Documento de formulación del proyecto.

Yolanda Gonzalez Herrera (100129489)

Santiago León Duran (100151578)

Jhon Sebastian Rodriguez Contreras (100159126)

Jeison Orlando baquero Gutierrez (100130741)

Facultad de Ingeniería

Corporación Universitaria Iberoamericana

Proyecto de Software

Tatiana Cabrera

2025

Fase de Planificación.

1. Contextualización de la necesidad.

En entornos laborales y académicos, la gestión eficiente de tareas en equipo es fundamental para alcanzar objetivos comunes. Las herramientas tradicionales de listas de tareas no siempre permiten una colaboración fluida ni una visualización clara del progreso. Por ello, se requiere una aplicación que facilite la asignación de tareas, seguimiento en tiempo real y comunicación integrada para mejorar la productividad y coordinación del equipo.

2. Planteamiento del problema.

En la actualidad, tanto a nivel personal como en entornos laborales, la gestión eficiente del tiempo y de las actividades se ha convertido en un desafío constante. Muchas personas y equipos enfrentan dificultades para organizar sus tareas, lo que genera problemas como:

- **Pérdida de información** al anotar tareas en diferentes medios (papel, notas en el celular, correos electrónicos).
- **Falta de seguimiento** en las actividades, ocasionando retrasos en la entrega de proyectos.
- **Dificultad para priorizar** tareas urgentes frente a tareas secundarias.

- **Escasa colaboración** entre miembros de un equipo, ya que no existen herramientas centralizadas para asignar, delegar o monitorear responsabilidades.
- **Baja productividad**, debido a la desorganización y ausencia de recordatorios efectivos.

Estas limitaciones afectan tanto a estudiantes, trabajadores independientes como a equipos de empresas, generando retrasos, duplicidad de esfuerzos y pérdida de control sobre los objetivos planteados.

Por lo tanto, se requiere una **aplicación de gestor de tareas** que permita:

- Centralizar la información de las actividades.
- Facilitar la asignación de responsabilidades.
- Priorizar y dar seguimiento a las tareas.
- Mejorar la comunicación y productividad de los usuarios.

3. Alcance del proyecto. (restricciones, criterios de aceptación del proyecto).

- Registro e inicio de sesión. (Autenticación de Google).
- Funcionalidades: Creación y asignación de tareas, tablero Kanban.
- Visualización del progreso individual y grupal.
- Asignación de tareas a usuarios específicos.
- Filtros y búsqueda de tareas por estado, prioridad o usuario.

- Definición de título, descripción, fecha límite y prioridad.
- Actualización de estado (pendiente, en progreso, completada).

4. Estructura del Desglose. (EDT)

DT – Estructura de Desglose de Trabajo

1. Inicio del proyecto

1.1 Definición del alcance

1.2 Identificación de requerimientos

1.3 Elaboración del plan del proyecto

2. Análisis y diseño

2.1 Levantamiento de requerimientos funcionales y no funcionales

2.2 Diseño de la arquitectura del sistema

2.3 Diseño de la base de datos

2.4 Diseño de la interfaz de usuario (mockups, prototipos)

3. Desarrollo

3.1 Configuración del entorno de desarrollo

3.2 Implementación del backend (API REST / lógica de negocio)

3.3 Implementación del frontend (interfaz de usuario)

3.4 Integración frontend-backend

3.5 Implementación de autenticación y seguridad

3.6 Gestión de notificaciones y recordatorios

4. Pruebas

4.1 Pruebas unitarias

4.2 Pruebas de integración

4.3 Pruebas de usabilidad

4.4 Corrección de errores

5. Despliegue

5.1 Configuración del entorno en la nube (ej. GCP, AWS o Azure)

5.2 Implementación de CI/CD

5.3 Lanzamiento de la aplicación

6. Documentación y capacitación

6.1 Documentación técnica (arquitectura, APIs, base de datos)

6.2 Manual de usuario

6.3 Capacitación a usuarios finales

7. Cierre del proyecto

7.1 Evaluación de resultados

7.2 Retroalimentación de usuarios

7.3 Plan de mejoras futuras

5. Objetivos (General y específicos)

Objetivos Generales.

