**DOCUMENTO DE PLAN DE MANTENIMIENTO DEL SISTEMA DE MAQUINAS RECREATIVAS**

Curso de Construcción de Software

(Castillo Merejildo Joshúa Javier,

Sabando Barberán Edú,

Castro Murillo Jean Fernando,

Gabino Villao Joel Fabian,

Quiroz Gómez Euro Johann

)

Contenido

[**1. Introducción 3**](#_heading=h.r2da4fl3rgtm)

[**2. Tipos de mantenimiento 3**](#_heading=h.25ovttunv0ha)

[2.1 Mantenimiento correctivo 3](#_heading=h.iwsnfncvd7d7)

[2.2 Mantenimiento adaptativo 4](#_heading=h.fm9q78wed7x8)

[**3. Ingeniería inversa 10**](#_heading=h.68xio9a5u6hi)

[**4. Implementación 11**](#_heading=h.c7pgipu0vt3p)

[4.1 Cronograma 11](#_heading=h.y7xxpwylpsgo)

[**5 Roles y responsabilidades 15**](#_heading=h.vsj37f938ie)

[**6. Conclusión 15**](#_heading=h.ohft4ngq7ip)

[**7 Documentos de referencia 16**](#_heading=h.8pyfc94piqk)

[**8. Aprobación 16**](#_heading=h.roisshebyzfh)

# 1. Introducción

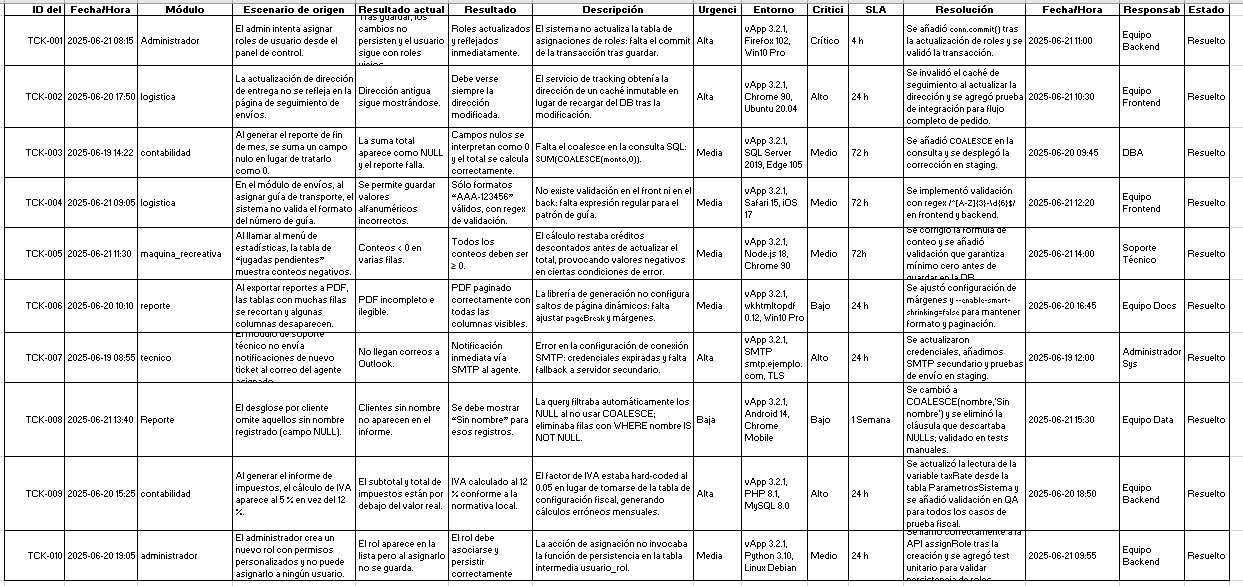
El sistema RECREA SYS es una aplicación web para administrar el ciclo de vida completo de máquinas recreativas, desde su montaje hasta la recaudación. Basado en los principios de Construcción de Software, este plan detalla las estrategias de mantenimiento para garantizar su operatividad continua y mejora progresiva.

