**Documentación del Proyecto: Sistema de Gestión de Tareas Avanzado**

**Descripción del Proyecto**

El proyecto consiste en desarrollar un Sistema de Gestión de Tareas Avanzado que permita a los usuarios organizar y administrar sus tareas de manera eficiente. El sistema estará disponible en plataformas web, Android y Windows.

**Estructura del Proyecto**

Proyecto\_Gestion\_Tareas/

│

├── 01\_Requerimientos/

│ ├── GoogleForms/

│ └── Meet/

│

├── 02\_Análisis\_y\_Diseño/

│ ├── Diagramas\_Flujos\_Casos/

│ │ └── Microsoft\_Visio/

│ ├── Diseño\_UI/

│ │ └── Adobe\_Photoshop/

│ └── Mockups\_Wireframes/

│ └── Adobe\_Dreamweaver/

│

├── 03\_Desarrollo/

│ ├── Frontend/

│ │ ├── HTML\_CSS/

│ │ ├── JavaScript/

│ │ └── Frameworks/

│ ├── Backend/

│ │ ├── PHP/

│ │ └── Frameworks/

│ └── Database/

│ ├── MySQL\_Workbench/

│ ├── Oracle/

│ └── MongoDB/

│

├── 04\_Pruebas\_y\_Validación/

│ ├── Unitarias/

│ ├── Integración/

│ └── Pruebas\_Unidad\_Negocio/

│

├── 05\_Seguridad\_y\_Pentesting/

│ ├── Kali\_Linux/

│ └── ...

│

├── 06\_Despliegue\_y\_Distribución/

│ ├── Local/

│ ├── Cloud/

│ └── Stores/

│

└── 07\_Control\_de\_Mantenimiento/

└── Microsoft\_Azure\_DevOps/

**Fases del Proyecto**

1. **Requerimientos:** Captura de requisitos del proyecto mediante Google Forms y reuniones en Google Meet.
2. **Análisis y Diseño:** Elaboración de diagramas de flujo y casos de uso en Microsoft Visio. Diseño de la interfaz de usuario utilizando Adobe Photoshop y creación de mockups y wireframes con Adobe Dreamweaver.
3. **Desarrollo:** Implementación del frontend, backend y base de datos utilizando las tecnologías especificadas.
4. **Pruebas y Validación:** Realización de pruebas unitarias, de integración y de negocio para garantizar la calidad del software.
5. **Seguridad y Pentesting:** Evaluación de la seguridad del sistema utilizando Kali Linux y técnicas de pentesting.
6. **Despliegue y Distribución:** Despliegue del sistema en entornos locales y en la nube, así como distribución en tiendas de aplicaciones.
7. **Control de Mantenimiento:** Gestión del mantenimiento del sistema utilizando Microsoft Azure DevOps.

**Conclusiones**

El proyecto se organiza en fases claramente definidas, cada una con sus respectivas tareas y entregables. Esto facilita la gestión y el seguimiento del progreso del proyecto, así como la colaboración entre los miembros del equipo.

Resultado de información

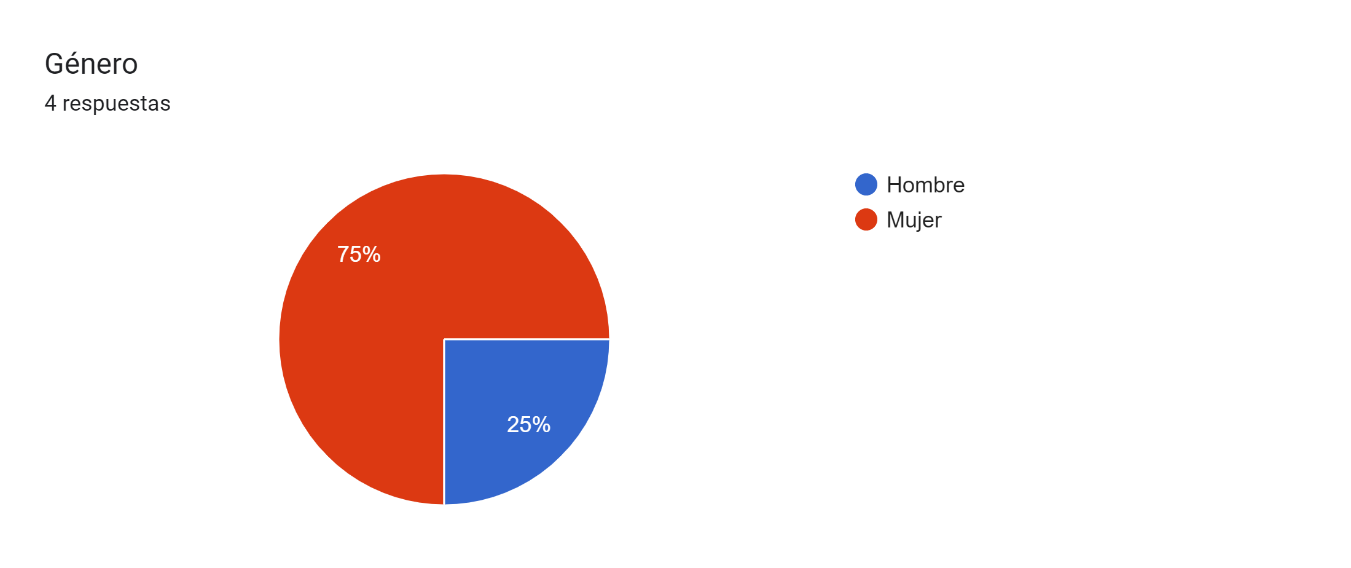
Edad (En número por favor)  
4 respuestas

