DOCUMENTACION DEL SISTEMA DE GESTIÓN DE TAREAS

|  |  |
| --- | --- |
| Nombre del proyecto: | Task Manager |
| Fecha: | 04 DE SEPTIEMBRE DEL 2024 |
| Versión: | 1.0 |
| Creado por: | Oscar Guillermo Sierra Lozano, Yeferson Esmid Heredia Perdomo, Karen Johana Caicedo Arias, Jhon Sebastián Molina Fierro y Nicolas Obregón Rojas |

Historial de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Autor | Descripción de la versión | Fecha de realización |
| 1.0 | Yeferson Heredia | Creación del modelado de mockup de la navegación en la página del sistema | 04/ 09 /2024 |
| 1.2 | Oscar Guillermo | Lanzamiento de las cualidades básicas y descrecimiento de los requerimientos | 08 / 09 / 2024 |
| 1.3 | Karen Caicedo | Descripción del sistema y limitación a los dispositivos que se piensa dar soporte del sistema general | 10/ 09 / 2024 |

Historial de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| Aprobador | Versión aprobada | Firma | Fecha |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Historial de aprobaciones | | | |
| Revisor | Versión revisada | Firma | Fecha |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Índice 3](#_Toc181777632)

[1.1. Alcance del producto 5](#_Toc181777633)

[1.2. Valor del producto 5](#_Toc181777634)

[a. Público objetivo 6](#_Toc181777635)

[b. Descripción general 6](#_Toc181777636)

[Requisitos no funcionales 11](#_Toc181777637)

# Índice



1. Introducción
   1. Alcance del producto
   2. Valor del producto
   3. Público objetivo
   4. Uso previsto
   5. Descripción general

2 Requisitos funcionales

1. Requisitos de la interfaz externa
   1. Requisitos de la interfaz de usuarios
   2. Requisitos de la interfaz de hardware
   3. Requisitos de la interfaz de software
2. Requisitos no funcionales
   1. Seguridad
   2. Capacidad
   3. Compatibilidad
   4. Confiabilidad
   5. Escalabilidad
   6. Mantenibilidad
   7. Facilidad de uso
   8. Otros requisitos no funcionales

5 Definiciones y acrónimos

1. Introducción

Llevar documentado el manejo del proyecto con dichas actualizaciones para buscar su mejora con el equipo de trabajo implementando dichas las sugerencias con lo que se recomiendo por el equipo y el administrador del proyecto.

## Alcance del producto

* Gestionar tareas.
* Gestionar trabajos.
* Manejo y control de datos para los trabajos creados.
* Control de tiempo respecto a entrega de trabajos.

## Valor del producto

1. **Datos:**

* **Horas al día por programador**: 8 horas
* **Semanas de trabajo por programador**: 16 semanas
* **Número de programadores**: 5
* **Valor de la hora**: 8.125 COP

1. **Total de horas por programador:**

El tiempo total trabajado por cada programador se calcula como:

1. **Total de horas trabajadas por el equipo:**

Para un grupo de 5 programadores, el total de horas trabajadas por el equipo sería:

1. **Costo total en COP:**
2. **Tiempo estimado de entrega:**

El proyecto tomará un total de **16 semanas** para completarse, ya que cada programador trabajará 8 horas a la semana durante las 16 semanas.

## Público objetivo

* Estudiantes.
* Grupos Empresariales.
* Instituciones educativas.
* Aulas virtuales.
* Grupos de marketing.
* Programadores.
  1. Uso previsto
* Gestión de tareas.
* Control de actividades.
* Control entrega.
* Seguimiento de cumplimiento de trabajos.

## Descripción general

Este software tiene como propósito hacer una gestión de tareas para sus usuarios que consiste en poder crear tareas y visualizarlas. Tiene la función de generar una lista donde organiza las tareas mostrando arriba las que más cerca este a vencer el plazo de entrega. Además, el sistema ofrecerá reportes y análisis sobre el progreso de las tareas con una gráfica en la que representa el tiempo restante para la entrega de la misma.

1. Requisitos funcionales

**Requisitos de diseño:**

* Diseño de la interfaz de usuario (UI).
* Arquitectura del sistema.
* Diseño de base de datos.
* Diseño de la interfaz de Programación (API).

**Requisitos Gráficos:**

* Paleta de colores.
* Iconografía.
* Imágenes y Gráficos.
* Espaciado y Márgenes.
* Animaciones y Transiciones.

**Requisitos de Sistema Operativo:**

* Compatibilidad con sistemas operativos de escritorio (Windows, macOS, Linux).
* Compatibilidad con sistemas operativos móviles (IOS, Android).
* Compatibilidad con navegadores web.
* Requisitos de red.

