
ICARO
Nuestro compromiso es con el futuro.

# Taller GIT & GITHUB

# Control de versiones

# Control de versiones

Git y GitHub son herramientas esenciales para el desarrollo de software colaborativo. Git es un sistema de control de versiones que permite a los desarrolladores trabajar juntos en el mismo código, mientras que GitHub es una plataforma en línea que aloja proyectos de código abierto y ofrece herramientas para la colaboración y el seguimiento de problemas.

# ¿Qué es Git?

Git es un sistema de control de versiones de código abierto que permite a los desarrolladores rastrear los cambios en su código a lo largo del tiempo. Es una herramienta esencial para la colaboración y el trabajo en equipo en proyectos de software.

# Beneficios de usar Git

# Beneficios de usar Git

Git es una herramienta poderosa para el desarrollo de software que ofrece muchos beneficios a los desarrolladores. Algunos de los beneficios más destacados de usar Git son:

- **Historial detallado del código**:

- **Colaboración**:

- **Rastreo y reversión de cambios**:

# Beneficios de usar Git

Git es una herramienta poderosa para el desarrollo de software que ofrece muchos beneficios a los desarrolladores. Algunos de los beneficios más destacados de usar Git son:

- **Historial detallado del código**: Git mantiene un historial detallado de todos los cambios que se han realizado en un archivo o proyecto. Esto significa que siempre puedes consultar el historial y ver qué cambios se han realizado en el código en cualquier momento dado.
- **Colaboración**: Git es excelente para la colaboración en equipo. Puedes trabajar en un proyecto con otros desarrolladores sin temor a sobrescribir el trabajo de los demás. También puedes compartir fácilmente tu código con otros desarrolladores y recibir comentarios y sugerencias.
- **Rastreo y reversión de cambios**: Git te permite rastrear y revertir cambios en tu código, lo que significa que puedes volver atrás en el tiempo y deshacer cambios si algo sale mal. Esto es especialmente útil cuando trabajas en equipo, ya que varios desarrolladores pueden trabajar en el mismo archivo al mismo tiempo.

# ¿Qué es GitHub?

# ¿Qué es GitHub?

GitHub es una plataforma en línea que aloja proyectos de código abierto y ofrece herramientas para la colaboración y el seguimiento de problemas. Es una de las herramientas más populares para la gestión de proyectos de software en equipo.

# Beneficios de usar GitHub

# Beneficios de usar GitHub

GitHub es una plataforma de alojamiento de código en línea que utiliza Git como sistema de control de versiones. Algunos de los beneficios de usar GitHub son:

- **Colaboración en equipo**:

- **Revisión de código**:

- **Seguimiento de problemas**:

# Beneficios de usar GitHub

GitHub es una plataforma de alojamiento de código en línea que utiliza Git como sistema de control de versiones. Algunos de los beneficios de usar GitHub son:

- **Colaboración en equipo**: GitHub es una excelente herramienta para la colaboración en equipo. Puedes invitar a otros desarrolladores a colaborar en tu proyecto y gestionar permisos de acceso a tu repositorio. También puedes trabajar en el mismo proyecto sin temor a sobrescribir el trabajo de los demás.
- **Revisión de código**: GitHub ofrece herramientas para la revisión de código, que te permiten solicitar revisiones de código a otros desarrolladores y gestionar comentarios y cambios sugeridos. Esto es especialmente útil cuando trabajas en equipo y quieres asegurarte de que el código cumpla con los estándares de calidad.
- **Seguimiento de problemas**: GitHub ofrece herramientas para la gestión de problemas, que te permiten gestionar problemas y solicitudes de características en tu proyecto. Puedes asignar problemas a diferentes desarrolladores, establecer prioridades y etiquetas, y hacer un seguimiento del progreso.

