Requerimientos, Organización y Soluciones

Software MyTask

Repositorio: GitHub:Proyecto-Gestionador-Tareas

Fecha de creación:

Viernes, 30 de 08 2024

Última modificación:

Domingo, 08 de 09 2024

Versión:

1.0

Estado:

Beta

Preparado por:

Nicolás Olivos Muñoz

Felipe Brauer Castillo

¿Cómo especificarías mejor el requerimiento?

Para especificar mejor los requerimientos del usuario, es importante documentar los requerimientos de manera adecuada y detallada para obtener una mejor comprensión de lo que se debe realizar. Esto se puede realizar mediante la escritura de historias de usuario junto con criterios de aceptación e idealmente validarlos con el cliente, ya que como desarrolladores debemos asegurar que los requerimientos reflejan correctamente las necesidades del cliente revisándolos hasta obtener una aprobación y un acuerdo.

A continuación, se detallan los requisitos del cliente junto con sus historias de usuario y criterios de aceptación:

1. **Gestión de Tareas:** Desarrollar un sistema que permita a los usuarios crear, consultar, actualizar y eliminar tareas a través de la línea de comandos. Cada tarea debe tener un título, una descripción, una fecha de vencimiento, y una etiqueta que facilite la organización (por ejemplo, "urgente", "trabajo", "personal", otros).

   Historia de usuario:

   Como usuario, **Quiero** crear, actualizar, consultar y eliminar tareas que contengan título, descripción, fecha de vencimiento y etiquetas representativas **Para** poder organizar mejor mis tiempos y recordar lo que debo realizar.

   Criterios de aceptación:

   Dado a que puedo consultar el estado de una tarea, **Cuando** complete una tarea o quiera modificar su contenido, **Entonces** tendré la posibilidad de actualizar la tarea en cualquier momento

2. **Filtrado y Búsqueda:** Incluir funcionalidades para que los usuarios puedan filtrar y buscar tareas en función de varios criterios, como la fecha de vencimiento, etiquetas o estado de la tarea (pendiente, en progreso, completada).

   Historia de usuario:

   Como usuario, **Quiero** poder filtrar y buscar las tareas según fecha, etiquetas o estado de la tarea **Para** ahorrarme tiempo y evitar consultar una por una.

   Criterios de aceptación:

   Dado a que puedo guardar tareas colocándoles fecha, etiqueta y estado, **Cuando** necesite encontrar una tarea específica, **Entonces** tendré la posibilidad de buscar la tarea deseada utilizando un filtro por título, fechas, etiqueta o estado.

   Dado a que puedo guardar tareas por fecha, **Cuando** necesite encontrar tareas en un rango de fechas específico, **Entonces** tendré la posibilidad de encontrar todas las tareas dentro de ese rango.

3. **Gestión de Estados de Tareas:** Implementar un sistema de gestión de estados para las tareas que permita a los usuarios marcar una tarea como "en progreso" o "completada". Las tareas completadas pueden ser archivadas y consultadas posteriormente.

   Historia de usuario:

   Como usuario, **Quiero** poder marcar mis tareas como “completadas” o en “progreso” **Para** obtener una mejor gestión y realizar consultas de manera más sencilla.

   Criterios de aceptación:

   Dado a que puedo marcar tareas con un estado como “completado”, **Cuando** necesite guardar el registro, **Entonces** tendré la posibilidad de archivar o eliminar la tarea.

   Dado a que al crear una tarea tiene estado en “pendiente”, **Cuando** termine de realizar una tarea, **Entonces** tendré la posibilidad de cambiar el estado a “completado”

4. **Autenticación:** Proteger el acceso al sistema con un mecanismo de autenticación mediante un nombre de usuario y contraseña.

   Historia de usuario:

   Como usuario, **Quiero** poder acceder al sistema de manera segura **Para** poder utilizar el gestionador de tareas y tener privacidad de las mismas.

   Criterios de aceptación:

   Dado a que debo registrarme en el sistema para acceder a él, **Cuando** intente ingresar con información errónea, **Entonces** el software no permitirá el ingreso y mostrará un mensaje de intento de inicio de sesión.

   Dado a que me registré en el sistema correctamente, **Cuando** alguien intente registrarse con mi usuario, **Entonces** el software no permitirá generar una nueva cuenta con el mismo usuario.

¿Cómo asegurarías que el programa cumpla el requerimiento?

