TaskMaster Pro

Plan de Migración

Versión: 0100

Fecha: 08/07/2025

HOJA DE CONTROL

Organismo	SENA		
Proyecto	TaskMaster Pro		
Entregable	Plan de migración		
Autor	Johan Felipe Garcia Salazar Erika Daniela Triana Bustos Nikole Camila Bernal Ávila Andrés Julián Garzón Perea		
Aprobado por		**Fecha Aprobación**	08/07/2025
		Nº Total de Páginas	13

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
0100	Versión inicial	Johan Felipe Garcia Salazar	08/07/2025

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos
Johan Felipe Garcia Salazar
Erika Daniela Triana Bustos
Nikole Camila Bernal Ávila
Andrés Julián Garzón Perea

1. Objetivo

El objetivo de este plan de migración es establecer un proceso estructurado, seguro y controlado para trasladar el sistema **TaskMaster Pro** hacia un nuevo entorno tecnológico o estructura funcional. Esto incluye preservar la integridad de los datos, minimizar riesgos operativos y garantizar la continuidad del servicio para todos los usuarios. El plan proporciona lineamientos claros para realizar respaldos, ejecutar cambios y validar que la migración se haya realizado con éxito.

2. Alcance

Este plan abarca todas las actividades relacionadas con la migración del sistema TaskMaster Pro, incluyendo:

- Migración de los componentes principales del sistema: frontend, backend y base de datos.
- Preservación y restauración de archivos de configuración, recursos estáticos, y archivos de usuario relevantes.
- Afectación parcial o total de los módulos del sistema, incluyendo autenticación, gestión de usuarios, proyectos, tareas, reportes y funcionalidades conectadas con servicios externos.
- Coordinación entre los equipos de desarrollo, control de calidad, infraestructura y soporte técnico.

3. Consideraciones Generales

Antes de ejecutar el proceso de migración, se deben tener en cuenta los siguientes aspectos:

- **Justificación técnica y funcional:** La migración responde a la necesidad de actualizar dependencias, mejorar el rendimiento del sistema y asegurar la compatibilidad con nuevos entornos.
- **Impacto previsto:** La migración puede afectar temporalmente algunos módulos del sistema. Por tanto, se deben planificar las ventanas de mantenimiento y comunicarlo a los usuarios.

- **Respaldo previo:** Es obligatorio realizar respaldos completos de la base de datos, archivos críticos, certificados y configuraciones antes de iniciar cualquier cambio.

- **Pruebas y entornos controlados:** Todas las modificaciones deben ser probadas previamente en entornos de staging, utilizando herramientas de automatización y control de versiones para asegurar su estabilidad.

- **Capacitación técnica:** El personal involucrado debe estar capacitado para aplicar el plan, monitorear la migración y actuar ante posibles contingencias.

4. Respaldos y Preparación

Para garantizar la continuidad del sistema y la recuperación efectiva ante cualquier incidente, se establece el siguiente **plan de respaldo** que abarca tanto la base de datos como los archivos críticos del sistema.

Base de datos: Dump completo

- Se deberá realizar un **respaldo completo (dump)** de la base de datos MySQL antes de cualquier actualización, cambio mayor o despliegue.
- Comando utilizado: mysqldump -u root taskmasterdb > respaldo_taskmaster_YYYY-MM-DD.sql
- Este archivo debe almacenarse en: D:\TaskMasterPro\db\backups\
- Se recomienda establecer una rutina de respaldo **semanal**, y obligatoriamente antes de cada cambio importante.

Archivos críticos a respaldar

Los siguientes elementos deben ser respaldados manual o automáticamente:

Tipo de archivo	Ejemplo de ubicación	Descripción
Archivos de configuración	backend_web/.env - frontend_web/.env	Contienen variables de entorno sensible.
Recursos estáticos	frontend_web/public/ - uploads/attachments/	Imágenes, íconos, archivos subidos.

Evidencia de respaldos

Para cada respaldo realizado, se deberá registrar:

- Fecha y hora del respaldo
- Usuario responsable
- Ubicación exacta del archivo

Ejemplo de registro:

[2025-07-09 15:20] - Respaldo completo base de datos y .env realizado por Andrés Garzon

Ubicación: D:\TaskMasterPro\db\backups\respaldo_taskmaster_2025-07-09.sql

5. Metodología y Objetivos

Este proceso tiene como finalidad asegurar que cualquier cambio en el sistema pueda ser controlado, validado, y revertido en caso de fallos.

Objetivos

- Evitar pérdida de información durante actualizaciones.
- Validar los cambios antes de afectar entornos productivos.
- Mantener trazabilidad técnica de todo el ciclo de cambios.

Metodología de trabajo establecida

1. **Respaldo obligatorio** de la base de datos y archivos críticos **antes de aplicar cualquier cambio** al sistema.
2. Uso de herramientas de **integración y entrega continua (CI/CD)** para compilar, probar y validar el sistema antes del despliegue.
3. **Documentación detallada** de cada cambio aplicado: versión, fecha, autor, archivos modificados, propósito.
4. Todos los despliegues deben realizarse **primero en entornos de prueba (staging)**, con validación por parte del equipo de QA.
5. Sólo tras aprobación de pruebas funcionales y técnicas, se permite el paso a producción.