**Limitaciones del Producto:**

* Escalabilidad limitada.
* Dependencia de conexión a internet.
* Compatibilidad con dispositivos antiguos.

1. Requisitos de la interfaz externa
   1. Requisitos de

Describe la lógica detrás de las interacciones entre los usuarios y el software (diseños de pantalla, guías de estilo, etc.).

la interfaz de usuario

**1. Lista de tareas:**

* **Usuario:** Ingresa a la sección Inicio.
* **Backend:** Obtiene las tareas de la base de datos.
* **Resultado:** Muestra en una lista en la interfaz de todas las tareas que se han creado hasta el momento.

**2. Creación y Gestión de tareas:**

* **Usuario:** Crea, edita o elimina tareas desde la interfaz.
* **Backend:** Procesa los datos enviados, valida y guarda las tareas en la base de datos.
* **Resultado:** Creación de tareas reflejada en tiempo real en la interfaz.

**3. Prioridad de tareas:**

* **Usuario:** Ingresa a la sección de prioridad de tareas.
* **Backend:** Toma las tareas de la base de datos y por medio de la fecha de plazo se organiza para que se muestren hasta arriba las tareas próximas a vencer.
* **Resultado:** Lista de tareas organizada en prioridad de entrega en la interfaz.

**4. Eliminación de Tareas:**

* **Usuario:** Solicita eliminar una tarea.
* **Backend:** Borra la tarea en la base de datos tras hacer confirmación.
* **Resultado:** La tarea desaparece de la lista.

**5. Gráfico:**

* **Usuario:** Ingresa a la sección de gráfico.
* **Backend:** Toma las tareas de la base de datos y se inserta en un gráfico tomando como referencia la fecha de entrega y muestra cuantos días restan para que se cumpla dicho plazo.
* **Resultado:** Gráfica en la interfaz que muestra los días restantes de entrega para cada tarea.

**6. Notificaciones:**

* **Usuario:** Recibe notificaciones acerca de las tareas pendientes próximas a vencer.
* **Backend:** Envía notificaciones y actualizaciones en tiempo real.
* **Resultado:** Los usuarios reciben notificaciones.
  1. Requisitos de

Incluye las conexiones entre tu producto y otros componentes de software, como el marco de front-end y back-end,

las bibliotecas, etc.

la interfaz de software

* **Compatibilidad con navegadores web (para versiones basadas en web):**

Cualquier navegador en su versión más reciente y estable ya sean Google Chrome, Edge, Mozilla Firefox, Safari para una mejor compatibilidad en los sistemas operativos de Windows, macOS y Linux.

* **Requisitos de red:**

Conectividad: Conexión a internet de alta velocidad para garantizar una buena sincronización de datos y uso de servicios en la nube.

Protocolos de red: Soporte para HTTP/HTTPS, y WebSocket si se requiere si se requieren comunicaciones en tiempo real.

Configuraciones de firewall: Permitir conexiones salientes de puertos especificos utilizados por la aplicación, como el puerto 443 para HTPPS.

# Requisitos no funcionales



4

* 1. Seguridad

Incluye las normas de privacidad y protección de datos que deben cumplirse.

Al momento de la creación de los usuarios tiene se le otorga un rol que conlleve a unas limitaciones del sistema que maneja. Se limita el manejo de los usuarios y los datos enlazados a ellos mismos para mantener la privacidad.

* 1. Capacidad

Describe la demanda de almacenamiento actual y futura de tu software.

El sistema asegura una alta compatibilidad con los navegadores web más recientes y populares, garantizando un rendimiento óptimo en múltiples plataformas, incluyendo Windows, macOS, Linux, iOS y Android. Hemos definido una capacidad de red que exige una conexión de alta velocidad, soportando protocolos esenciales como HTTP/HTTPS y WebSocket para comunicaciones en tiempo real. Además, hemos priorizado la seguridad de las conexiones a través de la configuración de firewalls para conexiones seguras por el puerto 443. A nivel de diseño gráfico, el sistema maneja una interfaz de usuario intuitiva y visualmente atractiva, con una paleta de colores cuidadosamente seleccionada, iconografía clara y animaciones suaves que no afectan el rendimiento. Si bien la escalabilidad está limitada por la infraestructura, hemos asegurado que el sistema pueda manejar adecuadamente un volumen moderado de usuarios y que dependa de una conexión estable para su funcionamiento eficiente.

* 1. Compatibilidad

Enumera los requisitos mínimos de hardware para el software.

**Compatibilidad para sistemas operativos de escritorio:**

* **Windows:**

Requisitos mínimos: Procesador de 64 bits, 4 GB de RAM, 2 GB de espacio en disco.