Desarrollar una aplicación web de gestión de tareas que permita a los usuarios organizar, priorizar y dar seguimiento a sus actividades de manera eficiente, fomentando la productividad individual y la colaboración en equipos de trabajo mediante una interfaz amigable, segura y accesible.

Objetivos Específicos.

- Diseñar y desarrollar un módulo de gestión de tareas que permita crear, editar, eliminar y actualizar el estado de las actividades según su prioridad y fecha límite.
- Desarrollar una interfaz de usuario intuitiva y responsiva, que garantice facilidad de uso en distintos dispositivos y entornos de trabajo.
- Asegurar la escalabilidad y seguridad del sistema, implementando buenas prácticas de desarrollo, almacenamiento de datos.
- **Implementar un sistema de autenticación de usuarios** que garantice el acceso seguro a la aplicación mediante registro, inicio de sesión con Google.

6. Metodología ágil seleccionada (https://n9.cl/7z1xoa)

Control de versiones y desarrollo colaborativo

- **GitHub / GitLab** → Repositorios de código, control de versiones y gestión de ramas.
- **Gestión de proyectos y tareas**
- **Trello** → Tableros Kanban para organizar tareas por columnas (Pendiente, En progreso, Hecho).

Comunicación y colaboración

- **Microsoft Teams** → Videollamadas, chat y almacenamiento colaborativo.
- **Google Workspace** → Documentos, hojas de cálculo y calendarios compartidos.

7. **Justificación alcance corto, medio, largo), descripciones soluciones al problema, respuesta a los stakeholders.**

Justificación del proyecto.

El desarrollo de una aplicación de gestión de tareas responde a la necesidad de organizar, priorizar y dar seguimiento a las actividades de usuarios individuales y equipos de trabajo, permitiendo aumentar la productividad, mejorar la comunicación y reducir la pérdida de información.

Actualmente, la falta de una herramienta centralizada genera retrasos, duplicidad de esfuerzos y baja eficiencia. Con este proyecto se busca ofrecer una solución accesible, escalable y adaptable a distintos contextos (académico, profesional o empresarial).

Corto plazo (primeros 6 meses)

- Desarrollar una versión mínima viable (MVP) con funciones básicas: registro de usuarios, creación/edición de tareas, estados (pendiente, en progreso, completada).
- Implementar un sistema seguro de autenticación.
- Permitir la gestión individual de tareas.

Largo plazo (más de 12 meses)

- Integrar la aplicación con calendarios externos (Google Calendar, Outlook).
- Incorporar aplicación móvil nativa para Android y iOS.
- Añadir inteligencia artificial para sugerencias de priorización y predicción de tiempos de entrega.
- Expandir el sistema hacia entornos empresariales con gestión de proyectos avanzada.

8. Mapa de Stakeholders y clasificación de los mismos.

Stakeholders

Stakeholder	Rol	Nivel de Influencia	Nivel de Interés	Estrategia de Gestión
Equipos de Desarrollo	Usuarios principales	Alto	Alto	Involucramiento continuo, feedback constante, priorización de sus necesidades
Líderes de Proyecto	Supervisores	Alto	Alto	Reuniones regulares, acceso prioritario a nuevas funcionalidades, soporte dedicado
Inversores	Financiamiento	Medio	Medio	Reportes periódicos de progreso,

				demostraciones de avances, gestión de expectativas
Departamento Legal	Cumplimiento normativo	Medio	Medio	Consultas regulares, revisiones de cumplimiento, documentación exhaustiva