6. Equipos Relacionados

Los siguientes equipos son responsables de ejecutar y supervisar las actividades descritas en esta sección:

Equipo	Responsabilidades principales
Equipo de Desarrollo y Migración (EDM)	Desarrollo de nuevas funciones, actualizaciones y ejecución de despliegues.
Equipo de Control de Calidad (QA)	Verificación de funcionalidades, ejecución de pruebas en entorno de staging.

7. Estrategia Técnica: Rama de Migración y Entrega Continua

Para garantizar una migración segura, controlada y sin interrupciones en el entorno de producción, se plantea la siguiente estrategia técnica:

Uso de Rama Secundaria (migración)

- Se ha creado una rama específica en el sistema de control de versiones (Git), denominada `migración`.
- Todo cambio estructural, de versiones o dependencias se realiza solo en esta rama.
- Los desarrolladores realizan pruebas locales y en un entorno de staging (preproducción) antes de hacer cualquier fusión.
- Las actualizaciones deben seguir el flujo de Pull Request con revisión por pares.
- Se asegura que la rama principal (main o production) no se vea afectada hasta la aprobación.

Entrega Continua

Se planea usar CI/CD formalmente para poder detectar errores más rápido y entregar versiones nuevas del software de forma segura y constante. Aunque no este implementado aun, se realizaria de la siguiente manera:

- Se implementa un flujo de Integración y Entrega Continua (CI/CD) a través de herramientas como GitHub Actions, GitLab CI o similar (ajusta esto según lo que usen).
- Cada vez que se suban cambios a las ramas migración, se ejecutarán automáticamente los siguientes pasos:
    - Validación de pruebas unitarias (si existen).

    - Revisión de dependencias y conflictos.
    - Despliegue automático en entorno de staging para pruebas funcionales.
- Solo después de aprobar las pruebas, se permitirá el despliegue en el entorno de producción.

Esta estrategia permite minimizar riesgos, garantizar la calidad del sistema antes de su despliegue final y facilitar la reversión en caso de fallos durante la migración.

8. Gestión de Riesgos

Riesgo	Impacto	Mitigación
Incompatibilidad con nuevas versiones	Alto	Pruebas en entorno de staging, documentación de breaking changes
Pérdida de datos durante la migración	Crítico	Copias de seguridad (backups) completas previas, validación post-migración.
Retardo en la migración por problemas técnicos	Medio	Cronograma con tiempos de contingencia y recursos técnicos disponibles.
Falta de capacitación del personal post-migración	Medio	Manuales actualizados, sesiones de capacitación técnica y funcional.
Interrupción del servicio en producción	Alto	Planificación de ventanas de mantenimiento, comunicación previa a usuarios.

Riesgo	Impacto	Mitigación
Problemas de compatibilidad con navegadores o dispositivos móviles	Alto	Pruebas de compatibilidad en distintos navegadores y tamaños de pantalla; diseño responsivo.

9. Plan de Validación y Pruebas (con Automatización)

Tipo	Descripción	Automatización	Herramientas
Pruebas unitarias	Validan métodos aislados	No	Pruebas en desarrollo
Pruebas de Integración	Verifican que los módulos del sistema funcionen correctamente entre sí	No	Postman, Insomnia
Pruebas Funcionales	Validan que las funciones del sistema cumplan los requerimientos	Si	Casos de Prueba, Excel y Cucumber + Serenity
Pruebas de Sistema	Evalúan el comportamiento del sistema completo como un todo	Si	Cucumber + Serenity
Pruebas de Aceptación	Evalúan si el sistema satisface los criterios del usuario final	Si, usando lenguaje Gherkin (Given-When-Then) y	Cucumber + Serenity

		Serenity	

Automatización con Cucumber + Serenity
Cualidades de la herramienta:

- Serenity genera reportes visuales detallados, incluyendo pasos, capturas de pantalla y evidencias.
- Permite escribir los casos de prueba en lenguaje natural (Gherkin), lo cual facilita la comprensión por parte de desarrolladores, testers y usuarios no técnicos.
- Puedes vincular fácilmente los escenarios de prueba con los requerimientos funcionales definidos, garantizando cobertura de pruebas.
- Cada ejecución de prueba queda documentada con capturas de pantalla, logs y resultados esperados vs. reales.
- Organiza las pruebas en features, escenarios y pasos, lo cual mejora la gestión de los casos.