# Configuración de Git

# Configuración de Git

Para empezar a usar Git, es necesario realizar una configuración inicial en tu máquina. La configuración incluye dos pasos principales:

1. **Instalación del software**:

2. **Configuración de tu identidad en Git**:

# Configuración de Git

Para empezar a usar Git, es necesario realizar una configuración inicial en tu máquina. La configuración incluye dos pasos principales:

1. **Instalación del software**: Antes de poder utilizar Git, debes instalarlo en tu máquina. Puedes descargar el instalador para tu sistema operativo desde la página oficial de Git. Una vez instalado, puedes abrir la consola de comandos y empezar a utilizar Git.
2. **Configuración de tu identidad en Git**: Después de instalar Git, es importante configurar tu identidad en Git. Esto significa establecer tu nombre y dirección de correo electrónico para que los cambios que realices en el código estén correctamente atribuidos. Para hacer esto, puedes utilizar los siguientes comandos de Git:

    *git config --global user.name "Tu nombre"*

    *git config --global user.email "tu.correo@ejemplo.com"*

# Comandos básicos de Git

# Comandos básicos de Git

- **git init**:
- **git add**:
- **git commit**:
- **git push**:

# Comandos básicos de Git

Git utiliza una serie de comandos para realizar operaciones sobre los repositorios. A continuación se describen algunos de los comandos básicos de Git:

- **git init**: Este comando inicializa un nuevo repositorio de Git en tu máquina. Este comando normalmente se utiliza una vez por proyecto.
- **git add**: Este comando añade un archivo al índice de Git. El índice es un área intermedia donde se preparan los cambios antes de confirmarlos.
- **git commit**: Este comando confirma los cambios que has realizado en el código. Al realizar un commit, es importante escribir un mensaje descriptivo que explique los cambios realizados.
- **git push**: Este comando envía los cambios confirmados a un repositorio remoto, como GitHub. Para utilizar este comando, debes tener permisos para enviar cambios al repositorio remoto.

# Comandos básicos de Git

Ejemplo para subir cambios:

- git add .
- git commit -m ‘nombre-commit’
- git push

# Comandos básicos de Git

Ejemplo para subir cambios:

- git add .
- git commit -m ‘nombre-commit’
- git push

Además de estos comandos, Git ofrece muchos otros comandos y opciones para realizar operaciones avanzadas sobre los repositorios. Es importante aprender bien los comandos básicos antes de avanzar a operaciones más complejas.

# Crear un repositorio en GitHub

# Crear un repositorio en GitHub

GitHub es una plataforma de alojamiento de repositorios de Git en línea que te permite compartir tu código con otros desarrolladores. Para crear un repositorio en GitHub, sigue los siguientes pasos:

1. Crea una cuenta en GitHub si no la tienes. Puedes crear una cuenta de forma gratuita en la página web de GitHub.
2. Inicia sesión en tu cuenta de GitHub.
3. Haz clic en el botón "New" (Nuevo) en la parte superior izquierda de la pantalla.
4. Introduce el nombre del repositorio y una descripción opcional.
5. Elige la visibilidad del repositorio: público o privado.
6. Si deseas añadir un archivo README.md (una descripción del proyecto), marca la casilla correspondiente.
7. Haz clic en el botón "Create repository" (Crear repositorio).
8. ¡Listo! Tu repositorio ha sido creado en GitHub.

# Clonar un repositorio

# Clonar un repositorio

Clonar un repositorio de GitHub te permite tener una copia local del repositorio en tu máquina. Para clonar un repositorio de GitHub, sigue los siguientes pasos:

1. Abre GitHub e inicia sesión en tu cuenta.
2. Busca el repositorio que deseas clonar.
3. Haz clic en el botón "Code" (Código) y copia la URL del repositorio.
4. Abre la línea de comandos en tu máquina.
5. Navega a la ubicación donde deseas clonar el repositorio.
6. Escribe el comando "git clone" seguido de la URL del repositorio y presiona Enter.
7. ¡Listo! El repositorio ha sido clonado en tu máquina.

Una vez que hayas clonado el repositorio, puedes trabajar en él en tu máquina y realizar cambios utilizando los comandos básicos de Git. Si deseas enviar tus cambios al repositorio en línea, utiliza los comandos "git add", "git commit" y "git push" para confirmar y enviar tus cambios al repositorio en GitHub.