* **MacOS:**

Requisitos mínimos: Procesador Intel o Apple M1, 4 GB de RAM, 2 GB en disco.

* **Linux:**

Requisitos mínimos: Procesador de 64 bits, 4 GB de RAM, 2 GB de espacio en disco.

**Compatibilidad para sistemas operativos móviles:**

* **IOS:**

Requisitos mínimos: iPhone XS o superior, 4 GB de RAM.

* **Android:**

Requisitos mínimos: Dispositivo con procesador Qualcomm Snapdragon 765G o equivalente, 4 GB de RAM.

* 1. Confiabilidad

Calcula cuál es el tiempo de fallo crítico esperado de tu producto con relación a un uso normal.

Dentro de la práctica prototipada para la evaluación del proyecto se utilizó un tiempo de 144 horas (6 días) dentro de un uso normal, teniendo un tiempo promedio entre fallos de λ=0.0069.

Tasa de fallos

Ahora, la tasa de media de tiempo entre fallos, analizando los tiempos y la cantidad de errores se llegó a la conclusión de 144.92 representa el tiempo promedio que esperas que pase del proyecto en uso normal antes de que ocurra un fallo crítico.

En el tiempo utilizado (2 semanas) se estima que el sistema se use en 12 horas durante 7 días a la semana. Dándonos un tiempo de 84 horas/semana, entonces sabiendo esto, se concluye que se puede esperar un fallo aproximadamente cada 1.72 semanas.

* 1. Escalabilidad

Calcula la cantidad máxima de trabajo con la que tu software puede funcionar según lo esperado.

Un software se puede evaluar y optimizar mediante un proceso de métricas y fórmulas. Primero, se mide el rendimiento, que se calcula de la siguiente manera:

Esta fórmula ayuda a determinar cuántas transacciones puede procesar el sistema en un período de tiempo determinado y proporciona una base para comprender su capacidad actual. Si medimos cuántas tareas pueden crearse, asignarse, actualizarse o completarse en un periodo determinado, Podríamos gestionar 500 tareas por minuto de forma eficiente, el rendimiento es de 500 tareas/minuto. Esto nos ayuda a saber la cantidad de trabajo que puede procesar la aplicación sin que se degrade el rendimiento.

que indica qué porcentaje de recursos del sistema (como CPU y memoria) se están utilizando. Si el servidor está usando el 80% de la CPU y la memoria cuando hay 100 usuarios activos gestionando tareas, significa que está cerca de los límites de capacidad y podría ser necesario añadir más recursos o servidores, para asegurar que el sistema no colapse bajo cargas mayores.

ya que un aumento en el tiempo de respuesta a medida que aumenta la carga de trabajo puede ser una señal de que el sistema necesita más capacidad. Si los usuarios notan que la aplicación se vuelve más lenta a medida que se incrementa el número de tareas o usuarios, esto indica que el sistema necesita mejoras de escalabilidad. Tenemos que Mantener un bajo tiempo de respuesta para ofrecer una experiencia fluida.

se puede utilizar para predecir el comportamiento del sistema bajo diferentes condiciones de carga y planificar mejoras. Esto nos permitirá estimar cuándo necesitaremos más servidores o aumentar la capacidad del sistema para garantizar que funcione correctamente con una carga mayor.

* 1. Mantenibilidad

Describe cómo se debe usar la integración continua para implementar funciones o reparar errores rápidamente.

El proyecto mejora significativamente mediante el uso de la Integración Continua (CI), ya que las soluciones se pueden identificar, corregir e implementar de manera rápida y eficiente. Siempre que hay un cambio en el código, CI ejecuta pruebas automatizadas que ayudan a detectar errores tempranamente. Cuando se detecta un problema, los desarrolladores pueden solucionarlo y volver a ejecutar el proceso de prueba comprometiéndose a garantizar que no se produzcan nuevos errores. Además, CI permite la implementación automática de la corrección en entornos de producción, acelerando el tiempo de entrega de la solución sin interrupciones innecesarias. En caso de errores críticos, el sistema puede volver automáticamente a la última versión estable, minimizando el impacto en los usuarios. De esta forma, la CI no sólo facilita la corrección rápida de errores, sino que también evita la acumulación de errores, dando como resultado un mantenimiento ágil, seguro y sostenible en el tiempo.

* 1. Facilidad de uso

Describe qué tan fácil resultará el uso del software a los usuarios finales.

El software se plantea de manera intuitiva para el usuario, ya sea a nivel de menús, gráficos, imágenes y tablas para tener una experiencia fluida y agradable.

* 1. Otro

Enumera cualquier otro requisito no funcional.