Stakeholder	Rol	Nivel de Influencia	Nivel de Interés	Estrategia de Gestión
Equipos Pequeños	Usuarios ocasionales	Bajo	Alto	Encuestas de satisfacción, canales de feedback, planes accesibles
Soporte Técnico	Atención al usuario	Bajo	Alto	Capacitación adecuada, documentación clara, herramientas de diagnóstico
Auditores Externos	Evaluación de seguridad	Bajo	Bajo	Mantener documentación actualizada, cumplir con estándares, transparencia limitada
Competidores	Otras empresas del sector	Bajo	Bajo	Monitoreo del mercado, diferenciación

				clara, protección de propiedad intelectual

9. Matriz de riesgos.

ID	Riesgo	Probabilidad	Impacto	Nivel de Riesgo	Plan de Mitigación
R1	Retrasos en el desarrollo por cambios en los requerimientos	Media	Alto	Alto	Definir requerimientos claros desde el inicio y aplicar metodologías ágiles para

					manejar cambios.
R2	Fallas de seguridad en la autenticación de usuarios	Baja	Alto	Medio-Alto	Implementar cifrado de contraseñas, autenticación JWT y pruebas de seguridad.
R3	Baja adopción por parte de los usuarios finales	Media	Medio	Medio	Realizar pruebas de usabilidad y encuestas piloto para mejorar la experiencia del usuario.
R4	Problemas de disponibilidad del sistema en la nube	Baja	Alto	Medio	Usar servicios cloud con SLA garantizado y configurar monitoreo constante.

R5	Sobrecarga del sistema por aumento inesperado de usuarios	Media	Alto	Alto	Diseñar una arquitectura escalable y realizar pruebas de carga.
R6	Pérdida de datos por fallos en la base de datos	Baja	Alto	Medio-Alto	Implementar backups automáticos y replicación de datos.
R7	Retrasos en la entrega por falta de experiencia técnica del equipo	Media	Medio	Medio	Capacitación del equipo y revisión continua con mentorías o code reviews.
R8	Costos mayores a lo presupuestado	Baja	Medio	Bajo-Medio	Establecer un control financiero con reportes periódicos.

R9	Errores en la integración frontend-backend	Media	Medio	Medio	Definir contratos claros en los endpoints de la API y realizar pruebas de integración continua.
R10	Dificultad para la adopción de metodologías ágiles en el equipo	Media	Medio	Medio	Capacitación en Scrum/Kanban y seguimiento con retrospectivas.
R11	Incompatibilidad entre navegadores o dispositivos	Alta	Medio	Medio-Alto	Realizar pruebas cross-browser y diseño responsive desde el inicio.
R12	Poca claridad en roles y	Media	Alto	Alto	Definir un RACI

	responsabilidades dentro del equipo				(responsables, aprobadores, consultados, informados) y mantener comunicación continua.

10. Presupuesto.

Detalle por Fases (4 meses)

Fase	Actividades	% presupuesto	Valor (COP)
Mes 1 – Instalación & Configuración	SonarQube, Jenkins, Snyk, ZAP, setup infraestructura	20%	$12,800,000
Mes 2 – Integración	Pipelines CI/CD, pruebas automáticas, integración con repositorio	25%	$16,000,000
Mes 3 – Ejecución & Ajustes	Ejecución de análisis, pruebas de seguridad, tuning herramienta	30%	$19,200,000
Mes 4 – Validación & Entrega	Reportes finales, documentación, capacitación interna	25%	$16,000,000
Total		100%	**$64,000,000**

Distribución por Programador

- $48,000,000 en 4 programadores durante 4 meses.
- Equivale a $12,000,000 por programador en todo el proyecto.
- Es decir, $3,000,000 mensuales aprox. por programador.

11. Levantamiento de información (herramientas y conclusiones)

1. SonarQube

Tipo: Análisis estático de código (SAST).

Características: Detecta bugs, vulnerabilidades, code smells y deuda técnica.

Ventajas: Open-source en su versión Community, soporta múltiples lenguajes, integración sencilla con pipelines CI/CD.

Desventajas: Configuración inicial compleja, requiere servidores con recursos adecuados.

Aplicación: Ideal para detectar problemas de seguridad y calidad en el código fuente antes de compilar.

2. OWASP ZAP

Tipo: Análisis dinámico de seguridad (DAST).

Características: Escáner especializado en vulnerabilidades de aplicaciones web.

Ventajas: Gratuito, mantenido por la comunidad OWASP, interfaz intuitiva.

