Material descargable

egg

GIT: Control de Versiones

¿Qué es GitHub?

GitHub es una plataforma en línea donde puedes guardar y manejar tus proyectos de software. Utiliza Git para llevar un registro de las versiones que se hacen en los archivos del proyecto. Esto es muy útil cuando varias personas trabajan juntas en el mismo proyecto, ya que pueden ver quién hizo qué cambios y cuándo.

Lo interesante de GitHub es que puedes elegir si tu proyecto es visible para todo el mundo (repositorio público) o sólo para ciertas personas que elijas (repositorio privado). Esto lo hace ideal tanto para proyectos que quieres compartir con otros como para aquellos en los que trabajas en privado o en equipo.

Además, GitHub proporciona una serie de herramientas y características adicionales que mejoran el flujo de trabajo de desarrollo de software, tales como:

- Reportar y seguir problemas: Si hay un error o algo que mejorar en tu proyecto, puedes anotarlo en GitHub para no olvidarlo.
- Revisar cambios: Antes de que los cambios se añadan oficialmente al proyecto, puedes revisarlos y discutirlos.
- Automatizar tareas: Hay herramientas en GitHub que te ayudan a hacer cosas automáticamente, como probar tu código para ver si funciona bien.
- Wiki y Páginas de GitHub: Ofrecen espacio para la documentación del proyecto y páginas web alojadas directamente desde el repositorio.

¿Cuáles son las características más relevantes de esta plataforma?

- **Repositorios**: En GitHub, puedes crear repositorios para tus proyectos web, mobile o de software. Estos repositorios contienen todos los archivos del proyecto, así como el historial de revisiones de cada archivo.
- **Control de versiones**: GitHub utiliza Git para el control de versiones. Esto significa que puedes hacer un seguimiento de los cambios en tus archivos

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

- **Colaboración**: GitHub facilita la colaboración en proyectos. Varias personas pueden trabajar en el mismo proyecto, cada una en su propia rama o versión del repositorio, y luego fusionar sus cambios en la rama principal cuando estén listos.
- **Pull Requests (PR)**: Los usuarios pueden hacer "pull requests" (solicitud de incorporación) para sugerir cambios en un repositorio.
- **Code Review**: Los colaboradores del proyecto pueden revisar los cambios de la PR, discutirlos y eventualmente fusionarlos en la rama principal del proyecto.
- **Issues**: GitHub también proporciona herramientas para el seguimiento de problemas y tareas, permitiendo a los equipos mantener un registro de los problemas pendientes, solicitudes de características y más.
- **Comunidad**: Es conocido por ser un centro para proyectos de código abierto, donde los desarrolladores pueden contribuir a proyectos existentes o utilizar proyectos de otros como base para sus propios proyectos.

¿Cuáles son las funciones de los colaboradores?

Los colaboradores en un repositorio de GitHub son usuarios a los que se les ha concedido acceso para contribuir al repositorio. Agregar colaboradores a un proyecto de GitHub es necesario en varias situaciones, particularmente cuando se trabaja en equipo o se busca la contribución externa. Desempeñan un papel importante:

- **Proyectos en equipo**: Cuando estás trabajando en un proyecto con otros desarrolladores, agregarlos como colaboradores es esencial para permitirles contribuir directamente al repositorio. Esto facilita la colaboración, ya que pueden empujar cambios, 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 contribuyan, agregar colaboradores es una forma de darles permisos más amplios para gestionar el repositorio. Esto es común en proyectos open source donde confías en colaboradores habituales.

- **Revisión de código y calidad**: En proyectos que requieren un control de calidad estricto, los colaboradores pueden ser agregados para realizar revisiones de código. Esto asegura que el código se revise adecuadamente antes de integrarse en la rama principal.
- **Roles específicos**: Para proyectos más grandes, podrías necesitar personas con roles específicos (como gestores de proyecto, diseñadores, etc.) que requieran acceso para gestionar ciertos aspectos del repositorio, como la documentación o issues.
- **Delegación y escalabilidad**: A medida que un proyecto crece, puede ser necesario delegar responsabilidades. Agregar colaboradores con permisos específicos ayuda a distribuir la carga de trabajo y mantener el proyecto escalable y manejable.
- **Mantenimiento continuo**: En proyectos que requieren mantenimiento continuo, como aquellos usados en producción, tener colaboradores adicionales asegura que haya personal disponible para hacer actualizaciones rápidas, arreglar bugs y responder a problemas de seguridad.

¿Qué son los permisos de un repositorio?

Los permisos en un repositorio de GitHub definen lo que un usuario puede o no hacer dentro del mismo. Pueden variar desde solo lectura (acceso para ver el código y descargarlo) hasta permisos de administración, que permiten modificar la configuración del repositorio, gestionar los accesos de otros colaboradores y más. Dependiendo del tamaño y la naturaleza del proyecto, se puede ajustar la configuración de permisos para ajustarse a las necesidades específicas del equipo y del proyecto.

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

El archivo .gitignore en un repositorio se utiliza para especificar los archivos y directorios que Git debe ignorar y no rastrear. Esto es útil para excluir archivos como configuraciones locales, archivos temporales, dependencias y datos sensibles que no se deben compartir en el repositorio. Facilita la gestión de un repositorio limpio y enfocado en el código fuente y recursos relevantes.

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

Los comandos que nos ayudarán con esta tarea son:

git clone <link github>: Este comando se utiliza para crear una copia local de un repositorio que ya existe en GitHub (o en cualquier otro servicio de alojamiento de Git). El <link github> es la URL del repositorio que deseas clonar. Al ejecutar este comando, Git descarga todo el contenido del repositorio en tu máquina local, incluyendo todos los archivos y el historial de commits.

git remote add origin <link github>: Este comando se utiliza para vincular tu repositorio local con un repositorio remoto. Aquí, origin es un nombre convencional que se da al repositorio remoto principal. Esto es útil cuando has iniciado un repositorio local y luego quieres conectarlo con un repositorio remoto (por ejemplo, uno que has creado en GitHub).

git push origin main: Sube la rama principal desde el repositorio local al repositorio remoto en "la nube". En un equipo real de programadores, salvo ocasiones particulares, nunca se suben los cambios directamente a la rama main/master.

git push origin nueva-rama: Sube la nueva-rama al repositorio remoto en "la nube". Este comando es útil para compartir la rama con otros colaboradores o para guardar el progreso en el repositorio remoto.

git push -u origin main: Este comando se utiliza para dos propósitos principales:

- Subir cambios al repositorio remoto: En este caso, estás subiendo los cambios de la rama master de tu repositorio local al repositorio remoto.
- Establecer una rama de seguimiento upstream: La opción -u establece la rama remota especificada como la rama de seguimiento "upstream" para la rama local actual. Esto significa que, para futuros git pull o git push desde o hacia la rama main/master, no necesitarás especificar el nombre del repositorio remoto y la rama; Git recordará esta configuración y la utilizará por defecto.

Cada uno de estos comandos es una herramienta esencial en el flujo de trabajo de Git, permitiendo a los desarrolladores gestionar eficientemente las diferentes líneas de desarrollo en sus proyectos.