexto Inicial

Linux comenzó como un proyecto personal de Linus Torvalds para crear un núcleo similar a Unix. Con el tiempo, atrajo la atención de miles de programadores interesados en contribuir al desarrollo de un sistema operativo libre y de código abierto. Sin embargo, mantener un proyecto de esta magnitud presentó varios desafíos desde el principio:

- **Creación de una base de código flexible:** El código del kernel debía ser modular para permitir la integración de nuevas funcionalidades sin comprometer la estabilidad.
- **Gestión de contribuciones:** El modelo abierto requería un sistema robusto para gestionar los aportes de miles de desarrolladores.

Torvalds adoptó el sistema de control de versiones Git, lo que permitió una colaboración efectiva a gran escala, facilitando la revisión y el mantenimiento del código.

2. Desafíos en el Mantenimiento

El mantenimiento del kernel de Linux ha implicado abordar los siguientes retos:

1. Adaptación tecnológica:

El kernel debe mantenerse relevante frente a los avances en hardware y nuevas arquitecturas. Por ejemplo, soportar procesadores ARM para dispositivos móviles fue un cambio crucial que requirió una reestructuración significativa.

2. Gestión de bugs y vulnerabilidades:

Dado su uso masivo, el kernel está continuamente expuesto a problemas de seguridad. Los parches deben desarrollarse y aplicarse rápidamente para minimizar el impacto de las vulnerabilidades.

3. Escalabilidad:

A medida que el kernel crecía, surgió la necesidad de garantizar que nuevas características no afectaran el rendimiento general.

4. Colaboración y conflicto:

Con miles de desarrolladores contribuyendo, coordinar y revisar los cambios sin introducir errores ha sido un desafío constante.

3. Soluciones Implementadas

A. Organización Jerárquica

El desarrollo y mantenimiento del kernel están organizados en una jerarquía de mantenedores:

- **Linus Torvalds:** Tiene la última palabra sobre los cambios integrados en el kernel principal.
- **Mantenedores de subsistemas:** Supervisan áreas específicas como controladores de dispositivos, sistemas de archivos, y redes.
- **Contribuidores:** Realizan cambios que son revisados antes de su integración.

Esta estructura asegura que las modificaciones pasen por varias capas de revisión antes de ser aceptadas.

B. Control de Versiones con Git

En 2005, Torvalds creó Git, un sistema de control de versiones diseñado específicamente para manejar la magnitud del desarrollo del kernel. Git permite:

- Seguimiento detallado de cambios.
- Gestión eficiente de ramas para probar nuevas funcionalidades.
- Resolución de conflictos entre contribuciones simultáneas.

C. Liberaciones Estables y LTS (Long-Term Support)

El kernel sigue un modelo de lanzamientos frecuentes:

- Versiones principales lanzadas cada 8-10 semanas.
- Versiones LTS (soporte a largo plazo) que reciben mantenimiento durante varios años, ideales para sistemas que necesitan estabilidad.

D. Comunicación y Transparencia

El uso de listas de correo públicas (como la Linux Kernel Mailing List) permite

que la discusión y revisión de cambios sea accesible para todos los interesados. Esto fomenta la transparencia y la colaboración.

4. Resultados del Mantenimiento

1. Crecimiento Sostenido

El kernel cuenta con millones de líneas de código, manteniendo su estabilidad y funcionalidad. Ha soportado diversas arquitecturas como x86, ARM, y RISC-V, siendo el núcleo de distribuciones como Ubuntu, Android y Red Hat Enterprise Linux.

2. Resolución Rápida de Vulnerabilidades

En 2018, se descubrieron vulnerabilidades críticas como Spectre y Meltdown, que afectaban tanto al software como al hardware. La comunidad de Linux respondió rápidamente con parches que mitigaron los riesgos en días.

3. Adaptación Continua

Linux ha integrado soporte para tecnologías emergentes como contenedores (Docker), sistemas de archivos avanzados (Btrfs), y computación en la nube.

Caso de estudio 4

Caso de Éxito en la Implementación del Mantenimiento Predictivo

Mediante el Uso de Tecnologías de la Industria 4.0

El uso de nuevas tecnologías ha permitido a las empresas mejorar la eficiencia y reducir costos operativos. Un caso de éxito es la implementación del mantenimiento predictivo en sistemas de refrigeración industrial, utilizando sensores IoT y análisis de datos en tiempo real.

La empresa "FríoTech" adoptó un sistema basado en monitoreo de temperatura, presión y vibración en sus equipos críticos. Gracias a la inteligencia artificial y la automatización, lograron reducir en un 40% las fallas imprevistas y optimizar el consumo energético.