Desventajas: Puede generar falsos positivos y requiere experiencia en ciberseguridad para interpretar resultados.

Aplicación: Detecta vulnerabilidades de seguridad OWASP Top 10 (XSS, SQLi, CSRF, etc.) durante pruebas dinámicas.

3. Snyk

Tipo: Análisis de dependencias y contenedores.

Características: Escaneo en tiempo real de librerías de terceros, imágenes Docker y configuraciones.

Ventajas: Base de datos de vulnerabilidades muy actualizada, integración directa con repositorios GitHub/GitLab.

Desventajas: El plan gratuito es limitado y los costos de la versión empresarial son altos.

Aplicación: Permite prevenir despliegues con librerías inseguras o configuraciones mal implementadas.

4. Jenkins

Tipo: Integración y despliegue continuo (CI/CD).

Características: Automatiza la compilación, ejecución de pruebas y despliegues en diferentes entornos.

Ventajas: Open-source, gran cantidad de plugin, adaptable a distintos proyectos.

Desventajas: Puede ser complejo de mantener en proyectos grandes y con muchas integraciones.

Aplicación: Orquesta herramientas como SonarQube, Snyk y Selenium dentro del pipeline de desarrollo.

5. Selenium

Tipo: Automatización de pruebas funcionales.

Características: Simula interacciones de usuario en navegadores, compatible con múltiples lenguajes.

Ventajas: Open-source, muy flexible, amplia comunidad de soporte.

Desventajas: Requiere mantenimiento constante de scripts, limitado para pruebas de Backend.

Aplicación: Automatiza pruebas de interfaz gráfica (UI) para validar la funcionalidad del sistema antes de producción.

📌 Conclusiones del Levantamiento de Herramientas

El análisis realizado muestra que cada herramienta cumple un rol específico dentro del ciclo de vida del software:

SonarQube y Snyk permiten detectar vulnerabilidades de manera temprana en el código y en las dependencias.

OWASP ZAP aporta seguridad en la etapa de pruebas dinámicas, evaluando la aplicación en ejecución.

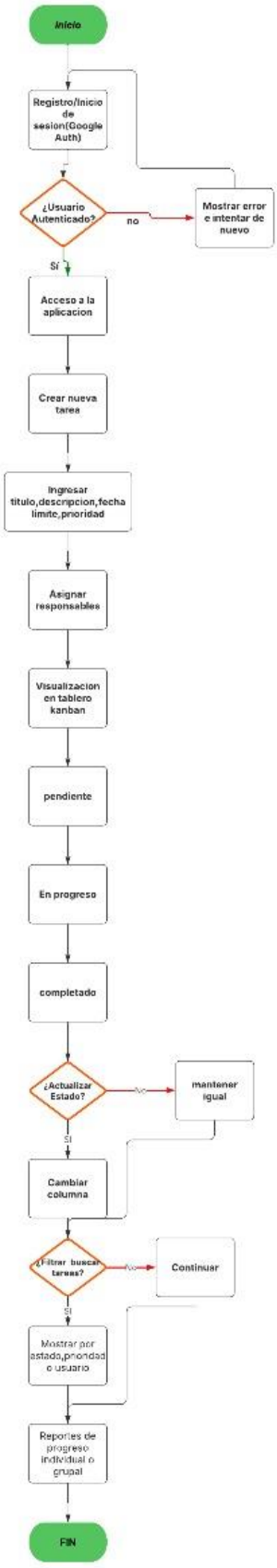
Jenkins y Selenium garantizan la automatización y la continuidad del proceso, evitando que las pruebas dependan únicamente de la intervención manual.

En conjunto, estas herramientas ofrecen una cobertura integral de calidad y seguridad, desde la codificación hasta el despliegue. Además, la mayoría cuenta con versiones gratuitas u open-source, lo que facilita su adopción en proyectos académicos y empresariales sin representar un alto costo.