Para asegurar que el programa cumpla con los requerimientos es importante evaluar el software revisando el código y principalmente realizando pruebas unitarias para cada una de las funcionalidades. Estas pruebas se deben realizar comparando el comportamiento del programa con los resultados esperados por el cliente o los criterios de aceptación, verificando que el software está funcionando correctamente y que estamos construyendo el producto correctamente.

Organización y el flujo de trabajo del proyecto.

El proyecto se organizó utilizando Slack y github, de la cual se realizó un flujo de trabajo similar a Git Flow, donde se fueron realizando varias ramas de Git, cada cual centrada o especificada en una funcionalidad específica del software, por ejemplo la rama “autenticacion” que posee la funcionalidad de login y registro de cuentas de usuarios con cifrado de contraseñas, la rama de “gestion-de-tareas” que posee las funcionalidades principales de las tareas, tales como añadir, eliminar, buscar, filtros, etc. Posteriormente, luego de revisar el código y hacer algunas pruebas, las ramas fueron integradas a la rama de desarrollo “develop” haciendo uso de pull request. Las revisiones de código se realizaron de forma individual por cada desarrollador haciendo pruebas y simulando casos de uso de un usuario, posterior a esto el desarrollador podía aprobar la decisión de integrar el código a la rama de desarrollo o rechazarla y continuar mejorando el código.

¿Por qué Git flow?

Se seleccionó un flujo de trabajo de tipo Git flow ya que es un flujo de trabajo ideal y bastante utilizado en los proyectos al mantener el código organizado con ramas separadas, evitando la inestabilidad en la rama principal del proyecto. Git flow nos permite y facilita el control de versiones, ayudándonos a tener una mejor organización y un mejor desarrollo en caso de errores. A pesar de que Git flow es mucho más utilizado en proyectos grandes o de desarrollos largos, es importante mantener las buenas prácticas y tener un código organizado, controlado y limpio.

Evidencia del Flujo de trabajo y configuración:

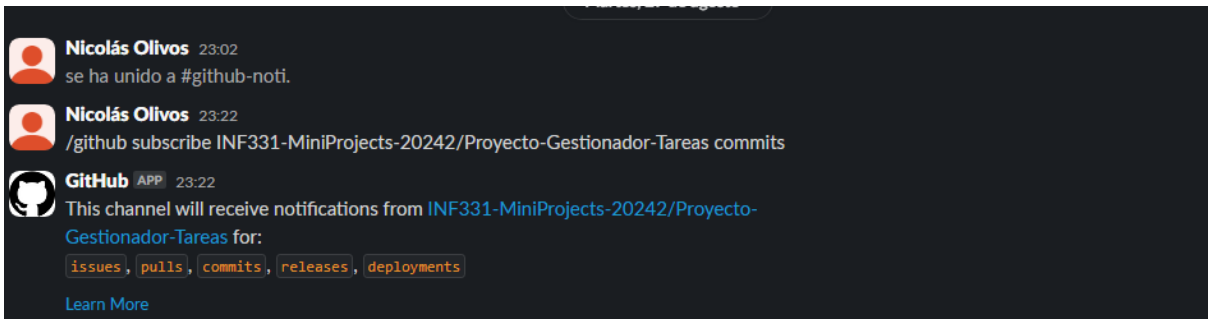

Imagen 1: Implementación de Github en Slack

Imagen 2: Implementación de Github en Slack, última configuración

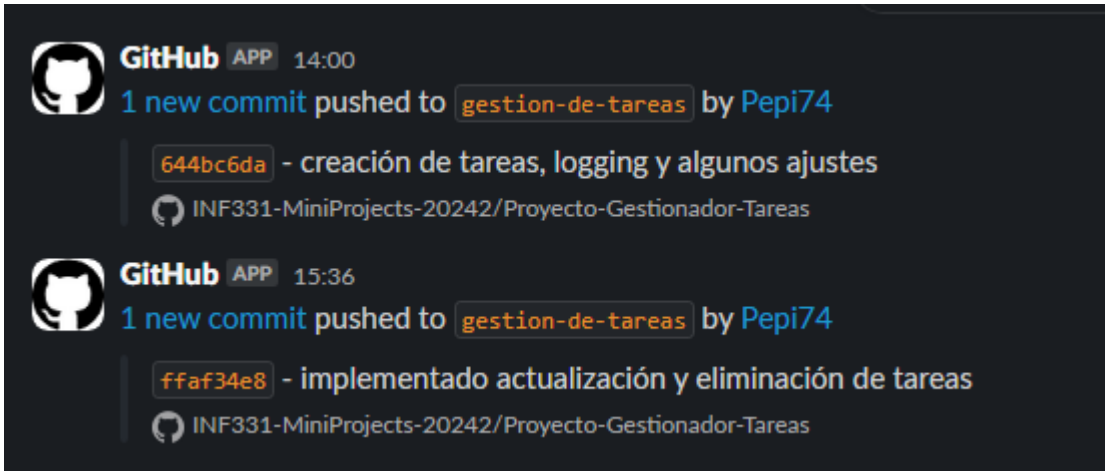

Imagen 3: Commits y push a la rama “gestión-de-tareas”