Antes, los módulos de refrigeración costaban 1,895,000 COP kWh/50. Ahora, estos módulos cuestan 1,320,000 COP kWh/50.

Implementaron un programa de automatización (IoT) que hace uso de reportes para minimizar el consumo.

Recabar y analizar datos ambientales que están directamente relacionados con el desempeño de los módulos de refrigeración permitió predecir fallas y establecer un mantenimiento proactivo en lugar de reactivo.

Además, se incorporó un sistema basado en Python que analiza datos históricos para detectar patrones de desgaste y prever fallos en los equipos. Esto ha tenido un impacto positivo en la productividad y ha reducido costos de mantenimiento.

Desarrollaron un sistema de control para el monitoreo y mantenimiento de los módulos de refrigeración, permitiendo analizar variables en tiempo real y generar alertas automáticas en caso de anomalías.

Gracias a estas iniciativas, lograron un ahorro significativo en el consumo energético y optimizaron los procesos de mantenimiento en la empresa.

Caso de estudio 5

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

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

Efecto

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

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 6

Caso de estudio: Sistema de Gestión de Inventarios para una Tienda de Ropa

Descripción del sistema

La tienda de ropa "Moda y Estilo" tiene un sistema de gestión de inventarios que se utiliza para registrar y controlar las existencias de productos en la tienda. El sistema se compone de una base de datos que almacena información sobre los productos, como descripción, precio, cantidad en stock, etc.

Problemas

El sistema no se ha actualizado en varios años y utiliza tecnologías obsoletas.

La base de datos es lenta y no puede manejar el volumen de datos que se genera diariamente. El sistema no tiene una interfaz de usuario intuitiva y es difícil de usar para los empleados. No hay un proceso de respaldo de datos establecido, lo que significa que si se produce un fallo en el sistema, se pueden perder datos importantes.

Análisis de lo que no funcionó

¿Qué no funcionó?

El sistema de gestión de inventarios no funcionó correctamente debido a una combinación de factores, incluyendo:

- Tecnologías obsoletas: El sistema se desarrolló utilizando tecnologías que ya no son compatibles con los estándares actuales.

- Falta de mantenimiento: El sistema no se ha actualizado ni mantenido adecuadamente durante varios años, lo que ha llevado a una acumulación de errores y problemas.
- Interfaz de usuario no intuitiva: La interfaz de usuario del sistema es difícil de usar y no proporciona una experiencia de usuario adecuada para los empleados.

¿Por qué no funcionó?

El sistema no funcionó correctamente debido a una falta de planificación y mantenimiento adecuados. La empresa no invirtió suficientes recursos en el desarrollo y mantenimiento del sistema, lo que llevó a una acumulación de errores y problemas.

¿Si funcionó en algún momento?

Sí, el sistema funcionó correctamente durante los primeros años después de su implementación. Sin embargo, a medida que la empresa creció y el volumen de datos aumentó, el sistema comenzó a mostrar signos de debilidad y eventualmente dejó de funcionar correctamente.

¿Cuáles fueron los errores?

Los errores que se produjeron en el sistema incluyeron:

- Errores de conexión a la base de datos
- Problemas de rendimiento y velocidad
- Errores de visualización de datos
- Fallos en la seguridad y autenticación de usuarios

¿Cómo mejoraría?

Para mejorar el sistema, se podrían implementar las siguientes soluciones:

- Actualizar el sistema a tecnologías más recientes y eficientes
- Mejorar la interfaz de usuario para proporcionar una experiencia de usuario más intuitiva y fácil de usar
- Implementar un proceso de respaldo de datos regular para garantizar la integridad de los datos
- Realizar pruebas y depuración del sistema para identificar y corregir errores

¿Qué se puede hacer para evitar que vuelva a suceder?

Para evitar que el sistema vuelva a fallar, se podrían implementar las siguientes medidas:

- Establecer un plan de mantenimiento regular para el sistema
- Asignar recursos suficientes para el desarrollo y mantenimiento del sistema
- Realizar pruebas y depuración del sistema de manera regular
- Implementar un proceso de respaldo de datos regular para garantizar la integridad de los datos.

Caso de estudio 7

Caso de Estudio: MCAS del Boeing 737 MAX

El rediseño del Boeing 737 MAX con motores más grandes y desplazados hacia adelante causaba que la nariz del avión se levantara en ciertas condiciones, afectando la estabilidad. Era necesario diseñar un sistema automatizado llamado MCAS para corregir esta tendencia y garantizar la estabilidad sin requerir modificaciones significativas en la estructura del avión o entrenamientos costosos para los pilotos.