12. Diagrama de flujo de solución selecciona.

Link diagrama de flujo (https://lucid.app/lucidchart/cd6052ec-e03e-4620-ad47-951309399339/edit?viewport_loc=277%2C443%2C1218%2C506%2C0_0&invitationId=inv_1987536e-fb51-4d4a-9965-267ab221240d)

Diagrama de flujo



Historias de Usuario en el tablero de la metodología Ágil seleccionada. Link (https://trello.com/invite/b/68d59cb7461e85b65a97f558/ATTI1329032e17cca957fadf00003137851317D2A48E/proyecto-de-software)

Autenticación y Acceso

1. Como usuario, quiero iniciar sesión en el sistema usando mi cuenta personal de Google para acceder de forma rápida y segura sin crear una nueva contraseña.
2. Como usuario, quiero cerrar sesión desde el sistema para proteger mi información cuando ya no necesite usar la aplicación.
3. Como administrador, quiero asegurar que solo los usuarios autenticados con Google puedan acceder al sistema para evitar accesos no autorizados.
4. Como administrador, quiero definir roles (administrador, miembro de equipo, observador) tras la autenticación para controlar qué acciones puede realizar cada usuario.
5. Como usuario, quiero que mi sesión permanezca activa mientras trabajo para no tener que iniciar sesión constantemente.
6. Como administrador, quiero poder revocar el acceso de un usuario autenticado para mantener la seguridad en caso de que alguien ya no deba usar la aplicación.

Gestión de tareas

7. Como usuario, quiero **crear** tareas con título, descripción, fecha límite y prioridad para organizar mis actividades de forma clara.
8. Como miembro de un equipo, quiero **asignar** tareas a usuarios específicos para distribuir responsabilidades y evitar confusión.

9. Como usuario, quiero **cambiar el estado** de mis tareas (**pendiente, en progreso, completada**) para reflejar el avance de mi trabajo.
10. Como usuario, quiero visualizar mis tareas en un tablero Kanban para tener una vista general de mi progreso.
11. Como miembro de un equipo, quiero ver el progreso grupal para evaluar el avance del proyecto en conjunto.
12. Como usuario, quiero **filtrar y buscar tareas** por estado, prioridad o usuario para encontrar fácilmente la información que necesito.
13. Como usuario, quiero **comentar dentro de las tareas** para mejorar la comunicación y resolver dudas sin depender de otras herramientas.
14. Como usuario, quiero **adjuntar documentos** a las tareas para centralizar la información necesaria para cumplirlas.
15. Como usuario, quiero **consultar el historial** de cambios de una tarea para saber quién hizo modificaciones y en qué momento.

13. Definición de RQF-RQNF manejar un formato para la documentación.

Requisitos Funcionales. (RF)

ID	Requerimiento	Descripción
RF01	**Análisis de calidad del código**	El sistema deberá permitir realizar análisis de calidad del código fuente para detectar errores y malas prácticas.
RF02	**Detección de vulnerabilidades de seguridad**	El sistema deberá identificar posibles vulnerabilidades de seguridad en la aplicación.
RF03	**Ejecución de pruebas automáticas**	El sistema deberá ejecutar pruebas

		automáticas que validen el correcto funcionamiento de las principales funciones del software.
RF04	**Generación de reportes básicos**	El sistema deberá generar reportes básicos con los resultados de los análisis y pruebas realizados.
RF05	Impedir paso a producción con fallas críticas	El sistema deberá impedir el paso a producción de versiones que presenten fallas críticas.

Requerimientos No funcionales. (RNF)

ID	Requerimiento	Descripción
RNF01	**Facilidad de uso**	El sistema deberá ser fácil de utilizar por el equipo de desarrollo, con una interfaz clara y comprensible.
RNF02	**Integración con repositorios de control de versiones**	El sistema deberá integrarse de manera sencilla con repositorios

		de control de versiones como GitHub o GitLab.
RNF03	**Disponibilidad continua**	El sistema deberá estar disponible de forma continua durante el ciclo de desarrollo del proyecto.
RNF04	**Claridad en los reportes**	El sistema deberá presentar reportes con información clara y comprensible para los programadores.
RNF05	**Escalabilidad**	El sistema deberá ser escalable, de modo que pueda adaptarse a futuros proyectos sin grandes modificaciones.