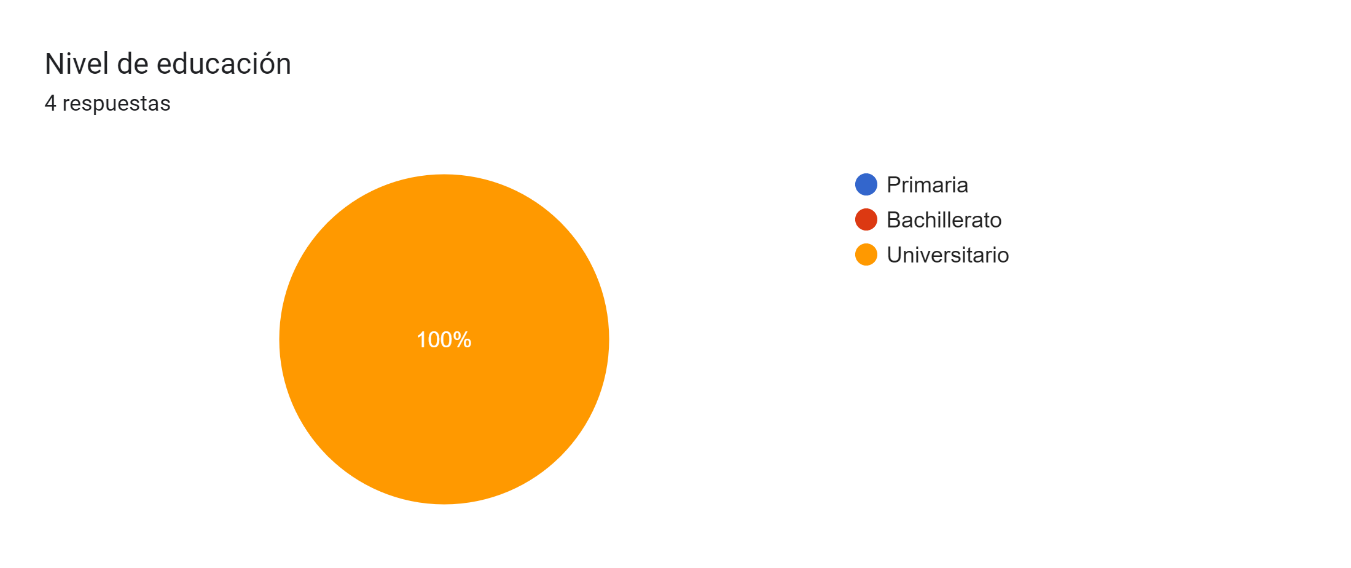
32

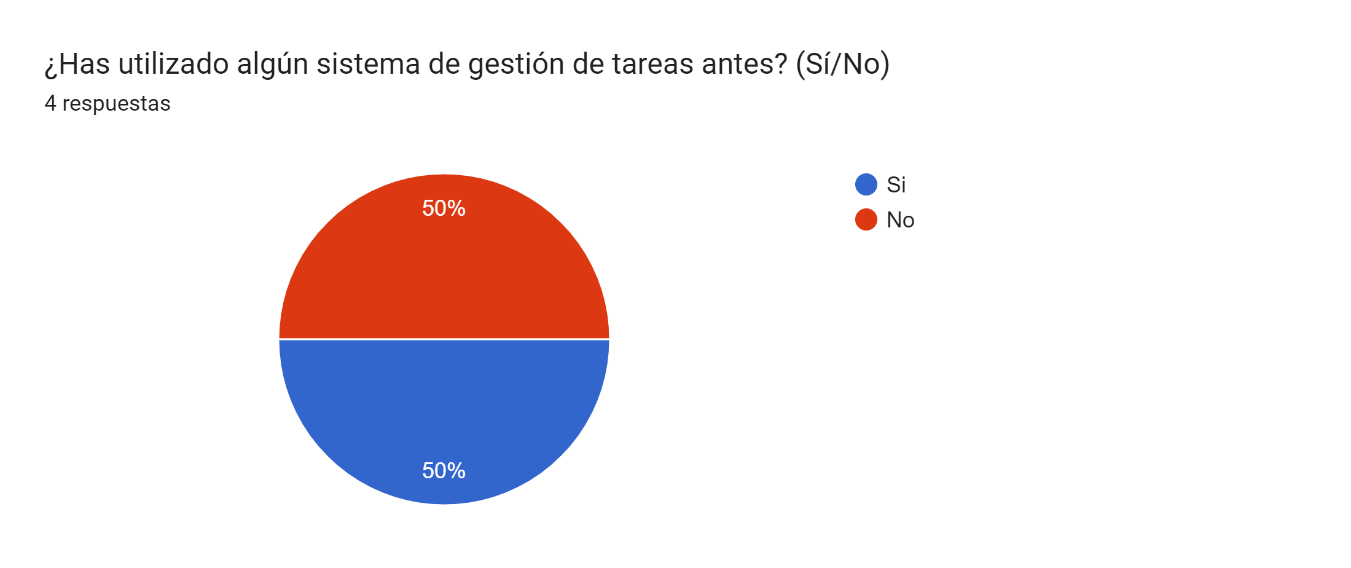
34

57