El MCAS presentaba fallas como la dependencia de un único sensor de ángulo de ataque (AoA), pruebas insuficientes que no consideraron escenarios de fallas del sensor, falta de redundancia y detección de errores en el sistema, además de documentación deficiente y falta de entrenamiento adecuado para los pilotos. Estas fallas resultaron en dos accidentes fatales en 2018 y 2019, causando la muerte de 346 personas. Se suspendió el modelo 737 MAX a nivel mundial durante más de un año, dañando la reputación de Boeing, que además enfrentó sanciones regulatorias y costos financieros significativos.

Se actualizó el MCAS para depender de múltiples sensores en lugar de uno solo. Se mejoraron las pruebas y simulaciones para contemplar escenarios extremos y fallas múltiples. Se revisó y mejoró la documentación técnica para pilotos y operadores. Se estableció una supervisión más estricta por parte de los organismos reguladores como la FAA.

De forma positiva, el Boeing 737 MAX fue recertificado en 2020 tras modificaciones, pruebas exhaustivas y la aprobación de reguladores globales. Los nuevos sistemas implementaron redundancia y controles adicionales para

garantizar la seguridad. Sin embargo, de forma negativa, la confianza en Boeing y la industria aeronáutica se vio severamente afectada, y la empresa enfrentó multas significativas, demandas y pérdida de clientes debido a la percepción de negligencia

Caso de estudio 8

El problema identificado en la investigación radicó en la ausencia de un procedimiento formal y estandarizado para el mantenimiento de software en pequeñas organizaciones, específicamente aplicado al sistema de gestión GSP utilizado en el Centro de Referencia para la Educación Avanzada (CREA) en Cuba. Este sistema es esencial para la gestión de actividades de superación pedagógica, pero su funcionamiento presentaba limitaciones debido a deficiencias en el mantenimiento, lo que afectaba su capacidad de adaptación a nuevos requisitos y su rendimiento general. Entre las principales dificultades se encontró la falta de documentación adecuada y una planificación insuficiente en las actividades de mantenimiento, lo que resultó en un manejo improvisado y poco eficiente de las modificaciones necesarias para mantener el sistema actualizado y funcional.

El problema comenzó a manifestarse con mayor claridad conforme las demandas organizacionales evolucionaban y el sistema necesitaba adaptarse a nuevos procesos, como la creación de proyectos productivos, la vinculación de temas de investigación y la gestión de grandes volúmenes de información. Las deficiencias del sistema generan una creciente necesidad de ajustes que, sin una metodología definida, no logran satisfacer las expectativas de los usuarios ni garantizar la calidad del software. Este escenario reflejaba un desafío significativo en términos de mantenibilidad y sostenibilidad del sistema, lo cual comprometía su vida útil y su capacidad de satisfacer las necesidades organizacionales de manera efectiva.

Como solución, se elaboró y aplicó un procedimiento específico para el mantenimiento de software, diseñado para pequeñas organizaciones, pero

basado en estándares internacionales como ISO/IEC 14764 e IEEE 1219. Este procedimiento se enfocó en proporcionar una guía clara y práctica para realizar actividades de mantenimiento que incluyen desde la evaluación inicial del estado del software hasta la implementación y revisión de las modificaciones realizadas. Durante la implementación, se analizaron los problemas existentes en el sistema, se definieron las modificaciones necesarias y se desarrollaron herramientas para facilitar la trazabilidad y documentación de las actividades. Además, se integraron prácticas para evaluar la calidad del software antes y después de aplicar las modificaciones, asegurando que los resultados cumplieron con los estándares esperados.

Los resultados de esta propuesta fueron significativos. El sistema GSP experimentó mejoras notables en su calidad y mantenibilidad, permitiendo extender su vida útil y adaptarse mejor a los cambios organizacionales. Las herramientas introducidas facilitaron el registro y seguimiento de los cambios, lo que contribuyó a una mayor trazabilidad y transparencia en el proceso de mantenimiento. Asimismo, los usuarios del sistema reportaron una mejora en la funcionalidad y adaptabilidad del software, incrementando su satisfacción y confianza en el uso del sistema. Este enfoque no solo resolvió los problemas existentes, sino que también desarrolló una base sólida para futuros trabajos de mantenimiento, garantizando que el sistema pueda evolucionar de manera sostenible en respuesta a las necesidades cambiantes de la organización.