# 2. Tipos de mantenimiento

## 2.1 Mantenimiento correctivo

**Proceso de gestión de incidencias:**

1. **Detección y reporte**:
   * Sistema de tickets categorizados por criticidad (Crítico, Alto, Medio, Bajo)
   * Monitoreo manual con alertas para:
     + Fallos en comunicación entre módulos
2. **Priorización**:
   * Matriz de impacto/urgencia
   * SLAs definidos:
     + Crítico: Resolución en 4 horas
     + Alto: 24 horas
     + Medio: 72 horas
     + Bajo: 1 semana
3. **Resolución**:
   * Protocolo de análisis de causa raíz
   * Sistema de rollback automatizado



## 2.2 Mantenimiento adaptativo

**2.2.1 Descripción y objetivos**

El mantenimiento adaptativo se encarga de modificar el software para preservar su operatividad frente a cambios en el entorno operativo, tales como actualizaciones de sistemas operativos, variaciones en hardware, nuevas versiones de librerías, regulaciones emergentes o integraciones con tecnologías recientes.

Este tipo de mantenimiento se activa específicamente cuando ocurren cambios externos inevitables que pueden comprometer la estabilidad, funcionalidad o seguridad del sistema, como la obsolescencia de tecnologías utilizadas, la finalización del soporte (EOL), la aparición de vulnerabilidades críticas o la exigencia de cumplimiento normativo.

De este modo, se garantiza la vigencia tecnológica, la continuidad operativa y la competitividad del sistema a lo largo del tiempo.

**Objetivos principales:**

• Asegurar compatibilidad con plataformas y dependencias externas.

• Incorporar tecnologías emergentes que aporten valor.

• Minimizar la obsolescencia técnica y los riesgos de seguridad

**2.2.2 Áreas clave de adaptación**

|  |  |
| --- | --- |
| **Área** | **Descripción** |
| **Tecnologías emergentes** | Evaluación e integración de nuevos lenguajes, servicios o métodos (p. ej. IA, microservicios). |
| **Roadmap de compatibilidad** | Plan de versiones para asegurarse de soportar entornos y frameworks clave. |
| **Frameworks frontend (React)** | Asegurar compatibilidad con versiones LTS y mayoristas de React; pruebas de regresión tras cada upgrade. |
| **Entornos de ejecución** | SO (Windows, Linux, macOS), contenedores Docker, Kubernetes. |
| **Dependencias de terceros** | Versiones de bases de datos, librerías, APIs externas. |
| **Regulaciones y estándares** | Cumplimiento de normativas locales (p. ej. GDPR, HIPAA, PCI-DSS). |
| **Hardware y dispositivo** | Adaptaciones para nuevas arquitecturas (ARM, x86\_64), dispositivos móviles o IoT. |

**2.2.3 Proceso de gestión**

**1.** **Detección de cambio**

La detección de cambio implica la supervisión constante de todas las piezas del ecosistema tecnológico, tales como versiones de React, navegadores web, sistemas operativos y librerías de terceros. Para ello se emplean herramientas automatizadas de monitoreo de dependencias y alertas de seguridad. Este proceso garantiza la identificación temprana de nuevas versiones o vulnerabilidades críticas, permitiendo al equipo de Arquitectura y DevOps anticipar posibles incompatibilidades antes de que afecten al entorno de producción.

**2.** **Análisis de impacto**

Una vez detectado un cambio relevante, el equipo conjunto de Desarrollo, QA y Seguridad lleva a cabo un análisis de impacto exhaustivo. En este informe se detallan los módulos afectados, las dependencias críticas y los riesgos técnicos, funcionales y de negocio asociados. La elaboración y validación de este documento, asegura que todas las partes interesadas comprendan el alcance del trabajo y las posibles consecuencias de la actualización.

**3.** **Planificación**

Con el análisis de impacto aprobado, los líderes de Frontend, Backend y DevOps programan la actividad en el roadmap de compatibilidad durante la sesión de planificación del sprint. Se definen prioridades, estimaciones de esfuerzo y recursos asignados, así como fechas de inicio y finalización para cada tarea. El resultado se refleja en la herramienta de gestión de, lo que permite una visibilidad completa del cronograma y facilita el seguimiento sistemático del avance.

