Git Collaboration

Guía de Gestión de Tareas y Trabajo en Equipo

GF

Grupo Flores

Versión 1.0 - Fernando Flores Junio 2026

1. Introducción

Cuando trabajamos solos es fácil recordar qué estamos haciendo. Sin embargo, cuando varias personas trabajan en el mismo proyecto, es necesario organizar las tareas para evitar trabajo duplicado, confusiones o cambios perdidos.

Para lograrlo utilizamos Github Projects, Issues y Pull Requests.

El objetivo de esta guía es mostrar el flujo básico de trabajo que debe seguir cualquier integrante de un proyecto.

2. ¿Qué es una Issue?

Una Issue representa una tarea específica dentro del proyecto.

Las Issues permiten saber qué se debe hacer, quién está trabajando en ello y cuándo la tarea puede considerarse terminada.

Ejemplos de Issues:

- Crear pantalla login
- Crear tabla usuarios
- Crear endpoint de productos
- Crear formulario para registrar productos

Una Issue debe representar una tarea concreta y facil de identificar

3. ¿Qué *NO* es una Issue?

Las siguientes tareas son demadias grandes y difíciles de controlar:

- Hacer el sistema
- Programar el backend
- Terminar el proyecto
- Crear toda la aplicación

Si una tarea parece demasiado grande, debe dividirse en varias issues más pequeñas. Mientras más específica sea una issue, más facil será trabajar en ella.

4. Github Projects

Github Projects es el tablero basico donde se organizan todas las tareas del proyecto. Cada tarea aparece como una tarjeta que puede moverse entre diferentes estados.

Estados recomendados:

> Recomiendo Github Projects para equipos pequeños e informales. Si el equipo crece mejor usar una plataforma como Jira

Backlog
Tareas pendientes que aún no serán desarrolladas

Ready
Tareas listas para comenzar

In Progress
Tareas que actualmente están siendo desarrolladas.

Review
Tareas terminadas que están esperando revisión

Done
Tareas finalizadas y aprobadas

4.1 Flujo de una tarea

Todas las tareas deben seguir el mismo proceso:

Issue → Crear Rama → Programar → Push → Pull Request → Merge → Issue Cerrada

Este flujo permite mantener organizado el proyecto y facilita el trabajo en equipo.

5. Tomar una tarea

Antes de comenzar a programar:

1. Abrir Github Projects
2. Elegir una Issue disponible
3. Asignarse la tarea

4. Mover la tarea a In Progress
5. Crear una rama para comenzar a trabajar

Una vez asignada la tarea, ningún otro integrante debe trabajar sobre ella al mismo tiempo.

5.1 Una tarea, una rama

Cada Issue debe tener su propia rama.

Ejemplo:

Issue: #15 Crear pantalla login

Rama: feat/15-login-screen

Esto permite mantener los cambios organizados y evitar mezclar diferentes funcionalidades en una misma rama.

No es recomendable utilizar una sola rama para varias tareas.

Incorrecto: feat/backend-completed

Correcto: feat/22-products-api
feat/31-db-tables

5.2 Trabajar en la tarea

Una vez creada la rama, se pueden realizar los cambios necesarios.

Cuando se complete una parte importante del trabajo, los cambios deben guardarse mediante commits.

```
git add .
git commit -m “Create login screen”
```

Los commits deben describir claramente qué se realizó.

Buenos ejemplos (Preferible sea en Ingles) :

- Crear pantalla login
- Agregar endpoint productos
- Corregir validación de usuario
- Actualizar estilos de mesas

5.3 Subir cambios al repositorio

Una vez realizados los commits, la rama debe enviarse a Github

```
git push origin feat/15-login-screen
```

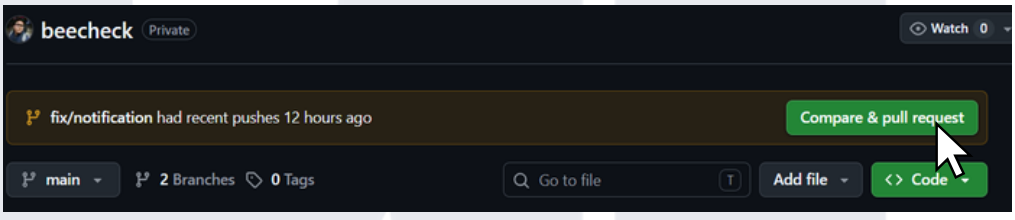
Esto permitirá crear posteriormente un Pull Request

6. ¿Qué es un Pull Request?

Un Pull Request es una solicitud para integrar una rama al proyecto principal.

En lugar de modificar directamente la rama principal, los cambios se envían mediante un Pull Request

Esto permite revisar el trabajo antes de incorporarlo al proyecto



6.1 Relacion entre Issue y Pull Request

Cada Pull Request debe estar relacionado con una Issue.