Evidencia 3

6.4.13.1

Propósito

- Mantener la capacidad del sistema para proporcionar sus servicios
- Realizar acciones correctivas, adaptativas, perfectivas y preventivas
- Integrar mantenimiento con gestión de configuración y gestión de activos de software

Resultados

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

Actividades

→ Preparar el mantenimiento

- Definir estrategia de mantenimiento:
 - Priorizar y programar cambios
 - Mantener necesidades de mantenimiento.
- Desarrollar la estrategia logística
 - Considerar almacenamiento, tasas de reemplazo y recuperación
 - Identificar restricciones del mantenimiento

Sistema habilitado:

b) Realizar el mantenimiento

Asignar identificar necesidades y evaluar cambios
Restaurar operación y aplicar correcciones
Mantenimiento preventivo y adaptativo

c) Soporte logístico

Asignar recursos y calidad de reemplazos
Gestionar distribución y calidad de reemplazos
Gestionar verificar cumplimiento de soporte

Evidencia 4

Correctivo \rightarrow Corregir bugs en código fuente

Adaptativo \rightarrow Actualizar la bd para ser compatible con nuevos formatos

Perfeccionista \rightarrow Mejorar la eficiencia del código

Preventivo \rightarrow Puebas unitarias para encontrar errores

6-4-14

Terminación del servicio cuando ya no es requerido
Este proceso elimina los archivos redundantes y los asigna a una condición aceptable

1. Requisitos de eliminación integrados en los elementos de diseño, arquitectura e implementación

2. Disponibilidad de sistemas o servicios habilitados para la eliminación

3. Elementos del sistema o archivos redundantes destruidos, almacenados, reducidos o recuperados según los requisitos de seguridad y protección

4. Restauración del entorno a su estado original o aceptado

5. Registro de las acciones y análisis de eliminación disponibles

Preparación para la eliminación

1. Definir una estrategia que contemple terminación permanente de funciones

Identificar restricciones relacionadas con los requisitos, diseño y técnicas de implementación

Planificar sistemas habilitados necesarios para soportar la eliminación
Continuar la actualización y tratamiento de riesgos para la seguridad

Evidencia 5

Elabora un glosario de los siguientes terminos:

Proceso iterativo Enfoque en el que las actividades se repiten en ciclos sucesivos, permitiendo realizar ajustes y mejoras.

Modelo Marco Estructurado que guía el mantenimiento mediante un conjunto de principios, prácticas y procesos.

Ejecución Implementación de tareas específicas dentro del proceso de mantenimiento.

Fase de planeación Etapa que define los objetivos, recursos, cronograma y estrategias para llevar a cabo el mantenimiento.

Evaluación Analizar y medir el estado del software para determinar su calidad, rendimiento, eficacia y cumplimiento de req. establecido.

Herramientas Programas o plataformas que facilitan las tareas de mantenimiento del software.

Técnicas Conjunto de procedimientos o enfoques específicos utilizados para realizar actividades de manera eficiente.

Métodos Estrategias que guían el proceso de mantenimiento, como el correctivo.

Recuperación Acciones orientadas a restaurar el funcionamiento normal del software tras una falla, error o incidente.

Evidencia 6

- Adaptativo

- La modificación de un producto de software, realizada después de su entrega, manteniéndolo utilizable en un entorno modificado o en constante cambio

- Correctivo

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

- Emergencia

- Modificación no programada

Mantenibilidad

Capacidad del producto para ser modificado

- Adaptativo - Perfectivo

MR

Se utiliza para formalizar la propuesta de un cambio en el software. Permite un proceso estructurado para evaluar, aprobar y ejecutar cambios necesarios.

Mantenimiento Prolongado Preventivo

La modificación de un SW después de su entrega para detectar y corregir fallos latentes en el SW.

PR: Término utilizado para identificar y corregir problemas detectados en un SW.

Mantenimiento del SW

Necesario para proporcionar soporte rentable a un sistema de SW.