36







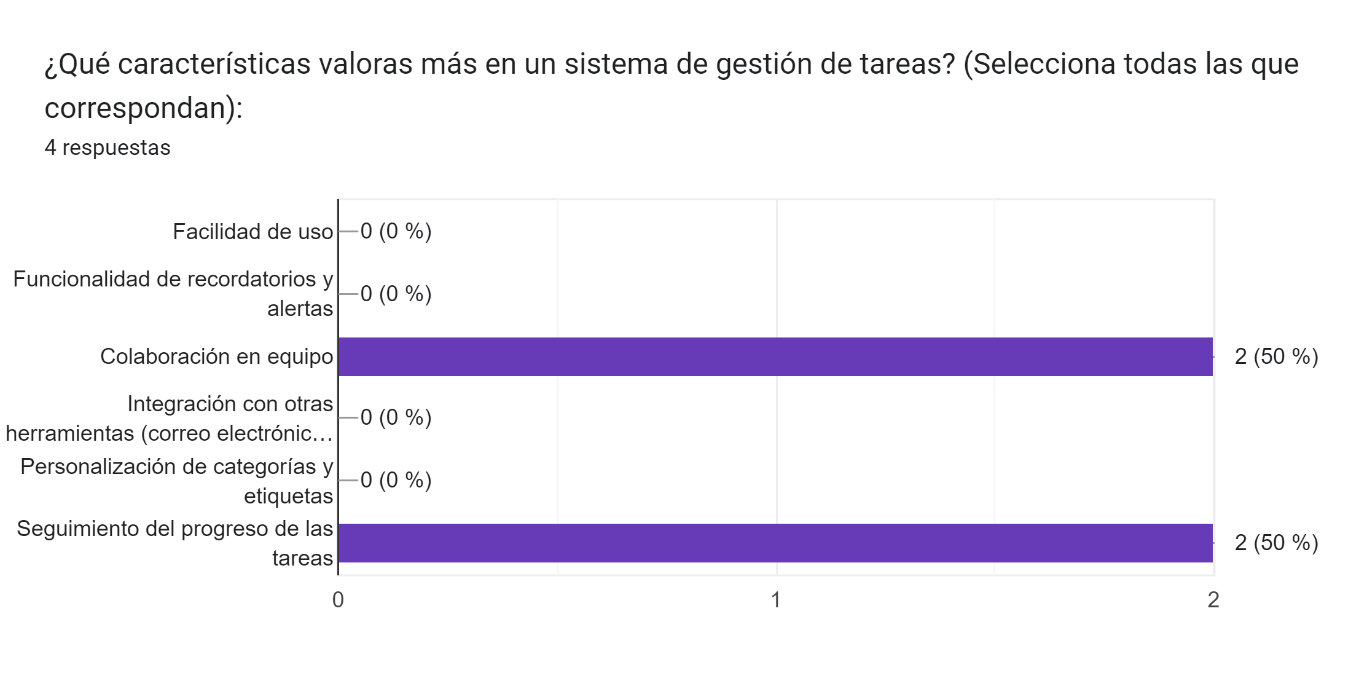
En caso afirmativo, ¿cuál(es)?  
4 respuestas

