



# GitHub

## **Ventajas y funcionalidades de GitHub**

GitHub es una plataforma de alojamiento de código basada en Git, que ofrece herramientas para la colaboración y el control de versiones. Algunas de sus **ventajas y funcionalidades principales** son:

---

### **Ventajas de GitHub**

1. **Gestión centralizada de repositorios** – Permite almacenar y gestionar proyectos en la nube.
2. **Colaboración en equipo** – Múltiples desarrolladores pueden trabajar en el mismo proyecto con facilidad.
3. **Historial y control de versiones** – Mantiene un registro completo de los cambios realizados en el código.
4. **Acceso desde cualquier lugar** – Al estar basado en la nube, puedes trabajar desde cualquier dispositivo.
5. **Integraciones con CI/CD** – Compatible con herramientas de automatización como GitHub Actions, Travis CI y Jenkins.
6. **Seguridad y copias de seguridad** – Repositorios protegidos y respaldos constantes del código.
7. **Documentación y wikis** – Permite agregar documentación en archivos README o en la Wiki del proyecto.
8. **Gestión de incidencias (Issues)** – Facilita la administración de errores y nuevas funcionalidades.
9. **Pull Requests** – Facilita la revisión y fusión de cambios en un flujo de trabajo colaborativo.
10. **GitHub Pages** – Permite desplegar sitios web estáticos directamente desde un repositorio.

11. **Marketplace e Integraciones** – Puedes conectar GitHub con múltiples herramientas externas.
12. **Seguridad avanzada** – Incluye escaneo de código para detectar vulnerabilidades.
13. **Control de acceso y permisos** – Administra roles y permisos para colaboradores en un proyecto.

## **Funcionalidades clave en GitHub**

Funcionalidad	Descripción
<b>Repositorios remotos</b>	Aloja código en la nube y permite colaborar con otros.
<b>Forks</b>	Crea una copia de otro repositorio para trabajar en paralelo.
<b>Pull Requests</b>	Solicita la integración de cambios en un proyecto.
<b>GitHub Actions</b>	Automatiza tareas como pruebas, despliegues y CI/CD.
<b>Issues</b>	Registra errores, mejoras y tareas pendientes.
<b>Proyectos</b>	Kanban integrado para gestión ágil de tareas.
<b>GitHub Pages</b>	Hospeda sitios web estáticos desde un repositorio.
<b>Wiki</b>	Permite documentar proyectos directamente en GitHub.
<b>Code Scanning</b>	Detecta vulnerabilidades en el código.
<b>Dependabot</b>	Mantiene dependencias actualizadas y seguras.
<b>Copilot</b>	Inteligencia artificial que ayuda a escribir código más rápido.
<b>Branch Protection</b>	Protege ramas críticas de cambios no autorizados.

## Tabla de comparación: Git vs GitHub vs GitHub Desktop

Aquí tienes una tabla con las principales similitudes y diferencias:

Característica	Git	GitHub	GitHub Desktop
<b>Definición</b>	Sistema de control de versiones distribuido.	Plataforma basada en Git para almacenar y gestionar repositorios en la nube.	Aplicación de escritorio para manejar Git de manera visual.
<b>Almacenamiento</b>	Local (en tu máquina).	En la nube.	Local (con sincronización con GitHub).
<b>Interfaz</b>	Línea de comandos (CLI).	Web y API.	Interfaz gráfica (GUI).
<b>Repositorios</b>	Solo locales.	Remotos (públicos o privados).	Sincroniza repositorios locales con GitHub.
<b>Colaboración</b>	Limitada, requiere compartir archivos manualmente.	Permite colaboración en equipo con Pull Requests e Issues.	Facilita el trabajo en equipo con sincronización fácil.
<b>Flujo de trabajo</b>	Usa comandos ( <code>git add</code> , <code>git commit</code> , <code>git push</code> ).	Usa UI web para administrar ramas, PRs, Issues.	Permite realizar commits, merges y push sin usar CLI.
<b>Hosting de código</b>	No incluye hosting.	Sí, permite almacenar código en repositorios públicos o privados.	No, pero permite sincronizar con GitHub.
<b>Integraciones CI/CD</b>	No incluye integración por defecto.	Sí, con GitHub Actions y otras herramientas.	No directamente, pero permite sincronizar código con GitHub.
<b>Ideal para</b>	Desarrolladores con experiencia en CLI.	Equipos colaborativos y proyectos públicos o privados.	Usuarios que prefieren interfaz gráfica sobre terminal.

## Pull Requests (PRs)

Un **Pull Request (PR)** es una solicitud para que los cambios en una rama sean revisados y fusionados en otra rama (generalmente en `main` o `develop`).

### ¿Para qué sirven?

- Facilitan la colaboración en equipo.
- Permiten revisar y aprobar cambios antes de integrarlos en la rama principal.
- Pueden incluir comentarios y revisiones por parte del equipo.


### Ejemplo de flujo de trabajo con un PR:

1. Creas una **nueva rama** para desarrollar una nueva funcionalidad:

```
git checkout -b feature-nueva-funcionalidad
```

2. Haces cambios en el código y los confirmas (commits).
3. Subes los cambios a GitHub:

```
git push origin feature-nueva-funcionalidad
```

4. Vas a GitHub y creas un **Pull Request** para fusionar ( `merge` ) tus cambios con la rama `main` o `develop`.
5. Tus compañeros pueden revisar el código y dejar comentarios.
6. Si todo está bien, aprueban el PR y se fusionan los cambios en `main`. 

◆ **En resumen:** Un PR permite fusionar código en equipo sin afectar la rama principal hasta que se aprueben los cambios.

### Issues (Problemas o tareas)

Un **Issue** en GitHub es una manera de **registrar problemas, errores, mejoras o tareas pendientes** en un proyecto.

### ¿Para qué sirven los Issues?

- Reportar errores (bugs).
- Sugerir nuevas características.

- Asignar tareas a miembros del equipo.
- Hacer seguimiento del progreso de un proyecto.

## Ejemplo de cómo usar un Issue en GitHub

1. Alguien encuentra un error en la aplicación.
2. Crea un **Issue** en GitHub con el título: "Error en la pantalla de inicio".
3. Describe el problema detalladamente (incluso con imágenes o logs).
4. Se asigna el Issue a un desarrollador del equipo.
5. El desarrollador corrige el error y crea un **Pull Request** con la solución.
6. Cuando el PR es aprobado y fusionado, se cierra el Issue.

◆ **En resumen:** Un Issue es como una "tarea pendiente" dentro de un proyecto en GitHub.

## Relación entre Issues y PRs

- Un **Issue** describe un problema o una mejora.
- Un **Pull Request** se usa para proponer cambios que solucionan un Issue.
- Cuando un PR se fusiona correctamente, el Issue relacionado se puede **cerrar automáticamente**.

👉 **Ejemplo:** Si un PR corrige el Issue #42, puedes escribir en la descripción del PR:

Closes #42

Cuando se fusione el PR, GitHub **cerrará automáticamente el Issue #42**. 🔥

## Resumen corto

Concepto	¿Qué es?	¿Para qué sirve?
<b>Pull Request (PR)</b>	Solicitud de cambios en el código.	Permite fusionar cambios después de revisión.
<b>Issue</b>	Un problema, error o tarea.	Facilita el seguimiento de bugs y mejoras.