Introducción a Control de versiones y repositorios remotos

Git

Definición: Git es un sistema de control de versiones distribuido y de código abierto.
Función: Permite a los desarrolladores realizar un seguimiento de los cambios en el código fuente de un proyecto a lo largo del tiempo.
Características:

- Registra cada cambio realizado en los archivos.
- Permite volver a versiones anteriores del código.
- Facilita la colaboración entre desarrolladores.
- Gestiona ramas para trabajar en nuevas características sin afectar la rama principal.

GitHub

- **Definición**: GitHub es una plataforma en línea que ofrece servicios de alojamiento de repositorios Git.
- **Función**: Permite a los desarrolladores almacenar, compartir y colaborar en proyectos de software utilizando Git.
- **Características**:
    - Aloja repositorios Git de forma remota.
    - Ofrece herramientas para la gestión de proyectos, seguimiento de errores y revisión de código.
    - Facilita la colaboración entre equipos de desarrollo.
    - Permite compartir proyectos de código abierto con la comunidad.

En resumen

- **Git**: Es la herramienta que permite realizar el control de versiones de tu código.
- **GitHub**: Es la plataforma que te permite almacenar tus repositorios Git en la nube y colaborar con otros desarrolladores.

Instalación de Git:

- Descarga Git desde el sitio web oficial: https://git-scm.com/
- Sigue las instrucciones de instalación para tu sistema operativo.
- Asegúrate de agregar Git a tu variable de entorno PATH durante la instalación.

1. Descarga e instalación de GitHub Desktop

- Ve al sitio web de GitHub Desktop: https://desktop.github.com/
- Descarga la versión para tu sistema operativo (Windows o macOS).
- Ejecuta el instalador descargado y sigue las instrucciones.

2. Inicio de sesión en GitHub Desktop

- Abre GitHub Desktop.
- Si ya tienes una cuenta de GitHub, haz clic en "Sign in to GitHub.com" e inicia sesión con tus credenciales.
- Si no tienes una cuenta, puedes crear una gratuitamente haciendo clic en "Create a GitHub account".

3. Configuración de Git

- Después de iniciar sesión, GitHub Desktop te pedirá que configures tu nombre de usuario y dirección de correo electrónico para Git. Esta información se utilizará para identificar tus commits.
- Ingresa tu nombre y correo electrónico y haz clic en "Continue".

4. Creación de un repositorio local

- Si quieres crear un nuevo repositorio para tu proyecto, haz clic en "Create a new repository on your computer".
- Elige la ubicación para tu repositorio local, el nombre del repositorio y una descripción (opcional).
- Si quieres agregar un archivo README al repositorio, marca la casilla correspondiente.
- Haz clic en "Create repository".

5. Publicación del repositorio en GitHub

- Si ya tienes un repositorio local y quieres publicarlo en GitHub, haz clic en "Publish repository".

- Elige el nombre para tu repositorio en GitHub y una descripción (opcional).
- Puedes elegir mantener el repositorio privado o hacerlo público.
- Haz clic en "Publish repository".

6. Uso de GitHub Desktop

- Ahora puedes usar GitHub Desktop para realizar diversas tareas, como:
    - **Hacer commits**: Selecciona los archivos que quieres incluir en el commit, escribe un mensaje descriptivo y haz clic en "Commit to main" (o el nombre de tu rama).
    - **Enviar cambios (push)**: Haz clic en "Push origin" para enviar tus commits al repositorio remoto en GitHub.
    - **Descargar cambios (pull)**: Haz clic en "Pull origin" para descargar los cambios desde el repositorio remoto a tu repositorio local.
    - **Crear ramas**: Haz clic en "New branch" para crear una nueva rama para trabajar en nuevas características sin afectar la rama principal.
    - **Cambiar de rama**: Selecciona la rama a la que quieres cambiar en el menú desplegable.

Consejos adicionales

- **.gitignore**: Crea un archivo ".gitignore" en la raíz de tu proyecto para especificar qué archivos y carpetas no quieres que se incluyan en el repositorio. Esto es útil para archivos temporales, archivos de configuración sensibles, etc.
- **Ramas**: Utiliza ramas para trabajar en nuevas características o solucionar errores. Esto te permite mantener tu rama principal limpia y estable.
- **Mensajes de commit**: Escribe mensajes de commit claros y descriptivos para que sea fácil entender los cambios que has realizado.
- **Actualizaciones**: Mantente al día con las últimas actualizaciones de GitHub Desktop para aprovechar las nuevas características y mejoras.