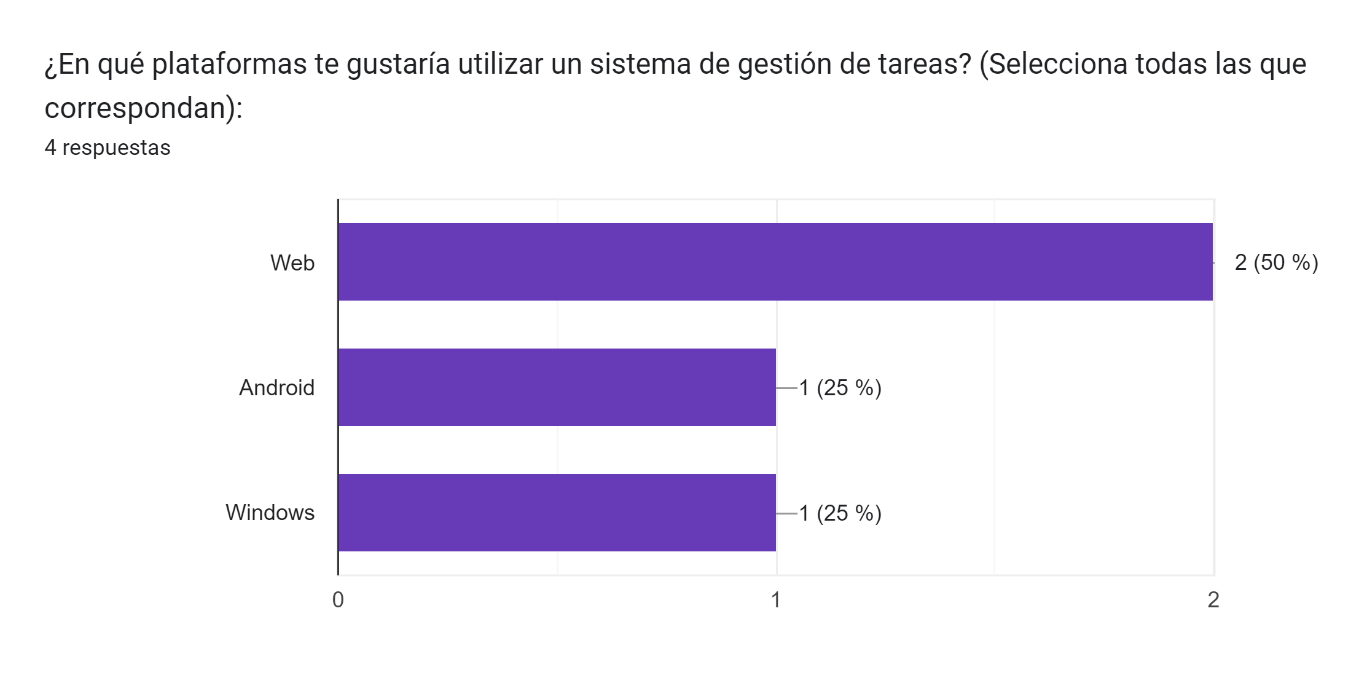
Si

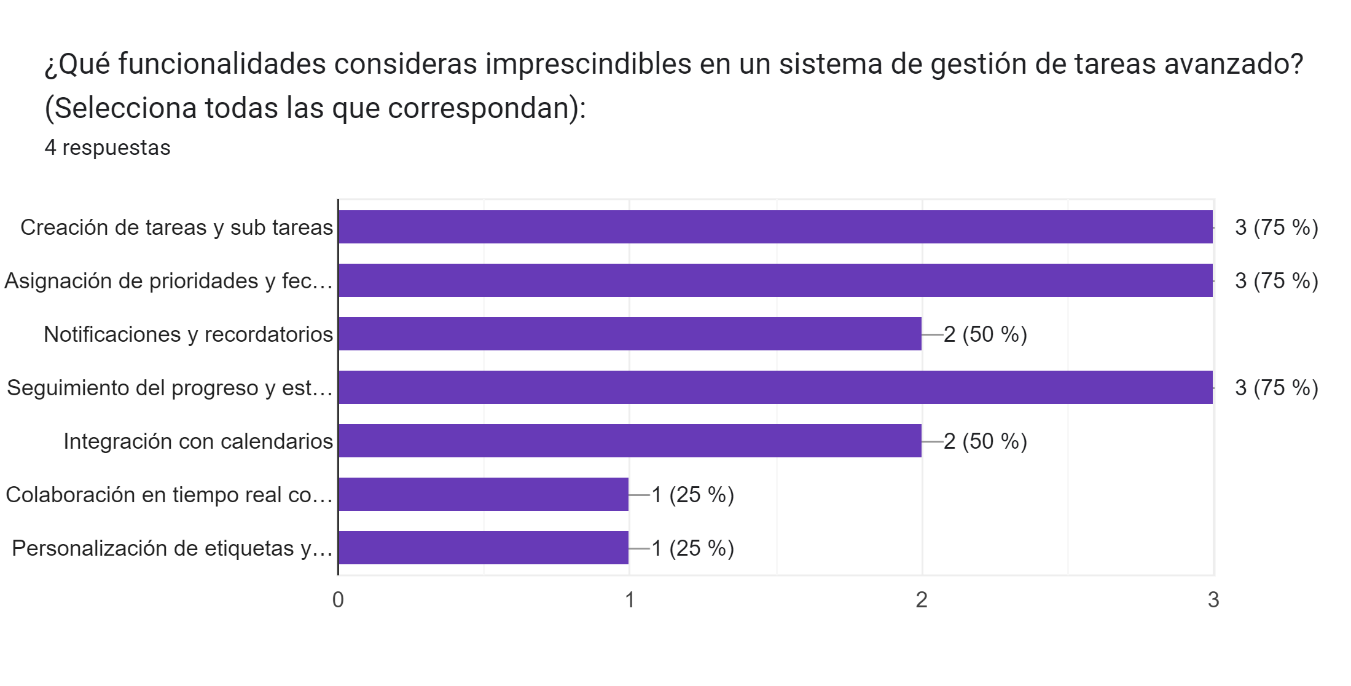
Trello

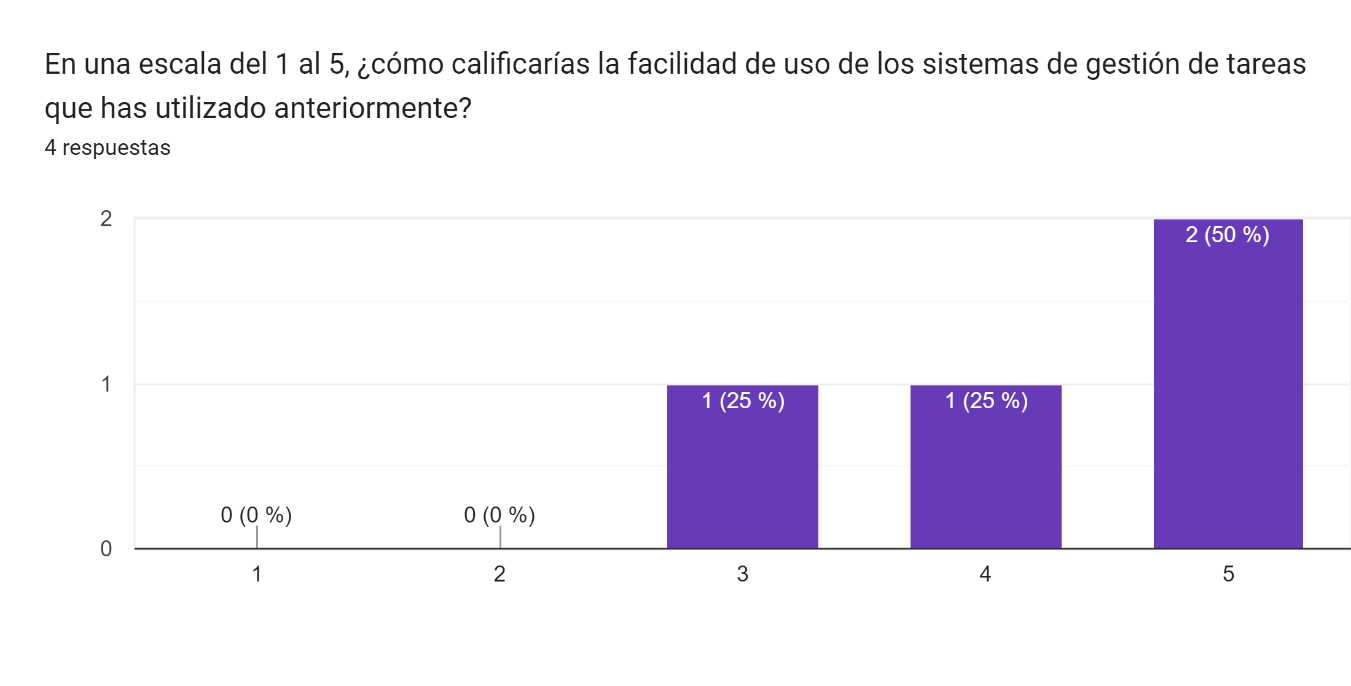
No

Google









¿Hay algo más que te gustaría añadir sobre tus necesidades o expectativas con respecto a un sistema de gestión de tareas avanzado?4 respuestas

Ninguna

Muy bueno

Tener en cuenta la discapacidad

Capacitar sobre web

Resultados:

1. **Demografía**:
   * Edad: La mayoría de los encuestados tienen entre 32 y 36 años.
   * Género: Hay una distribución equitativa entre hombres y mujeres en la muestra.
   * Nivel de educación: Todos los encuestados tienen educación universitaria.
2. **Experiencia con sistemas de gestión de tareas**:
   * La mitad de los encuestados ha utilizado sistemas de gestión de tareas antes.
3. **Plataformas preferidas**:
   * La mayoría prefiere utilizar un sistema de gestión de tareas en la web, seguido por Android y Windows.
4. **Funcionalidades consideradas imprescindibles**:
   * La creación de tareas y sub tareas, así como el seguimiento del progreso de las tareas, son las funcionalidades más importantes para los encuestados.