**4.** **Implementación**

Durante la fase de implementación, los desarrolladores actualizan el código y las dependencias pertinentes, así como refactorizan los componentes que requieran adaptarse a nuevas APIs o patrones de diseño. Todo cambio se incorpora mediante pull requests, sujetos a revisiones de al menos dos miembros del equipo, y se verifica su correcta integración y compilación en el entorno de integración continua. Este enfoque basado en control de versiones y revisiones de pares minimiza la introducción de errores y mantiene la coherencia del repositorio.

**5.** **Pruebas y validación**

Posteriormente, el equipo de QA ejecuta en un entorno espejo de producción (staging) tanto pruebas de regresión automatizadas como casos manuales representativos. Solo cuando el ciento por ciento de las pruebas supera los criterios de aceptación establecidos y se firma el checklist de validación, se genera un informe de resultados que respalda la transición a la siguiente etapa. Este riguroso proceso de pruebas garantiza la detección temprana de posibles regresiones y la conformidad con los requisitos funcionales y de rendimiento.

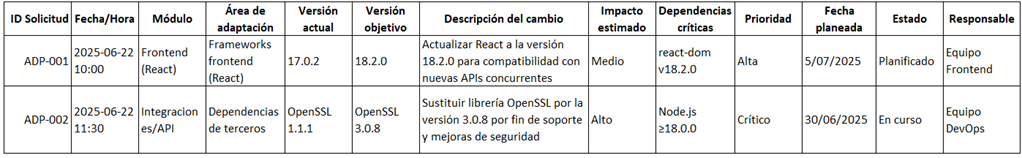
**6.** **Despliegue**

El despliegue en producción se efectúa de forma escalonada mediante estrategias canary o blue‑green. Durante y tras la migración, se monitorizan indicadores de salud del sistema y métricas de rendimiento para confirmar que la actualización no degrade la experiencia de usuario ni comprometa la disponibilidad. En caso de emerger anomalías, se activa el procedimiento de rollback documentado en el runbook de emergencia, asegurando una rápida restitución del servicio.

**7.** **Cierre y documentación**

Finalmente, el Project Manager y los líderes técnicos cierran la actividad de mantenimiento adaptativo registrando los tiempos reales consumidos, las incidencias detectadas y las lecciones aprendidas. Se actualiza la documentación de arquitectura, los manuales de usuario y el inventario de dependencias en el repositorio central. Este informe de cierre constituye un insumo fundamental para la mejora continua, enriqueciendo la base de conocimiento del equipo y optimizando futuros ciclos de mantenimiento.

**2.2.4 Plantilla de registro para Mantenimiento Adaptativo**



**Notas de registro de mantenimiento**

* **ID Solicitud**: Convención “ADP-XXX”.
* **Área de adaptación**: coincide con las “Áreas clave” definidas más arriba.
* **Impacto estimado**: tiempo y esfuerzo, riesgos de ruptura.
* **Dependencias críticas**: versiones mínimas requeridas en librerías o plataformas.
* **Prioridad**: según la urgencia de mantenerse al día con tecnologías o requisitos de negocio.

**2.2.5 Riesgos comunes en el mantenimiento adaptativo**

**Incompatibilidades de versiones**Cuando se actualizan librerías o frameworks, existe el riesgo de que dependencias internas o externas dejen de funcionar entre sí.  
Para evitar esto debemos mantener un inventario actualizado de todas las dependencias; usar entornos de pruebas completamente aislados; ejecutar una batería de pruebas de regresión tras cada cambio.

**Regresiones funcionales**Las modificaciones introducidas para soportar nuevas versiones pueden generar fallos en áreas previamente estables.  
Por lo que se debe automatizar pruebas unitarias y de integración continuas; realizar revisiones de código antes de fusionar cualquier cambio

**Demoras en el roadmap**La planificación de actualizaciones puede verse retrasada por falta de recursos, imprevistos técnicos o dependencia de terceros.  
Evaluar periódicamente la capacidad de los equipos; aplicar metodologías ágiles (sprints cortos, revisiones frecuentes) para reajustar prioridades y tiempos.