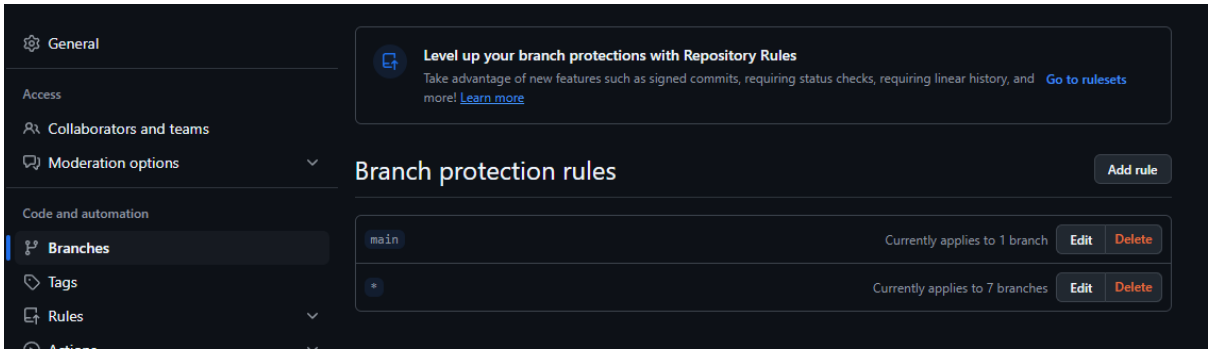

Imagen 4: Regla de protección para ramas

Imagen 5: Regla aplicada a todas las ramas del repositorio

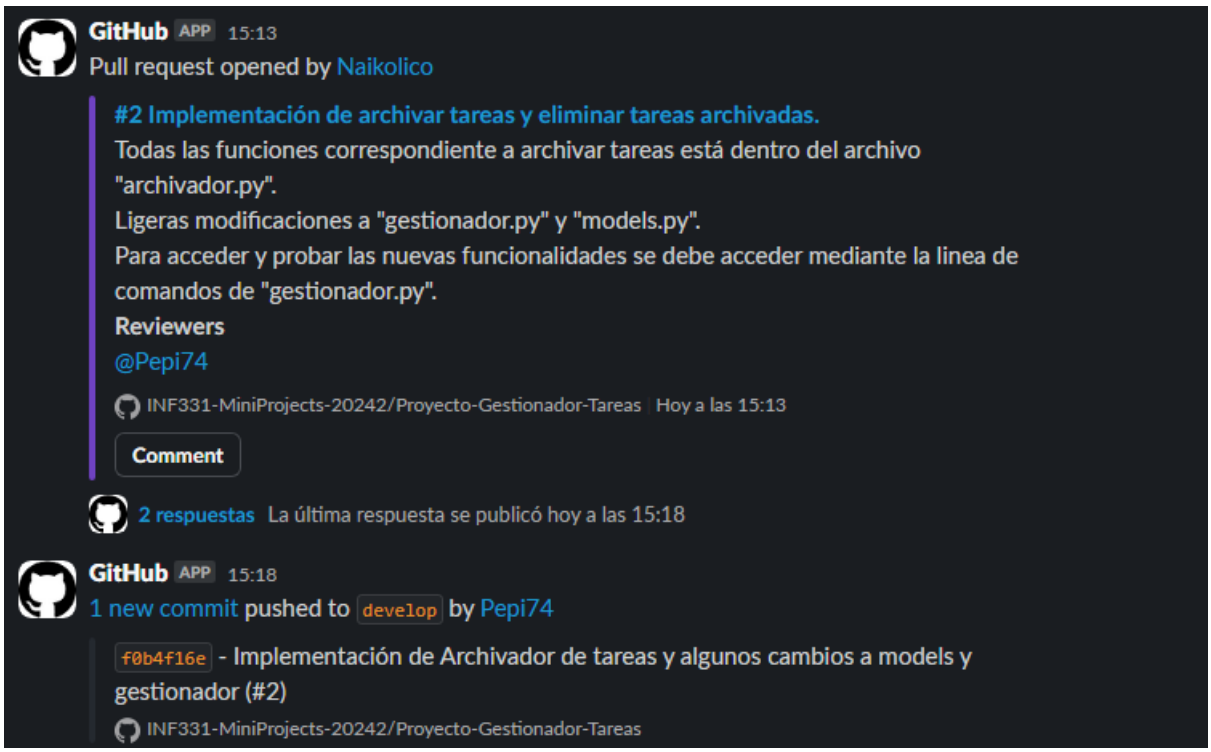

Imagen 6: Pull request de la rama “archivador” a la rama “develop”

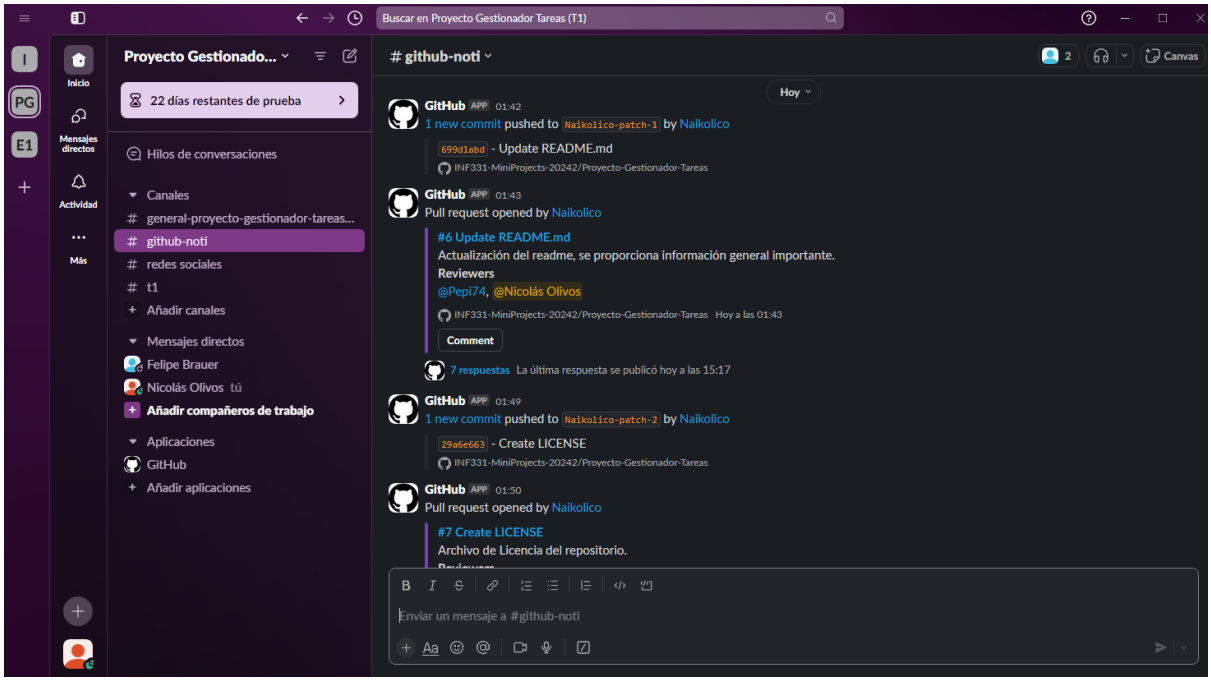

Imagen 7: Slack de la organización del proyecto con sus respectivos miembros y canales

Descripción de evidencias:

La **Imagen 1 y 2** muestran un chat dentro de la aplicación Slack con el bot de GitHub, donde se realizan las configuraciones necesarias para llevar a cabo el desarrollo del proyecto.

La **Imagen 3** muestra uno de los cuantos commits realizados a una de las ramas del repositorio, demostrando que GitHub y Slack están perfectamente configurados para cualquier commit hecho en el repositorio.