5. **Calificación de la facilidad de uso**:
   * La mayoría de los encuestados califican la facilidad de uso de los sistemas de gestión de tareas anteriores con una calificación de 3, 4 o 5 en una escala del 1 al 5.
6. **Comentarios adicionales**:
   * Algunos encuestados expresaron que el sistema debería tener en cuenta la accesibilidad para personas con discapacidad y ofrecer capacitación sobre su uso.

Requerimientos:

**Requerimientos No Funcionales:**

1. **Facilidad de Uso:**
   * El sistema debe ser intuitivo y fácil de usar, incluso para usuarios con poca experiencia tecnológica.
   * Debe tener una interfaz de usuario clara y bien organizada.
2. **Accesibilidad:**
   * El sistema debe ser accesible para personas con discapacidades, cumpliendo con los estándares de accesibilidad web.
   * Debe ofrecer opciones de navegación y uso que sean accesibles para todos los usuarios.
3. **Rendimiento:**
   * El sistema debe ser rápido y responder de manera eficiente, incluso con un gran volumen de datos y usuarios.
   * Debe tener tiempos de carga mínimos y proporcionar una experiencia fluida de usuario.
4. **Seguridad:**
   * Debe garantizar la seguridad de los datos de los usuarios, utilizando medidas de cifrado y protección de la información.
   * Debe tener controles de acceso para proteger la información confidencial y evitar el acceso no autorizado.
5. **Escalabilidad:**
   * El sistema debe ser escalable para poder crecer y adaptarse a medida que aumenta el número de usuarios y la cantidad de datos almacenados.
   * Debe poder manejar eficientemente un mayor volumen de usuarios y tareas sin comprometer el rendimiento.

**Configuración del Entorno de Desarrollo:**

1. **Herramientas y Recursos:** Asegúrate de que todos los miembros del equipo tengan acceso al entorno de desarrollo necesario, incluyendo herramientas, bibliotecas y recursos adicionales.
2. **Instalación y Configuración:** Ayuda a los miembros del equipo a instalar y configurar las herramientas necesarias en sus entornos de desarrollo.