Act. previos a la entrega

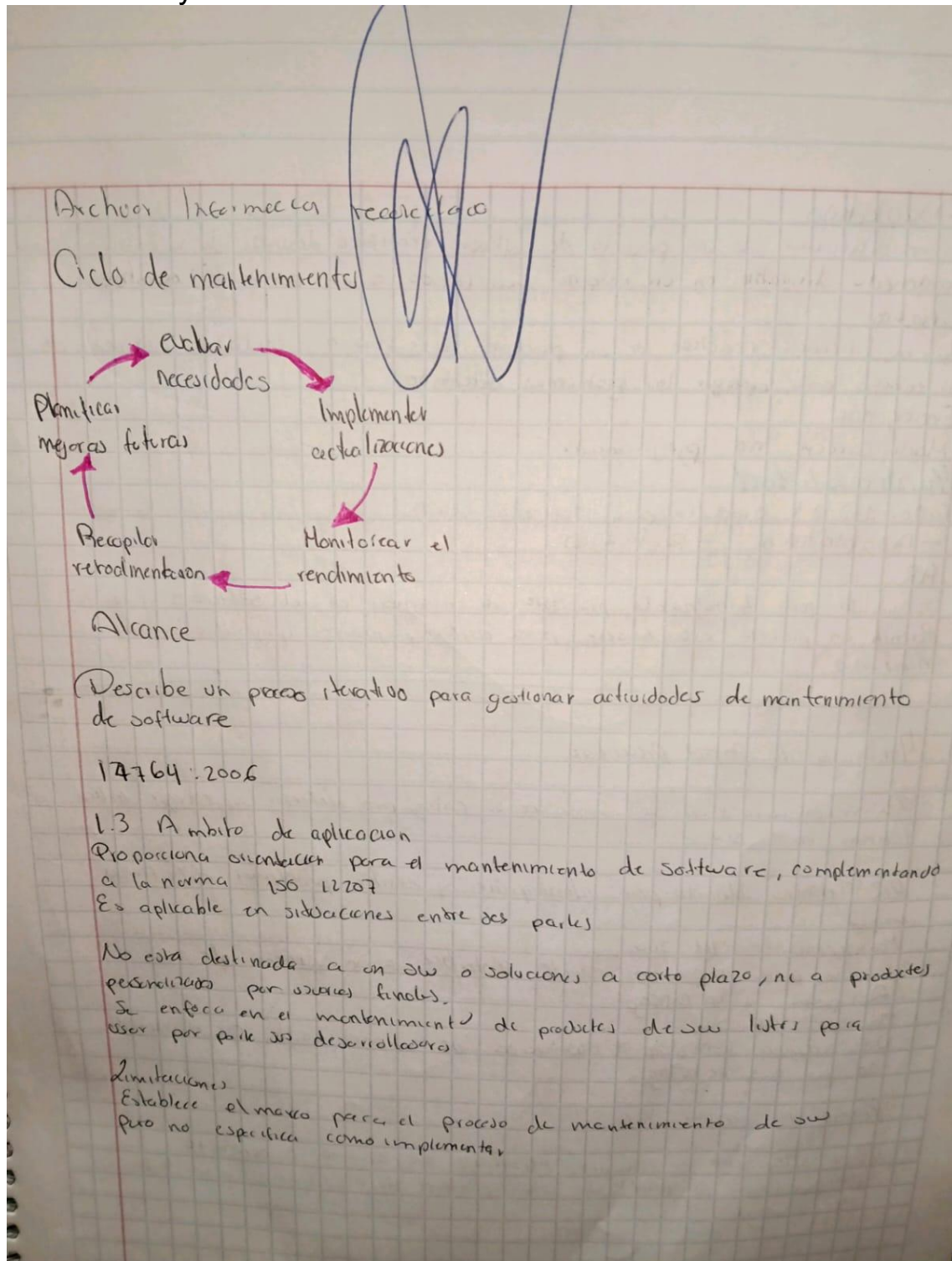
Transición del SW

Una secuencia controlada y coordinada de acciones en la que el desarrollo de SW pasa de la org.

Proceso de mantenimiento

- Por adquisición o subcontrato. Mediante un acuerdo o contrato.
- Por operación: A través de una solicitud de modificación o un informe de problemas.

Evidencia 7 y 8



Evidencia 9

Proceso de mantenimiento

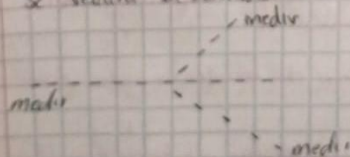
- Proceso de Implementación
- Analisis del problema y la modificación
- Implementación de la modificación
- Revisión / ~~hasta~~ aceptación
- Migración
- Retiro

1. Cuando se ~~realiza~~ los me o pr para evaluar si se hace un proceso de mantenimiento y de que tipo de mantenimiento sea

2. Se busca el problema y solución

3. Realizar las modificaciones (llevar a cabo) en proceso de trabajo

4. Se aplica la modificación permanente, si hay cambio en el código se realiza el cambio



EVIDENCIA 10 EJEMPLO DE UN PLAN DE MANTENIMIENTO PREVENTIVO DE SOFTWARE

1. INTRODUCCIÓN

El presente documento establece el plan de mantenimiento preventivo para garantizar el correcto funcionamiento del software desarrollado en ASP.NET Core. Su objetivo es minimizar fallas, mejorar el rendimiento y asegurar la disponibilidad del sistema.

2. OBJETIVOS

- Prevenir fallos y errores en la aplicación.
- Garantizar el correcto funcionamiento y estabilidad del sistema.
- Optimizar el rendimiento del software.
- Asegurar la integridad de los datos almacenados en la base de datos.
- Reducir tiempos de inactividad y mejorar la experiencia del usuario.