# Trabajar con ramas

# Trabajar con ramas

- **git branch**:
- **git branch [nombre de la rama]**:
- **git checkout [nombre de la rama]**:
- **git merge [nombre de la rama]**:
- **git push [nombre del repositorio] [nombre de la rama]**:

# Trabajar con ramas

Las ramas en Git te permiten trabajar en diferentes versiones de tu código al mismo tiempo. Puedes crear nuevas ramas para trabajar en nuevas características o correcciones de errores sin afectar la rama principal. Algunos comandos útiles para trabajar con ramas son:

- **git branch**: muestra una lista de todas las ramas en tu repositorio y resalta la rama actual.
- **git branch [nombre de la rama]**: crea una nueva rama con el nombre especificado.
- **git checkout [nombre de la rama]**: cambia a la rama especificada.
- **git merge [nombre de la rama]**: fusiona la rama especificada con la rama actual.
- **git push [nombre del repositorio] [nombre de la rama]**: envía una rama específica al repositorio remoto. El nombre del repositorio se suele abreviar como “**origin**”.

Es importante tener en cuenta que al trabajar con ramas, es una buena práctica crear una nueva rama para cada nueva característica o corrección de errores. De esta manera, puedes trabajar en diferentes partes de tu código sin afectar la rama principal, y también puedes colaborar con otros desarrolladores en diferentes ramas.

# Trabajar con repositorios remotos

# Trabajar con repositorios remotos

Git te permite trabajar con repositorios remotos, como los alojados en GitHub. Puedes enviar tus cambios a un repositorio remoto usando el comando git push y descargar cambios desde un repositorio remoto usando el comando git pull. Para asociar un repositorio remoto con tu repositorio local, debes usar el comando git remote add. Luego, puedes enviar tus cambios a ese repositorio remoto con el comando git push y descargar los cambios del repositorio remoto con el comando git pull. Al trabajar con repositorios remotos, es importante tener cuidado para evitar sobrescribir los cambios de otros desarrolladores. Por lo tanto, es recomendable trabajar en ramas separadas y fusionarlas solo cuando estén completamente probadas y listas para ser integradas.

# Fusionar ramas

# Fusionar ramas

Cuando hayas terminado de trabajar en una rama, puedes fusionarla con la rama principal usando el comando git merge. Esto te permitirá combinar los cambios de ambas ramas y actualizar la rama principal con los cambios realizados en la rama secundaria. En algunos casos, la fusión se realizará automáticamente, pero en otros casos es posible que debas resolver conflictos manualmente.

# Resolver conflictos de fusión

# Resolver conflictos de fusión

A veces, Git no puede fusionar los cambios automáticamente y se producen conflictos de fusión. Esto ocurre cuando dos ramas han modificado la misma sección del código de forma diferente. Para resolver estos conflictos, debes revisar los cambios realizados en ambas ramas y decidir cómo se debe combinar el código. Esto se hace manualmente editando el código para eliminar las secciones conflictivas y conservar las secciones relevantes de ambas ramas. Una vez que hayas resuelto los conflictos, puedes realizar la fusión.

# Revisión de código

# Revisión de código

GitHub ofrece herramientas para la revisión de código, que te permiten solicitar revisiones de código a otros desarrolladores y gestionar comentarios y cambios sugeridos. Los desarrolladores pueden comentar directamente en el código y hacer sugerencias para mejorarlo.

# Gestión de problemas

# Gestión de problemas

GitHub ofrece herramientas para la gestión de problemas, que te permiten gestionar problemas y solicitudes de características en tu proyecto. Puedes asignar problemas a diferentes desarrolladores, establecer prioridades y etiquetas, y hacer un seguimiento del progreso.

# Recursos adicionales

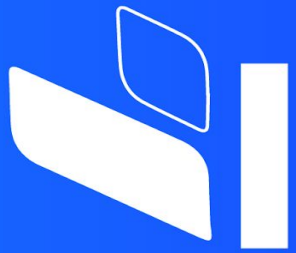
Hay muchos recursos disponibles en línea para aprender más sobre Git y GitHub, como tutoriales, documentación oficial y foros de discusión.

- Documentación oficial Git
- Documentación oficial GitHub
- Git y GitHub Mozilla Developers

# Conclusión

Git y GitHub son herramientas esenciales para el desarrollo de software en equipo. Como desarrolladores de software, es importante que aprendamos a usar estas herramientas para colaborar eficazmente en proyectos de software.

# ¡Muchas gracias!



ICARO Asociación Civil
CUIT 30716564815
info@icaro.org.ar
www.icaro.org.ar