Documentación final

# Documentación del Proyecto de Gestión de Tareas Avanzado

## 1. Introducción

### Descripción del Proyecto

El sistema de gestión de tareas avanzado es una aplicación web diseñada para facilitar la creación, administración y seguimiento de tareas. Proporciona a los usuarios una interfaz intuitiva para gestionar sus actividades diarias, establecer prioridades y realizar un seguimiento del progreso.

### Objetivos

* **Eficiencia:** Mejorar la eficiencia en la gestión de tareas personales y de equipo.
* **Organización:** Proporcionar una herramienta que ayude a los usuarios a mantenerse organizados.
* **Colaboración:** Facilitar la colaboración entre miembros de un equipo.

### Tecnologías Utilizadas

* **Frontend:** React, Bootstrap
* **Backend:** Node.js, Express.js
* **Base de Datos:** MySQL, MongoDB
* **Despliegue:** Servidor Ubuntu, Microsoft Azure
* **Control de Versiones:** GitHub
* **Pruebas:** Postman, SoapUI
* **Pentesting:** Kali Linux
* **IDE:** Visual Studio Code
* **Bibliotecas Adicionales:** Axios, React Bootstrap

## 2. Guía de Instalación

### Requisitos Previos

* Node.js y npm
* MySQL o MongoDB
* Git
* Servidor web (Apache o Nginx)
* Cuenta en Microsoft Azure para el despliegue en la nube

### Instrucciones de Instalación

#### Instalación en Local

1. **Clonar el repositorio:**

sh

Copy code

git clone https://github.com/tu-usuario/proyecto-gestion-tareas.git cd proyecto-gestion-tareas

1. **Instalar dependencias:**

sh

Copy code

npm install

1. **Configurar la base de datos:**
   * Crear una base de datos MySQL:

sql

Copy code

CREATE DATABASE empleados\_crud;

* + Configurar las credenciales de la base de datos en **backend/config.js**.

1. **Iniciar el servidor:**

sh

Copy code

cd backend node index.js

1. **Iniciar el frontend:**

sh

Copy code

cd frontend npm start

#### Despliegue en la Nube

1. **Configurar el entorno de Azure:**
   * Crear un recurso de aplicación en Microsoft Azure.
   * Configurar las variables de entorno y las credenciales de la base de datos en el portal de Azure.
2. **Desplegar la aplicación:**

sh

Copy code

git push azure master

#### Distribución en Tiendas (si aplica)

* **Google Play Store:** Sigue el proceso de publicación de aplicaciones en la Google Play Store.
* **Microsoft Store:** Sigue el proceso de publicación de aplicaciones en la Microsoft Store.

## 3. Manual de Usuario

### Introducción al Sistema

El sistema de gestión de tareas permite a los usuarios crear, administrar y monitorear tareas de manera eficiente. La interfaz es intuitiva y está diseñada para facilitar la navegación y el uso.

### Guía de Uso

#### Crear y Administrar Tareas

1. **Crear Tarea:**
   * Navegar a la sección "Crear Tarea".
   * Rellenar los campos obligatorios y hacer clic en "Guardar".
2. **Administrar Tareas:**
   * Ver la lista de tareas en la sección "Mis Tareas".
   * Editar o eliminar tareas según sea necesario.

#### Configuración de Usuario

1. **Perfil de Usuario:**
   * Acceder a la sección "Perfil".
   * Actualizar información personal y preferencias.

#### Reportes y Análisis

1. **Generar Reportes:**
   * Ir a la sección "Reportes".
   * Seleccionar los filtros deseados y generar el reporte.

## 4. Documentación del Código

### Estructura del Código

* **Frontend:**

java

Copy code

proyecto-gestion-tareas/ ├── public/ ├── src/ │ ├── components/ │ ├── pages/ │ ├── App.js │ ├── index.js └── package.json

* **Backend:**

go

Copy code

backend/ ├── controllers/ ├── models/ ├── routes/ ├── index.js └── package.json

### Comentarios en el Código

Asegúrate de que el código esté bien comentado, explicando claramente la lógica y el propósito de cada sección del código.

### Generación de Documentación Automática

Utiliza herramientas como JSDoc para generar documentación automática a partir de los comentarios en el código.