**Incremento de deuda técnica**Las actualizaciones frecuentes sin dedicar tiempo al refactorizado pueden acumular código difícil de mantener. Por ello se debe programar tareas regulares de limpieza de código (refactoring) y medir la deuda técnica con herramientas de análisis estático.

**Conflictos de licenciamiento**Incorporar nuevas dependencias puede chocar con las políticas de licencias de la organización.

Revisar la compatibilidad de licencias antes de adoptar cualquier componente y utilizar herramientas de escaneo de licencias.

**Sobrecarga de entornos de testing**La ejecución simultánea de múltiples suites de pruebas puede saturar los recursos de staging o integración continua. Se tiene que dimensionar adecuadamente los entornos de pruebas, empleando contenedores efímeros o infraestructuras en la nube para escalar dinámicamente según la demanda.

**2.2.6 Roadmap de compatibilidad**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Hito** | **Módulo / Área** | **Versión actual** | **Versión objetivo** | **Ultima actualización** | **Responsable** |
| **R1 – Análisis de compatibilidad** | **Todas** | **N/A** | **N/A** | **2025-06-28** | **Equipo Arquitectura** |
| **R2 – React 18 Upgrade** | **Frontend (React)** | **17.0.2** | **18.2.0** | **2025-07-10** | **Equipo Frontend** |
| **R3 – OpenSSL 3.0** | **Integraciones/API** | **1.1.1** | **3.0.8** | **2025-07-05** | **Equipo DevOps** |
| **R4 – Node.js LTS** | **Backend general** | **16.x** | **18.x LTS** | **2025-07-12** | **Equipo Backend** |
| **R5 – SO Linux (Debian)** | **Infraestructura / Contenedores** | **Debian 9** | **Debian 11** | **2025-07-17** | **Equipo Infra** |
| **R6 – Safari / iOS** | **QA / Móvil** | **iOS 15** | **iOS 17** | **2025-07-20** | **Equipo QA** |
| **R7 – Windows 11** | **QA / Escritorio** | **Win10 Pro** | **Win11 Pro** | **2025-07-22** | **Equipo QA** |
| **R8 – Kubernetes 1.23 1.27** | **Infraestructura / Contenedores** | **1.23** | **1.27** | **2025-07-25** | **Equipo DevOps** |
| **R9 – PCI‑DSS v4.0** | **Seguridad / Auditoría** | **v3.2** | **v4.0** | **2025-07-30** | **Equipo Seguridad** |
| **R10 – Revisión y cierre** | **Todas** | **N/A** | **N/A** | **2025-08-02** | **Equipo Proyecto** |

# 3. Ingeniería inversa

**Metodología:**

1. **Documentación**:
   * Wiki técnica con convenciones de código. Esto puede incluir:

* Identificación de módulos o componentes principales.
* Análisis de la arquitectura del software.
* Seguimiento de flujos de datos y procesos internos.
* Detección de patrones de diseño utilizados.
* Buenas prácticas para desarrollo y mantenimiento que han sido aplicadas al proyecto