**Rendimiento Bajo Conexiones Lentas**

Es necesario optimizar el sistema para que funcione eficientemente incluso en malas condiciones de conexión, que son comunes en algunas regiones de Colombia. Debe poder cargar sus funciones principales en redes con un ancho de banda mínimo de 1 Mbps sin afectar gravemente la experiencia del usuario. Además, es necesario implementar una función de modo sin conexión que permita a los usuarios trabajar en tareas y proyectos sin conexión y sincronizar datos cuando se restablezca la conectividad a Internet.

1. Definiciones y acrónimos

|  |  |
| --- | --- |
| **Requisitos funcionales** | Son los requerimientos que describen **qué debe hacer el sistema**. Especifican las funciones, tareas o comportamientos que el software debe ser capaz de realizar. Estos requisitos suelen responder a preguntas como: ¿qué hará el sistema? y ¿cómo interactuarán los usuarios con él?  **Ejemplo**: El sistema debe permitir a los usuarios crear, editar y eliminar tareas en el gestor de proyectos. |
|  |  |
| **Requisitos no funcionales** | Son características que definen **cómo** debe funcionar el sistema, pero no están directamente relacionados con las funcionalidades. Incluyen aspectos como rendimiento, escalabilidad, seguridad, usabilidad y tiempo de respuesta.  **Ejemplo**: El sistema debe tener un tiempo de respuesta inferior a 2 segundos bajo una carga máxima de 1000 usuarios concurrentes. |
|  |  |
| **Software** | Conjunto de programas, instrucciones y reglas que permiten el funcionamiento de una computadora o dispositivo para realizar tareas específicas. Puede incluir aplicaciones, sistemas operativos y herramientas de desarrollo.  **Ejemplo**: Un software de gestión de proyectos. |
|  |  |
| **Interfaz de usuario** | Es el espacio a través del cual un usuario interactúa con un sistema o aplicación. La interfaz incluye elementos gráficos, botones, menús y cualquier otro componente con el que el usuario puede interactuar.  **Ejemplo**: La pantalla principal del gestor de tareas donde los usuarios ven y organizan sus proyectos. |
|  |  |
| **Backend** | Es la parte "trasera" de una aplicación o sistema. Incluye la lógica del servidor, las bases de datos y la infraestructura que permite que el sistema funcione, pero no es visible para el usuario.  **Ejemplo**: El backend gestiona las peticiones de creación de tareas, las guarda en la base de datos y asegura que los datos sean correctos. |
|  |  |
| **Fronted** | Es la parte "frontal" de una aplicación que es visible e interactiva para el usuario. Incluye todo lo que el usuario ve y utiliza directamente, como botones, menús y formularios.  **Ejemplo**: El diseño del tablero de tareas con el que el usuario interactúa para crear y gestionar proyectos. |
|  |  |
| **Interfaz interna** | Es una interfaz diseñada para ser usada por el equipo interno de desarrollo, administración o soporte técnico, y no está disponible para los usuarios finales. Facilita la gestión del sistema y los recursos internos.  **Ejemplo**: Un panel de administración donde los administradores gestionan usuarios y permisos. |
|  |  |
| **Sistema Operativo** | El software base que gestiona el hardware de una computadora o dispositivo móvil y permite que los programas se ejecuten. Algunos ejemplos comunes son **Windows**, **Linux**, y **macOS**.  **Ejemplo**: El sistema operativo del servidor que aloja el backend del gestor de proyectos. |
|  |  |
| **Integración continua (ci)** | Es una práctica de desarrollo de software que implica la automatización de la integración del código de múltiples desarrolladores en un repositorio compartido. Las pruebas automáticas y la compilación se ejecutan de forma continua para asegurar que el código sea correcto.  **Ejemplo**: Usar herramientas de CI como Jenkins o Travis CI para ejecutar pruebas automáticas cada vez que se sube un nuevo código. |
|  |  |
| **Interfaz de Programación (API)** | Un conjunto de definiciones y protocolos que permite a diferentes programas y sistemas interactuar entre sí. Las APIs permiten que diferentes aplicaciones compartan datos o funcionalidades.  **Ejemplo**: Una API que permite a otras aplicaciones acceder a las tareas almacenadas en el gestor de proyectos. |
|  |  |
| **Hardware** | Es el conjunto de componentes físicos que forman una computadora o sistema, como el procesador, la memoria, y el disco duro. Todo el software requiere hardware para ejecutarse.  **Ejemplo**: El servidor físico donde se aloja la base de datos del gestor de proyectos. |
|  |  |
| **Requisitos Gráficos** | Son los requerimientos relacionados con el diseño visual y la representación gráfica de la interfaz de usuario, como resolución, colores, tipografía y animaciones.  **Ejemplo**: El sistema debe soportar una resolución mínima de 1280x720 píxeles y debe ser compatible con pantallas de alta definición. |