## 5. Guía de Desarrollo

### Configuración del Entorno de Desarrollo

1. **Instalar Visual Studio Code.**
2. **Instalar extensiones recomendadas:**
   * ESLint
   * Prettier
   * Live Server

### Estándares de Código

* **JavaScript:**
  + Seguir las convenciones de estilo de Airbnb.
  + Utilizar ESLint para mantener la consistencia del código.

### Guía para Contribuidores

1. **Hacer un fork del repositorio.**
2. **Crear una nueva rama para cada característica o corrección de errores.**
3. **Enviar pull requests con una descripción clara de los cambios realizados.**

## 6. Pruebas y Validación

### Estrategia de Pruebas

* **Pruebas Unitarias:** Pruebas individuales para cada componente.
* **Pruebas de Integración:** Asegurar que todos los componentes funcionen juntos.
* **Pruebas de Usuario:** Pruebas con usuarios reales para validar la usabilidad.

### Resultados de las Pruebas

Resumen de los resultados de las pruebas realizadas, destacando las áreas donde se encontraron y corrigieron errores.

### Cómo Ejecutar Pruebas

1. **Configurar el entorno de pruebas:**

sh

Copy code

npm install --save-dev jest

1. **Ejecutar las pruebas:**

sh

Copy code

npm test

## 7. Seguridad

### Medidas de Seguridad Implementadas

* **Autenticación y Autorización:** Utilización de JWT para manejar sesiones de usuario.
* **Validación de Entradas:** Validar todas las entradas del usuario para evitar inyecciones SQL y XSS.
* **Encriptación:** Encriptar contraseñas y datos sensibles.

### Guía de Pentesting

1. **Herramientas Utilizadas:** Kali Linux, OWASP ZAP.
2. **Pasos Realizados:**
   * Realizar escaneo de vulnerabilidades.
   * Intentar explotar vulnerabilidades encontradas.
   * Corregir vulnerabilidades y realizar re-escaneo.

## 8. Mantenimiento y Soporte

### Plan de Mantenimiento

* **Actualizaciones Periódicas:** Programar actualizaciones de seguridad y funcionalidad.
* **Monitoreo Continuo:** Utilizar herramientas de monitoreo para mantener la aplicación.

### Soporte Técnico

* **Canal de Soporte:** Proporcionar un canal de soporte para que los usuarios puedan reportar problemas y obtener ayuda.

## 9. Conclusiones

### Resumen del Proyecto

El sistema de gestión de tareas avanzado ha sido diseñado y desarrollado para mejorar la eficiencia y organización de los usuarios. La aplicación ha pasado por pruebas rigurosas y se ha optimizado para ofrecer una experiencia de usuario de alta calidad.

### Futuras Mejoras

* **Integración con otras herramientas de productividad.**
* **Aplicaciones móviles nativas para iOS y Android.**
* **Funciones de inteligencia artificial para recomendaciones y análisis de tareas.**

## Ejemplo de Estructura de Documentación en el Repositorio

css

Copy code

Proyecto\_Gestion\_Tareas/ ├── README.md ├── docs/ │ ├── introduccion.md │ ├── guia\_instalacion.md │ ├── manual\_usuario.md │ ├── documentacion\_codigo/ │ ├── guia\_desarrollo.md │ ├── pruebas\_validacion.md │ ├── seguridad.md │ └── mantenimiento\_soporte.md └── src/ └── (código fuente del proyecto)

## Pasos para Asegurarte de que Toda la Documentación esté Actualizada

1. **Revisa y Actualiza Documentos Existentes:**
   * Revisa cada documento y asegúrate de que la información esté actualizada y sea precisa.
2. **Completa cualquier Falta de Información:**
   * Añade cualquier información faltante que pueda ser útil para desarrolladores y usuarios.
3. **Genera Documentación Automática:**
   * Si usas herramientas de documentación automática, asegúrate de que estén configuradas y generen la documentación correcta.
4. **Realiza una Revisión por Pares:**
   * Pide a otros miembros del equipo que revisen la documentación para asegurarse de que esté completa y sea comprensible.
5. **Sube la Documentación al Repositorio:**
   * Asegúrate de que todos los documentos estén subidos al repositorio y sean accesibles desde la raíz del proyecto o en un directorio **docs**.
6. **Actualiza el Archivo README:**
   * Asegúrate de que el archivo **README.md** contenga enlaces a toda la documentación relevante y un resumen claro del proyecto.