Identifica herramientas de versionamiento GA7-220501096-AA1-EV03

Aprendiz:

Valentina Vargas Sanchez

Instructor

Yerman Augusto Hernández Sáenz

CENTRO DE SERVICIOS FINANCIEROS

SENA- REGIONAL DISTRITO CAPITAL

ANALISIS Y DESARROLLO DEL SOFTWARE

FICHA: 2627062

2024

Introducción

Git y GitHub son herramientas fundamentales en el desarrollo de software, proporcionando un robusto sistema de control de versiones y una plataforma de colaboración en la nube. Git, un sistema distribuido, permite a los desarrolladores gestionar cambios de manera eficiente en proyectos locales. En contraste, GitHub, una plataforma basada en la web, sirve como un repositorio remoto que facilita la colaboración, seguimiento de problemas, y gestión de proyectos. A continuación, se presentan algunos comandos básicos que ayudan a iniciar con Git y GitHub, permitiendo controlar versiones y colaborar efectivamente en los proyectos de desarrollo de software.

Git	GitHub
Diferencias	
• Git es un sistema de control de versiones distribuido. • Gestiona y controla las versiones del código fuente en un proyecto de desarrollo de software. • Funciona localmente en la máquina del desarrollador, lo que significa que no requiere conexión a internet para realizar operaciones de control de versiones.	• GitHub es una plataforma de alojamiento para proyectos que utilizan Git como sistema de control de versiones. • Proporciona servicios centralizados en la nube para alojar, colaborar y gestionar proyectos de software basados en Git. • Funciona en línea y permite a los desarrolladores colaborar en proyectos de forma remota. Ofrece características adicionales.
Características	

• **Distribuido:** Git es un sistema de control de versiones distribuido, lo que significa que cada desarrollador tiene una copia completa del historial del proyecto en su máquina local. • **Eficiencia:** Es eficiente en el manejo de grandes proyectos y conjuntos de datos, ya que solo guarda las diferencias entre las versiones. • **Ramas (Branches):** Permite la creación de ramas para desarrollar nuevas características de manera aislada antes de fusionarlas con la rama principal. • **Historial de cambios:** Mantiene un historial completo de cambios, lo que facilita la identificación de quién hizo qué y cuándo. • **Flexibilidad:** Ofrece flexibilidad en el flujo de trabajo y permite a los desarrolladores trabajar de diversas maneras según sus necesidades. • **Velocidad:** Las operaciones locales en Git son rápidas, ya que la mayoría de las operaciones se realizan en la máquina del desarrollador.	• **Almacenamiento remoto:** GitHub proporciona un lugar centralizado en la nube para almacenar y gestionar repositorios Git de forma remota. • **Colaboración:** Facilita la colaboración entre desarrolladores permitiendo el trabajo conjunto en proyectos a través de solicitudes de extracción, problemas, y comentarios en el código. • **Seguimiento de problemas (Issue tracking):** Permite el seguimiento y la gestión de problemas y tareas relacionadas con el proyecto. • **Solicitudes de extracción (Pull Requests):** Facilita la integración de cambios realizados en ramas secundarias de un repositorio a la rama principal, permitiendo revisión y discusión antes de la fusión. • **Integración continua:** Permite la integración continua con servicios de CI/CD (Continuous Integration/Continuous Deployment) para automatizar pruebas y despliegues. • **Visibilidad:** Proporciona una plataforma pública para proyectos de código abierto, permitiendo la visibilidad y la contribución de la comunidad. • **Gestión de equipos:** Ofrece herramientas para gestionar y asignar roles a los colaboradores, lo que facilita la administración de equipos de desarrollo.

Comandos	
`git init` **Descripción:** Inicializa un nuevo repositorio Git en el directorio actual. ***`git clone <URL>`*** **Descripción:** Clona un repositorio remoto existente en la máquina local. ***`git add <archivo>`*** **Descripción:** Agrega cambios de archivos al área de preparación (staging) para ser incluidos en el próximo commit. ***`git commit -m "Mensaje del commit"`*** **Descripción:** Crea un nuevo commit con los cambios en el área de preparación y agrega un mensaje descriptivo. ***`git status`*** **Descripción:** Muestra el estado actual del repositorio, incluyendo archivos modificados y pendientes de commit. ***`git log`*** **Descripción:** Muestra el historial de commits del repositorio. ***`git Branch`*** **Descripción:** Muestra las ramas disponibles en el repositorio. ***`git checkout <nombre-de-rama>`*** **Descripción:** Cambia a la rama especificada. ***`git merge <nombre-de-rama>`*** **Descripción:** Fusiona la rama especificada con la rama actual.	***`git remote add origin <URL>`*** **Descripción:** Asocia el repositorio local con un repositorio remoto en GitHub. ***`git remote -v`*** **Descripción:** Muestra las URLs de los repositorios remotos asociados. ***`git push -u origin <nombre-de-rama>`*** **Descripción:** Configura el seguimiento de una rama local con su contraparte remota. ***`git pull origin <nombre-de-rama>`*** **Descripción:** Descarga los cambios más recientes desde la rama remota específica en GitHub a la rama local. ***`git clone <URL>`*** **Descripción:** Clona un repositorio remoto desde GitHub en la máquina local. ***`git fork`*** **Descripción:** Crea una copia del repositorio de otro usuario en tu propia cuenta de GitHub. ***`git pull-request`*** **Descripción:** Crea una solicitud de extracción en GitHub para proponer cambios en un repositorio. ***`git issues`*** **Descripción:** Lista y muestra los problemas (issues) en un repositorio de GitHub.

`git pull` **Descripción:** Descarga los cambios más recientes desde el repositorio remoto a la rama actual. ***`git push`*** **Descripción:** Sube los cambios locales al repositorio remoto.	

Conclusiones

En resumen, Git y GitHub forman una combinación poderosa para el desarrollo de software. Git ofrece un sistema de control de versiones distribuido, mientras que GitHub proporciona una plataforma en la nube para colaboración y gestión de proyectos. Al comprender y utilizar los comandos básicos de Git y GitHub, los desarrolladores pueden controlar eficazmente versiones, colaborar de manera eficiente y aprovechar las herramientas que estas tecnologías ofrecen para mejorar el flujo de trabajo en el desarrollo de software.