El trabajo colaborativo en Git se basa en la creación de ramas (*branches*), los *commits* que registran cambios y las fusiones (*merges*) que integran desarrollos paralelos en la rama principal. A continuación, se describe este proceso paso a paso usando GitHub Desktop.

Creación de ramas para

En un entorno colaborativo, cada desarrollador trabaja en su propia rama. Esto permite desarrollar nuevas funcionalidades o resolver errores sin modificar directamente la rama principal (*main* o *master*).

Pasos para crear una rama en GitHub Desktop:

1. Abrir GitHub Desktop y seleccionar el repositorio.
2. Estando en la rama principal, hacer clic en **“Current branch”** y seleccionar **“New branch”**.
3. Asignar un nombre significativo a la nueva rama (ej. `feature-login` o `mi-rama`).
4. Publicar la rama con el botón **“Publish branch”** para hacerla visible en el repositorio remoto.

Una vez creada y publicada, la rama puede usarse para registrar cambios sin afectar el estado del proyecto en la rama principal.

Realizar cambios en una rama

Con la rama activa, el desarrollador puede modificar archivos del proyecto en su editor de código (por ejemplo, Visual Studio Code). Luego, los cambios se gestionan desde GitHub Desktop.

Flujo básico:

1. Editar el archivo correspondiente en el editor.
2. Volver a GitHub Desktop, donde se visualizarán los cambios.
3. Escribir un mensaje descriptivo para el *commit* (ej. "Cambio en mi rama personal").
4. Confirmar el *commit* y hacer **push** para subirlo al repositorio remoto.

Cada *commit* representa una unidad de trabajo, lo que facilita el seguimiento de cambios y la colaboración (Loeliger & McCullough, 2012).

Integración de cambios a la rama principal (merge)

Una vez finalizado el desarrollo o la funcionalidad en una rama, esta puede fusionarse (*merge*) con la rama principal.

Pasos para fusionar:

1. Cambiar a la rama principal desde GitHub Desktop.
2. Hacer clic en **“Branch” > “Merge into current branch”**.
3. Seleccionar la rama que contiene los cambios.
4. GitHub Desktop mostrará las diferencias entre ramas.
5. Si no hay conflictos, se realiza el *merge* mediante un *merge commit*.
6. Subir los cambios al remoto con un **push**.

Este procedimiento asegura que la rama principal se actualice solo con cambios verificados y completos (GitLab Docs, s.f.-a).

Trabajo en equipo: múltiples ramas colaborativas

Cuando se trabaja en equipo, cada colaborador debe crear su propia rama para desarrollar funcionalidades específicas. El procedimiento es el mismo:

1. Crear y publicar una nueva rama (ej. `rama-colega`).
2. Realizar los cambios, hacer *commits* y subirlos.
3. Una vez terminado el trabajo, fusionar la rama con *main* siguiendo el mismo procedimiento anterior.

Este modelo distribuye el trabajo y minimiza conflictos, ya que cada desarrollador trabaja de forma aislada hasta integrar sus avances (Chacon & Straub, 2014).

Historial y trazabilidad

El historial del repositorio permite verificar qué cambios fueron realizados, en qué ramas y por quién. Tras cada *merge*, la rama principal conserva el registro de todos los *commits* integrados, permitiendo reconstruir el flujo de trabajo y auditar el desarrollo del proyecto (GitHub Docs, s.f.-b).

Referencias

Chacon, S., & Straub, B. (2014). *Pro Git* (2.ª ed.). Apress. https://git-scm.com/book/en/v2

GitHub Docs. (s.f.-a). *Cloning and forking repositories*. GitHub. https://docs.github.com/en/desktop/contributing-and-collaborating-using-github-desktop

GitHub Docs. (s.f.-b). *Managing branches in GitHub Desktop*. GitHub. https://docs.github.com/en/desktop/overview/getting-started-with-github-desktop

GitLab Docs. (s.f.-a). *Git branching strategies*. GitLab. https://docs.gitlab.com/ee/topics/git/

Loeliger, J., & McCullough, M. (2012). *Version Control with Git* (2.ª ed.). O'Reilly Media.