Material descargable

egg

Gestión de Proyectos con Git y GitHub

¿Qué es GitHub?

GitHub es una plataforma en línea para almacenar y gestionar proyectos de software. Utiliza Git para llevar un registro de las versiones de los archivos, lo que resulta útil cuando varias personas colaboran en un mismo proyecto, ya que pueden ver quién hizo qué cambios y cuándo.

Lo interesante de GitHub es que te permite decidir si tu proyecto es público, accesible para todo el mundo, o privado, al cual solo pueden acceder las personas que tú elijas. Esta flexibilidad lo convierte en una herramienta ideal tanto para proyectos abiertos al público como para aquellos en los que trabajas de manera privada o en equipo.

Además, GitHub ofrece varias funcionalidades que mejoran el flujo de trabajo en el desarrollo de software, tales como:

- **Reportar y seguir problemas:** Si encuentras un error o una mejora pendiente en tu proyecto, puedes crear un "issue" para no olvidarlo.
- **Revisión de cambios:** Antes de integrar cambios al proyecto, puedes revisarlos y discutirlos con otros colaboradores.
- **Automatización de tareas:** GitHub cuenta con herramientas que permiten automatizar procesos como la ejecución de pruebas para verificar que el código funcione correctamente.
- **Wiki y Páginas de GitHub:** Te proporciona espacio para documentar tu proyecto y crear páginas web directamente desde tu repositorio.

¿Cuáles son las características más relevantes de GitHub?

- **Repositorios:** En GitHub, puedes crear repositorios para proyectos web, móviles o de software. Estos repositorios almacenan todos los archivos del proyecto, así como el historial de cambios de cada archivo.
- **Control de versiones:** GitHub utiliza Git para gestionar el control de versiones. Esto te permite hacer un seguimiento de los cambios realizados en

los archivos a lo largo del tiempo, revertir a versiones anteriores si es necesario y gestionar distintas versiones de tu proyecto.

- **Colaboración:** GitHub facilita la colaboración entre equipos. Varias personas pueden trabajar en el mismo proyecto, cada una en su propia rama, y luego fusionar sus cambios con la rama principal cuando estén listos.
- **Pull Requests (PR):** Los usuarios pueden hacer "pull requests" (solicitudes de incorporación) para sugerir cambios en un repositorio. Este proceso permite revisar, discutir y validar los cambios antes de integrarlos al proyecto.
- **Revisión de código:** Los colaboradores del proyecto pueden revisar los cambios propuestos en las PR, discutirlos y, si todo es correcto, fusionarlos con la rama principal del proyecto.
- **Issues:** GitHub ofrece herramientas para el seguimiento de problemas y tareas, lo que permite a los equipos registrar incidencias, solicitudes de nuevas funcionalidades y más.
- **Comunidad:** GitHub es una plataforma reconocida por su enfoque en proyectos de código abierto, donde los desarrolladores pueden contribuir a proyectos existentes o usar proyectos ajenos como base para sus propias iniciativas.

¿Cuáles son las funciones de los colaboradores?

Los colaboradores en un repositorio de GitHub son usuarios a quienes se les ha otorgado acceso para contribuir al repositorio. Agregar colaboradores es esencial en diversas situaciones, especialmente cuando se trabaja en equipo o se busca recibir contribuciones externas. Sus funciones son clave en los siguientes aspectos:

- **Proyectos en equipo:** Cuando trabajas en un proyecto junto con otros desarrolladores, agregar colaboradores es fundamental para permitirles contribuir directamente al repositorio. Esto facilita la colaboración, ya que pueden realizar commits, crear ramas y participar en revisiones de código.
- **Contribuciones externas en proyectos open source:** Si tu proyecto es de código abierto y deseas que otros desarrolladores participen, agregar colaboradores les otorga permisos más amplios para gestionar el repositorio. Esto es común en proyectos open source donde se confía en colaboradores habituales para que contribuyan de forma constante.
- **Revisión de código y calidad:** En proyectos que requieren un control de calidad riguroso, los colaboradores pueden ser responsables de realizar revisiones de código. Esto garantiza que el código sea examinado adecuadamente antes de ser integrado a la rama principal.
- **Roles específicos:** En proyectos grandes, puede ser necesario contar con

colaboradores que desempeñen roles específicos, como gestores de proyecto, diseñadores o encargados de la documentación. Estos roles necesitan permisos para gestionar ciertos aspectos del repositorio, como los issues o la documentación.

