# Plataforma de gestión de tareas para equipos.

## 1. Fase de Análisis

### Recolección de Requisitos:

* + Entrevistas con los usuarios clave (gerentes, empleados).
  + Identificación de funcionalidades necesarias: creación de tareas, asignación de tareas, seguimiento de progreso, notificaciones automáticas.
  + Definición de requisitos no funcionales: rendimiento, seguridad, escalabilidad.

### Documentación de Requisitos:

* + Creación del Documento de Especificación de Requisitos (SRS) que incluye todos los requisitos funcionales y no funcionales.
  + Revisión y aprobación del SRS con las personas interesadas.

### Análisis del Sistema:

* + Definición de los casos de uso para cada funcionalidad.

## 2. Fase de Diseño

### Diseño de Arquitectura:

* + Elección de la arquitectura basada en microservicios para permitir la escalabilidad y la modularidad.
  + Diseño de componentes: Servicio de gestión de usuarios, Servicio de gestión de tareas, Servicio de notificaciones.

### Diseño de la Base de Datos:

* + Modelo entidad-relación (ERD) que incluye tablas como Usuarios, Tareas, Proyectos, Comentarios.
  + Normalización para evitar redundancias.

### Diseño de la Interfaz de Usuario (UI):

* + Creación de wireframes para la interfaz: panel de control, lista de tareas, formulario de creación de tareas.
  + Diseño de la experiencia del usuario asegurando una navegación intuitiva.

### Modelado:

* + Diagramas UML: Diagrama de clases, diagrama de secuencia.
  + Identificación de interacciones entre módulos.

## 3. Fase de Implementación

### Desarrollo Frontend:

* + Uso de React.js para la creación de la interfaz de usuario interactiva.
  + Implementación de componentes reutilizables como botones, formularios, listas dinámicas.

### Desarrollo Backend:

* + Uso de Node.js y Express para construir la API RESTful.
  + Implementación de endpoints: POST /tasks, GET /tasks/:id, PUT /tasks/:id, DELETE /tasks/:id.

### Integración de la Base de Datos:

* + Conexión del backend con una base de datos PostgreSQL.
  + Implementación de ORM (Object-Relational Mapping)

### Integración Continua (CI):

* + Configuración de pipelines en Jenkins para pruebas automáticas y despliegue continuo.

## 4. Fase de Pruebas

### Pruebas Unitarias:

* + Creación de pruebas para funciones individuales en el backend y frontend utilizando Mocha y Jest.

### Pruebas de Integración:

* + Verificación de la interacción entre módulos (por ejemplo, cómo el sistema de notificaciones interactúa con la gestión de tareas).

### Pruebas de Aceptación del Usuario (UAT):

* + Realización de pruebas con usuarios reales para verificar que el sistema cumple con los requisitos definidos.

### Pruebas de Rendimiento:

* + Uso de herramientas como JMeter para simular la carga y medir el rendimiento bajo diferentes escenarios.

## 6. Fase de Implantación

### Preparación del Entorno:

* + Configuración del entorno de producción en AWS.

### Despliegue:

* + Implementación del software en el entorno de producción utilizando Docker y Kubernetes para la gestión de contenedores.

### Migración de Datos:

* + Si existiera un sistema anterior, migración de los datos al nuevo sistema.

### Capacitación:

* + Formación de los usuarios clave en el uso de la nueva plataforma mediante sesiones de capacitación y manuales.

## 7. Fase de Mantenimiento

### Mantenimiento Correctivo:

* + Resolución de bugs detectados después del despliegue.

### Mantenimiento Adaptativo:

* + Implementación de nuevas funcionalidades solicitadas por los usuarios, como la integración con herramientas de terceros (Slack, Google Calendar).

## 8. Evaluación Post-Implantación

### Revisión del Proyecto:

* + Análisis del éxito del proyecto, medido por la satisfacción del usuario y el cumplimiento de los plazos y presupuestos.

### Lecciones Aprendidas:

* + Documentación de lo que funcionó bien y lo que podría mejorarse en futuros proyectos.