3. ALCANCE

Este plan aplica a todas las aplicaciones desarrolladas en ASP.NET Core, incluyendo sus bases de datos en PostgreSQL, APIs, interfaces web y servidores donde se ejecuta.

4. FRECUENCIA DE MANTENIMIENTO

Actividad	Frecuencia
Revisión de logs del sistema	Semanal
Actualización del framework y paquetes	Mensual
Copias de seguridad de la base de datos	Diaria
Revisión de bases de datos	Trimestral

Pruebas de rendimiento y carga	Semestral
Auditoría de seguridad	Anual

5. ACTIVIDADES DEL MANTENIMIENTO PREVENTIVO

5.1. Copias de Seguridad

- Realizar backups diarios de la base de datos y archivos críticos.
- Almacenar las copias en un servidor seguro y en la nube.

5.2. Actualización del Framework y Dependencias

- Aplicar actualizaciones de seguridad y nuevas versiones de ASP.NET Core.
- Evaluar la compatibilidad de nuevas versiones con los sistemas actuales.
- Actualizar paquetes de NuGet y verificar dependencias obsoletas.

5.3. Revisión de Bases de Datos

- Optimizar consultas y eliminar datos redundantes en PostgreSQL.
- Verificar índices y realizar mantenimiento en tablas.

5.4. Monitoreo y Revisión de Registros (Logs)

- Analizar logs en busca de errores y anomalías usando Serilog o Application Insights.
- Generar reportes de incidencias y corregir problemas detectados.

5.5. Pruebas de Rendimiento y Carga

- Evaluar la carga del sistema y ajustar recursos según necesidad.
- Realizar pruebas de estrés en componentes críticos con herramientas como JMeter o k6.

5.6. Auditoría de Seguridad

- Revisar accesos y permisos de usuarios en Identity Server o Azure AD.
- Identificar vulnerabilidades y aplicar medidas correctivas como el uso de OWASP ZAP.

6. RESPONSABLES

- Administrador del sistema: Supervisión general y ejecución del plan.
- Equipo de desarrollo: Aplicación de actualizaciones y mejoras.
- Departamento de seguridad informática: Auditorías y pruebas de seguridad.
- Soporte técnico: Monitoreo de incidencias y solución de problemas.

7. CONCLUSIONES

El mantenimiento preventivo de aplicaciones en ASP.NET Core es esencial para garantizar la operatividad del sistema, minimizar fallos y optimizar recursos. La implementación de este plan permitirá una mejor administración y disponibilidad del software, beneficiando a usuarios y clientes.

8. REGISTRO Y SEGUIMIENTO

Todas las actividades de mantenimiento deberán ser documentadas en un informe detallado, con registro de incidencias y acciones correctivas implementadas.

PLANTILLA PARA UN PLAN DE MANTENIMIENTO PREVENTIVO DE SOFTWARE

1. INTRODUCCIÓN

(Descripción del plan de mantenimiento y su objetivo)

2. OBJETIVOS

3. ALCANCE

(Descripción del alcance del plan de mantenimiento)

4. FRECUENCIA DE MANTENIMIENTO

Actividad Frecuencia

5. ACTIVIDADES DEL MANTENIMIENTO PREVENTIVO

5.1. Copias de Seguridad

5.2. Actualización del Framework y Dependencias

5.3. Revisión de Bases de Datos

5.4. Monitoreo y Revisión de Registros (Logs)

5.5. Pruebas de Rendimiento y Carga

5.6. Auditoría de Seguridad

6. RESPONSABLES

7. CONCLUSIONES

(Conclusiones sobre la importancia del mantenimiento preventivo)

8. REGISTRO Y SEGUIMIENTO

(Instrucciones para la documentación y seguimiento de actividades)

EVIDENCIA 11 EJEMPLO DE PLAN DE MANTENIMIENTO ADAPTATIVO DE SOFTWARE

EJEMPLO: Sistema de Gestión de Documentos Institucionales*

Una institución educativa ha implementado un sistema de gestión de documentos que automatiza la creación, almacenamiento y administración de archivos. Dado que el sistema se encuentra en constante evolución debido a nuevas necesidades y cambios en regulaciones, se requiere un Plan de Mantenimiento Adaptativo para modificar el sistema según estos requerimientos.