- **Delegación y escalabilidad:** A medida que un proyecto crece, delegar responsabilidades se vuelve crucial. Agregar colaboradores con permisos específicos ayuda a distribuir la carga de trabajo y a mantener el proyecto organizado, escalable y manejable.
- **Mantenimiento continuo:** En proyectos que requieren un mantenimiento constante, como aquellos en producción, tener colaboradores adicionales asegura que haya personal disponible para hacer actualizaciones rápidas, corregir errores y abordar problemas de seguridad de manera oportuna.

¿Qué son los permisos de un repositorio?

Los permisos en un repositorio de GitHub determinan qué acciones puede realizar un usuario dentro del mismo. Estos permisos varían desde solo lectura, que permite ver y descargar el código, hasta permisos de administración, que permiten modificar la configuración del repositorio, gestionar el acceso de otros colaboradores, y más. La configuración de estos permisos puede adaptarse según las necesidades del proyecto y del equipo, lo que facilita la gestión de roles y responsabilidades dentro del repositorio.

¿Para qué sirve el archivo “.gitignore” de un repositorio?

El archivo `.gitignore` se utiliza para especificar qué archivos y directorios Git debe ignorar y no rastrear. Esto es útil para excluir archivos temporales, configuraciones locales, dependencias y datos sensibles que no deben compartirse en el repositorio. De esta manera, se mantiene el repositorio limpio y enfocado en los archivos esenciales, como el código fuente y los recursos relevantes para el proyecto.

¿Cuáles son los comandos para trabajar con el repositorio remoto?

A continuación se describen algunos de los comandos más utilizados para trabajar con un repositorio remoto:

- **`git clone <link github>`**: Este comando se utiliza para crear una copia local de un repositorio existente en GitHub (o en cualquier otro servicio de alojamiento Git). La URL del repositorio (`<link github>`) debe proporcionarse para clonar el repositorio, lo que descarga todo el contenido y el historial de commits en tu máquina local.

- **`git remote add origin <link github>`**: Este comando vincula un repositorio local con un repositorio remoto. El nombre `origin` es un alias convencional para el repositorio remoto principal. Este comando es útil cuando has creado un repositorio local y deseas conectarlo a uno remoto (como el que has creado en GitHub).
- **`git push origin main`**: Este comando sube los cambios de la rama principal del repositorio local al repositorio remoto en la nube. Es importante mencionar que, en equipos de trabajo reales, los cambios generalmente no se suben directamente a la rama `main` o `master`, sino que se gestionan mediante ramas específicas.
- **`git push origin nueva-rama`**: Este comando sube una nueva rama del repositorio local al repositorio remoto en la nube. Es útil para compartir ramas con otros colaboradores o para guardar el progreso del trabajo en el repositorio remoto.
- **`git push -u origin main`**: Este comando tiene dos propósitos:
    1. Sube los cambios de la rama `main` del repositorio local al repositorio remoto.
    2. Establece una rama remota como la "rama de seguimiento upstream" para la rama local actual. Esto significa que, para futuros comandos `git pull` o `git push`, no será necesario especificar el nombre del repositorio remoto ni la rama; Git recordará esta configuración por defecto.

Cada uno de estos comandos es fundamental en el flujo de trabajo de Git, permitiendo gestionar de manera eficiente el desarrollo de diferentes líneas en los proyectos y mantener la sincronización entre los repositorios locales y remotos.

GitHub tiene una **documentación oficial** completa y detallada. En ella encontrarás guías, manuales y recursos que cubren desde los conceptos básicos hasta los temas más avanzados, como la gestión de repositorios, colaboración en equipo, y estrategias de flujo de trabajo. La documentación se actualiza regularmente para incluir las últimas funciones, herramientas y mejoras de la plataforma.