10. Documentación y Control de Versiones

La documentación del sistema se ha organizado de forma estructurada para facilitar su consulta y mantenimiento. Entre los documentos más relevantes se encuentran:

- Manual Técnico: Describe la arquitectura, componentes del sistema, configuraciones y dependencias.
- Manual de Usuario: Instrucciones claras para el uso del sistema por parte de los usuarios finales.
- Modelo de Datos: Diagramas y descripciones de la base de datos.
- Casos de Uso: Listado detallado de las funcionalidades con su descripción, actores involucrados y flujos.
- Plan de pruebas: Tabla de tipos de prueba, métodos, resultados esperados y herramientas utilizadas.
- Plan de migración: Documento que guía el paso del sistema al entorno productivo de forma segura.

La documentación se encuentra almacenada en una carpeta llamada docs en la rama main del repositorio

El control de versiones del código fuente del sistema se gestiona utilizando **Git**, mediante la plataforma **GitHub**. El proyecto cuenta con una estructura de ramas que facilita el trabajo colaborativo y la estabilidad del sistema:

- `main`: Rama principal con el código estable y listo para producción.
- `Felipe`: Rama de desarrollo personal.
- `Erika`: Ramas de desarrollo personal.
- `Andres`: Rama de desarrollo personal.
- `Nikole`: Rama de desarrollo personal.
- `Migración`: Rama usada exclusivamente para la implementación del plan de migración.

Cada cambio en el sistema se documenta mediante **commits descriptivos**, se revisa mediante *pull requests* y se relaciona con tareas específicas del equipo de desarrollo.

11. Capacitación y Comunicación

Usuarios finales
Para asegurar que los usuarios aprovechen al máximo TaskMaster Pro, se implementarán las siguientes acciones de formación y acompañamiento:

Talleres presenciales y virtuales
Sesiones de 1 hora en las que se muestran los flujos de trabajo clave (crear proyectos, asignar tareas, registrar tiempos) y se resuelven dudas en tiempo real.

Videos tutoriales breves
Clips de 3–5 minutos disponibles en la plataforma que muestran paso a paso funcionalidades específicas (por ejemplo, cómo comentar en una tarea o subir un archivo a la nube).

Manuales de usuario actualizados
Documentos PDF y guías interactivas en línea con capturas de pantalla, ejemplos prácticos y atajos de uso que cubren cada módulo de la herramienta.

Ejercicios prácticos guiados
Actividades diseñadas para que cada aprendiz realice en su equipo un conjunto de tareas predeterminadas (por ejemplo, crear un proyecto de prueba y asignar tareas entre compañeros).

Encuestas de retroalimentación
Formularios cortos al finalizar cada módulo para evaluar la comprensión, recoger sugerencias de mejora y medir la satisfacción de los participantes.

Equipos técnicos

El personal encargado de la administración y soporte del sistema recibirá una formación técnica especializada para garantizar el correcto funcionamiento de la plataforma:

1. Desarrolladores

- Capacitación en el uso y consumo de APIs del sistema.
- Explicación del flujo de trabajo con herramientas de CI/CD para automatizar pruebas y despliegues.
- Procedimientos de rollback en caso de fallos durante actualizaciones.
- Estructura del código y convenciones de desarrollo para facilitar mantenimiento y escalabilidad.

2. Personal de soporte

- Guía para atención a usuarios: solución de errores comunes, recuperación de contraseñas, carga de archivos, etc.
- Acceso y monitoreo de logs para el análisis de errores técnicos.
- Protocolo de escalamiento de incidentes a desarrollo o infraestructura.

3. Infraestructura tecnológica

- Configuración y administración de entornos (desarrollo, staging y producción).
- Implementación de medidas de seguridad (control de accesos, backups

automáticos, cifrado de datos).

- Uso de herramientas de monitoreo del sistema y rendimiento del servidor.
- Plan de recuperación ante desastres (caídas de sistema, pérdida de datos, ciberataques).

Canal de comunicación

Se establecerá un canal técnico unificado (como correo, chat corporativo o sistema de tickets) para facilitar:

- La resolución de dudas técnicas.
- La gestión de incidencias.
- La actualización de documentación técnica en tiempo real.

12. Post-migración

Una vez finalizada la migración del sistema, se implementarán medidas de control para garantizar la estabilidad, el correcto funcionamiento del software y la detección temprana de posibles fallos.

Monitoreo intensivo (primeras 24–72 horas)

Durante este periodo crítico, el equipo técnico realizará una supervisión constante del sistema en producción para:

- Verificar todos los módulos (proyectos, tareas, tiempos, usuarios, comentarios, notificaciones, etc.) funcionen correctamente.
- Evaluar el rendimiento del sistema (tiempos de carga, consumo de recursos, estabilidad del servidor).
- Comprobar la correcta integración con servicios en la nube (Google Drive, OneDrive, etc.).
- Confirmar que los accesos de usuarios y roles están habilitados según lo definido.

Registro de errores o inconsistencias

Cualquier error detectado será documentado en un reporte de incidentes que incluirá:

- Fecha y hora del incidente.

- Módulo afectado.
- Descripción del fallo o comportamiento inesperado.
- Usuario afectado.
- Capturas de pantalla o mensajes de error.
- Acción tomada por el equipo técnico**.**