Elementos del Plan de Mantenimiento Adaptativo

1. Identificación del Sistema

- Nombre del sistema: Sistema de Gestión Documental Institucional (SGDI)
- Versión actual: 2.3
- Fecha de implementación: Enero de 2024
- Tecnologías utilizadas: .NET Core, Angular, SQL Server
- Responsable del mantenimiento: Equipo de Desarrollo de TI

2. Justificación del Mantenimiento

El mantenimiento adaptativo es necesario para *ajustar el sistema a cambios en regulaciones gubernamentales, integración con nuevas plataformas y mejoras en la seguridad*.

3. Objetivos del Mantenimiento

- Adaptar el sistema a nuevas normativas de almacenamiento de documentos electrónicos.
- Integrar una API de firma digital para documentos oficiales.
- Mejorar los permisos de acceso según los roles de los usuarios.

4. Alcance del Mantenimiento

- Módulos afectados:

Módulo de gestión de documentos

Módulo de autenticación y roles

API de integración

- Áreas impactadas:

Seguridad y control de acceso

Interoperabilidad con otras plataformas

Cumplimiento normativo

5. Requerimientos y Recursos

-Equipo de desarrollo: 3 desarrolladores backend, 2 frontend, 1 analista de seguridad.

-Herramientas necesarias: Visual Studio, Postman, Azure DevOps.

-Presupuesto estimado: \$5,000 USD

-Tiempo estimado: 2 meses

6. Procedimiento del Mantenimiento

FASE	ACTIVIDAD	RESPONSABLE	TIEMPO ESTIMADO
Análisis	Identificación de nuevos requisitos y normativas	Analista de TI	2 semanas
Diseño	Planificación de cambios en la arquitectura del sistema	Equipo de desarrollo	2 semanas

Implementación	Desarrollo e integración de nuevas funciones	Desarrolladores	4 semanas
Pruebas	Verificación de compatibilidad y seguridad	QA y Seguridad	2 semanas
Despliegue	Implementación en el entorno de producción	Equipo de TI	1 semana
Monitoreo	Seguimiento y corrección de posibles fallos	Soporte técnico	Continuo

7. Evaluación y Validación

○ Indicadores de éxito:

- Implementación correcta de nuevas reglas de acceso
- Integración de firma digital sin errores
- Cumplimiento de normativas al 100%

○ Métodos de evaluación:

- Pruebas de carga y seguridad
- Auditorías internas

8. Plan de Contingencia

Si se presentan fallos graves en la integración, se mantendrá una versión estable anterior y se programarán correcciones inmediatas.

PLANTILLA

PLAN DE MANTENIMIENTO ADAPTATIVO

1. Identificación del Sistema

- Nombre del sistema: [Nombre del sistema]
- Versión actual: [Versión]
- Fecha de implementación: [Fecha]
- Tecnologías utilizadas: [Tecnologías]
- Responsable del mantenimiento: [Equipo responsable]

2. Justificación del Mantenimiento

- [Razones del mantenimiento adaptativo]

3. Objetivos del Mantenimiento

- [Objetivos específicos]

4. Alcance del Mantenimiento

- Módulos afectados: [Módulos]
- Áreas impactadas: [Áreas]

5. Requerimientos y Recursos

- Equipo de trabajo: [Roles y cantidad]
- Herramientas necesarias: [Software y hardware]
- Presupuesto estimado: [Presupuesto]
- Tiempo estimado: [Tiempo]

6. Procedimiento del Mantenimiento

FASE	ACTIVIDAD	RESPONSABLE	TIEMPO ESTIMADO
Análisis	[Descripción]	[Responsable]	[Tiempo]
Diseño	[Descripción]	[Responsable]	[Tiempo]

Implementación	[Descripción]	[Responsable]	[Tiempo]
Pruebas	[Descripción]	[Responsable]	[Tiempo]
Despliegue	[Descripción]	[Responsable]	[Tiempo]
Monitoreo	[Descripción]	[Responsable]	[Tiempo]

7. Evaluación y Validación

- Indicadores de éxito: [Indicadores]
- Métodos de evaluación: [Métodos]

8. Plan de Contingencia

- [Medidas en caso de fallos]

EVIDENCIA 12 EJEMPLO PLAN DE MANTENIMIENTO

PERFECTIVO EN RED SOCIAL

Propósito del plan

Mejorar el rendimiento, funcionalidad y experiencia del usuario mediante la optimización de consultas, rediseños a las interfaces y la inclusión de nuevas características.

Alcance

- Mejora de los tiempos de carga de las páginas principales.
- Optimización de consultas SQL para reducir tiempos de respuesta.
- Rediseñar las interfaces para una fácil navegación.