La **Imagen 4** muestra que el repositorio github posee una regla en la rama main que la protege de commits directos, por lo que es necesario que se realicen pull request con al menos la aprobación de 2 revisores como también se restringe la realización de commits directos a la rama principal.

La **Imagen 5** muestra específicamente una regla aplicable a todas las ramas, esta regla establece que todos los pull request que se realicen necesitará de al menos 1 revisor (Establecido de esta manera al ser un equipo de 2 personas) antes de que pueda ser fusionados. Asegurando la calidad del código y siguiendo buenas prácticas del trabajo.

La **Imagen 6** muestra una notificación de github sobre un pull request creado por un desarrollador, dado a las reglas creadas en el repositorio era necesario que el desarrollador restante aprobará la solicitud para realizar un merge, en este caso a la rama develop.

La **Imagen 7** muestra como Slack fue utilizado por los desarrolladores para organizarse y enterarse de todo lo que sucede dentro del repositorio gracias a la aplicación de GitHub instalada en el espacio de trabajo Slack.

Pruebas del software:

El equipo decidió llevar a cabo las pruebas del software de la siguiente manera:

En el primer ciclo de prueba, cada desarrollador realizó aproximadamente nueve pruebas enfocadas en componentes o conjunto de componentes que consideraban prioritarios para garantizar el correcto funcionamiento del software y satisfacer las necesidades del usuario. Estas pruebas se realizaron simulando escenarios de uso real del software.

Tras completar el primer ciclo de prueba, el equipo discutió los resultados de cada prueba para identificar posibles fallos o discrepancias con los resultados esperados, llegando a algún acuerdo de solución. Luego de la discusión, se corrigió cada componente afectado, dando comienzo al segundo ciclo de pruebas.

En el segundo ciclo de pruebas, se evaluaron tanto los componentes corregidos como también los componentes o funcionalidades que no fueron testeadas en el primer ciclo, para asegurar la calidad total del software.

Problemas encontrados y soluciones:

Los problemas encontrados luego de haberse realizado el primer ciclo de pruebas fueron los siguientes:

- Registro de Usuario sin input:

  El problema trata de que un usuario puede registrarse sin ningún problema a pesar de no haber colocado nada en ninguno de los campos necesarios, resultando en la creación de un usuario inválido.

  Para resolver este problema se modificó dentro del archivo “autenticador” la función “desplegar_registro”, la cual se encarga de desplegar por consola la interfaz de registro de usuarios y recibir los inputs de estos. Se implementó una validación donde el usuario debe escribir al menos 2 caracteres en los campos “Nombre”, “Usuario” y “Contraseña”, desplegando un mensaje de aviso cada vez que cometa un error. Esta solución fuerza al usuario a crear cuentas con al menos 2 caracteres, evitando la creación de cuentas sin información ni contraseña.

- Buscar tarea sin título “” (Vacío):

  El problema trata sobre la posibilidad de filtrar una tarea por título sin siquiera haber escrito uno, provocando que se muestren todas las tareas del usuario. Esto por un error lógico dentro del archivo “filtrador”, en específico en la función “filtrar_por_titulo”, la cual verificar por cada una de las tareas que posee el usuario

si el título contiene un carácter vacío (“”) por lo que todas las tareas del usuario cumplirían la condición.

Para resolver este problema se modificó la función main del archivo que contiene el input título que coloca el usuario antes de llamar la función “filtrar_por_titulo” para que detectara el input del usuario cuando estuviese vacío, desplegando un mensaje de aviso para que intente nuevamente escribir un título para filtrar. Esta solución fuerza al usuario a escribir un título de al menos un carácter en el campo correspondiente para llevar a cabo el filtro de tareas sin problemas.

- Crear tarea con título y descripción vacíos, pero con alguna etiqueta:

  El problema trata sobre la posibilidad de crear tareas sin contenido, es decir, si las entradas proporcionadas por el usuario están vacías sin tener en cuenta la etiqueta de la tarea, se estaría generando una tarea invalida.

  Para resolver este problema se modificó dentro del archivo “gestionador” la función “crear_tarea”, la cual se encarga de pedirle los inputs necesarios al usuario para la correcta creación de tareas. Se implementó una validación donde el usuario debe escribir al menos un carácter para el campo título como el de descripción, desplegando un mensaje cada vez que cometa un error. Esta solución fuerza al usuario a crear tareas con al menos 1 carácter en su título y descripción, evitando la creación de tareas sin información.