PROYECTO GIT Y GITHUB

Esteban David Hernandez Parra - 0000307597

Manuel José Cardozo Vanegas - 0000303159

Amy Jeanine Nossa Ramirez - 296077

Contenido

1. ¿Qué es Git y cuál es su propósito principal en el desarrollo de software?

Git es un sistema de control de versiones distribuido que permite a los desarrolladores gestionar el historial de cambios en el código fuente de un proyecto. Su propósito principal es facilitar la colaboración en equipo, el rastreo de modificaciones y la posibilidad de volver a versiones anteriores si se presentan problemas.

2. ¿Cómo se diferencia GitHub de Git en términos de funcionalidad?

Git es la herramienta que se utiliza para llevar el control de versiones a nivel local, mientras que GitHub es una plataforma en la nube que actúa como un repositorio remoto para almacenar proyectos basados en Git. GitHub también ofrece funcionalidades adicionales como revisión de código, gestión de proyectos y colaboración en equipo.

3. ¿Qué es un repositorio en GitHub y para qué se utiliza?

Un repositorio en GitHub es un espacio donde se almacena todo el código de un proyecto, junto con su historial de versiones y archivos relacionados. Se utiliza para organizar proyectos, colaborar con otros desarrolladores y compartir el trabajo de forma pública o privada.

4. ¿Qué es GitHub Desktop?

GitHub Desktop es una aplicación que permite usar Git y gestionar repositorios de GitHub desde una interfaz gráfica, sin necesidad de usar comandos en la terminal. Es útil para los que prefieren una experiencia más visual al trabajar con Git, este mismo incluye funciones adicionales como colaboración, gestión de problemas, revisión de código y visibilidad publico/ privada de los proyectos.

5. ¿Qué es "pull" y "push"?

Pull: Descarga los cambios más recientes desde un repositorio remoto hacia tu copia local.
Push: Sube los cambios que has hecho localmente al repositorio remoto.
Ambos son esenciales para sincronizar el trabajo entre tu máquina y el repositorio remoto.

6. ¿Qué es un "commit" y cuándo deberías realizarlo?

Un *commit* es un registro de los cambios realizados en los archivos de un proyecto. Se debe realizar un commit cada vez que se completa una tarea o se alcanza un punto importante en el desarrollo, asegurando que los cambios sean claros y tengan un mensaje descriptivo.

7. ¿Por qué no es lo mismo hacer "commit" que guardar el archivo?

Guardar un archivo simplemente actualiza su contenido en tu máquina, mientras que hacer un commit registra esos cambios en el historial de versiones de Git, permitiendo rastrear y revertir cambios si es necesario.

8. ¿Qué es la herramienta "blame" en GitHub y cuándo sería útil?

La herramienta *blame* en GitHub muestra quién hizo cada cambio en un archivo específico y en qué línea. Es útil para identificar el autor de un cambio, resolver dudas o buscar la raíz de un error en el código.

9. ¿Qué es una rama en Git y cuál es su función?

Una rama en Git es una línea separada de desarrollo que permite trabajar en características nuevas o correcciones de errores sin afectar la rama principal. Facilita que varios desarrolladores trabajen en paralelo.

10. ¿Por qué es recomendable utilizar ramas en lugar de hacer commits directamente en la rama principal?

Usar ramas reduce el riesgo de introducir cambios inestables o con errores en la rama principal. Además, permite que diferentes características o fixes se desarrollen y prueben por separado antes de fusionarse con la rama principal.

11. ¿Qué es un “merge” en Git? ¿Cuántos tipos de merge hay?

Un *merge* es el proceso de combinar los cambios de una rama en otra. Los tipos principales son:

- *Merge directo*: Cuando no hay conflictos y los cambios se integran automáticamente.
- *Merge con conflictos*: Ocurre cuando hay cambios contradictorios entre ramas, y es necesario resolverlos manualmente.

12. ¿Qué es un conflicto en Git?

Un conflicto ocurre cuando Git no puede fusionar automáticamente los cambios de dos ramas porque afectan la misma parte del código de manera diferente. En estos casos, el desarrollador debe resolver el conflicto manualmente.

13.¿Qué pasa si se envía un cambio no deseado a una rama en Git?

Se puede corregir utilizando herramientas como *revert*, que crea un nuevo commit para deshacer los cambios, o *reset* para eliminar el commit del historial (aunque este último debe usarse con cuidado).

14.¿Qué es un Pull Request y para qué beneficios tiene?

Un *Pull Request* (PR) es una solicitud para fusionar los cambios de una rama en otra. Permite a los miembros del equipo revisar el código, discutirlo y aprobarlo antes de integrarlo, asegurando mejor calidad y colaboración.

15. ¿Cuál es la importancia de asignar tareas específicas a cada miembro del equipo al trabajar en un proyecto colaborativo?

Asignar tareas específicas evita la duplicación de esfuerzos, mejora la organización, y asegura que todos sepan qué deben hacer. También permite aprovechar las fortalezas de cada miembro y hace que el equipo trabaje de manera más eficiente.

Referencias

- Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress.
- GitHub Docs. (n.d.). *GitHub Documentation*. Retrieved January 24, 2025, from https://docs.github.com