# 4. Implementación

## 4.1 Cronograma

**6.3 Implementación en Tiempo Real**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Fecha* | *Módulo Afectado* | *Descripción del Problema* | *Análisis y*  *Causa Técnica* | *Solución*  *Implementada* |
| *04-jul-2025* | **Perfil de Usuario** | Pantalla blanca al cargar perfil | Se accedía al perfil sin verificar si el UUID estaba presente o bien formado. | Se implementó validación de UUID con regex y redirección automática a login. |
| *07-jul-2025* | **Perfil de Usuario** | Error: 'Datos de usuario incompletos' | El backend retornaba datos sin campos esenciales como 'ci' o 'email'. | Se añadió verificación explícita y se lanza error controlado. |
| *11-jul-2025* | **Header Admin** | Cerrar sesión no eliminaba datos correctamente | Se eliminaba solo el token, sin notificar al backend. | Se agregó llamada a /api/usuario/logout y limpieza de localStorage. |
| *16-jul-2025* | **Menú Perfil** | Error en botón 'Editar Perfil' | Se usaba currentUser.uuid sin verificar si currentUser era null. | Se ajustó la condición para navegar solo si currentUser existe. |
| *21-jul-2025* | **Chat Usuarios** | Chat no cargaba al ingresar desde URL | No se pasaba correctamente el userId desde la URL. | Se implementó lectura desde currentUser?.ID\_Usuario y fallback desde URLSearchParams. |
| *25-jul-2025* | **Chat Usuarios** | No se podían enviar comentarios sin reporte previo | El sistema asumía que ya existía un reporte entre usuarios. | Se agregó verificación y creación dinámica del reporte si no existe. |
| *30-jul-2025* | **Chat Usuarios** | Comentarios no se actualizaban tras enviar | No se refrescaba el estado comentarios después del envío. | Se llama a cargarComentariosReporte() tras enviar el comentario. |
| *08-ago-2025* | **Header Admin** | Menú perfil mostraba opciones sin usuario cargado | Error en currentUser.usuario\_asignado cuando currentUser era undefined. | Se validó que currentUser exista antes de renderizar menú. |
| *15-ago-2025* | **Chat Usuarios** | Mensajes duplicados con múltiples reportes | El estado comentarios no se reiniciaba correctamente. | Se limpia comentarios antes de cargar nuevos. |
| *24-ago-2025* | **Chat Usuarios** | Falta de indicador de conversación activa | El usuario no sabía si estaba en una conversación activa. | Se agregó clase visual activa al usuario seleccionado. |
| *04-sep-2025* | **Chat Usuarios** | Errores no visibles al usuario | Los errores iban a consola, no se mostraban en la interfaz. | Se añadió estado error y div visual para mostrar mensajes. |
| *13-sep-2025* | **BD y Backend** | Reportes no se asociaban bien en nuevos chats | No se consultaban ambos sentidos emisor/destinatario. | Se implementó lógica que busca chats entre ambos sentidos. |
| *20-sep-2025* | **Login y Navegación** | Acceso al sistema sin sesión activa | Faltaba verificación de sesión en varios useEffect. | Se redirige a login si no hay usuario identificado. |
| *28-sep-2025* | **Base de Datos** | 'usuario\_asignado' y 'contrasena' impedían logins. | Campos por defecto no permitían logins funcionales en usuarios sin asignar. | El backend valida si los campos son reales antes de permitir inicio de sesión. |

# 

# 5 Roles y responsabilidades

**Equipo especializado:**

* **Arquitecto**: Supervisión integral de la evolución y consistencia del modelo arquitectónico basado en el patrón Model-View-Controller (MVC)**,** asegurando que los cambios estructurales del sistema sigan principios de bajo acoplamiento y alta cohesión.
* **Líder técnico**: Coordinación diaria
* **Desarrolladores junior**: Componentes periféricos
* **Analista QA**: Planificación de pruebas

# 6. Conclusión

Este plan detallado establece un marco completo para el mantenimiento evolutivo de RECREA SYS, incorporando las mejores prácticas de ingeniería de software aprendidas. Se enfoca en:

1. **Sostenibilidad**: Ciclos de vida definidos para componentes
2. **Escalabilidad**: Arquitectura preparada para crecimiento
3. **Adaptabilidad**: Mecanismos para cambios futuros

El plan será revisado semestralmente, ajustándose a las necesidades emergentes del negocio y los avances tecnológicos, garantizando así la longevidad y eficacia continua del sistema.

# 7 Documentos de referencia

|  |
| --- |
| **DOCUMENTO VERSIÓN** |
| Documento de Diseño Detallado - Grupo F |
| Documento de Alcance - Grupo F |
| Guía de instalación paso a paso - Grupo F |
| Documento de Requerimientos - Grupo F |
| Documento de plan de pruebas – Grupo F |

# 8. Aprobación

(Fecha)

|  |  |
| --- | --- |
| Elaborado por: | Revisado por: |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Nombre  Cargo | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Nombre  Cargo |
| Aprobado por:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Nombre  **Líder del Proyecto** | Aprobado por:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Nombre  **Director de la Unidad de Tecnologías de la Información y Comunicaciones de la Empresa X** |