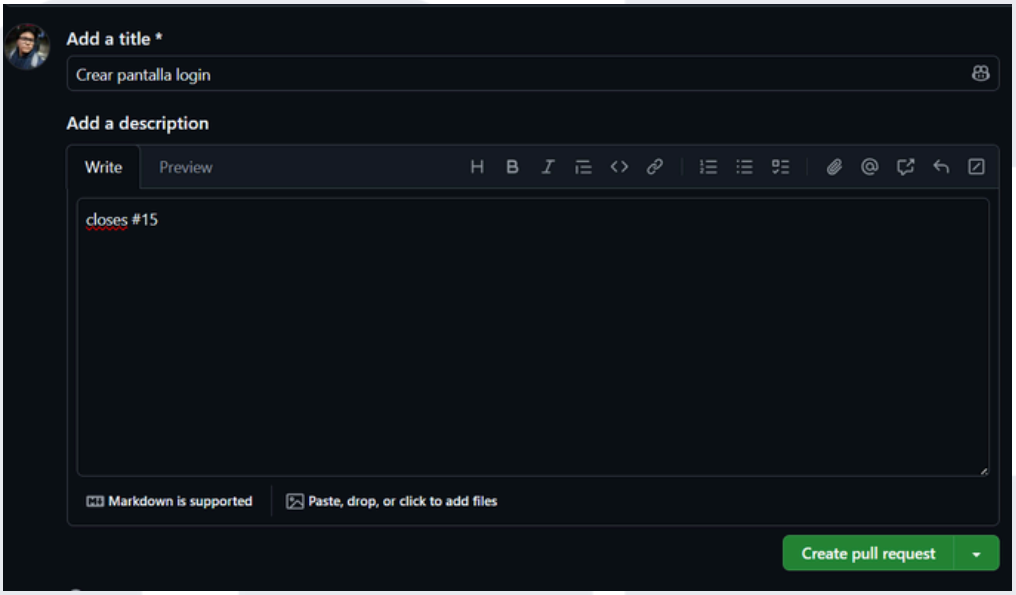
Ejemplo:

Issue #15 Crear pantalla login

Pull Request: Agregar pantalla login

Dentro de la descripcion del Pull Request se debe indicar la Issue correspondiente

Ejemplo: Closes #15



7. Cerrar una Issue

Github puede cerrar una Issue automáticamente cuando un Pull Request es aceptado.

Dentro del Pull Request se debe escribir: Closes #15 o Fixes #15 (El numero “15” es variable, ese es el ID de la issue y cada issue tiene uno diferente)

Cuando el Pull Request sea fusionado con la rama principal, Github cerrará autoáticamente la issue relacionada.

No es necesario cerrar la issue manualmente

8. Revisión de una tarea

Antes de aceptar un Pull Request es importante revisar:

- Que la tarea funciona correctamente.
- Que cumpla con los requisitos de la Issue
- Que no rompa otras funcionalidades
- Que el código sea entendible

La revisión ayuda a detectar errores antes de que lleguen al proyecto principal

9. Reglas básicas del proyecto

1. No trabajar directamente sobre la rama principal
2. Toda tarea debe tener una Issue
3. Toda Issue debe tener una rama
4. Toda rama debe terminar en un Pull Request
5. No cerrar Issues manualmente si existe un Pull Request asociado
6. Si una tarea no está registrada en Github Projects, la tarea no existe
7. Solo una persona debe trabajar en una misma Issue al mismo tiempo
8. Antes de comenzar una tarea, actualizar siempre la rama principal

10. Flujo Completo

Supongamos que existe la siguiente tarea:

#15 Crear pantalla login

Paso 1: El desarrollador se asigna la tarea

Paso 2: Mueve la Issue a In Progress

Paso 3: Crea una rama “feat/15-login-screen”

Paso 4: Realiza los cambios necesarios

Paso 5: Guarda los cambios mediante commits

Paso 6: Sube la rama a Github

Paso 7: Crea un Pull Request

Paso 8: En la descripción escribe “Closes #15”

Paso 9: El Pull Resquest es revisado y aprobado

Paso 10: Se realiza el Merge

Resultado:

- La funcionalidad queda integrada al proyecto
- La Issue se cierra automáticamente
- La tarea puede moverse a Done

11. Conclusión

Trabajar en equipo no consiste únicamente en programar. También implica organizar tareas, comunicar avances y mantener un flujo de trabajo ordenado.

Utilizar correctamente Issues, ramas y Pull Requests permite que cualquier integrante del equipo pueda colaborar sin generar conflictos o perder cambios importantes.

Mientras todos sigan el mismo flujo de trabajo, el proyecto será más fácil de mantener, revisar y mejorar con el paso del tiempo.