Fases

1. Análisis y diagnostico

- Recopilar feedback de los usuarios
- Evaluación de métricas actuales de rendimiento
- Revisión del código fuente para identificar cuellos de botella

2. Diseño y planificación

- Diseñar una nueva arquitectura para las consultas SQL
- Prototipado de mejoras en la interfaz de usuario
- Definición de nuevas funcionalidades
- Documentar los cambios planificados

3. Implementación

- Optimizar consultas SQL
- Desarrollo de nuevas funcionalidades
- Ajuste del frontend para mejorar la experiencia de usuario
- Realizar pruebas unitarias y de integración durante el desarrollo

4. Pruebas y validación

- Pruebas de rendimiento antes y después de los cambios
- Pruebas de usabilidad con usuarios finales
- Corrección de errores encontrados

5. Despliegue y evaluación

- Implementación en el entorno de producción
- Encuestas de satisfacción de los usuarios

Evaluación de resultados

- Comparación de tiempos de carga y consulta antes y después del mantenimiento
- Informe final con el impacto del mantenimiento perfectivo

PLANTILLA

[Título del plan de mantenimiento perfectivo]

Propósito del plan

[Describir le objetivo del plan, por ejemplo, las mejoras de rendimiento, funcionalidad y experiencia del usuario]

Alcance

[Describir los aspectos que se abordaran]

Fases del plan

Análisis y diagnostico

[Identificar oportunidades de mejora]

Diseño y planificación

[Diseñar las mejoras propuestas]

Implementación

[Aplicar las mejoras planificadas]

Pruebas y validación

[Confirmar que los cambios cumplen con los objetivos del plan]



Despliegue y evaluación

[Poner en producción las mejoras y evaluar su impacto]

Evaluación de resultados

[Analiza el éxito del plan mediante indicadores claros]

EVIDENCIA 13 EJEMPLO DE FORMATO PR

	Plan de Mantenimiento Preventivo y Correctivo de equipos informáticos	Código: EPN-DGIP-OP-001-PMPC	
		Versión: 01	
		Fecha de Aprobación:	
		Hoja: Página 1 de 20	

1 **Elaborado por:**

NOMBRE	CARGO	FIRMA
Pablo Ortiz	Especialista Técnico 1	

2 **Revisado por:**

NOMBRE	CARGO	FIRMA
Wilson Delgado	Especialista Técnico 2	

3 **Aprobado por:**

NOMBRE	CARGO	
Msc. Roberto Andrade	Director DGIP	

4 **HOJA DEL ESTADO DEL DOCUMENTO**

VERSIÓN	FECHA	ÍTEM	ASPECTO CAMBIADO	RAZONES	PERSONA QUE SOLICITÓ EL CAMBIO
01	04/08/16				

EVIDENCIA 14 EJEMPLO DE FORMATO MR

Fecha de emisión	29-ene-2018	Prioridad del problema	Alta
Emitido por	Tamara Agafonov	Cargo	Especialista Jefe de la Mesa de Ayuda

Problemas reportados	<ul style="list-style-type: none"> • Cuelgues frecuentes en teléfonos de escritorio (12 usuarios en 1 semana). • Sin tono de marcado en teléfonos de escritorio (9 usuarios en 1 semana). • Llamada finalizada al intentar transferir una llamada a otra línea (3 usuarios en 1 semana).
Descripción del problema	El directorio telefónico está desalineado y el cableado no está equipado para manejar el volumen actual de llamadas.
Solución propuesta	Actualizar el directorio actual a uno que sea capaz de manejar 200 llamadas simultáneas (actualmente el

	<p>número máximo de llamadas es 127).</p> <p>Esta solución costará \$12,300, y cualquier futura actualización para duplicar la capacidad costará \$1,800 cada una.</p>
Resultado esperado	Esperamos que todos los incidentes reportados se disipen una vez que se haya instalado el nuevo directorio
Lecciones aprendidas	<p>Verificar periódicamente el equipo administrativo para determinar si están equipados para manejar el número creciente de empleados.</p> <p>Esto incluye, pero no se limita a, el servidor de correo electrónico, el sitio web de pedidos de almuerzo, el plan de cobertura de teléfonos celulares, el programa de arrendamiento de automóviles y el sitio web interno de beneficios para empleados.</p>
Frecuencia de uso	Una vez cada 6 meses, o cada vez que la empresa crezca al doble de su número anterior de empleados (lo que ocurra primero).
Categoría del problema	Directorio de teléfonos de escritorio, cuelgues, sin tono de marcado.

CONCLUSIÓN

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

pdfcoffee.com_iso12207-2017-pdf-free (1).pdf

10.1109_ieeestd.2006.235774 (1).pdf

SWM_samples (1).pdf

19880003572 (1).pdf

SCADA Preventative Maintenance_ Reducing the Potential of Unexpect.pdf